Nieliniowe systemy rozpraszające

Weronika Smagór, Aleksandra Motor, Patryk Polczyk, Krzysztof Konieczny
Fizyka Techniczna III rok
Wydział Inżynierii Materiałowej i Fizyki Politechniki Krakowskiej

Czerwiec 2021
Kraków

Spis treści

1 Punkty stałe i stabilność

Zakładamy, że dynamiczne zachowanie układu jest modelowane przez krzywe rozwiązania równań różniczkowych (lub układu dynamicznego), gdzie U jest otwartym podzbiorem R^n:

$$\frac{du}{dt} = f(u), \qquad f : U \longrightarrow R^n$$

Załóżmy, że f jest równe C^1. Punkt $u^* \in U$ jest nazywany punktem stałym (punktem równowagi lub punktem stacjonarnym) systemu dynamicznego, jeżeli:

$$f(u^*) = 0$$

Ze względu na wyjątkowość rozwiązania żadna inna krzywa rozwiązania nie może przejść przez u.

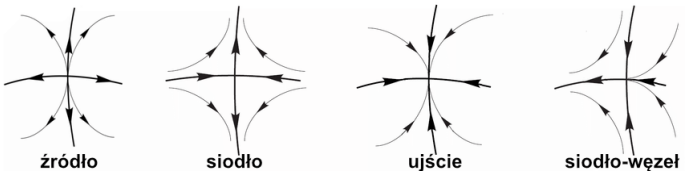
Niech $\Phi_t : U \longrightarrow R^n$ będzie przepływem związanym z układem dynamicznym. Zbiór $U \subset R^n$ jest zbiorem otwartym, a dla każdego $u \in U$ odwzorowanie $t \longrightarrow \Phi(t, u) = \Phi_t(u)$ jest rozwiązaniem przechodzącym przez u, gdy t=0; jest określony dla t w pewnym przedziale otwartym. Jeżeli u* jest punktem stały, to:

$$\Phi_t(u^*) = u^*$$

dla wszystkich $t \in R$. u* nazywamy także punktem osobliwym pola wektorowego f. Zakładając, że f jest liniowe, zapisujemy f(u)=Au, gdzie A jest liniowym operatorem na R^n. Wtedy początek $0 \in R^n$ jest punktem stałym. Gdy $\lambda < 0$ jest większa od rzeczywistych części wartości własnych A, rozwiązania $\Phi_t(u)$ zbliżają się do 0 wykładniczo dla pewnych $C > 0$.

$$|\Phi_t(u)| \leqslant Ce^{\lambda t}$$

Teraz załóżmy, że t jest wektorem pola C^1 z punktem stałym $0 \in R^n$. Zakładamy, że pochodna $Df(0) = A$ z f w 0 jest liniowym polem wektorowym, które zbliża się do f w pobliżu 0. Jeśli wszystkie wartości własne Df(0) mają ujemne części rzeczywiste, 0 nazywamy ujściem.



Rysunek 1: Rodzaje punktów stałych.

Ogólnie mówiąc, punkt stały u* jest ujściem, jeśli wszystkie wartości własne Df(u*) mają ujemne części rzeczywiste. Mówimy również, że przepływ liniowy e^{tA} jest skróceniem. Można wykazać, że 0 jest ujściem, wtedy i tylko wtedy, gdy każda trajektoria dąży do 0 gdy $t \longrightarrow \infty$. Nazywamy to stabilnością asymptotyczną. Wynika z tego, że trajektorie zbliżają się wykładniczo do ujścia. Nieliniowe ujście u* zachowuje się lokalnie jak ujście liniowe: pobliskie rozwiązania zbliżają się do u* wykładniczo.

Jako przykład rozważmy stabilność punktów stałych modelu Lorenza. Model Lorenza otrzymuje się jako przybliżenie do równań różniczkowych cząstkowych opisujących konwekcję w podgrzanej warstwie płynu (problem Bernarda). Model Lorenza podaje:

$$\frac{du_1}{dt} = \sigma(u_2 - u_1)$$

$$\frac{du_2}{dt} = -u_1u_3 + ru_1 - u_2$$

$$\frac{du_3}{dt} = u_1u_2 - bu_3$$

gdzie σ, r i b są dodatnimi stałymi.
Punkty stałe są wyznaczane przez układ równań algebraicznych:

$$u_2^* - u_1^* = 0$$

$$-u_1^*u_3^* + ru_1^* - u_2^* = 0$$

$$u_1^*u_2^* - bu_3^* = 0$$

Zakładamy, że $u_1^* = u_2^* = u_3^* = 0$ jest punktem stałym dla wszystkich σ, r i b. Ten stały punkt wychodzi dla wszystkich wartości parametrów σ, r i b. Jeśli $r < 1$ ten stały punkt jest przyciągany (ujście). Jeśli r staje się większe niż 1, ten stały punkt traci swój przyciągający charakter (jedna wartość własna staje się dodatnia) i pojawiają się dwa nowe stałe punkty.
Dla $r > 1$ znajdujemy punkty stałe:

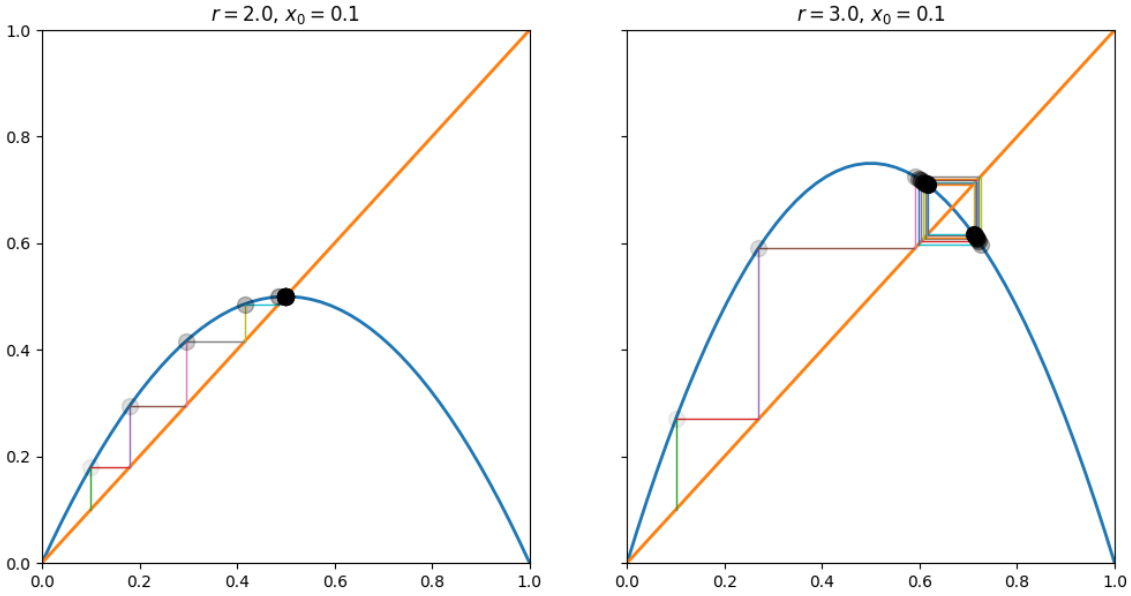
$$u_1^* = u_2^* = \pm\sqrt{b(r-1)}, \qquad u_3^* = r - 1$$

Nasz program znajduje równanie charakterystyczne dla wartości własnych z równań wariacyjnych i wyznacza stabilność punktu stałego (0,0,0). Wartości parametrów to r=40, σ=16 i b=4. Dla $r > 1$ początek (0,0,0) staje się niestabilny, tzn. jedna z trzech wartości własnych staje się dodatnia (podczas gdy pozostałe dwie pozostają ujemne).

dynamicFunc.py

```
1  def matFun(r, x):
2      #definiujemy funkcje matemtyczną
       dla ktorej bedziemy liczyli
       bifurkacje
3      # f(x) = rx(1 - x)
4      return r * x * (1 - x)
```

Rysunek 2: Program "dynamicFunc.py"

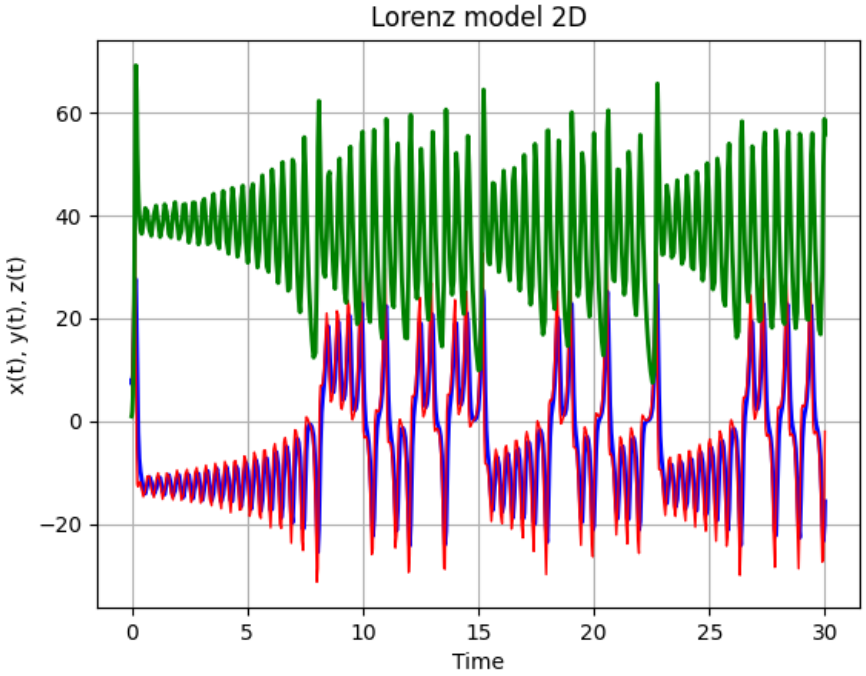


Rysunek 3: Wizualizacja zachowania funkcji dynamicznej.

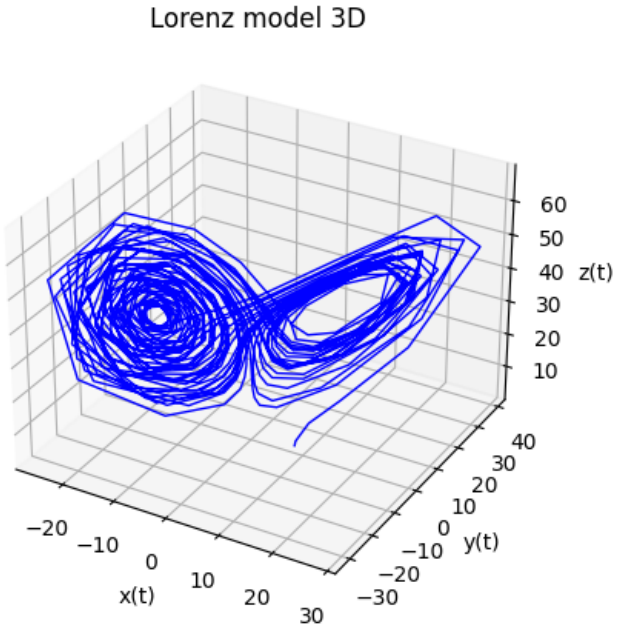
lorenzODE.py

```python
import numpy as np
from scipy.integrate import solve_ivp
import matplotlib.pyplot as plt
import matplotlib as mpl
from mpl_toolkits.mplot3d import Axes3D

def lorenzDeriv(t, xyz):
    #parametry zalezne od czasu
    beta, rho, sigma, t0, y0 = lorenzParameters()
    #inicjacja pustej macierzy z zerowymi parametrami
    dxdt = np.zeros(3)
    #model lorenza
    dxdt[0] = sigma * ( xyz[1] - xyz[0] )
    dxdt[1] = xyz[0] * ( rho - xyz[2] ) - xyz[1]
    dxdt[2] = xyz[0] * xyz[1] - beta * xyz[2]

    return dxdt

def lorenzPlot():
    #wywolywanie funkcji rysujacej wykresy
    t, x, y, z = lorenzSolveIvp()
    lorenz2dPlot(t, x, y, z)
    lorenz3dPlot(t, x, y, z)
    return

def lorenzSolveIvp ( ):
    #solve_ivp funkcja rozwiazuje problem wartosci poczatkowych
    beta, rho, sigma, t0, xyz0 = lorenzParameters()
    #tspan to wektor, ktory pobiera poczatkowa i koncowa wartosc parametru.
    #i liczy po nim wszystkie mozliwe pochodne y' = f(t,y)
    tspan = np.array([t0, 30.0])
    sol = solve_ivp(lorenzDeriv, tspan, xyz0)

    t = sol.t
    x = sol.y[0,:]
    y = sol.y[1,:]
    z = sol.y[2,:]

    return t, x, y, z

def lorenz2dPlot (t, x, y, z):

    plt.plot (t, x, linewidth = 2, color = 'b')
    plt.plot (t, y, linewidth = 1, color = 'r')
    plt.plot (t, z, linewidth = 2, color = 'g')
    plt.grid (True)
    plt.xlabel ('Time')
    plt.ylabel ('x(t), y(t), z(t)')
    plt.title ('Lorenz model 2D')
    plt.savefig('Plot2dLorenz.png')
    plt.show()


    return

def lorenz3dPlot ( t, x, y, z ):
    fig = plt.figure()
    ax = fig.gca(projection = '3d')
    ax.plot (x, y, z, linewidth = 1, color = 'b')
    ax.grid (True)
    ax.set_xlabel ('x(t)')
    ax.set_ylabel ('y(t)')
    ax.set_zlabel ('z(t)')
    ax.set_title ('Lorenz model 3D')
    plt.savefig ('Plot3dLorenz.png')
    plt.show ()
    plt.clf ()

    return

def lorenzParameters():
    beta = 4.0
    rho = 40.0
    sigma = 16.0

    t0 = 0.0
    y0 = np.array([8.0, 1.0, 1.0])

    return beta, rho, sigma, t0, y0

if ( __name__ == '__main__' ):
    lorenzPlot()
```

Rysunek 4: Program ”lorenzODE”



Rysunek 5: Wynik działania programu "Lorenz model 2D" - kolorem czerwonym zaznaczono współrzędne Z, niebieskim-Y, a zielonym-X.



Rysunek 6: Wynik działania programu "Lorenz model 3D"

2 Wykładniki Liapunova

Wykładniki Liapunova zapewniają sensowny sposób scharakteryzowania asymptotycznego zachowania nieliniowego układu dynamicznego, w którym f jest w sposób ciągły różniczkowalne.

$$\frac{du}{dt} = f(u), \qquad u(0) = u_0, \qquad u \in R$$

Stanowią one uogólnienie analizy stateczności liniowej punktów stałych. W przypadku ergodycznych układów dynamicznych wykładniki Liapunova są takie same dla prawie wszystkich warunków początkowych u_0 w odniesieniu do dowolnej miary niezmiennego przepływu. Oznacza to, że ich wartości nie zależą od określonej trajektorii. Dla danej trajektorii rozwiązania u(t) bierze się pod uwagę liniowe równanie wariacyjne, w którym $A(t) = \frac{\partial f}{\partial u}$ jest jakobianem w u(t), a Y jest macierzą $n \times n$ zależną od czasu.:

$$\frac{dY}{dt} = Df(u)Y = A(t)Y, \quad Y(0) = I$$

Wówczas dla n podstawowej macierzy rozwiązań Y(t) symetryczna dodatnia macierz:

$$\Lambda := \lim_{t \longrightarrow \infty} \Lambda_{u_0}(t) = \lim_{t \longrightarrow \infty} (Y^T(t)Y(t))^{\frac{1}{2t}}$$

jest dobrze zdefiniowana, a ^{T} oznacza transpozycję.

Jako przykład rozważamy model Lorenza, aby znaleźć największy jednowymiarowy wykładnik Liapunova. Równanie wariacyjne modelu Lorenza

$$\frac{du_1}{dt} = \sigma(u_2 - u_1)$$

$$\frac{du_2}{dt} = -u_1u_3 + ru_1 - u_2$$

$$\frac{du_3}{dt} = u_1u_2 - bu_3$$

jest dane przez

$$\frac{dv_1}{dt} = \sigma(v_2 - v_1)$$

$$\frac{dv_2}{dt} = (-u_3 + r)v_1 - v_2 - u_1v_3$$

$$\frac{dv_3}{dt} = u_2v_1 + u_1v_2 - bv_3$$

Największy jednowymiarowy wykładnik Liapunova wyraża się wzorem:

$$\lambda(u_1(0), u_2(0), u_3(0), v_1(0), v_2(0), v_3(0)) = \lim_{T \longrightarrow \infty} \frac{1}{T} \ln ||v(T)||$$

gdzie ||.|| oznacza dowolną normę w R^3. Wybieramy normę:

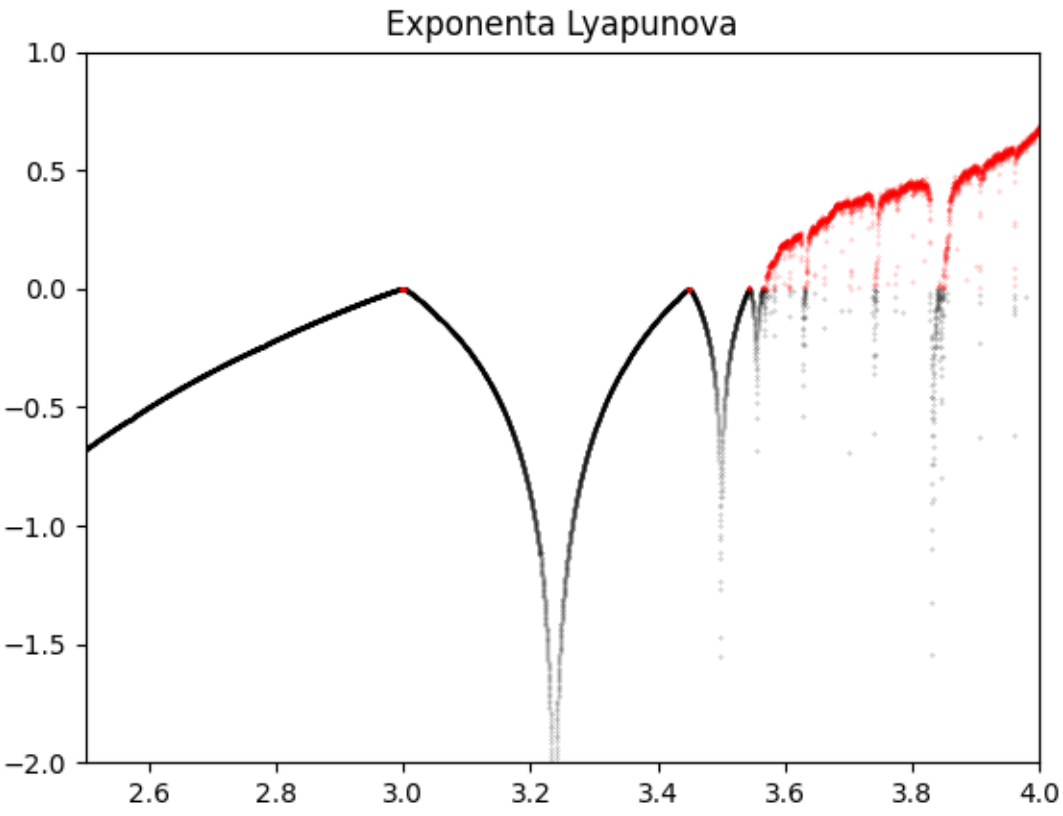
$$||v|| = |v_1| + |v_2| + |v_3|$$

W programie obliczamy wykładnik Liapunova dla wartości parametrów r=40, σ=16, b-4. Przypuszcza się, że metoda daje maksymalny jednowymiarowy wykładnik Liapunova. Dokładność jednowymiarowego wykładnika Liapunova można poprawić, gdy stany nieustalone ulegną rozpadowi.

lyapunov.py

```python
import numpy as np
import matplotlib.pyplot as plt
from dynamicFunc import matFun

n = 10000
r = np.linspace(2.5, 4.0, n)
iterations = 1000
x = 1e-5 * np.ones(n)
lyapunov = np.zeros(n)

for i in range(iterations):
    x = matFun(r, x)
    lyapunov += np.log(abs(r - 2 * r * x))

plt.plot(r[lyapunov < 0],lyapunov[lyapunov < 0] / iterations, '.k', alpha=.5, ms=.5)
plt.plot(r[lyapunov >= 0],lyapunov[lyapunov >= 0] / iterations, '.r', alpha=.5, ms=.5)
plt.xlim(2.5, 4)
plt.ylim(-2, 1)
plt.title("Exponenta Lyapunova")
plt.savefig("PlotLyapunov.png")
plt.show()
```

Rysunek 7: Program "Exponenta Lyapunova"



Rysunek 8: Wynik działania programu "Exponenta Lyapunova"

3 Rozwidlenie Hopfa

Rozwidlenie Hopfa odgrywa kluczową rolę w badaniu nieliniowych układów dynamicznych.
Twierdzenie
Niech U będzie otwartą spójną dziedziną w R^n $c > 0$ i niech f będzie rzeczywistą funkcją analityczną zdefiniowaną na $U \times [-c, c]$. Rozważmy autonomiczny układ równań różniczkowych zwyczajnych pierwszego rzędu:

$$\frac{du}{dt} = f(u, r), \quad gdzie \quad u \in U, \quad |r| < c$$

Załóżmy, że istnieje analityczna, rzeczywista funkcja wektorowa g zdefiniowana na [-c,c] taka, że:

$$f(g(r), r) = 0$$

Zatem można rozwinąć f(u,r) o g(r) w postaci, w której L_r jest $n \times n$ macierzą rzeczywistą, która zależy tylko od r, a f*(u*,r) jest nieliniową częścią f. Załóżmy, że istnieją dokładnie dwie zespolone sprzężone wartości własne $\alpha(r), \bar{\alpha}(r)$ L_r o właściwościach:

$$\Re(\alpha(0)) = 0 \quad i \quad \Re(\alpha'(0)) \neq 0 \quad (' \equiv \frac{d}{dr})$$

Wtedy istnieje rozwiązanie okresowe $P(t, \in)$ o okresie $T(\in)$ z $r = r(\in)$, takie, że $r(0) = 0$, $P(t, 0) = g(0)$ i $P(t, \in) \neq g(r(\in))$ dla wszystkich wystarczająco małych $\in \neq 0$ i

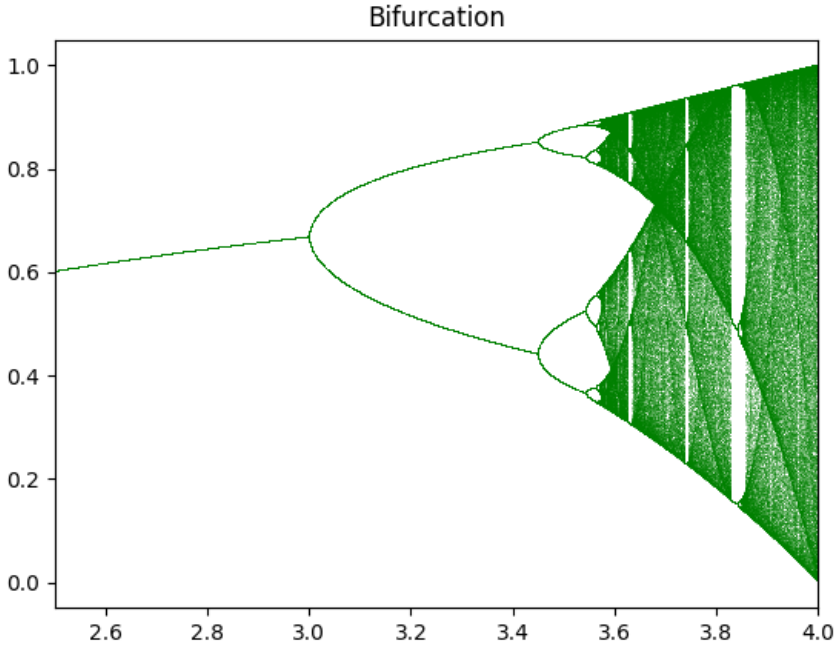
$$T(0) = \frac{2\pi}{|\Im\alpha(0)|}$$

Te "małe" okresowe rozwiązania istnieją dokładnie w jednym z trzech przypadków: albo tylko dla $r > 0$, albo tylko dla $r < 0$, albo tylko dla $r = 0$.

bifurcationDiag.py

```
1  import numpy as np
2  import matplotlib.pyplot as plt
3  from dynamicFunc import matFun
4
5  n = 10000
6  r = np.linspace(2.5, 4.0, n)
7  iterations = 1000
8  last = 100
9  x = 1e-5 * np.ones(n)
10 lyapunov = np.zeros(n)
11 for i in range(iterations):
12     x = matFun(r, x)
13     lyapunov += np.log(abs(r - 2 * r * x))
14     if i >= (iterations - last):
15         plt.plot(r, x, ',g', alpha=.25)
16 plt.xlim(2.5, 4)
17 plt.title("Bifurcation")
18 plt.savefig("PlotBifurcation.png")
19 plt.show()
```

Rysunek 9: Program "Bifurcation".



Rysunek 10: Wynik działania programu "Bifurcation".

4 Podsumowanie

Model Lorenza, który pierwotnie został użyty przez twórcę do przewidywaniu pogody, znalazł również wiele innych zastosowań. Ponieważ układ jest bardzo wrażliwy na dane wejściowe i już niewielka niedokładność wprowadzonych danych ma ogromny wpływ na wynik końcowy, mówi się o tak zwanym efekcie motyla. Podobnie dzieje sie w oscylacyjnych reakcjach chemicznych ciał oddzialujacych grawitacyjnie. Wykładnik Lapunowa określa chaotyczność układu, natomiast bifurkacja Hopfa ukazuje jak układ dochodzi do zachowania chaotycznego. Nasza praca pokazuje, że przewidywanie chaosu jest niemal niemożliwe. Możemy jedynie tworzyć modele o dużej dokładności na początku modelowania, jednak w krótkim czasie układ zaczyna zachowywać się w sposób nieprzewidywalny.

Literatura

[1] "The Nonlinear Workbook", Willi-Hans Steeb