Fern

Emil Śmiech,
Amelia Królczyk,
Michał Palwal,
Sabina Adamska

Czerwiec 2021

Wstęp

Układy iteracyjne funkcji mogą być używane do tworzenia wzorów, które swoją strukturą przypominają paprocie. Stworzony przez M. Barnsleya system iteracji funkcji paproci wytwarza paproć zwaną "Black Spleenwort". Tworzą ją cztery następujące transformacje afiniczne:

$$w_j(\mathbf{x}) = \begin{pmatrix} r_j\cos(\alpha_j) & -s_j\sin(\beta_j) \\ r_j\sin(\alpha_j) & s_j\cos(\beta_j) \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + \begin{pmatrix} a_{1j} \\ a_{2j} \end{pmatrix}$$

dla $j = 1, 2, 3, 4$.

Mamy cztery przemiany:

$$\begin{aligned} w_1(\mathbf{x}) &= \begin{pmatrix} 0 & 0 \\ 0 & 0.16 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \end{pmatrix} \\ w_2(\mathbf{x}) &= \begin{pmatrix} 0.85 & 0.04 \\ -0.04 & 0.85 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + \begin{pmatrix} 0 \\ 1.6 \end{pmatrix} \\ w_3(\mathbf{x}) &= \begin{pmatrix} 0.2 & -0.26 \\ 0.23 & 0.22 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + \begin{pmatrix} 0 \\ 1.6 \end{pmatrix} \\ w_4(\mathbf{x}) &= \begin{pmatrix} -0.15 & 0.28 \\ 0.26 & 0.24 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + \begin{pmatrix} 0 \\ 0.44 \end{pmatrix}. \end{aligned}$$

Normy macierzowe dla macierzy są mniejsze niż 1. Zatem wszystkie odwzorowania ω_1, ω_2, ω_3 i ω_4 są skróceniami. Jakie jest ich znaczenie? ω_1 odwzorowuje całość paproci na paproć składającą się z najniższych gałęzi, ale bez najniższej części łodygi. Stałym punktem ω_1 jest czubek paproci. ω_2 przekształca całą paproć w gałąź z prawego dolnego rogu; kurczy się, zwęża i przewraca. Podobnie, ω_3 przekształca całą paproć w gałąź z lewego dolnego rogu. Wreszcie, ω_4 odwzorowuje całą paproć na odcinek linii na osi x_2; jego stałym punktem jest początek i jest używany do generowania łodygi.

Kod programu

Kod programu Fern.java

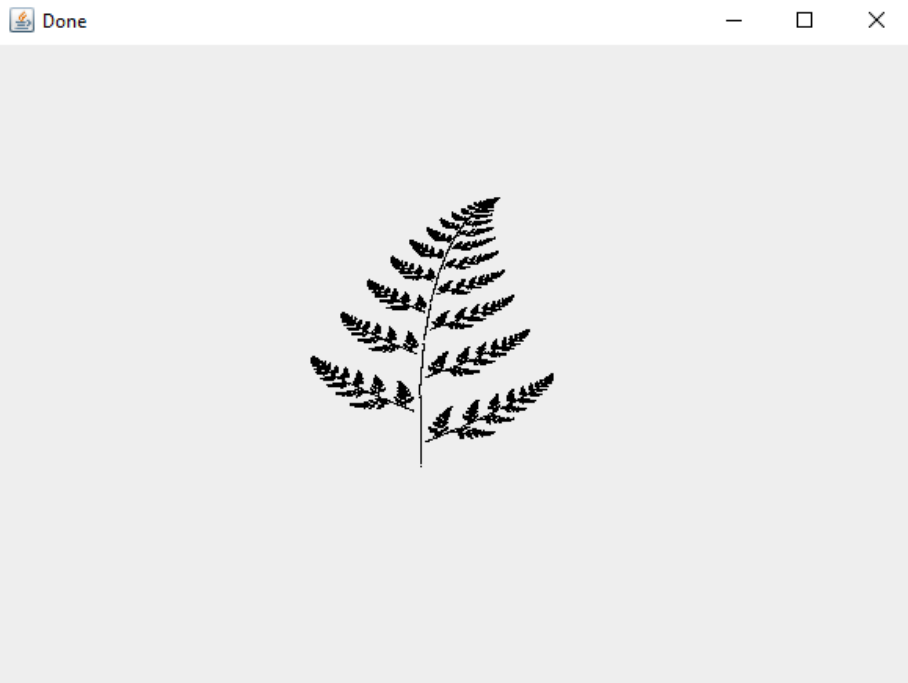
```java
import java.awt.*;
import java.awt.Graphics;
import javax.swing.JFrame;
import javax.swing.JPanel;

public class Fern extends JPanel
{
    double mxx, myy, bxx, byy;

    public void plot(Graphics g)
    {
        double x, y, xn, yn, r;
        int pex, pey;
        int max = 15000;x = 0.5; y = 0.0;
        convert(0.0, 1.0, 1.0, -0.5);
        setBackground(Color.white);
        g.setColor(Color.black);
```

```
18          for(int i = 0; i ⩽ max; i++)
19          {
20              r = Math.random();
21              if(r ⩽ 0.02)
22              {
23                  xn = 0.5;                                   // Odwzorowanie 1
24                  yn = 0.27 * y;
25              }
26              else if((r > 0.02) && (r ⩽ 0.17))
27              {
28                  xn = -0.139 * x + 0.263 * y + 0.57;         // Odwzorowanie 2
29                  yn =  0.246 * x + 0.224 * y - 0.036;
30              }
31              else if((r > 0.17) && (r ⩽ 0.3))
32              {
33                  xn = 0.17 * x - 0.215 * y + 0.408;          // Odwzorowanie 3
34                  yn = 0.222 * x + 0.176 * y + 0.0893;
35              }
36              else
37              {
38                  xn = 0.781 * x + 0.034 * y + 0.1075;        // Odwzorowanie 4
39                  yn = -0.032 * x + 0.739 * y + 0.27;
40              }
41              x = xn; y = yn;
42              pex = (int) (mxx * x + bxx);
43              pey = (int) (myy * y + byy);
44                  g.drawLine(pex, pey, pex, pey);
45          }
46      }
47
48      void convert(double xiz,double ysu,double xde,double yinf)
49      {
50          double maxx, maxy, xxfin, xxcom, yyin, yysu;
51          maxx = 600; maxy = 450;
52          xxcom = 0.15 * maxx; xxfin = 0.75 * maxx;
53          yyin = 0.8 * maxy; yysu = 0.2 * maxy;
54          mxx = (xxfin - xxcom)/(xde - xiz);
55          bxx = 0.5 * (xxcom + xxfin - mxx * (xiz + xde));
56          myy = (yyin - yysu)/(yinf - ysu);
57          byy = 0.5 * (yysu + yyin - myy * (yinf + ysu));
58      }
59
60      public void paint(Graphics g)
61      {
62          plot(g);
63      }
64
```

```
        Run | Debug
65      public static void main(String[] args)
66      {
67          JFrame window = new JFrame("Done");
68          Fern panel = new Fern();
69          window.add(panel);
70          window.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
71          window.setSize(600, 450);
72          window.setVisible(true);
73      }
74  }
```



Rysunek 1: Rysunek przedstawia wynik działania programu Fern.java

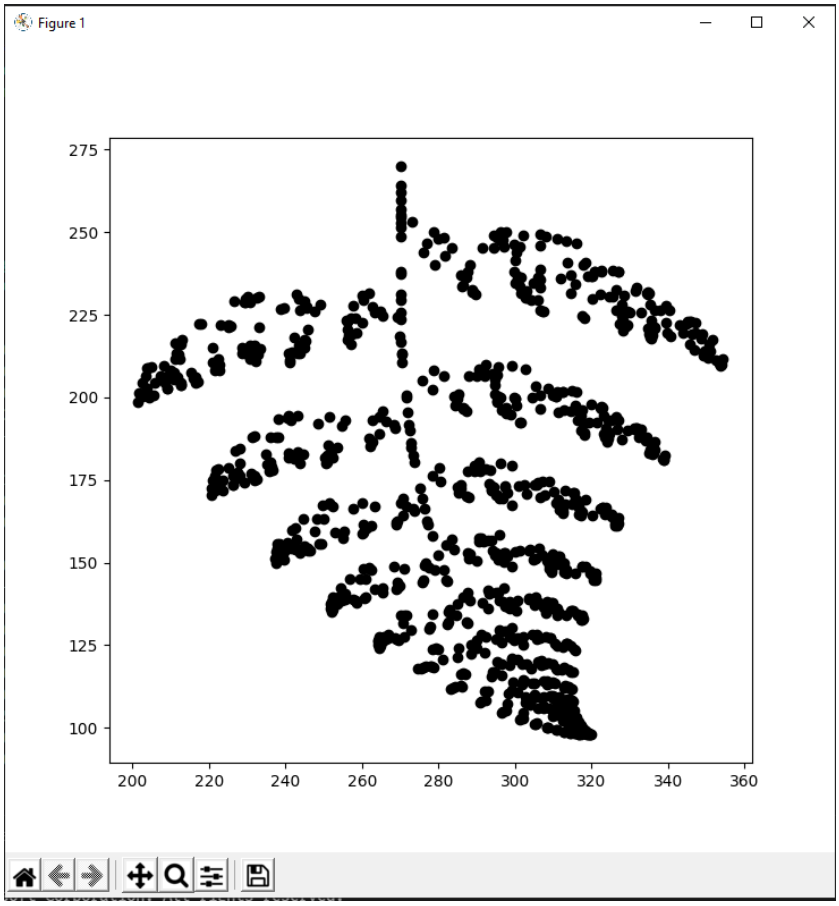
Kod programu Fern.py

```python
import matplotlib.pyplot as plt
import random

max = 1000
x = 0
y = 0
pex_tab = []
pey_tab = []

for i in range(max):
    r = random.random()
    xn = x
    yn = y
    if r <= 0.02:
        x = 0.5                                  # map 1
        y = 0.27 * yn
    elif 0.02 < r <= 0.17:
        x = -0.139 * xn + 0.263 * yn + 0.57      # map 2
        y =  0.246 * xn + 0.224 * yn - 0.036
    elif  0.17 < r <= 0.3:
        x = 0.17 * xn - 0.215 * yn + 0.408  # map 3
        y = 0.222 * xn + 0.176 * yn + 0.0893
    else:
        x = 0.781 * xn + 0.034 * yn + 0.1075    # map 4
        y = -0.032 * xn + 0.739 * yn + 0.27
```

```
26      xiz = 0.0
27      xde = 1.0
28      ysu = 1.0
29      yinf = -0.5
30      maxx = 600
31      maxy = 450
32      xxcom = 0.15 * maxx
33      xxfin = 0.75 * maxx
34      yyin = 0.8 * maxy
35      yysu = 0.2 * maxy
36      mxx = (xxfin - xxcom)/(xde - xiz)
37      bxx = 0.5 * (xxcom + xxfin - mxx * (xiz + xde))
38      myy = (yyin - yysu)/(yinf - ysu)
39      byy = 0.5 * (yysu + yyin - myy * (yinf + ysu))
40      pex = mxx * x + bxx
41      pex_tab.append(pex)
42      pey = myy * y + byy
43      pey_tab.append(pey)
44
45
46  plt.scatter(pex_tab, pey_tab, color='black')
47  plt.show()
```



Rysunek 2: Rysunek przedstawia wynik działania programu Fern.py