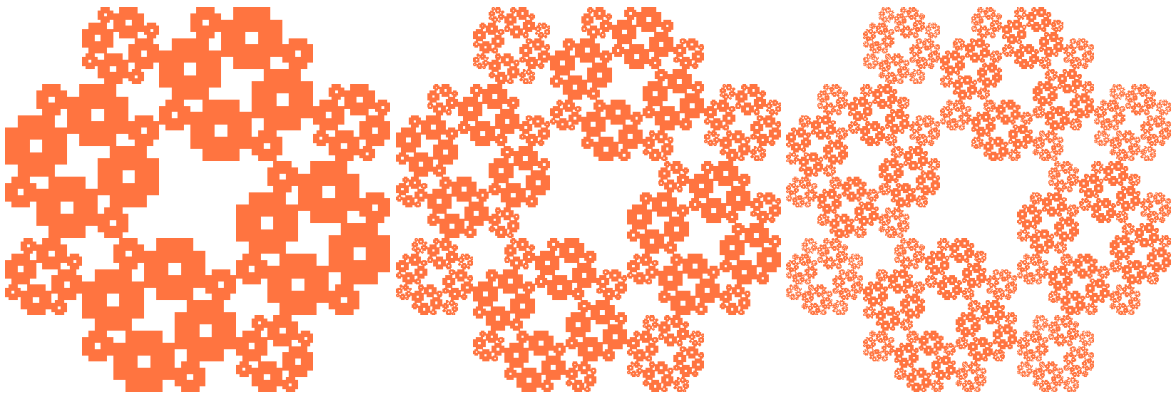
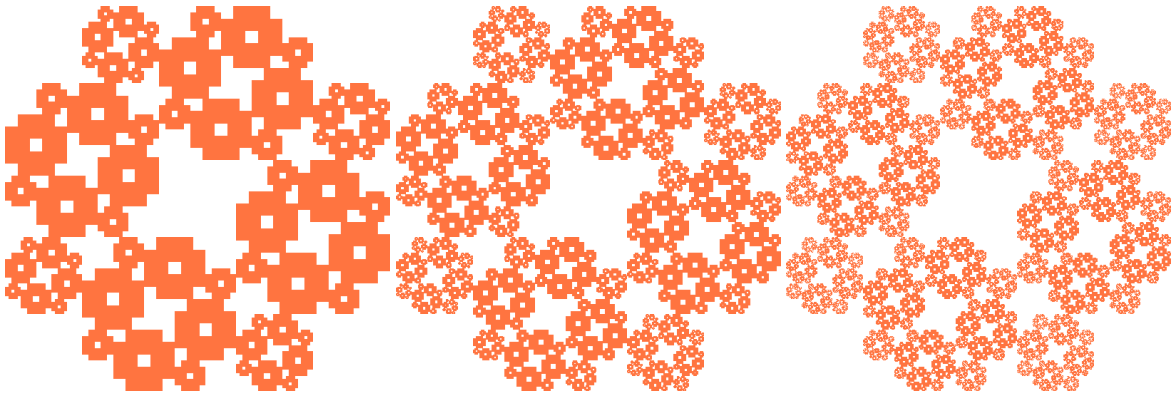
Wybrane fraktale na płaszczyźnie i ich wymiar Hausdorffa





Dawid Dąbkowski
Wydział Matematyki, Informatyki i Mechaniki
Uniwersytet Warszawski
dd345130@students.mimuw.edu.pl

20.03.2017

Konkurs na projekt w programie *Mathematica*
Wydział Matematyki, Informatyki i Mechaniki UW

Oświadczam, że niniejszy projekt został wykonany przeze mnie samodzielnie

Opis problemu

Założenia projektu

Celem projektu jest stworzenie interaktywnej wizualizacji fraktali wraz z obliczaniem ich wymiaru Hausdorffa. Dostępna jest pewna klasa fraktali, które tworzone są na dwuwymiarowej siatce kwadratów przez wyświetlanie kolejnych iteracji. Podając odpowiednie parametry można generować zarówno znane fraktale jak i wiele innych. Program wyświetla pierwsze iteracje oraz podaje dokładny i przybliżony wymiar Hausdorffa zbiorów granicznych.

Do funkcjonalności programu należy także sprawdzanie poprawności danych wejściowych i zastosowanie ewentualnych poprawek. Wizualizacja zawiera kilka efektów, w tym animację przybliżania wybranego fragmentu fraktala.

Projekt może być wykorzystywany do popularyzacji matematyki i tematyki związanej z fraktalami, a ponadto przydatny w nauce do badania podstawowych własności fraktali i związanego z nimi wymiaru Hausdorffa.

Czym są fraktale?

Fraktale to obiekty matematyczne o bardzo niezwykłych cechach. Typowo, mówiąc “fraktal”, mamy na myśli zbiór punktów w R^n, który cechuje:
- nieskończenie bogata budowa: dowolne przybliżenie części fraktala też jest fraktalem i nie da się opisać typową geometrią ani wyrazić typowym równaniem,
- samo-podobieństwo: składa się z kilku (a zatem nieskończenie wielu!) kopii samego siebie,
- wymiar Hausdorffa jest większy niż wymiar topologiczny (do tego wymiar ten zwykle jest niecałkowity!),
- jest zerowej miary w wymiarze n, lecz nieskończonej w wymiarze n-1, mimo że jest ograniczony.

Fraktale i inne dziwne, niegładkie zbiory początkowo były niedoceniane i traktowane jako swego rodzaju matematyczne “patologie”, którymi nie warto się zajmować. Uważano, że wszystko w świecie można opisać za pomocą gładkich struktur, linii prostych, elips czy równań algebraicznych. Około połowy XIX wieku fraktale zaczęły odgrywać rolę w matematyce, lecz raczej jako pojedyncze kontrprzykłady do znanych twierdzeń, nikt natomiast nie próbował stworzyć teorii fraktali. Dużo zmieniło się około lat siedemdziesiątych XX wieku za sprawą francuskiego matematyka B. Mandelbrota, który to właśnie ukuł termin “fraktal” i dał podstawy współczesnej teorii fraktali. W niektórych swoich pracach Mandelbrot pisał, że świat złożony jest wręcz z samych fraktali! Rzeczy takie jak drzewiasta struktura roślin, układu krwionośnego, nerwowego czy krwionośnego zwierząt; poszarpane szczyty górskie, nieskończenie zawiłe linie wybrzeża, delty rzek albo chmury, płatki śniegu, to wszystko do złudzenia przypomina matematyczne fraktale!

Obecnie fraktale stanowią obiekt badań i mają wiele zastosowań także w różnych dziedzinach nauk: informatyce, biologii, fizyce, technice... Sprawdzają się w algorytmach kompresji danych i w generowaniu grafiki komputerowej w oparciu o bardzo proste, oszczędne pamięciowo algorytmy. Z powodzeniem można używać ich do opisu różnych obiektów w naukach przyrodniczych, które mają samopodobną strukturę. Wymiar Hausdorffa jest wówczas realnym parametrem, który pozwala zmierzyć na przykład przepuszczalność materiału (ropa przeciskająca się przez porowate skały przypominające gąbkę Mengera) czy właściwości powierzchni (powierzchnia zardzewiałego metalu, której przekrój przypomina krzywą Kocha). Fraktale są na tyle popularne, że wkroczyły do współczesnej kultury. Samo ich oglądanie i odkrywanie jest ciekawym zajęciem. Można to robić dzięki licznym programom do generowania fraktali. A jednym z takich programów jest niniejszy projekt!

Czym jest wymiar Hausdorffa?

Jak już wcześniej pisałem, fraktale zwykle mają zerowe lub nieskończone miary Lebesgue'a, a wymiar topologiczny zdaje się zaniżać ich wymiar, mimo że są tak skomplikowane. Istnieje jednak dobry parametr, który jest pewnym uogólnieniem wymiaru topologicznego i dobrze nadaje się do opisu fraktali. Mowa o wymiarze Hausdorffa, który dla regularnych zbiorów jest równoważny z wymiarem topologicznym, a dla fraktali przyjmuje wartości rzeczywiste z przedziału [0,n]. Wymiar fraktala może być zatem niecałkowity (i zwykle taki jest)! O wymiarze Hausdorffa można myśleć jako o stopniu zapełnienia przestrzeni (czy też odwrotnie o stopniu "dziurawości" zbioru). Formalnie, dla zbioru $F \in R^n$, s≥0, δ>0 zdefiniujemy miarę Hausdorffa
$H^s{}_\delta$:

$$\boldsymbol{H^s_\delta(F) := \inf\{\sum_{i=1}^{\infty} \mid U_i \mid^s : \{U_i\} \text{ jest } \delta \text{ – pokryciem zbioru } F\}}$$

Mówiąc prościej, ustalamy δ i s i pokrywamy zbiór F zbiorami o średnicy nie większej niż δ. Z wszystkich takich pokryć wybieramy te, które minimalizują wyrażenie $\sum_{i=1}^{\infty} \mid U_i \mid^s$.

Teraz zobaczmy co stanie się, gdy będziemy zbiegać z δ do *0*. Klasa możliwych pokryć zbioru F będzie się zmniejszać, zatem infimum będzie rosnąć. Okazuje się, że wyrażenie będzie zbiegać do granicy H^s:

$$\boldsymbol{H^s(F) := \lim_{\delta \to 0} H^s_\delta(F)}$$

Wyrażenie to nazywa się *s*-wymiarową miarą Hausdorffa zbioru F i jest jednoznacznie określone dla dowolnego zbioru. Przy tym miara ta przyjmuje wartość stale ∞ aż do pewnego *s*, a powyżej tego *s* wartość *0* (i tylko w tym jednym punkcie może być skończona i dodatnia). To graniczne *s* nazywamy właśnie wymiarem Hausdorffa $\dim_H F$:

$$\boldsymbol{\dim_H F := \inf\{s : H^s(F) = 0\} = \sup\{s : H^s(F) = \infty\}}$$

Jak będziemy wyznaczać wymiar Hausdorffa?

Niestety, obliczanie wymiaru Hausdorffa, korzystając z definicji, zwykle jest bardzo trudne. Na szczęście w projekcie rozpatrujemy tylko szczególną klasę fraktali, dla której wyznaczenie wymiaru Hausdorffa będzie znacznie prostsze, gdy skorzystamy z odpowiedniego twierdzenia. Nasze fraktale powstawać będą rekurencyjnie wewnątrz kwadratu podzielonego na równą siatkę. W pierwszym kroku wczytywana jest lista współrzędnych i wielkości kwadratów, które zaznaczane są na siatce. Kwadraty mogą się na siebie nachodzić co najwyżej krawędzią lub wierzchołkiem i nie mogą wystawać poza obszar ani poza linie siatki, lecz mogą mieć różną wielkość. Następnie cały obraz jest skalowany i rysowany w każdym z zaznaczonych kwadratów i tak dalej. Możemy więc mówić o funkcjach podobieństw $S_1, \ldots, S_m : R^2 \to R^2$, które to przekształcają cały zbiór na jeden z mniejszych kwadratów, ze skalą podobieństwa c_i. Formalnie, możemy wówczas mówić o naszym fraktalu jako o *atraktorze układu iterowanych odwzorowań*. Możemy też skorzystać z następującego twierdzenia:

Twierdzenie:
Załóżmy, że podobieństwa S_i ze skalami c_i (dla $1 \le i \le m$) spełniają **warunek otwartego zbioru***. Wówczas, jeśli F jest zbiorem niezmienniczym ze względu na S_i (spełniającym $F = \cup_{i=1}^{m} S_i(F)$), to wymiar Hausdorffa $\dim_H F$ jest skończony i zadany rozwiązaniem równania $\sum_{i=1}^{m} c_i^t = 1$ ze względu na parametr **t**.

*Warunek otwartego zbioru zapewnia, że obrazy

podobieństw s_i nie pokrywają się zbyt mocno. Dokładnie, mówi on, że istnieje niepusty ograniczony zbiór otwarty V, taki że $V \supset \cup_{i=1}^{m} s_i(V)$, przy czym suma jest rozłączna. W naszym przypadku jako V wystarczy wziąć wnętrze dużego kwadratu. Wówczas obrazy podobieństw zbioru początkowego leżą we wnętrzach małych kwadratów, które są rozłączne i które w sumie zawierają się we wnętrzu dużego kwadratu.

Kod źródłowy

Definicje zbiorów

Przykładowe dane wejściowe.

```
point = {{2, 2, 1}};

line = {{1, 1, 1}, {2, 2, 1}, {3, 3, 1}};

cantor = {{1, 2, 1}, {3, 2, 1}};

cantorDust = {{1, 1, 1}, {3, 1, 1}, {1, 3, 1}, {3, 3, 1}};

sCarpet = {{1, 1, 1}, {1, 2, 1}, {1, 3, 1},
    {2, 1, 1}, {2, 3, 1}, {3, 1, 1}, {3, 2, 1}, {3, 3, 1}};

sTriangle = {{1, 1, 1}, {2, 1, 1}, {2, 2, 1}};

flowers = {{2, 1, 2}, {4, 2, 2}, {1, 3, 2},
    {3, 4, 2}, {4, 1, 1}, {1, 2, 1}, {2, 5, 1}, {5, 4, 1}};
```

Definicje funkcji dotyczących danych

Funkcja sprawdzająca, czy dane wejściowe pasują do wzorca listy trójek. W razie braku zgodności funkcja próbuje poprawić lub zresetować dane.

```
checkPattern[s_] := Module[{s1 = s},
  s1 = If[MatchQ[s1, List[__]], s1, point];
  For[i = 1, i ≤ Length[s1], i++,
   If[MatchQ[s1[[i]], List[_Integer, _Integer, _Integer]] == False,
    s1 = Delete[s1, i]; i--, Nothing]
  ]; s1
 ]
```

Funkcja wyłapująca i usuwająca z danych wejściowych niezgodne z założeniami obiekty, czyli kwadraty które nachodzą na inne lub wychodzą poza obszar.

```
prepareInput[s_, d_] := Module[{s1 = s, d1 = d},
  For[i = 1, i ≤ Length[s1], i++,
   If[(s1[[i, 1]] < 1 || s1[[i, 1]] + s1[[i, 3]] > 1 + d1 ||
      s1[[i, 2]] < 1 || s1[[i, 2]] + s1[[i, 3]] > 1 + d1),
    s1 = Delete[s1, i]; i--,
    For[j = 1, j < i, j++,
     If[(((s1[[j, 1]] ≤ s1[[i, 1]] < s1[[j, 1]] + s1[[j, 3]]) &&
           (s1[[j, 2]] ≤ s1[[i, 2]] < s1[[j, 2]] + s1[[j, 3]])) ||
         ((s1[[j, 1]] < s1[[i, 1]] + s1[[i, 3]] ≤ s1[[j, 1]] + s1[[j, 3]]) &&
           (s1[[j, 2]] < s1[[i, 2]] + s1[[i, 3]] ≤ s1[[j, 2]] + s1[[j, 3]])) ||
         ((s1[[i, 1]] <= s1[[j, 1]]) && (s1[[j, 1]] + s1[[j, 3]] <=
             s1[[i, 1]] + s1[[i, 3]]) &&
           ((s1[[i, 2]] <= s1[[j, 2]]) && (s1[[j, 2]] + s1[[j, 3]] <=
               s1[[i, 2]] + s1[[i, 3]])))),
      s1 = Delete[s1, i]; i--; Break,
      Nothing]
    ]
   ]
  ]; s1
 ]
```

Funkcja modyfikująca współrzędne z danych wejściowych na format wygodny do obsługi w programie.

```
normalize[s_, d_] := Module[{s1 = s, d1 = d},
  s1 = ((# - {1, 1, 0}) & /@ s1) / d1;
  s1]
```

Kod źródłowy

Definicje funkcji rysujących i liczących

Wygodna funkcja do rysowania kwadratu z jednego elementu listy.

```
square[{x_, y_, r_}] := Rectangle[{x, y}, {x + r, y + r}]
```

Funkcja, która dokonuje wszelkich obliczeń ze zbioru wejściowego i zwraca listę kwadratów do narysowania.

```
fractal[setNorm_, iter_] := Module[{inputSet = setNorm},
  (* shift dla danego kwadratu zwraca wspolrzedne
   kwadratow wewnatrz niego, w nastepnej iteracji *)
  shift[s_, i_] := Table[{
     s[[i, 1]] + inputSet[[k, 1]] * s[[i, 3]],
     s[[i, 2]] + inputSet[[k, 2]] * s[[i, 3]],
     s[[i, 3]] * inputSet[[k, 3]]},
    {k, 1, Length[inputSet]}];
  (* scor zwraca wspolrzedne wszystkich kwadratow
  w nastepnej iteracji *)
  scor[list_] :=
   Flatten[
    Table[
     shift[list, i],
     {i, 1, Length[list]}
    ], 1];
  (* fcor zwraca wspolrzedne wszystkich kwadratow
  po przejsciu wszystkich iteracji *)
  fcor[list_, it_] := Nest[scor, list, it - 1];
  square /@ fcor[inputSet, iter]
 ]
```

Funkcje liczące wymiar Hausdorffa do wyświetlenia (dokładny i przybliżony).

```
dimension[s_] := Module[{s1 = s},
  a[t_] := Sum[s1[[j, 3]]^t, {j, 1, Length[s1]}];
  b = (t /. Solve[a[t] == 1, t, Reals])[[1]];
  If[MatchQ[b, Root[_]] == True, Text[t : Simplify[a[t] == 1]], b]
 ]
```

```
ndimension[s_] :=
 (t /. NSolve[Sum[s[[j, 3]]^t, {j, 1, Length[s]}] == 1, t, Reals])[[1]]
```

Funkcja ograniczająca liczbę iteracji.

```
iterations[l_] := Min[12, Floor[If[(l == 0 || l == 1), 12, Log[l, 8^5]]]]
```

Główny program

```
Manipulate[
 div = Switch[mode, point, 3, line, 3, cantor, 3,
   cantorDust, 3, sCarpet, 3, sTriangle, 2, flowers, 5, custom, div];
 set = Switch[mode, point, point, line, line, cantor, cantor, cantorDust,
   cantorDust, sTriangle, sTriangle, sCarpet, sCarpet, flowers, flowers,
   custom, prepareInput[checkPattern[set], div]];
 setNorm = normalize[set, div];
 size = 540;
 len = Max[1, Length[setNorm]];
 zoomAt = If[zoomAt > len, 1, zoomAt];
 lim = iterations[len];
 iter = If[iter > lim, 1, iter];
 haus = dimension[setNorm];
 nhaus = ndimension[setNorm];
 Graphics[{Dynamic[If[inverse == False, fractalcolor, background]],
   fractal[setNorm, iter]},
  Background → Dynamic[If[inverse == False, background, fractalcolor]],
  GridLines → Dynamic[If[grid == True,
     {Range[0, 1 + 1 / div, 1 / div], Range[0, 1 + 1 / div, 1 / div]}, None]],
  PlotRange → Dynamic[{
     {zoomVal * setNorm[[zoomAt, 1]] + (1 - zoomVal) * 0,
      zoomVal * (setNorm[[zoomAt, 1]] + setNorm[[zoomAt, 3]]) + (1 - zoomVal) * 1},
     {zoomVal * setNorm[[zoomAt, 2]] + (1 - zoomVal) * 0, zoomVal *
        (setNorm[[zoomAt, 2]] + setNorm[[zoomAt, 3]]) + (1 - zoomVal) * 1}}],
  ImageSize → {size, size}],
 Delimiter, Style["Parameters", 12, Bold],
 {{mode, flowers, "preset"},
  {point → "Point", line → "Line", cantor → "Cantor set",
   cantorDust → "Cantor dust", sTriangle → "Sierpinski triangle",
   sCarpet → "Sierpinski carpet", flowers → "Flowers", custom → "Custom"}},
 {{div, 2, "division"}, 2, 12, 1},
 {{iter, 1, "iterations"}, Thread[Range[lim] → Range[lim]]},
 {{set, "", "initial set"}},
 Delimiter, Style["Animation", 12, Bold],
 {{grid, True, "grid"}, {False, True}},
 {{zoomAt, 1, "zoomed square"}, Thread[Range[len] → Range[len]]},
 {{zoomVal, 0, "zoom value"}, 0, 1, Appearance → "Open"},
 Delimiter, Style["Coloring", 12, Bold],
 {{inverse, False, "inverse"}, {False, True}},
 {fractalcolor, Black},
 {background, White},
 Delimiter, Style["Calculations", 12, Bold],
 {{len, len, "input length"}, 1, 144, 1, Opacity[1], Appearance → "Labeled"},
 {{lim, lim, "iterations limit"}, 1, 12, 1, Opacity[1], Appearance → "Labeled"},
 Delimiter, Style["Hausdorff dimension", 12, Bold],
 {{haus, haus, "exact"}, 0, 2, 0, Opacity[1], Appearance → "Labeled"},
 {{nhaus, nhaus, "approx"}, 0, 2, 0, Opacity[1], Appearance → "Labeled"},
 LabelStyle → Directive[Medium],
 ControlType →
  {Automatic, PopupMenu, Setter, Setter, InputField, Automatic, Automatic, Setter,
   Automatic, Automatic, Automatic, ColorSlider, ColorSlider, Automatic},
 ControlPlacement → Left, SaveDefinitions → True
]
```

Parameters

preset: Sierpinski triangle

division: 2 3 4 5 6 7 8 9 10 11 12

iterations: 1 2 3 4 5 6 7 8 9

initial set: {{1, 1, 1}, {2, 1, 1}, {2, 2, 1}}

Animation

grid: ☑

zoomed square: 1 2 3

zoom value: 0.135962

Coloring

inverse: ☐

fractalcolor

background

Calculations

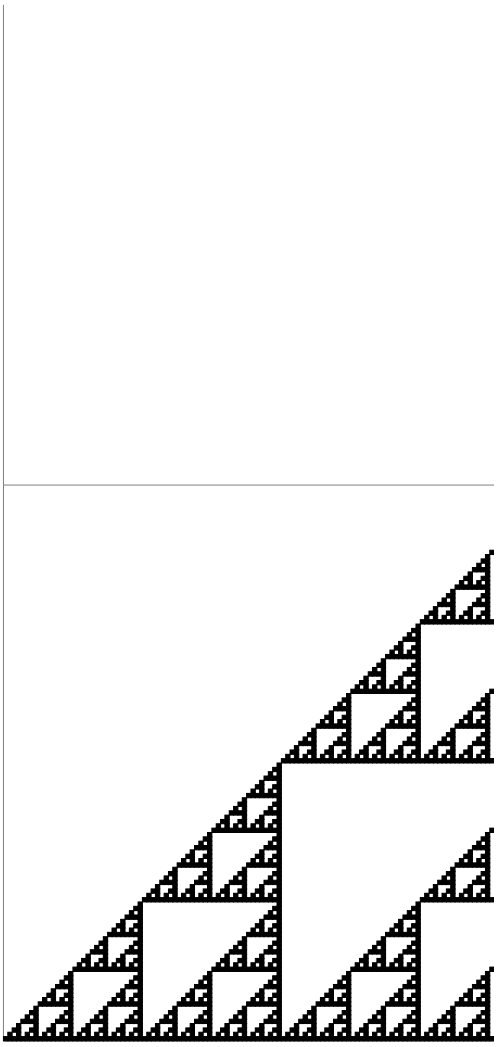
input length 3

iterations limit 9

Hausdorff dimension

exact $\frac{\text{Log}[3]}{\text{Log}[2]}$

approx 1.58496



Podsumowanie

Dalsze kroki

Ewentualne rozszerzenia, które mogłyby stanowić rozszerzenie projektu:

- zastosowanie algorytmów losowych do generowania fraktali trudnych wydajnościowo
- uogólnienie klasy fraktali (szerszy zakres współrzędnych, inne figury, inne przekształcenia jak obroty i odbicia)
- dopasowanie limitu iteracji do trudności obliczeniowej zadania (a zatem do stopnia użycia procesora)

Bibliografia

Pracę wykonałem w oparciu o materiały kursowe przedmiotu Matematyka z Mathematicą i Wolfram Alfa na stronie *https://moodle.mimuw.edu.pl* oraz dokumentację programu Mathematica (w szczególności podręczniki “Introduction to Manipulate” i “Advanced Manipulate Functionality”). Dane merytoryczne zaczerpnąłem z poniższych pozycji bibliograficznych.

- Falconer K., Fractal Geometry: Mathematical Foundations and Applications, John Wiley & Sons Ltd., Chichester 1990
- Kudrewicz J., Fraktale i chaos, Wydawnictwa Naukowo-Techniczne, Warszawa 1996
- Tempczyk M., Teoria chaosu dla odważnych, Wydawnictwo Naukowe PWN SA, Warszawa 2002