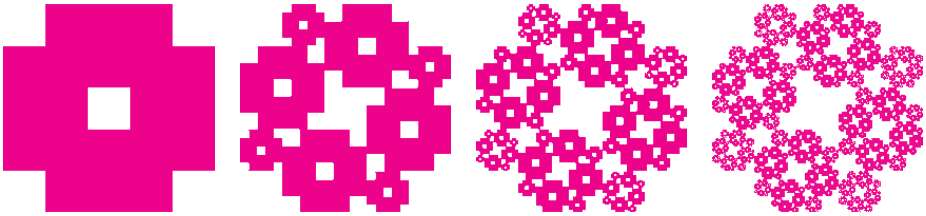
Mathematica i fraktale

*Dawid DĄBKOWSKI**

Wolfram Mathematica to popularny, nie tylko wśród studentów matematyki, system obliczeniowy, który umożliwia rozwiązywanie zadań z dziedzin, takich jak matematyka, fizyka czy ekonomia. Mamy tu oczywiście rachunek różniczkowy i całkowy, algebrę i statystykę, lecz także najróżniejsze metody z zakresu od matematyki czysto teoretycznej aż po zastosowania w data science, biznesie, inżynierii czy medycynie. Łącznie mamy prawie 5000 wzajemnie zintegrowanych, wbudowanych funkcji. Mathematica używa własnego języka programowania *Wolfram Language*, który cechuje się wydajnym operowaniem na listach. Zaletą systemu Mathematica jest także przyjazny interfejs z rozbudowaną dokumentacją każdej z funkcji, a także szerokie możliwości interaktywnej wizualizacji obliczeń.

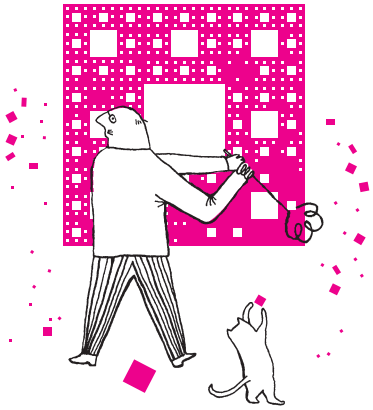
Choć zastosowania Mathematici są bardzo szerokie, tutaj skupię się na tych czysto teoretycznych. Pokażę bowiem, jak Mathematicę zastosować do elementarnej teorii fraktali. Opiszę założenia teoretyczne oraz pokażę funkcje, dzięki którym użytkownik może sprawdzać poprawność danych wejściowych, obliczać analitycznie i numerycznie wymiar Hausdorffa i wreszcie generować piękne wizualizacje fraktali!



Rys. 1. Wizualizacja kolejnych iteracji fraktala, stworzona w Mathematice

Fraktale na kartce

Zanim zaczniemy programować, przybliżmy nieco temat fraktali. Różne są definicje fraktali. Mówiąc o fraktalach, mamy zwykle na myśli zbiory w przestrzeni $\mathbb{R}^n$, które są samopodobne, a więc składają się z wielu przeskalowanych kopii samych siebie.



Matematyk zwykle powie, że fraktal to zbiór, którego *wymiar Hausdorffa* jest większy od jego wymiaru topologicznego. O wymiarze Hausdorffa można myśleć jako o uogólnieniu wymiaru topologicznego dla zbiorów, których nie da się łatwo sparametryzować. O fraktalach możemy myśleć jako o zbiorach (zwykle) niecałkowitego wymiaru!

Formalnie, dla zbioru $F \subset \mathbb{R}^n$ i $s \geqslant 0$ definiujemy s-wymiarową miarę Hausdorffa:

$$H^s(F) := \lim_{\delta \to 0^+} \inf\Big\{\sum_{i=1}^{\infty} |U_i|^s : \{U_i\}_{i=1}^{\infty} \text{ jest } \delta\text{-pokryciem } F\Big\},$$

gdzie $|V|$ oznacza średnicę zbioru v. Mówiąc prościej, ustalamy δ i s i pokrywamy zbiór F zbiorami o średnicy nie większej niż δ. Ze wszystkich takich pokryć wybieramy to, które minimalizuje wyrażenie $\sum_{i=1}^{\infty} |U_i|^s$. Następnie zmniejszamy δ. Klasa możliwych pokryć zbioru będzie się zmniejszać, zatem infimum będzie rosnąć. Okazuje się, że zbiegając z δ do zera, otrzymamy konkretną granicę $H^s(F) \in [0, \infty]$. O mierze tej wiemy, że przyjmuje stale wartość ∞ aż do pewnego $\bar{s}$, powyżej którego jest stale równa 0. Właśnie to $\bar{s}$, dla którego następuje swoista „nieciągłość” miary, to nasz wymiar Hausdorffa:

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

Wyznaczanie wymiaru Hausdorffa z tej definicji jest zwykle bardzo trudne. W naszym programie rozpatrywać będziemy jednak tylko szczególną klasę fraktali, dla których szczęśliwie zachodzi pewne dogodne twierdzenie.

Na wydziale Matematyki, Informatyki i Mechaniki Uniwersytetu Warszawskiego co roku odbywa sie konkurs na projekt stworzony w Mathematice. W roku 2017 nagrodzono program, o którym piszę w tym artykule. Pliki źródłowe do tego i innych programów można pobrać ze strony: `http://mathematica.mimuw.edu.pl/competition.html`

Mathematica jest programem płatnym, lecz niektóre jej funkcjonalności dostępne są za darmo. Za pomocą darmowego programu Wolfram CDF Player można przeglądać programy zapisane w rozszerzeniu .cdf. Istnieje dedykowana strona, na której można znaleźć wiele ciekawych demonstracji. Pod tym adresem znaleźć można wersję demonstracyjną mojego programu o fraktalach: `http://demonstrations.wolfram.com/FractalsAndTheirHausdorffDimension/`

*student, Wydział Matematyki, Informatyki i Mechaniki, Uniwersytet Warszawski

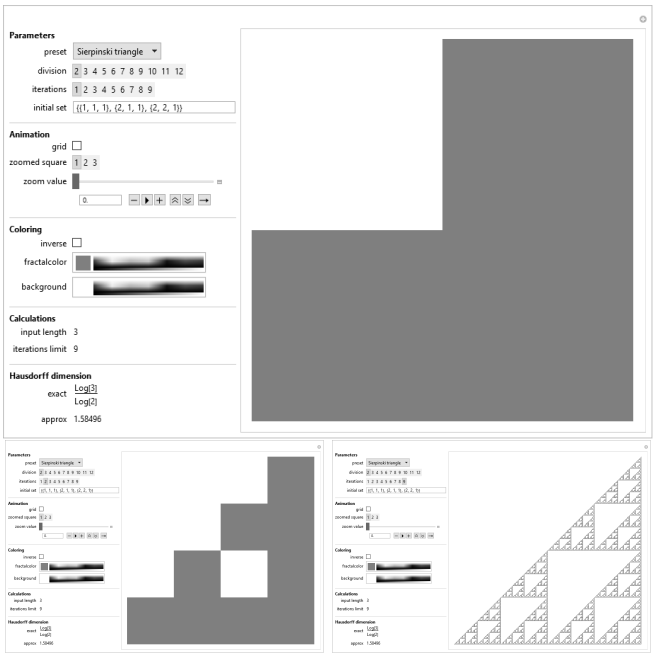
Fraktale powstawać będą rekurencyjnie wewnątrz kwadratu podzielonego na równą siatkę. W pierwszym kroku wczytywana jest lista współrzędnych i wielkości kwadratów, które zaznaczane są na siatce. Kwadraty mogą mieć wspólną co najwyżej krawędź lub wierzchołek i nie mogą wystawać poza linie siatki. Cały obraz jest następnie skalowany i rysowany w każdym z zaznaczonych kwadratów. Procedurę tę powtarza się kilka razy, aż do określonej liczby rekurencji. Mamy więc funkcje podobieństw $S_1, \ldots, S_m : \mathbb{R}^2 \to \mathbb{R}^2$, które przekształcają cały zbiór na jeden z mniejszych kwadratów, ze skalami podobieństw $c_1, \ldots, c_m$. Formalnie nasz fraktal jest tzw. *atraktorem układu iterowanych odwzorowań*. Zachodzi wówczas następujące twierdzenie:

Załóżmy, że podobieństwa S_i ze skalami c_i (dla $1 \leqslant i \leqslant m$) spełniają warunek otwartego zbioru. Wówczas, jeśli F jest zbiorem niezmienniczym ze względu na S_i (tzn. 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 zanadto. Dokładnie mówi on, że istnieje taki niepusty ograniczony zbiór otwarty V, że $V \supset \bigcup_{i=1}^{m} S_i(V)$, przy czym suma ta jest rozłączna. U nas jako V wystarczy wziąć wnętrze dużego kwadratu. Obrazy podobieństw zbioru początkowego leżą wówczas we wnętrzach małych kwadratów, które są rozłączne i zawierają się we wnętrzu dużego kwadratu.

Fraktale w Mathematice

Mając już zaplecze teoretyczne, przejdźmy do Mathematici. Na rysunku 2 widzimy trzy przykładowe okna końcowego wyniku programu, który napisałem w Mathematice.



Rys. 2. Interaktywne okna programu

Mamy tutaj okno z interaktywnym interfejsem oraz parametrami (po lewej) i wizualizacją (po prawej). Z rozwijanej listy *preset* możemy wybrać domyślny zbiór początkowy, taki jak *sierpinski triangle*, lub wybrać *custom* i wpisać własny. Robimy to, ustalając parametr *division* (rozdzielczość kwadratu) i wpisując współrzędne i wielkości kwadratów w listę w obszarze *initial set*. Kolejne iteracje zbioru możemy potem generować, zwiększając parametr *iteration*. Za pomocą przycisku *grid* możemy wyświetlić siatkę, a za pomocą *zoomed square* i *zoom value* uruchomić animację przybliżania wybranego fragmentu fraktala. Poniżej znajduje się przycisk *inverse* do inwersji kolorów i suwaki kolorów: *fractal* i *background*. Dalej mamy pokazane parametry fraktala: długość danych wejściowych *input length* i limit iteracji *iterations* (obliczany za pomocą prostej funkcji ze względów wydajnościowych). Wreszcie, na samym dole mamy wyświetlony wymiar Hausdorffa: dokładny – *exact* (o ile analityczne wyznaczenie jest możliwe) oraz przybliżony – *approx*.

Widać, że oprócz obszaru wyświetlania mamy tu kontrole najróżniejszych typów: listy rozwijane, pola wyboru, pola tekstowe, suwaki i palety kolorów. W Mathematice taki interfejs można łatwo stworzyć za pomocą funkcji *Manipulate*, która opisuje dynamiczne środowisko do obliczeń. Pierwszym argumentem funkcji jest wyrażenie, które ma być dynamicznie aktualizowane. W naszym przypadku są to wszystkie niezbędne obliczenia i rekurencyjna formuła generująca grafikę. Drugim argumentem jest lista kontrolowalnych parametrów i ich ustawienia. Oprócz tego możemy też podać opcje, które pomogą nam dostosować zachowanie i wygląd środowiska. Wszystkie te trzy elementy występują na schemacie 1.

```
Manipulate[[...]
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[{[...], fractal[setNorm, iter]}, [...]],
[...],
{{div, 2, "division"}, 2, 12, 1},
{{iter, 1, "iterations"},
Thread[Range[lim] -> Range[lim]]},
{{set, "", "initial set"}},
[...]
ControlType -> {Automatic, PopupMenu, Setter,
  Setter, InputField, Automatic, Automatic, Setter,
  Automatic, Automatic, Automatic, ColorSlider,
  ColorSlider, Automatic},
ControlPlacement -> Left]
```

Schemat 1. Fragment głównej funkcji programu.

Omówimy teraz funkcję obliczającą współrzędne kwadratów do wyświetlenia przedstawioną na schemacie 2. Funkcja ta podzielona jest na cztery mniejsze funkcje. Pierwsza z nich, nazwana *shift*,

bierze listę kwadratów i numer jednego z nich. Za pomocą obliczeń algebraicznych zwraca listę kwadratów o współrzędnych zaczepionych w danym kwadracie. Następna funkcja to *scor*, która zwraca współrzędne wszystkich kwadratów w następnej iteracji, a więc zwraca długą listę kwadratów potraktowanych funkcją *shift*. Korzystamy tutaj ze wbudowanej instrukcji *Table* tworzącej listę oraz wbudowanej funkcji *Flatten*, która listę list spłaszcza do jednej długiej listy. Ostatnia z naszych funkcji to *fscor*, która zagnieżdża funkcję *scor* aż do limitu iteracji. Zagnieżdżanie funkcji robimy za pomocą wbudowanej funkcji *Nest*. Otrzymujemy więc bardzo długą listę współrzędnych małych kwadratów. Na koniec zamieniamy ją na obiekty graficzne typu *Rectangle* za pomocą funkcji *square*. Ostatecznie przykładamy funkcję *square* do wszystkich elementów listy, za pomocą operatora /@. Na wyjściu mamy więc listę kwadratów, które wyświetlamy potem w głównej funkcji wewnątrz środowiska *Graphics*.

```
fractal[setNorm_, iter_] :=
 Module[{inputSet = setNorm},
 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[list_] :=
  Flatten[Table[shift[list, i],
   {i, 1, Length[list]}], 1];
 fcor[list_, it_] := Nest[scor, list, it - 1];
 square[{x_, y_, r_}] :=
  Rectangle[{x, y}, {x + r, y + r}]
 square /@ fcor[inputSet, iter]]
```

Schemat 2. Funkcja licząca współrzędne kwadratów.

Mamy już wizualizację, omówmy zatem funkcje obliczające wymiar Hausdorffa przedstawione na schemacie 3. Są to funkcje *dimension* oraz *ndimension*, które obliczają odpowiednio dokładny i przybliżony wymiar. Pierwsza z nich korzysta ze wbudowanej funkcji *Solve*, która w analityczny sposób próbuje rozwiązać układ równań. Funkcja ta nie zawsze prowadzi do rozwiązania. W takim przypadku za pomocą instrukcji *If* powodujemy, aby zwróciła odpowiedni napis. Druga funkcja używa bardzo podobnej wbudowanej funkcji *NSolve*. Różnica jest taka, że wynik wyznaczony jest numerycznie z odpowiednią precyzją. Zawsze otrzymamy wynik jako przybliżoną liczbę wymierną w zapisie dziesiętnym. W Mathematice występuje wiele funkcji z dodaną w nazwie literą *N*. W szczególności sama funkcja *N* zwraca zaokrągloną wartość w systemie dziesiętnym. Takie funkcje są szczególnie przydatne, gdy nie mamy pewności, że rozwiązanie analityczne nie istnieje, tak jak w naszym przypadku.

```
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]]
```

Schemat 3. Funkcje liczące wymiar Hausdorffa.

W programie mamy także kilka funkcji sprawdzających poprawność danych wprowadzonych przez użytkownika (aby zapobiec nieprzewidzianym zachowaniom programu). Jedną z nich jest funkcja *checkPattern*, która sprawdza, czy wejście spełnia podstawowe założenia, tzn. jest listą trójek złożonych z liczb całkowitych. W funkcji tej wykorzystywane są tzw. wzorce, czyli zapytania dotyczące liczby i rodzaju argumentów. Przykładowo wzorzec *List[___]* oznacza listę z dowolną liczbą argumentów, a *List[_Integer, _Integer, _Integer]* oznacza listę złożoną z trzech liczb całkowitych. Wzorce są sprawdzane za pomocą wbudowanej funkcji *MatchQ*. U nas sprawdzamy najpierw, czy na wejściu otrzymaliśmy listę (jeśli nie, to zwracamy domyślny zbiór). Następnie, za pomocą pętli *For*, przechodzimy po całej liście, sprawdzając, czy są to trójki liczb całkowitych. Elementy niepasujące do tego wzorca są usuwane za pomocą wbudowanej funkcji *Delete*.

```
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]
```

Schemat 4. Jedna z funkcji sprawdzająca poprawność danych wejściowych.

Program składa się jeszcze z kilku elementów, których nie pokazaliśmy. Przedstawione tu funkcje pokazują jednak główną ideę działania programu w Mathematice. Wykorzystaliśmy wbudowane funkcjonalności Mathematici, takie jak środowisko *Dynamic* (funkcja *Manipulate*), funkcje operujące na listach (*Flatten*, *Table*, operatory), zagnieżdżanie funkcji (funkcja *Nest*), analityczne i numeryczne obliczanie równań (funkcje *Solve* i *NSolve*) oraz różne funkcje operujące na wzorcach. Użyliśmy także konstrukcji typowych dla języków programowania, takich jak *If* czy *For*. Za pomocą kilku średniej długości funkcji stworzyliśmy rozbudowane narzędzie do wizualizacji fraktali i obliczenia wymiaru Hausdorffa, co jest przecież zadaniem dość nietrywialnym.

Literatura

[1] http://www.wolfram.com/mathematica/
[2] Falconer K., Fractal Geometry: Mathematical Foundations and Applications, John Wiley & Sons Ltd., Chichester 1990
[3] Kudrewicz J., Fraktale i chaos, Wydawnictwa Naukowo-Techniczne, Warszawa 1996
[4] Tempczyk M., Teoria chaosu dla odważnych, Wydawnictwo Naukowe PWN SA, Warszawa 2002