Przykładowe rozwiązania laboratoriów 3 z Modelowania Matematycznego

Zbiór Cantora

Zapisujemy wartości N oraz l odpowiadające kolejnym metodom pokryca zioru Cantora odcinkami:

In[∘]:=
```
lista = {{1, 1}, {1/3, 2}, {1/9, 4}, {1/27, 8}}
```

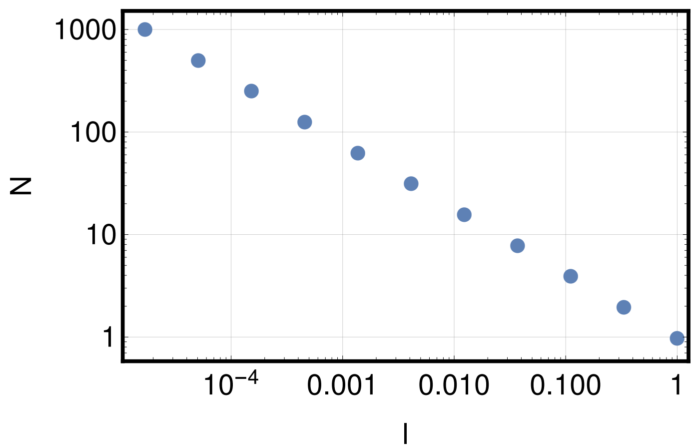
Out[∘]= $\{\{1, 1\}, \{\frac{1}{3}, 2\}, \{\frac{1}{9}, 4\}, \{\frac{1}{27}, 8\}\}$

Zgadujemy ogólny przepis postaci

In[∘]:=
```
lista = Table[{3^-n, 2^n}, {n, 0, 10}];
```

In[∘]:=
```
llp = ListLogLogPlot[lista, Frame → True, FrameStyle → Directive[Thick, 15],
  GridLines → Automatic, FrameLabel → {"l", "N"}, PlotMarkers → {Automatic, Medium}]
```

Out[∘]=



Dopasowujemy relację potęgową

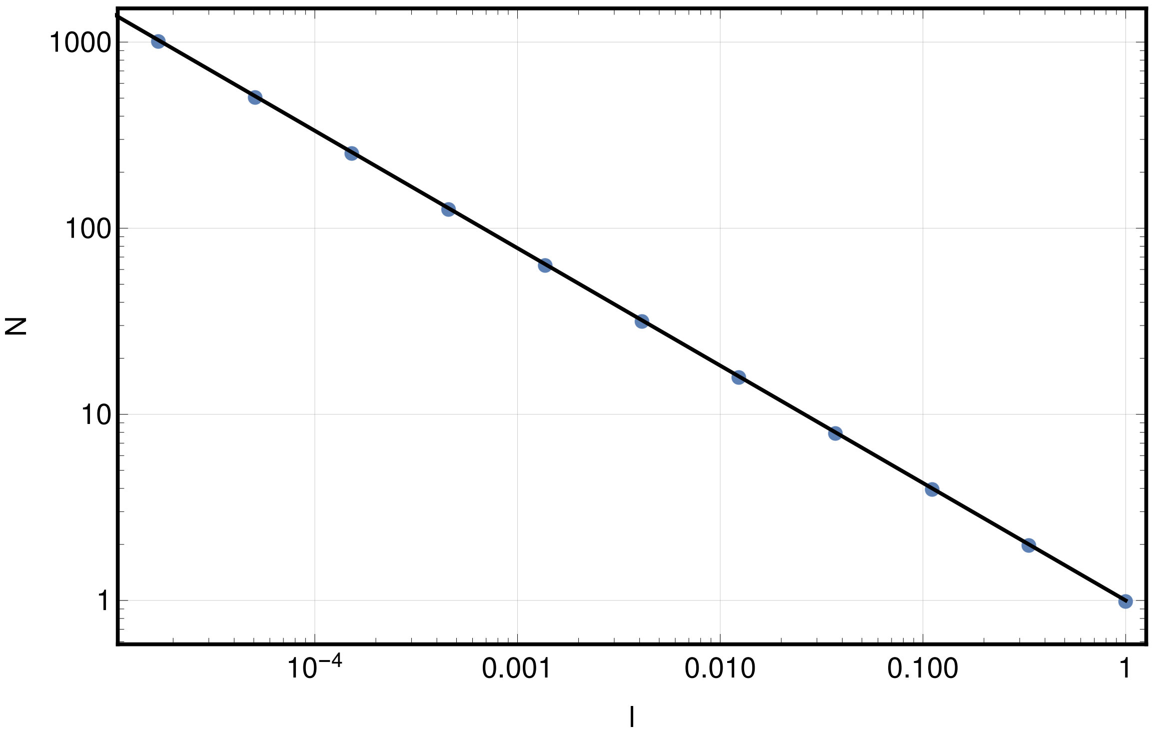
In[∘]:=
```
fit = FindFit[lista, a x^b, {a, b}, x]
```

Out[∘]= {a → 1., b → -0.63093}

Sprawdzamy czy przewidywanie jest dobre

In[•]:=
```
Show[llp,
 LogLogPlot[Evaluate[a x^b /. fit], {x, 10^-6, 1}, PlotStyle → Directive[Black, Thick]]]
```

Out[•]=



Sprawdźmy wymiar fraktalny z teoretycznym

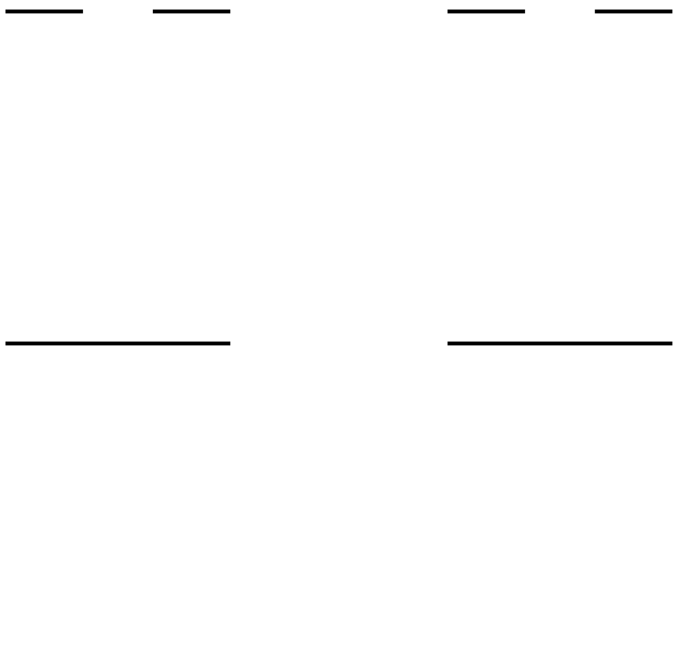
In[•]:=
```
Log[2] / Log[3] // N
```

Out[•]=
```
0.63093
```

Narysujmy zbiór Cantora: najpierw podejście naiwne, w którym ręczni wpisujemy kolejne etapy

In[◦]:=
```
Graphics[{Thick, Line[{{0, 1/2}, {1, 1/2}}],
  Line[{{0, 2/2}, {1/3, 2/2}}], Line[{{2/3, 2/2}, {3/3, 2/2}}],
  Line[{{0, 3/2}, {1/9, 3/2}}], Line[{{2/9, 3/2}, {3/9, 3/2}}],
  Line[{{6/9, 3/2}, {7/9, 3/2}}], Line[{{8/9, 3/2}, {9/9, 3/2}}]}]
```

Out[◦]=



Metoda bardziej zautomatyzowana (zachęcam do jej przeanalizowania, dla czytelności pozostawiam etapy posrednie, ale ze średnikiem na końcu, żeby się dało czytać)

In[•]:=
```
RandomReal[{0, 1}, {40}];
RealDigits[#, 3, 3, -1] & /@ RandomReal[{0, 1}, {40}];
RealDigits[#, 3, 1, -1][[1]] & /@ RandomReal[{0, 1}, {40}];
Select[RandomReal[{0, 1}, {100}], FreeQ[RealDigits[#, 3, 1, -1][[1]], 1] &]
Graphics[{LightGray, Rectangle[{-0.1, -0.1}, {1.1, 0.1}], Black, Point[{#, 0} & /@
    Select[RandomReal[{0, 1}, {100}], FreeQ[RealDigits[#, 3, 1, -1][[1]], 1] &]}]
```

Out[•]=
```
{0.958439, 0.276597, 0.760456, 0.0574913, 0.820713, 0.254615, 0.871778, 0.791844,
 0.833977, 0.0312574, 0.829756, 0.281618, 0.118817, 0.280521, 0.0437835,
 0.769174, 0.295871, 0.701031, 0.239369, 0.163857, 0.117388, 0.332793,
 0.0198804, 0.873407, 0.0772406, 0.235403, 0.152083, 0.210648, 0.668668,
 0.211963, 0.720511, 0.943981, 0.67972, 0.0443203, 0.947878, 0.709915,
 0.692548, 0.184474, 0.014869, 0.28539, 0.0727415, 0.753994, 0.0727776,
 0.10781, 0.976124, 0.145366, 0.841362, 0.00109985, 0.272986, 0.747274,
 0.980131, 0.133139, 0.156244, 0.807356, 0.0984021, 0.0187227, 0.255321,
 0.298421, 0.199664, 0.738847, 0.711779, 0.172265, 0.897501, 0.984916, 0.999632,
 0.140185, 0.201683, 0.729924, 0.89361, 0.913346, 0.789339, 0.672218, 0.250507}
```

Out[•]=

Zmieniamy teraz trzeci z parametrów funkcji RealDigits i obserwujemy kolejne “pokolenia”

In[1]:=
```
Table[Graphics[{LightGray, Rectangle[{-0.1, -0.1}, {1.1, 0.1}], Black,
   PointSize[0.01], Point[{#, 0} & /@ Select[RandomReal[{0, 1}, {1000}],
      FreeQ[RealDigits[#, 3, n, -1][[1]], 1] &]]}], {n, 6}]
```

Out[1]=

In[2]:=
```
Graphics[{LightGray, Rectangle[{-0.01, -0.01}, {1.01, 0.01}],
  Black, PointSize[0.001], Point[{#, 0} & /@
    Select[RandomReal[{0, 1}, {1000}], FreeQ[RealDigits[#, 3, 6, -1][[1]], 1] &]]}]
```

Out[2]=

Wymiar pudełkowy dla rzeczywistej grafiki

Wczytujemy ulubioną grafikę

In[•]:=
```
pl = CountryData["Poland", "Shape"]
```

Out[•]=

Sprawdzamy rozmiar obrazka

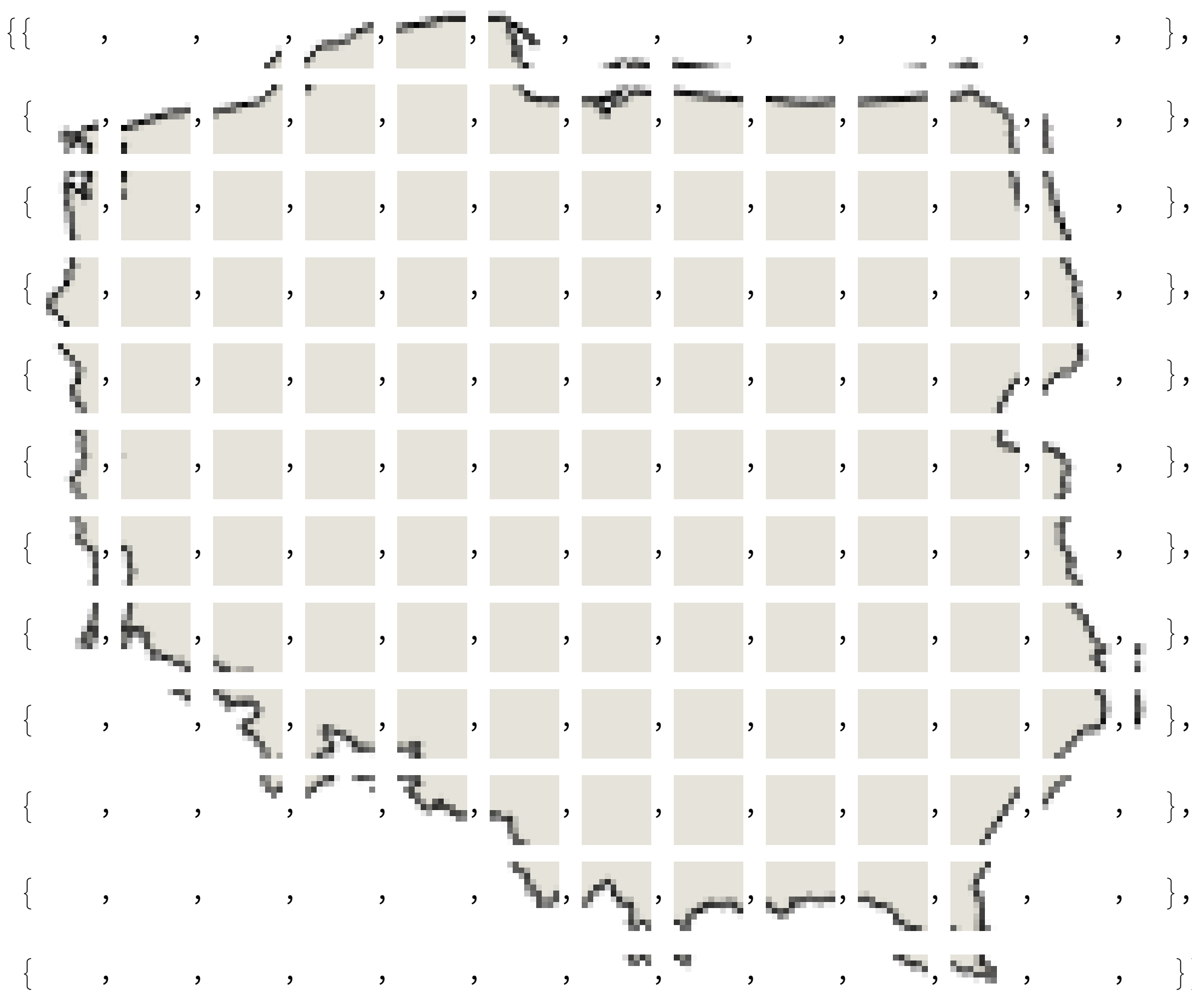
In[•]:=
```
dimpl = ImageDimensions[pl]
```

Out[•]=
```
{124, 117}
```

Przyjmujemy za parametr δ rozmiar “plasterka” na jakie kroimy grafikę

In[•]:=
```
δ = 10;
Table[ImageTrim[pl, {{i, j}, {i + δ, j + δ}}],
 {j, dimpl[[2]] - δ, -δ, -δ}, {i, 0, dimpl[[1]], δ}]
```

Out[•]=



Wyszukujemy plasterki, które mają w sobie tylko kolor szary

In[∘]:=
```
szary = Mean@Flatten[Map[Total, ImageData[
```

```
], {2}]]
δ = 10;
Count[Mean@Flatten[Map[Total, ImageData[#], {2}]] & /@
  (Flatten@Table[ImageTrim[pl, {{i, j}, {i + δ, j + δ}}],
     {j, dimpl[[2]] - δ, -δ, -δ}, {i, 0, dimpl[[1]], δ}]), szary]
```

Out[∘]= 2.64314

Out[∘]= 63

```
2.6430000000000002`
```

Out[∘]= 0

Zauważmy, że możemy wszystkie obliczenia przyspieszyć od razu analizując macierz, a nie obrazek Drobne różnice wynikają z różnic w uśrednianiu wartości pikseli, nie powinny nas one niepokoić.

```
δ = 10; pltab = Map[Total, ImageData[pl], {2}];
RepeatedTiming@Count[Mean /@
   (Flatten[#] & /@ Flatten[Partition[pltab, {δ, δ}, δ, 1, 10.], 1]), u_ /; u == szary]
RepeatedTiming@Count[Mean@Flatten[Map[Total, ImageData[#], {2}]] & /@
   (Flatten@Table[ImageTrim[pl, {{i, j}, {i + δ, j + δ}}],
      {j, dimpl[[2]] - δ, -δ, -δ}, {i, 0, dimpl[[1]], δ}]), szary]
```

Out[∘]= {0.0018, 65}

Out[∘]= {2.38, 63}

Sprawdźmy o ile przyspieszyło (dla $\delta = 10$)

In[∘]:=
```
2.38 / 0.001840526119402985`2.
```

Out[∘]= 1293.11

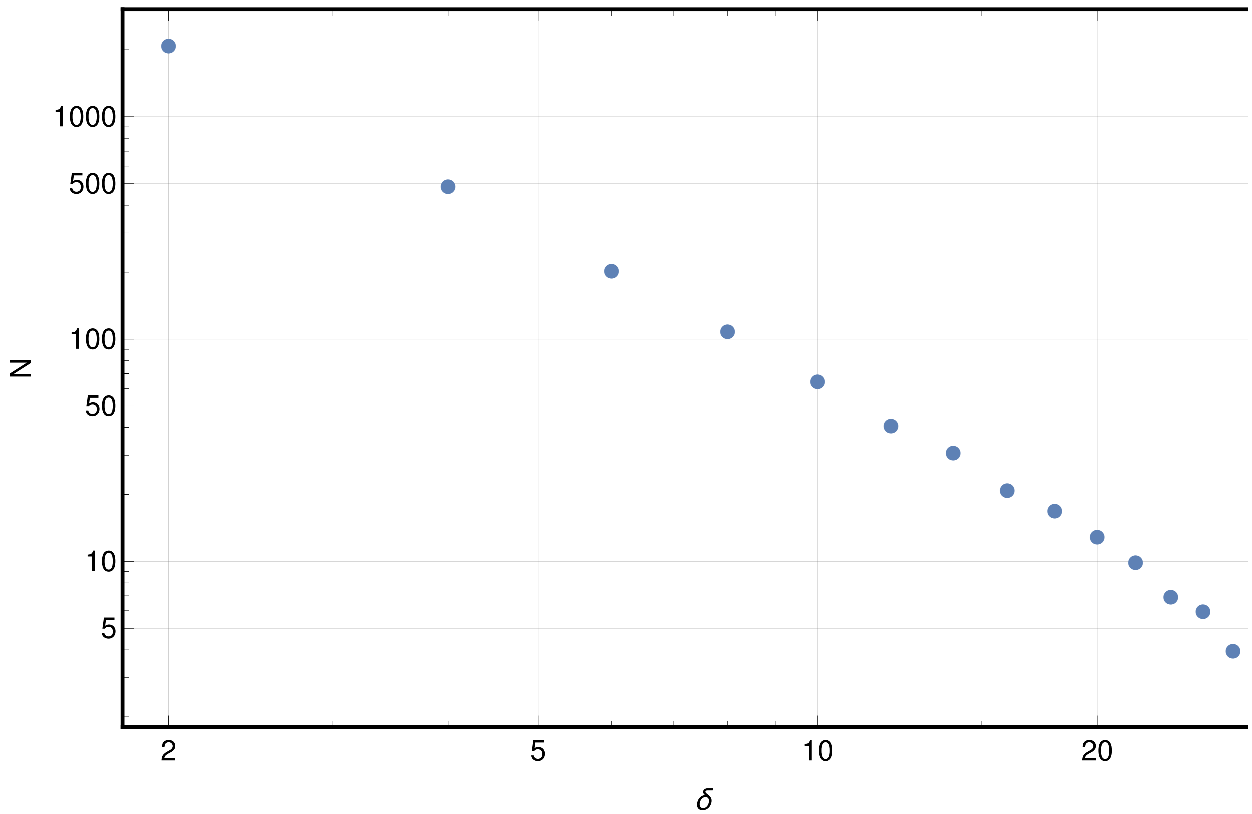
Skupmy się zatem na metodzie szybszej: policzymy N dla różnych rozmiarów l=δ

In[∘]:=
```
tab = Table[
  {δ, Count[Mean /@ (Flatten[#] & /@ Flatten[Partition[pltab, {δ, δ}, δ, 1, 10.], 1]),
    u_ /; u == szary]}, {δ, 2, 30, 2}]
```

Out[∘]=
```
{{2, 2097}, {4, 491}, {6, 204}, {8, 109}, {10, 65}, {12, 41}, {14, 31},
 {16, 21}, {18, 17}, {20, 13}, {22, 10}, {24, 7}, {26, 6}, {28, 4}, {30, 3}}
```

In[∘]:=
```
llp = ListLogLogPlot[tab, Frame → True, FrameStyle → Directive[Thick, 15],
  GridLines → Automatic, FrameLabel → {"δ", "N"}, PlotMarkers → {Automatic, Medium}]
```

Out[∘]=



Dopasowujemy relację potęgową

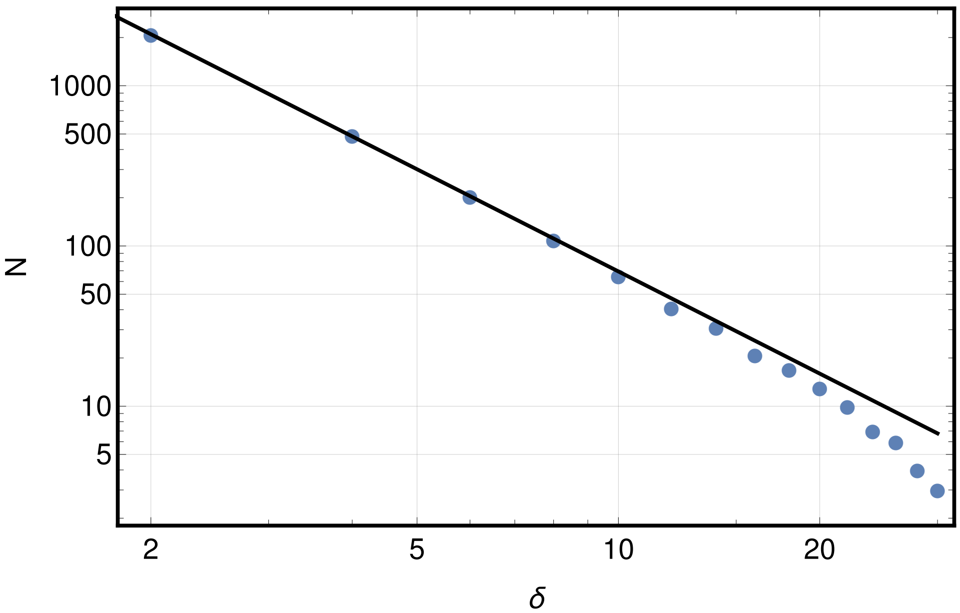
In[∘]:=
```
fit = FindFit[tab, a x^b, {a, b}, x]
```

Out[∘]=
```
{a → 9110.35, b → -2.11843}
```

Sprawdzamy czy przewidywanie jest dobre

In[ ]:=
```
Show[llp,
 LogLogPlot[Evaluate[a x^b /. fit], {x, 1, 30}, PlotStyle → Directive[Black, Thick]]]
```

Out[ ]=

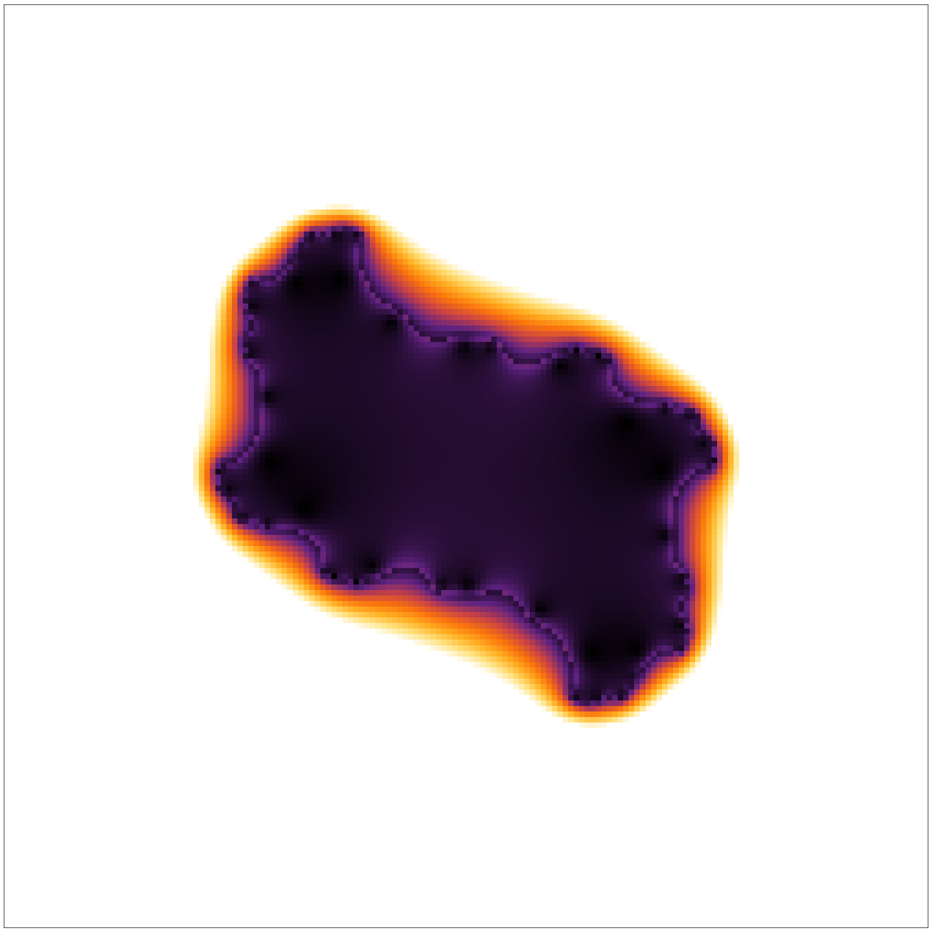


Wymiar ok. 2.118 :)

Zbiór Julii i Mandelbrota

In[3]:=
```
z = Table[x + I y, {x, -2, 2, 0.025}, {y, -2, 2, 0.025}];
c = 0.2 - 0.5 I;
ArrayPlot[Map[If[Abs[#] > 2, Min[Log@Abs[#], 10], Abs[#]] &,
  Nest[#^2 + c &, z, 6], {2}], ColorFunction → "SunsetColors"]
```

Out[5]=



In[6]:=

```
z = ParallelTable[If[Abs[#] > 2, Min[Log@Abs[#], 10], Abs[#]] &@
    Nest[(x + I y) + #^2 &, 0, 7], {x, -2.5, 1, 0.025}, {y, -1.5, 1.5, 0.025}];
ArrayPlot[Transpose@z, ColorFunction → "AvocadoColors"]
```

Out[7]=