

Fraktale in der Natur

Folgen und Häufungspunkte

Prof. Dr. Rebekka Axthelm (HTWG)



Tutorial

-1-

Von Natur aus fraktal: iterierte Funktionensysteme (IFS)

1.1 Arbeitsanleitung

Schritt 1: Erzeugen Sie eine Funktion `fraktale()` in der Datei `fraktale.m`.

Schritt 2: Schreiben Sie eine Funktion `IFS()`, die das Feld

```
Q=[ [0      0      0      0.16  0  0      0.01];
     [0.20  -0.26  0.23  0.22  0  1.60  0.07];
     [-0.15  0.28  0.26  0.24  0  0.44  0.07];
     [0.85  0.04  -0.04  0.85  0  1.60  0.85]];
```

zurückgibt. Als Argument können Sie später zwischen verschiedenen Zahlentabelle wählen. Im Moment haben wir nur das eine. Wenn Sie `fraktale` selbst als Funktion schreiben, können Sie `IFS()` als Subfunktion in der gleichen Datei definieren.

```
Datei fraktale.m

function fraktale()

    ...

    % Aufruf
    Q=IFS()

    ...

end

function Q=IFS()

    Q = ...

end
```

Jede weitere Funktion kann dann einfach unten angefügt werden¹.

Schritt 3: Q enthält in jeder Zeile

$$Q = \begin{pmatrix} A_{11}^1 & A_{12}^1 & A_{21}^1 & A_{22}^1 & b_1^1 & b_2^1 & P_1 \\ A_{11}^2 & A_{12}^2 & A_{21}^2 & A_{22}^2 & b_1^2 & b_2^2 & P_2 \\ A_{11}^3 & A_{12}^3 & A_{21}^3 & A_{22}^3 & b_1^3 & b_2^3 & P_3 \\ A_{11}^4 & A_{12}^4 & A_{21}^4 & A_{22}^4 & b_1^4 & b_2^4 & P_4 \end{pmatrix}$$

eine affine Abbildung

$$\mathcal{A}^i(x) = A^i x + b^i, \quad i \in \{1, 2, 3, 4\}.$$

Die letzte Spalte von Q ist ein Feld mit Wahrscheinlichkeiten $P = (0.01, 0.07, 0.07, 0.85)$. Daraus bilden wir eine implizite Folge von Zahlen in \mathbb{R}^2

$$x^0 = \begin{pmatrix} 0.5 \\ 0.5 \end{pmatrix}$$

für $k = 0, 1, 2, \dots$

$$\begin{pmatrix} x_1 \\ x_2 \end{pmatrix}^{k+1} = \begin{pmatrix} A_{11}^i & A_{12}^i \\ A_{21}^i & A_{22}^i \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}^k + \begin{pmatrix} b_1^i \\ b_2^i \end{pmatrix}$$

$i \in \{1, 2, 3, 4\}$, wobei die Auswahl der Abbildung \mathcal{A}^i gemäß der in P definierten Wahrscheinlichkeit erfolgt. Zu 1 % Abbildung \mathcal{A}^1 , zu je 7 % die Abbildung \mathcal{A}^2 und \mathcal{A}^3 , und schließlich zu 85 % die Abbildung \mathcal{A}^4 . Klar, oder?

Alle Punkte

$$\{x^0, x^1, x^2, \dots\} \subset \mathbb{R}^2$$

plotten wir in ein Bild. Das ist schon alles.

Schritt 4: Die Ausführung der i-ten Abbildung können Sie direkt mit Q durchführen:

```
x=[0.5 0.5];
xd(1) = Q(i,1:2)*x';
xd(2) = Q(i,3:4)*x'; % = A^i * x
x = xd+Q(i,5:6); % = A^i * x + b^i
```

¹Wollen Sie fraktale.m als Matlabskript belassen so müssen Sie die Funktion IFS in eine eigene Datei namens IFS.m definieren; und auch jede weitere Funktion. Diese sollten im gleichen Verzeichnis liegen, andernfalls muss ein Pfad gesetzt werden. Der Befehl dazu lautet addpath. Diese Variante hat den Vorteil, dass noch alle Variablen vorhanden sind. Schreibt man das Skript als Funktion sind die Variablen, nach Verlassen der Funktion, nicht mehr vorhanden. Variablen in Funktionen sind lokal, sofern man sie nicht als global definiert hat.

Versuchen Sie das mal.

Dann wollen Sie diese Abbildung mehrfach hintereinander ausführen, wobei die Zeile i zufällig gewählt werden soll.

Schritt 5: Schreiben Sie eine Funktion `RandP(P)`, die eine zufällige Zahl von 1 bis 4 ($=\text{length}(P)$) ausgibt, wobei die Wahrscheinlichkeit der einzelnen Werte in P definiert ist².

```
function i = RandP(P)
CP = cumsum(P)*100; % = [1,8,15,100]
ir = round(rand*100-1); % Zufallsz. aus [0,99]
i=1;
for l=length(CP)-1:-1:1
    if ir>=CP(l)
        i=l+1;
        break;
    end
end
```

Schritt 6: Mit `RandP` können Sie schon den eigentlichen Algorithmus fertigstellen:

```
x=[0.5 0.5];

for loop=1:LOOPmax % LOOPmax zu Beginn definieren
    i = RandP(Q(:,7))
    xd(1) = Q(i,1:2)*x';
    xd(2) = Q(i,3:4)*x';
    x = xd+Q(i,5:6);
end
```

Schritt 7: Nun wollen Sie natürlich was sehen. Plotten Sie den Startwert mit einem `hold on` und in der Schleife jeden neu berechneten Punkt. Probieren sie das mit `LOOPmax=400` oder weniger.

²Zum Test können Sie in einer Schleife 1000 Mal eine solche Zufallszahl berechnen und sich das Histogramm dazu anschauen. Sie sollten darin die Werte in P wiedererkennen.

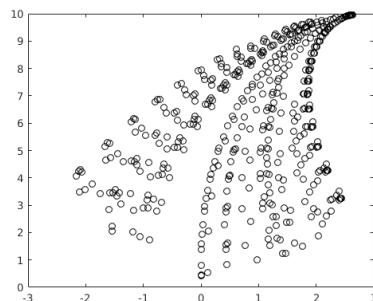
```
close all % schließt plots

x=[0.5 0.5];

plot(x(1),x(2), 'ko')
hold on

for loop=...
    ...
    x = ...
    plot(x(1),x(2), 'ko')
end
```

Es sollte ungefähr so aussehen:

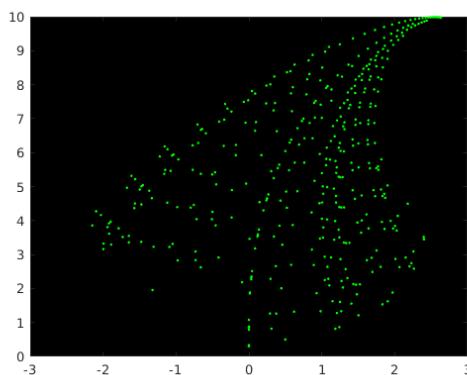


Schritt 8: Sie können die Graphik etwas verfeinern:

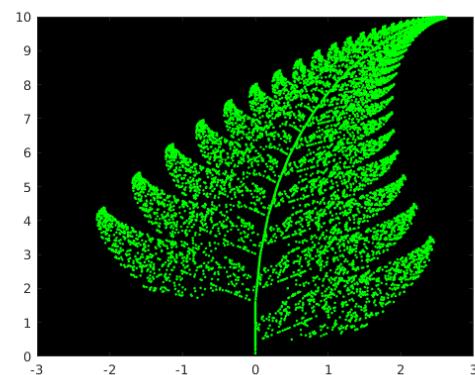
```
% feine dots in dunklem grün
plot(x(1),x(2),'.','color',[0 0.8 0])
% mit schwarzem Hintergrund
set(gca,'color',[0 0 0]);
```

Sowas sollten Sie nun sehen können

LOOPmax = 400



LOOPmax = 10000



Das ist schon ziemlich schick und man hat das Bedürfnis den Wert für LOOPmax weit hochzustellen. Das sollen Sie auch tun dürfen, nur werden Sie schnell bemerken, dass die Rechenzeit auch weit "hochgestellt" wird. Sie können mit tic in der ersten Zeile der Funktion fraktale und toc in der letzten Zeile eine Zeitangabe des Programmablaufs anfordern. Vergleichen Sie die Zeiten mit und ohne Plots. Einfach mal so.

Schritt 9: Was also tun? Am besten Sie plotten das Ergebnis in ein hochauflöste digitales Bild und speichern es als .png. Da das nicht das wesentliche Thema ist, aber den besseren "Wow"-Effekt liefert, gebe ich Ihnen hier eine Anleitung³:

```
% Urbild und Bildbereich
% einmal berechnen mit weniger
% loops oder im plot anschauen
Xminmax = [-2.152858 0.500000 2.653972 9.964024];

M = ... % Anzahl Zeilen des Bildes,
    % Tipp: =round(sqrt(LOOPmax))*3;
N = round(M*(Xminmax(3)-Xminmax(1))...
           /(Xminmax(4)-Xminmax(2)));
Pic = zeros(M,N); % schwarzes Bild als Init.

% in der Iterationsschleife
...
x = ...

raw = M+1-min(M,max(1,RawCol(x(2),...
    Xminmax(2),Xminmax(4),M)));
col = min(N,max(1,RawCol(x(1),...
    Xminmax(1),Xminmax(3),N)));

Pic(raw,col)=1;

% nach der Iterationsschleife

PicRGB = zeros(M,N,3); % farbiges Bild
PicRGB(:,:,2)=0.8*Pic; % in Dunkelgrün
% imshow(PicRGB) % zeigt das Bild direkt an
imwrite(PicRGB, 'fractal.png')
```

Dabei wurde die Funktion

³ Alle zum Plot gehörenden Zeilen jetzt auskommentieren.

```
function ij=RawCol(x,xmin,xmax,L)
% ordnet eine Koordinate einer Zeile oder
% Spalte (je nachdem) im Bild zu

ij = round(1+(x-xmin)/(xmax-xmin)*(L-1));

end
```

verwendet.

Hat es geklappt? Das Bild auf der Titelseite wurde mit folgenden Parametern erstellt:

LOOPmax = 10^7
M = 9486
Rechenzeit = 39.5 sec

Wenn Sie im Internet nach

“ifs fraktale”

suchen finden Sie noch weitere Beispiele. Sie müssen nur das Q austauschen und dann neu nach $xmin$, $xmax$, $ymin$, $ymax$ Ausschau halten.

1.2 Aufgaben

Nachdem Sie sich eine Weile an dem Anblick erfreut haben lade ich Sie nun dazu ein, ein wenig darüber nachzudenken, was Sie da sehen. Die Punkte, die wir geplottet haben sind Folgenglieder einer impliziten Folge.

Aufgabe 1: Welche Folgeneigenschaften können Sie erkennen?

Aufgabe 2: Welches sind die Teilfolgen?

Aufgabe 3: Untersuchen Sie die Teilfolgen separat. Sie könnten etwa Einheitsquadrate $[0, 1]^2$ abbilden, so wie wir Abbildungen in Mathe 1 untersucht haben. Was stellen Sie fest? Um welche Abbildungen handelt es sich (Drehung, Scherung, Spiegelung, etc)? Machen Sie Skizzen. Die Skizzen können Sie gerne mit Matlab erzeugen.

Aufgabe 4: Berechnen Sie die explizite Darstellung der impliziten Teilfolgen.

Tipp 1: Rechnen Sie mit allgemeinen Abbildungen $Ax + b$. Die Rechnung ist für alle Fälle von gleicher Art.

Tipp 2: Geometrische Reihe für Matrizen:

$$\sum_{k=0}^n A^k = (E - A)^{-1}(E - A^{n+1})$$

Welche Eigenschaft der Matrix führt auf Konvergenz der Reihe? Welche Eigenschaften der beteiligten Matrizen führen zu Konvergenz der Teilfolgeneigenschaften? Denken Sie dabei gerne auch an die Determinante einer Abbildung, bzw. Matrix.

Aufgabe 5: Wenn Sie nun die Abbildungen in ihren Eigenschaften ergründet haben können Sie sie ein wenig variieren und damit andere Formen kreieren. Probieren Sie das und schicken Sie Ihr spektakulärstes Ergebnis. :)