Chaotische dynamische Systeme mit SAS

Tomás Cámara, Tsunuun Consulting GmbH

KSFE 2022
Wiesbaden, 15. September 2022

0. Einführung

Zur Person

- Name: Tomás Cámara
- Gebürtiger Mexikaner, seit 15 Jahren in Deutschland
- Studium der angewandten Mathematik und Statistik, in Mexiko und Kanada
- Berufserfahrung in verschiedenen Sektoren: Zürich, Pemex, SAS, SAP, Deloitte unter anderem
- Seit 2017 selbständig; Projekte bei dmk in Bremen; Risikoabteilung Deutsche Bank; zur Zeit leite ich zwei Entwicklungsgruppen in einem Großprojekt bei der EZB
- Erfahrung mit SAS: seit 1996; ich lernte auf einer Mainframe in der Transition von Lochkarten(!) auf magnetische Tapes(!)

Zum Thema

Chaotische Dynamische Systeme wurden insbesondere in den 1980er Jahren vertieft untersucht

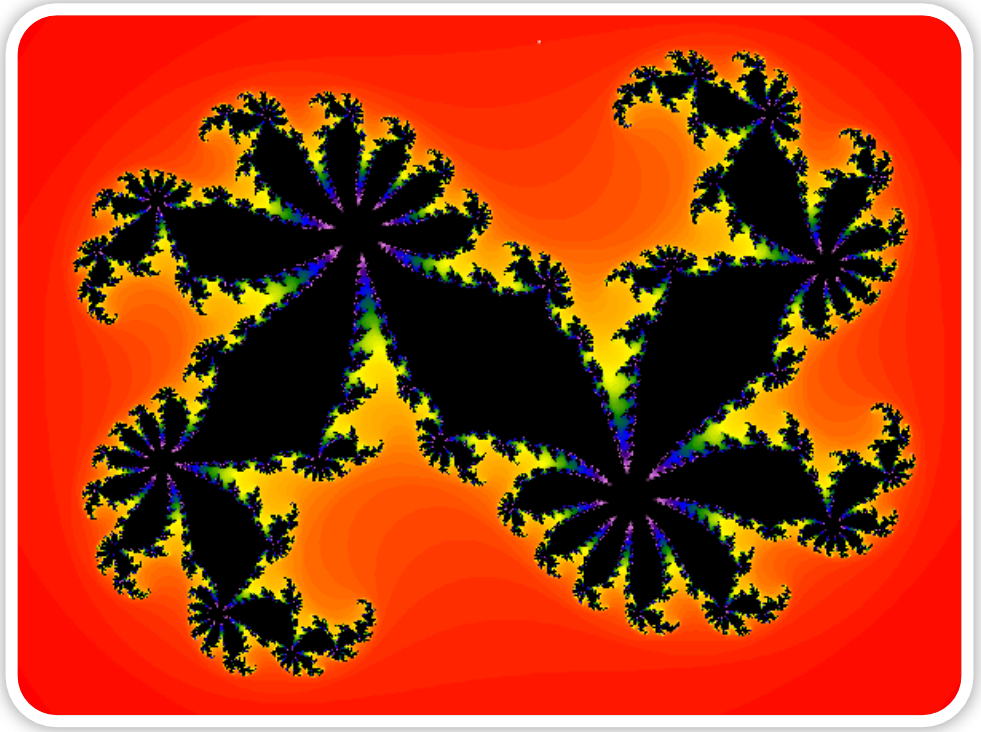
Das Konzept war nicht neu: Gaston Julia und Pierre Fatou veröffentlichen bereits 1918 Forschungs-ergebnisse zum Thema

In den späten 1970er Jahren war die Computertechnik rechnerisch soweit die Forschung zu unterstützen (John Hubbard und Adrien Douady)

Chaotische Dynamische Systeme ≠ Fraktale… aber: aus chaotischen Systemen entstehen Fraktale

Motivation es mit SAS zu machen:

- Es hilft, proc FCMP zu verstehen und vertiefen
- SAS / Graph kann seine Stärken zeigen
- … und: weil die Ergebnisse schön sind und Spaß machen



Julia set
c = 0.360284 + 0.100376i

1. Fraktale

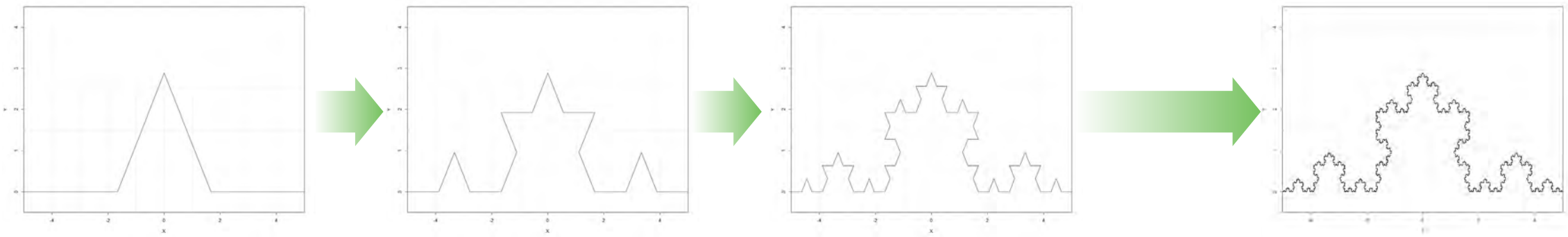
1. Fraktale: Definition und Beispiel

Eine lockere Definition

Ein Fraktal ist eine geometrische Menge mit folgenden Eigenschaften:

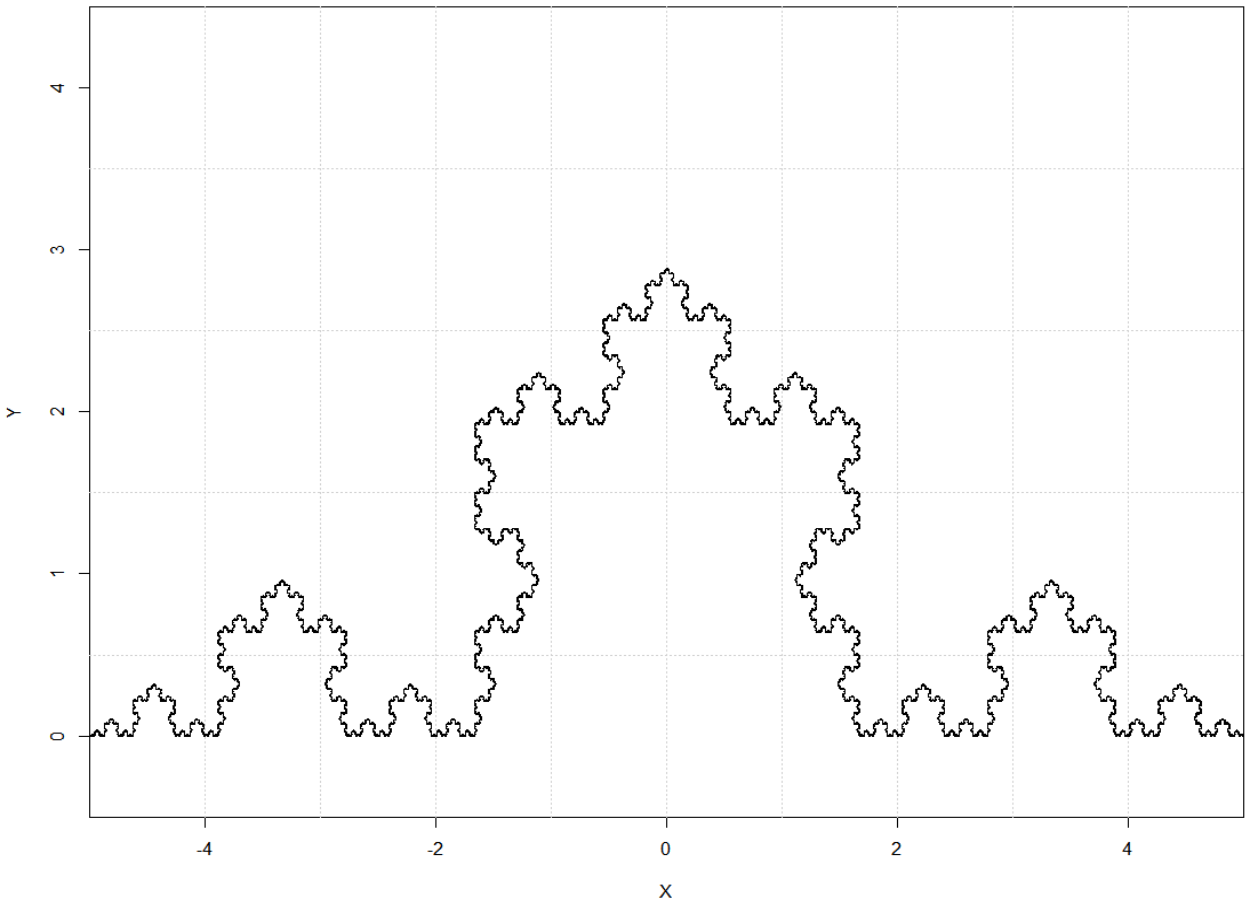
- Sie besitzt eine Feinstruktur auf beliebig kleinen Skalen
- Sie wird rekursiv definiert
- Oftmals besitzt sie Selbstähnlichkeit
- Sie besitzt eine fraktale Dimension, üblicherweise größer als ihre topologische Dimension

Beispiel: Koch'sche Kurve (die Schneeflocke)



n = 1 n = 2 n = 3 ... n = 9

1. Fraktale: Definition und Beispiel



- Feinstruktur auf beliebigen Skalen
- Rekursive Definition
- Selbstähnlichkeit
- Fraktale Dimension (nicht Teil des Vortrags)

Weitere Beispiele

- Barnsley Farn
- Cantor Menge
- Cantor Fläche
- Sierpinsky Dreieck
- Julia sets → später
- Mandelbrot set → später

2. Chaotische Dynamische Systeme

2. Chaotische Dynamische Systeme

Einige Definitionen im Baukasten

Benannte Systeme sind:

a. nichtlineare komplexwertige Funktionen,
b. die iterativ angewendet werden und…
c. deren Ergebnis auf der Gaußschen Ebene abgebildet werden kann

$\mathbb{C}$ ist die Menge der komplexen Zahlen, der Form $z = a + b \cdot i$, wo $a, b \in \mathbb{R}, i = \sqrt{-1}$

$f(z) \to \mathbb{C}$ ist eine komplexwertige Funktion, wenn $z, f(z) \in \mathbb{C}$

$f^n(z) \to \mathbb{C}$ ist die n-iterative Anwendung der Funktion f mit Anfangspunkt z

Beispiel: $f^3(z) = f(f(f(z)))$

Die Norm einer komplexen Zahl $z = a + bi$ ist definiert als $|z| = \sqrt{a^2 + b^2}$

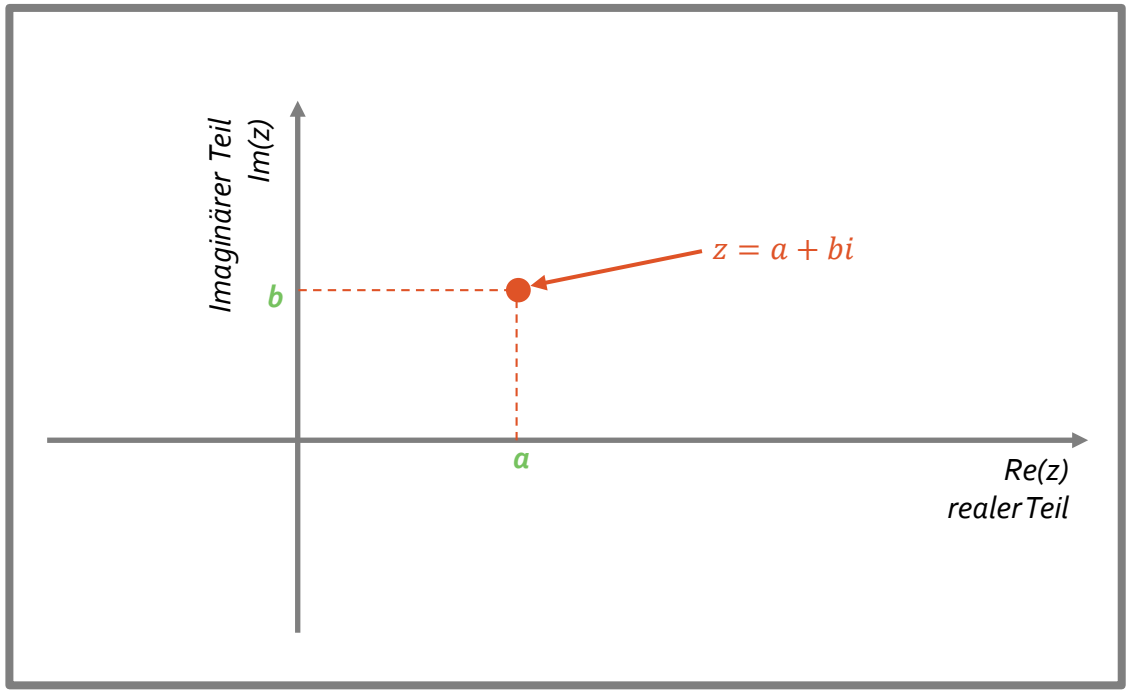
2. Chaotische Dynamische Systeme

Abbildung

Komplexe Zahlen $z \in \mathbb{C}$ lassen sich auf der Gaußschen Ebene abbilden, indem man den realen Teil und den imaginären Teil als Zahlenpaar abbildet

Bei der Analyse chaotischer dynamischer Systeme passieren nun drei Sachen:

1. Man definiert eine Funktion $f(z)$, deren Verhalten zu analysieren ist.
2. Auf allen Punkten der Gaußschen Ebene wird sie jezt n-iteriert (siehe vorherige Folie).
3. Abhängig von der Anzahl der Iterationen und der Norm des Ergebnisses, wird entsprechend eines Farbschemas der Punkt auf der Ebene gefärbt.



2. Chaotische Dynamische Systeme

Julia sets

- Julia sets werden durch die generische Funktion definiert

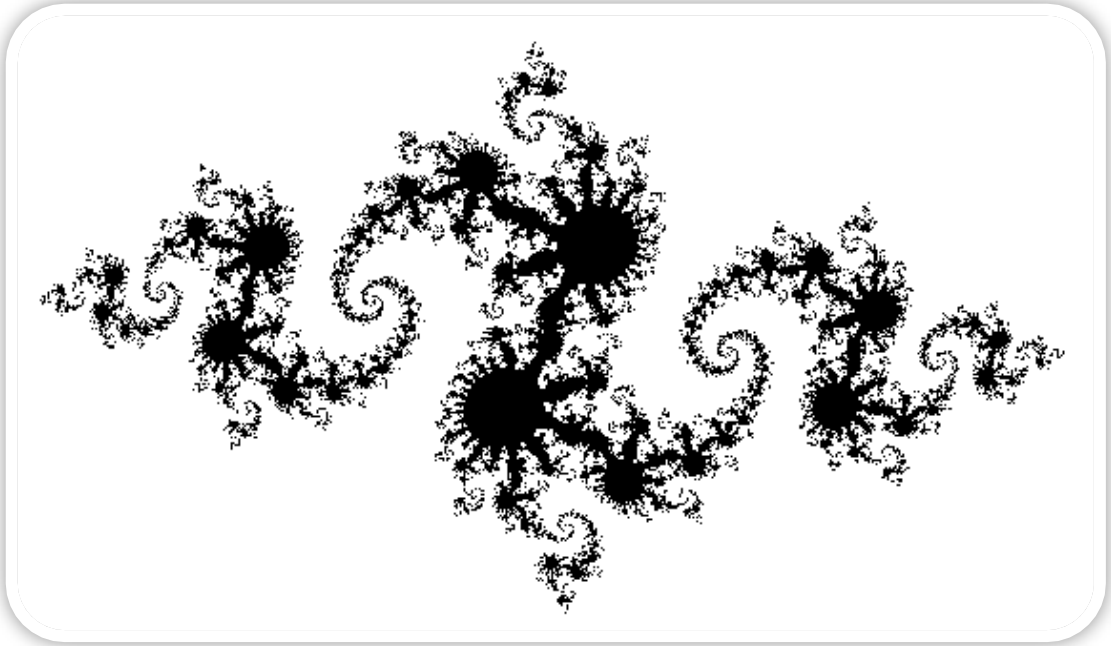
$$f(z) = z^2 + c; \;\; z, c \in \mathbb{C}$$

wo c eine beliebig gewählte konstante komplexe Zahl ist

2. Chaotische Dynamische Systeme

Beispiel 1

1. Definieren wir $f(z) = z^2 - 0{,}8 + 0.156i$, das Julia-Set mit $c = -0{,}8 + 0{,}156i$
2. Iterieren wir maximal 100 und nur dann wenn die Norm des Ergebnisses unter 2 bleibt
3. Farbschema: Schwarz wenn die maximale Iterationsnummer erreicht wurde, weiße Färbung für den Rest der Punkte



Interpretation: Schwarze Punkte bleiben unterhalb der Norm, weiße reißen sie

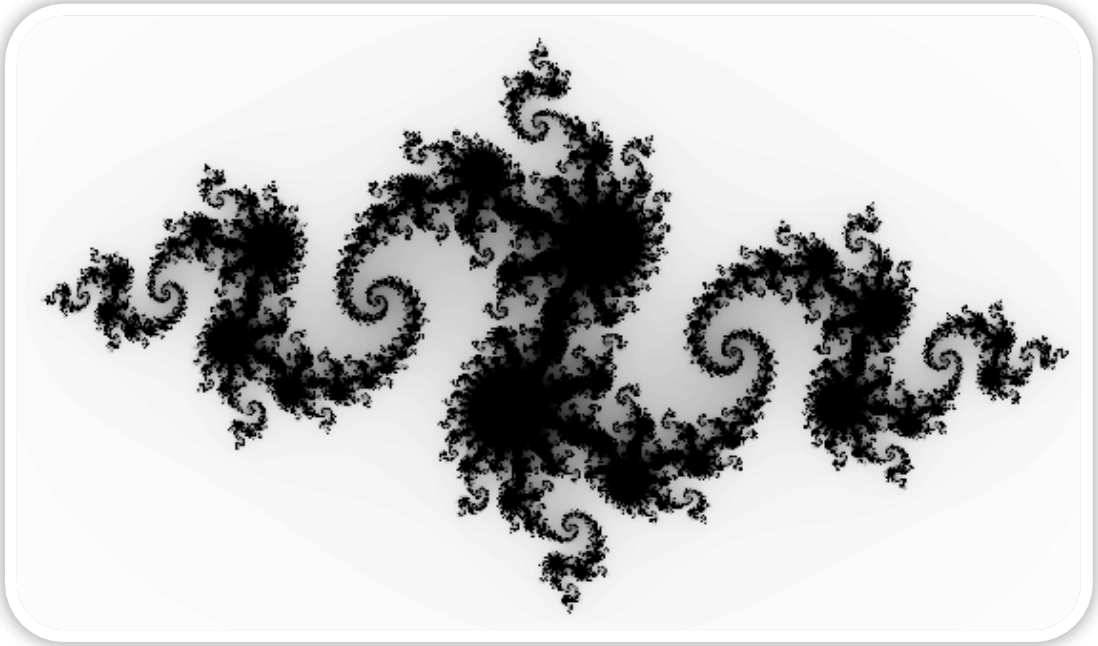
Dann erhalten wir … ein Fraktal!

2. Chaotische Dynamische Systeme

Beispiel 2

Für das vorgegangene Beispiel können wir aber auch die Anzahl der möglichen Iterationen unter Einhaltung der Grenznorm in skalierenden Weiß- bis Schwarztönen färben

3. Farbschema: Schwarz bis Weiß skalierend, abhängig von der möglichen Anzahl



Links, das entsprechende Bild

2. Chaotische Dynamische Systeme

Beispiel 3

Genauso wie beim Vorgänger, könnten wir die Anzahl der möglichen Iterationen unter Einhaltung der Grenznorm jetzt aber mit einem Farbschema abbilden

3. Farbschema: Bunt

Links das Ergebnis

3. Chaotische Dynamische Systeme mit SAS

3. Chaotische Dynamische Systeme mit SAS

Julia sets

Disclaimer: SAS erlaubt üblicherweise mehrere Lösungswege! Hier nur einer der möglichen.

Wie behandeln wir die Berechnung von Julia sets in SAS?

$$f(z) = z^2 + c = (Re(z) + i * Im(z))^2 + a + bi = Re(z)^2 - Im(z)^2 + a + i(2 * Re(z) * Im(z) + b)$$

Das heisst, in jeder Iteration berechnen wir:

- Den reellen Wert als $Re(z)^2 - Im(z)^2 + a$
- Den imaginären Wert als $2 * Re(z) * Im(z) + b$

Im SAS Code sieht das so aus:

```
31          temp=real_part;
32          real_part=real_part**2-imaginary_part**2 + a;
33          imaginary_part=2*temp*imaginary_part+b;
```

3. Chaotische Dynamische Systeme mit SAS

Julia sets

Wie behandeln wir die Iterationen mit SAS?

- Mit proc FCMP können vom Nutzer definierte Funktionen deklariert werden, die auch iteratives Verhalten erlauben.
- Das sieht in diesem Beispiel wie rechts aus.*

```
22 proc fcmp outlib=tomasito.funx.complex;
23     function julia_sets (x, y, a, b, threshold_n, threshold_r);
24     anzahl=0;
25     real_part=x;
26     imaginary_part=y;
27
28     do while (anzahl<threshold_n and
29             sqrt(real_part**2 + imaginary_part**2) <
30             max(threshold_r, sqrt(a**2 + b**2)));
31         temp=real_part;
32         real_part=real_part**2-imaginary_part**2 + a;
33         imaginary_part=2*temp*imaginary_part+b;
34         anzahl=anzahl+1;
35     end;
36     return(anzahl);
37     endsub;
38 run;
```

* Das Stoppkriterium ist aus dem Buch „A First Course in Chaotic Dynamical Systems“, Devaney, Addison-Wesley, 1992 entnommen.

3. Chaotische Dynamische Systeme mit SAS

Julia sets

Und SAS / GRAPH?

Mit der Vorgängerfunktion benötigen wir nur noch:

1. Ein Dataset mit einem engmaschigen Datenteppich
2. Ein Farbschema
3. **proc sgplot** und **scatter** um die Datenpunkte und den Iterationswert anzuzeigen

1. Der Datenteppich wird hier generiert

```
44 data julia_teppich;
45     do x=-&R to &R by 0.0025;
46         do y=-&R to &R by 0.0025;
47             *julia_wert=julia_sets(x, y, -0.8, 0.156, &iterations, &R);
48             julia_wert=julia_sets(x, y, 0.360284, 0.100376, &iterations, &R);
49             * julia_wert=julia_sets(x, y, -1, 0.0, &iterations, &R);
50             * julia_wert=julia_sets(x, y, +0.488, -0.20, &iterations, &R);
51             output julia_teppich;
52         end;
53     end;
54 run;
```

3. Chaotische Dynamische Systeme mit SAS

Julia sets

2. Farbschemata stehen hier zur Verfügung

```
5  %let mode_0_a = (black yellow);
6  %let mode_0_b = (white black);
7  %let mode_1= (red darkorange orange
8          red darkorange orange darkmagenta violet darkviolet magenta darkmagenta
9          lightgray gray black);
10 %let mode_2= (white whitesmoke gray darkgreen olivedrab green lightgreen white);
11 %let mode_3= (yellow lightgoldenrodyellow
12     lightyellow darkyellow CXFFFFF0 orange orangered red lightred whitesmoke
13     lightgray gray darkgray black);
14 %let mode_4= (gray darkgray gray lightgray whitesmoke white yellow orange);
15 %let mode_5= (darkblue darkorange orange yellow whitesmoke white black);
16 %let mode_6 = (lightblue blue green red yellow orange gray black
17                 lightgreen darkgreen orange black);
18 %let mode_7 = (black darkgray blue gray lightgray white);
19 %let devaney_scheme = (red orange yellow green blue indigo violet black);
```

Ich empfehle das *devaney_schema*, sicherlich aber eine persönliche Präferenz

3. Chaotische Dynamische Systeme mit SAS

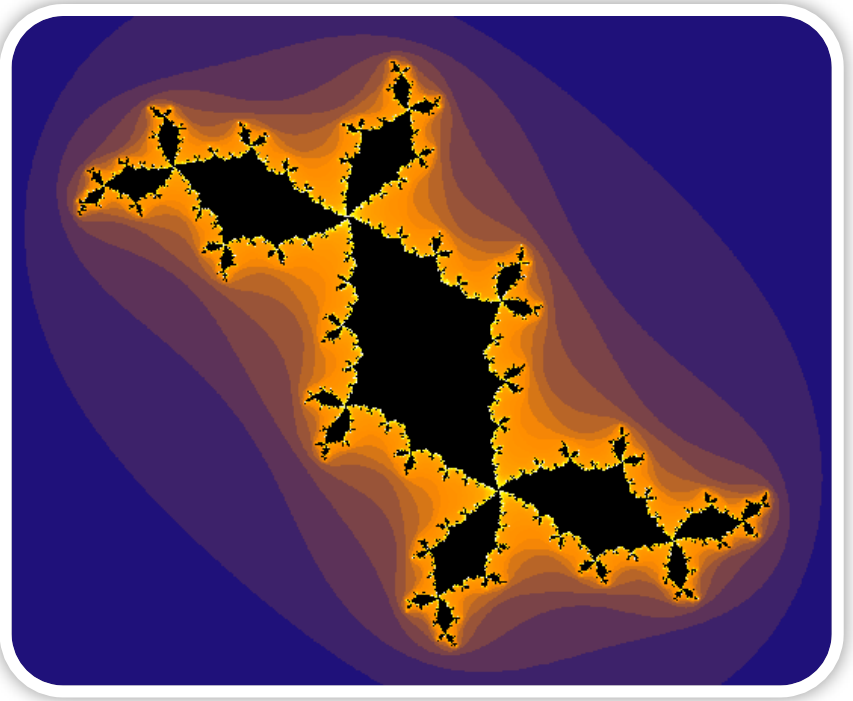
Julia sets

3. proc sgplot und scatter machen jetzt den Rest

```
58 proc sgplot data=julia_teppich noautolegend noborder nowall;
59     scatter x=x y=y / colorresponse=julia_wert colormodel=&devaney_scheme;
60     xaxis display=none;
61     yaxis display=none;
62 run;
```

Und somit lassen sich flexibel unterschiedliche Julia Sets austesten…

Hier rechts: c = -0,1 + 0,8i
und *colormodel=&mode_5*



3. Chaotische Dynamische Systeme mit SAS

Das Mandelbrot Set

Und das Mandelbrot Set…? Bonus! Wir haben bereits alles, was wir brauchen.

Ein Julia Set wird über ein fixes c und die entsprechende Funktion $f(z) = z^2 + c$ definiert.

- Dadurch ist die 2. Iteration der Funktion $f\big(f(z)\big) = (z^2 + c)^2 + c$, wo c vorgegeben ist, z beliebig.

Das Mandelbrot Set beschäftigt sich im Gegensatz dazu mit der Frage: für welches c und einen bestimmten Anfangspunkt z, reisst das dynamische System aus? Fixieren wir also $z = 0$.

- Hier ist die 2. Iteration der Funktion $f\big(f(0)\big) = c^2 + c$, für beliebiges c.

3. Chaotische Dynamische Systeme mit SAS

Das Mandelbrot Set

Wie sieht das mit SAS aus?
Wir brauchen einen anderen Datenteppich.

```
68 data mandelbrot_teppich;
69     do x=-&R-0.5 to &R-0.5 by 0.0025;
70         do y=-&R to &R by 0.0025;
71             mandel_wert=julia_sets(0, 0, x, y, &iterations, &R);
72             output mandelbrot_teppich;
73         end;
74     end;
75 run;
```

konstant

Zum Vergleich, als Erinnerung, hier der Julia Teppich.

```
44 data julia_teppich;
45     do x=-&R to &R by 0.0025;
46         do y=-&R to &R by 0.0025;
47             *julia_wert=julia_sets(x, y, -0.8, 0.156, &iterations, &R);
48             julia_wert=julia_sets(x, y, 0.360284, 0.100376, &iterations, &R);
49             * julia_wert=julia_sets(x, y, -1, 0.0, &iterations, &R);
50             * julia_wert=julia_sets(x, y, +0.488, -0.20, &iterations, &R);
51             output julia_teppich;
52         end;
53     end;
54 run;
```

konstant

3. Chaotische Dynamische Systeme mit SAS

Das Mandelbrot Set

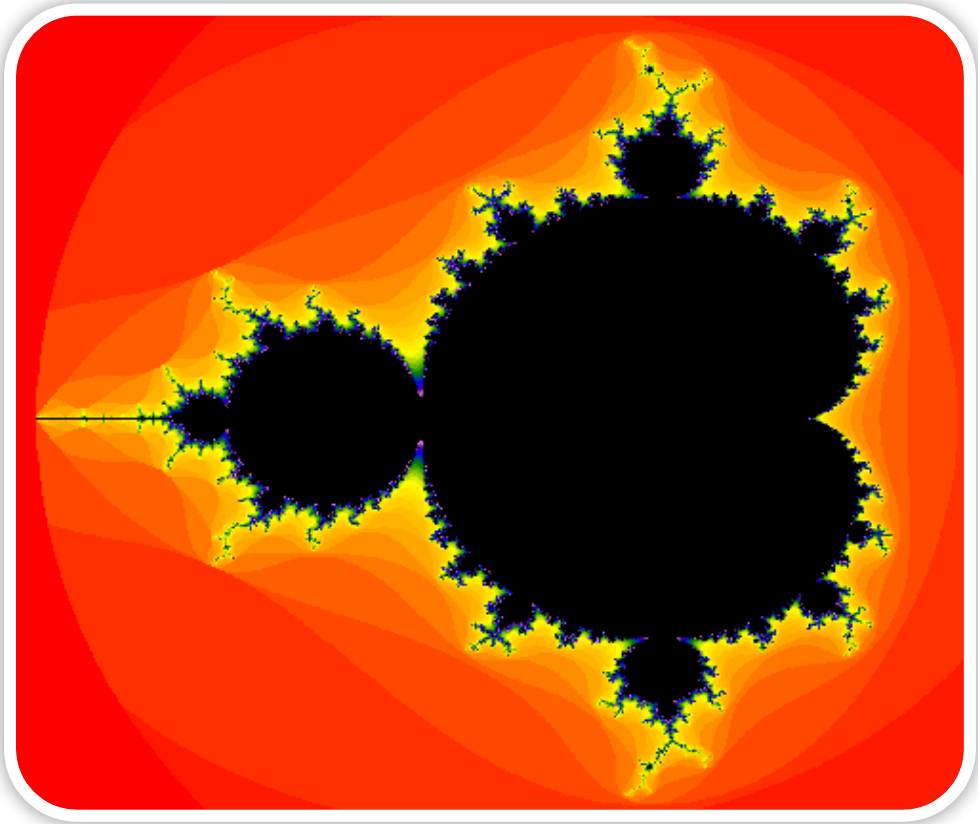
Und hier das fertige Ergebnis mit dem „Devaney“ Farbschema:

Fazit:

- Die Funktionalität von FCMP erlaubt es eigene iterative Funktionen problemfrei anzulegen
- ***proc sgplot*** und ***scatter*** sind sehr stark um detaillierte Heatmaps zu erstellen

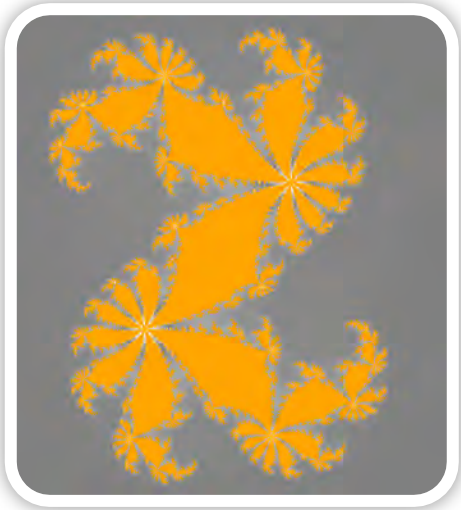
Dies wurde hier am Beispiel der Chaotischen Dynamischen Systeme dargestellt, es lässt sich aber auf viele andere Anwendungen erweitern.

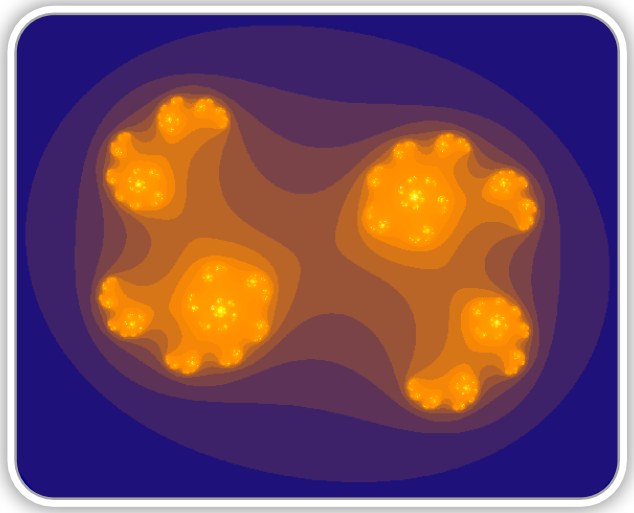
SAS Code wird zusammen mit der Präsentation geliefert. Viel Spaß beim eigenen Experimentieren!

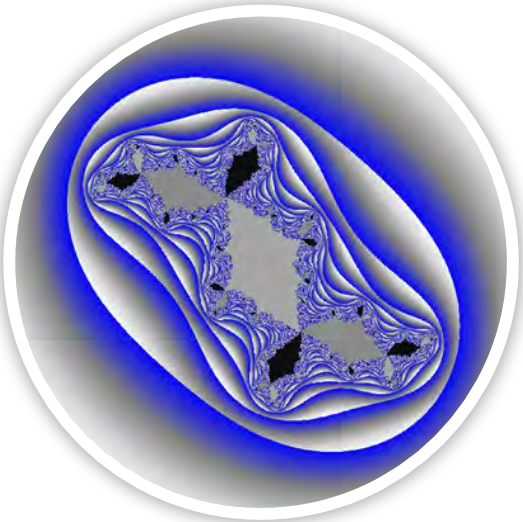
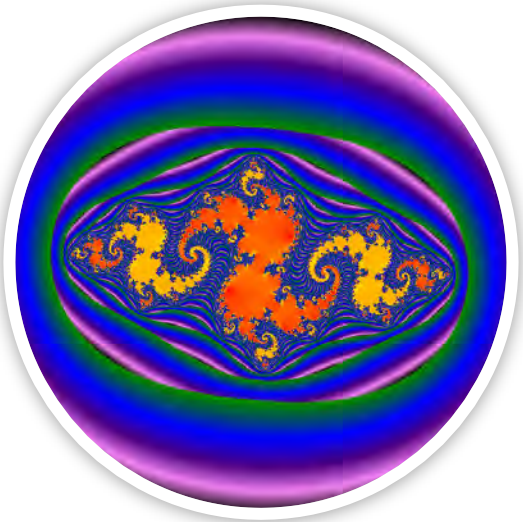


Tsunuun
Consulting GmbH

Hier noch einige kleine Spielereien…




Fürs Zuhören, danke!

Tsunuun
Consulting GmbH

www.tsunuun.com