

Iterationen

vom Repetitiven ins Chaos

gym | LERBERMATT
fms



Inhaltsverzeichnis

1. Erste Beispiele	5
1.1. Sierpinski Dreieck	5
1.2. Ein weiteres Beispiel	7
1.3. Notizen zu den Übungen	9
2. Iteration reellwertiger Funktionen	11
2.1. Grundbegriffe	11
2.2. Fixpunkte	13
2.3. Notizen zu den Übungen	19
3. Ausblick	22
A. Rettet die Wale	23
A.1. Situation	23
A.2. Das Modell	23
A.3. Die Analyse	24
A.4. Notizen zu den Walen	25
B. Starker Anstieg bei Masern	29
B.1. Gekoppelte Populationen	29
B.2. Masernepidemien	29
B.3. Ein Modell	29
B.4. Allgemeine Modellierung	30
B.5. Notizen zu Masern	32
C. Populationen in Wechselwirkung	34
C.1. Populationen in Symbiose	34
C.2. Notizen zu Symbiose	35
C.3. Räuber-Beute-Modelle	36
C.4. Notizen zu Lotka-Voltera	38
D. SIR-Modell	40
E. Integrierter Pflanzenschutz am Weihnachtsbaum	64
E.1. Die Situation	64
E.2. Das Modell	64
E.3. Die Analyse	65
E.4. Notizen zum Weihnachtsbaum	66
F. Von den Walen zur logistischen Gleichung	68
G. Von der logistischen Gleichung zur Mandelbrotmenge	85

Inhaltsverzeichnis

H. Der Kreis schliesst sich	105
------------------------------------	------------

1. Erste Beispiele

1.1. Sierpinski Dreieck

Wir spielen folgendes Spiel (Barnsley's Chaos Game):

Gegeben sei ein Dreieck mit den Eckpunkten 1, 2, 3. Wir wählen einen beliebigen Startpunkt x_0 , sowie einen beliebigen Punkt P , und berechnen mit der Vorschrift

$$x_1 = \frac{x_0 + P}{2}$$

den Punkt x_1 . Fahren Sie nach diesem Schema fort, um weitere Punkte zu bestimmen.

Am einfachsten geht's, wenn man sich rasch ein Programm schreibt, das obiges Verfahren ausführt.

```
import random
import matplotlib.pyplot as plt

def sierpinski(n):
    vertices = [(0.0,0.0), (0.5,0.85), (1.0,0.0)]
    points = []
    x,y = random.choice(vertices) #Startpunkt

    for i in range(n):
        vx,vy = random.choice(vertices) #waehle zufaellige Ecke

        x = (vx+x)/2.0 #Mittelung
        y = (vy+y)/2.0

        points.append((x,y))

    plot(points)
```

Noch ein Hinweis zur verblüffenden Tatsache, dass das Random Game gegen den Attraktor Sierpinski-Dreieck konvergiert. Klar und mit einfachen Argumenten wird dies im YouTube Video von Robin Truax erklärt.

Es gibt eine weitere, erstaunliche Beziehung zum Pascal'schen Dreieck, die ich von Agnes Scott habe. Und zwar kriegt man das Sierpinski-Dreieck-Muster, wenn man im Pascal'schen Dreieck alle geraden Zahlen mit einer Farbe markiert.

Dies sieht man wie folgt ein: Bezeichne P_n die ersten 2^n Zeilen des Pascal-Dreiecks (PD). Wir zeigen, dass P_{n+1} aus 3 Kopien von P_n besteht, die ein Dreieck mit geraden Koeffizienten umgeben. Die ersten 2^n Zeilen von P_{n+1} sind natürlich identisch mit den ersten 2^n Zeilen von P_n . Wir wollen beweisen, dass für jeden Koeffizienten in P_n die Koeffizienten an den entsprechenden Positionen in den unteren linken und unteren rechten Dreiecken von P_{n+1} denselben Wert ($\text{mod } 2$) haben wie der Koeffizient in P_n .

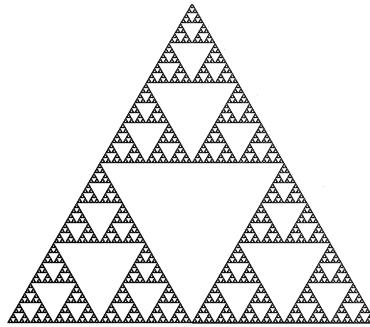


Abbildung 1: Sierpinski Dreieck, Struktur im Zufall

Die Beweisgrundlage bildet das Theorem von Lucas. Das folgende klassische Theorem wurde 1890 von Édouard Lucas bewiesen:

Satz 1.1: Lucas' Theorem

sei p prim und r, c in p -adischer Notation:

$$r = r_k \dots r_1 r_0 = r_0 + r_1 p + \dots + r_k p^k \quad (0 \leq r_i < p)$$

$$c = c_k \dots c_1 c_0 = c_0 + c_1 p + \dots + c_k p^k \quad (0 \leq c_i < p)$$

Dann gilt:

$$\binom{r}{c} = \binom{r_0}{c_0} \binom{r_1}{c_1} \dots \binom{r_k}{c_k} \mod (p)$$

Ferner gilt wie üblich $\binom{r}{c} = 0$, falls $c > r$.

Um das Pascalsche Dreieck modulo 2 zu untersuchen, setzen wir $p = 2$. Der Binomialkoeffizient $\binom{r}{c}$ in Zeile r und Spalte c von P_n kann in binärer Darstellung geschrieben werden, da $r, c < 2^n$:

$$r = r_{n-1} \dots r_1 r_0, \quad c = c_{n-1} \dots c_1 c_0 \quad (r_i, c_i \in \{0, 1\}).$$

In P_n entspricht der Koeffizient an der unteren linken Ecke $\binom{2^n+r}{2^n+c}$ und an der unteren rechten Ecke $\binom{2^n+r}{c}$. Nach Lucas' Theorem gilt:

$$\binom{2^n+r}{c} \equiv \binom{r}{c} \mod (2), \quad \binom{2^n+r}{2^n+c} \equiv \binom{r}{c} \mod (2).$$

Alle drei Binomialkoeffizienten haben somit denselben Wert modulo 2 und damit dieselbe Farbe im Pascalschen Dreieck ($\mod 2$). Dies beweist den ersten Teil des rekursiven Verhaltens.

Wir zeigen nun, dass die drei äquivalenten Eckdreiecke von P_{n+1} ein Dreieck aus geraden Zahlen umgeben. Die letzte Zeile von P_n entspricht $r = 2^n - 1$. Die nächste Zeile, $r = 2^n$,

hat Spalten $0 \leq c \leq 2^n$. Die Werte in den äußersten Spalten (0 und 2^n) sind gleich 1 , während für $1 \leq c \leq 2^n - 1$ mindestens ein Binärbit in c gleich 1 ist. Daher ist:

$$\binom{2^n}{c} = \binom{10 \dots 0}{0c_{n-1} \dots c_0} = \binom{1}{0} \dots \binom{0}{c_0} \equiv 0 \pmod{2} \quad \text{für } 1 \leq c \leq 2^n - 1.$$

Die Zeile beginnt und endet mit einer ungeraden Zahl, alle anderen Werte sind gerade. In der nächsten Zeile gilt:

$$\binom{r+1}{c} = \binom{r}{c} + \binom{r}{c-1}.$$

Die Summe einer geraden und einer ungeraden Zahl ist ungerade, die Summe zweier gerader Zahlen ist gerade. Daher bleibt die Kette gerader Zahlen erhalten, jedoch reduziert sich ihre Länge um 1 . In der letzten Zeile von P_{n+1} gibt es keine geraden Zahlen mehr, da diese Zeile $2^{n+1} - 1$ nur aus ungeraden Zahlen besteht:

$$\binom{2^{n+1} - 1}{c} = \binom{11 \dots 11}{c_n \dots c_0} = \binom{1}{c_n} \dots \binom{1}{c_0} \equiv 1 \pmod{2}.$$

Dies zeigt, dass P_{n+1} das gewünschte Muster aus drei Kopien von P_n und einem Dreieck aus geraden Zahlen besitzt.

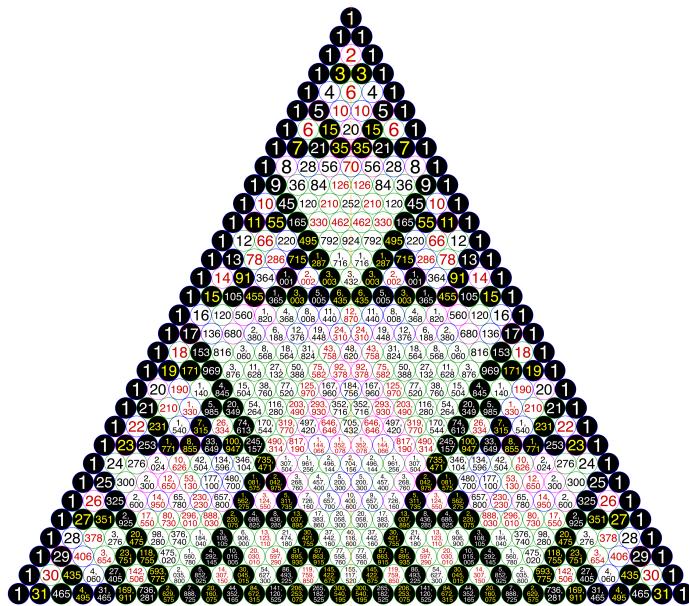


Abbildung 2: Pascal'sches Sierpinski Dreieck

1.2. Ein weiteres Beispiel

Wir kneten einen Blätterteig: Bei der Blätterteigherstellung wird folgendermassen vorgegangen: Ein Stück der Länge, sagen wir 1 , wird auf die doppelte Länge ausgewallt,

1. Erste Beispiele

2, dann zusammengefaltet auf die Länge 1, dann wiederum auf die doppelte Länge 2 ausgewallt, dann zusammengefaltet auf Länge 1, dann auf die doppelte Länge Und so weiter ...

Übung 1.1.



- a) Gibt es Fixpunkte?
- b) Was könnte mit Vorfixpunkte gemeint sein?
- c) Findest du Vorfixpunkte?
- d) Gibt es Zyklen der Länge 2?
- e) Gibt es Zyklen beliebiger Länge?
- f) Wie sieht der Orbit von $\frac{\sqrt{2}}{2}$ aus?

1.3. Notizen zu den Übungen

Notizen zu Übung 1.1. Für die Funktion brauchen wir eine Fallunterscheidung:

$$f(x) = \begin{cases} 2x & 0 \leq x \leq 0.5 \\ 2 - 2x & 0.5 \leq x \leq 1 \end{cases}$$

a) Die Fixpunktbedingungen liefern

$$\begin{aligned} 2x &\stackrel{!}{=} x \\ x &= 0 \end{aligned}$$

bzw.

$$\begin{aligned} 2 - 2x &\stackrel{!}{=} x \\ 2 &= 3x \\ x &= \frac{2}{3} \end{aligned}$$

Also sind $x_1 = 0$ und $x_2 = \frac{2}{3}$ Fixpunkte.

- b) Ich vermute, dass ein Vorfixpunkt ein Punkt ist, der nach unendlich vielen Iterations schritten gegen einen Fixpunkt konvergiert.
- c) Zum Beispiel 1, denn $f(1) = 2 - 2 = 0$ ist nach einer Iteration beim Fixpunkt 0. Mit Reverse Engineering findet man unendlich viele weitere: $\frac{1}{2^k}, k \in \mathbb{N}$, da $f^{-1}(1) = \frac{1}{2}$. Auch $\frac{1}{3}$ ist Vorfixpunkt und somit $\frac{1}{3 \cdot 2^k}, k \in \mathbb{N}$. Für $n \in \mathbb{N}, n > 3$ ist $f^{-1}\left(\frac{n-1}{n}\right) = 2 - 2 \cdot \frac{n-1}{n} = \frac{2n-2(n-1)}{n} = \frac{2}{n} \leq \frac{1}{2}$ und mündet vermutlich in Vorfixpunkten. Interessant ist die Untersuchung für Werte mit primen Nenner, $\frac{1}{p}$ oder Stammbrüche $\frac{1}{n}$. Rationale Zahlen scheinen hierbei erfolgsversprechend.
- d) Gibt es Fixpunkte der Periode 2? Dazu müssen wir 4 Fälle anschauen. Der Startwert liegt unter 0.5 bzw. oberhalb von 0.5; er bleibt nach einer Iteration in diesem Bereich oder er wechselt den Fall. Nehmen wir zuerst an, dass Startwert x_0 sowie Funktionswert $f(x_0)$ beide kleiner als 0.5 sind:

$$\begin{aligned} f(f(x)) &\stackrel{!}{=} x \\ 4x &= x \end{aligned}$$

und es folgt $x = 0$, den wir bereits kennen. Oberhalb 0.5:

$$\begin{aligned} f(f(x)) &\stackrel{!}{=} x \\ 2 - 2(2 - 2x) &= x \\ 4x - 2 &= x \\ x &= \frac{2}{3} \end{aligned}$$

1. Erste Beispiele

Den kennen wir auch. Unten oben:

$$\begin{aligned} f(f(x)) &\stackrel{!}{=} x \\ 2 - 2(2x) &= x \\ 2 &= 5x \end{aligned}$$

Wir haben einen echten gefunden, nämlich $x_3 = \frac{2}{5}$. Schliesslich noch die letzte Variante, obwohl wir auch die Bahn von x_3 verfolgen könnten:

$$\begin{aligned} f(f(x)) &\stackrel{!}{=} x \\ 2(2 - 2x) &= x \\ 4 &= 5x \end{aligned}$$

Es ist also $x_4 = \frac{4}{5}$.

- e) Dazu rechnet man am besten ein paar Beispiele durch und sucht nach Struktur. Ich habe beispielsweise Dreierzyklen ($x = \frac{4}{9}$ oder $x = \frac{6}{7}$), Viererzyklen ($x = \frac{2}{15}$, $x = \frac{6}{17}$ oder $x = \frac{2}{17}$) und weitere gefunden. Exemplarisch sei die Suche nach Viererzyklen kommentiert: Von den $2^4 = 16$ Fallunterscheidungen müssen nur die wenigsten im Detail gerechnet werden, da man Zweierzyklen wieder erkennt und Fälle durch zyklische Fortsetzung identisch sind. Man findet im wesentlichen die neuen Kandidaten (oben, unten): **uuuo**, **uuoo**, **ooou**. Kombinatorisch folgt also, dass es sicher immer mindestens 2 neue k -Zyklen gibt, nämlich **oooo...o** und **ouuu...u** mit den Werten $\frac{2}{2^k+1}$ bzw. $\frac{2}{2^k-1}$.

- f) Den Orbit von $\frac{\sqrt{2}}{2}$ lässt man am bequemsten berechnen. Zum Beispiel

```
x0 = 1/20
n = 10
def f(x=0):
    if x<0.5:
        return 2*x
    else:
        return 2-2*x
print(x0)
for k in range(n):
    x0 = f(x0)
    k = k+1
    print(x0)
```

$\frac{\sqrt{2}}{2} \notin \mathbb{Q}$ wird nie zyklisch.

2. Iteration reellwertiger Funktionen

2.1. Grundbegriffe

Wir betrachten eine reellwertige Funktion

$$f : \mathbb{R} \longrightarrow \mathbb{R},$$

beispielsweise eine Proportionalität

$$f(x) = \lambda x,$$

und wählen einen Startwert x_0 . Nun erzeugen wir rekursiv eine Folge mit

$$\begin{aligned} f^1(x_0) &= f(x_0) &= x_1 \\ f^2(x_0) &= f(x_1) &= x_2 \\ f^3(x_0) &= f(x_2) &= x_3 \\ &\vdots &\vdots \\ f^k(x_0) &= f(x_{k-1}) &= x_k \end{aligned}$$

Diesen Vorgang nennen wir **Iteration** von f mit Startwert x_0 . Iterieren einer Funktion heisst also:

1. Zu einem gegebenen Startwert x_0 den Funktionswert $x_1 = f(x_0)$ berechnen.
2. den Wert x_1 als neuen Startwert nehmen und bei 1. fortfahren.



Definition 2.1: Orbit

ie Menge

$$\{x_0, x_1, x_2, \dots\}$$

wird als Orbit oder Bahn von x_0 bezeichnet.

Beispiele. Wir betrachten einige Beispiele.

- $f(x) = \frac{x}{2}, \quad x_0 = 3 \dots$

Die Folge ist konvergent gegen 0.

- $f(x) = 3x, \quad x_0 = 1 \dots$

Die Folge divergiert.

•

$$f(x) = x, \quad x_0 \dots$$

Hier haben wir lauter Fixpunkte.

•

$$f(x) = x + c, \quad x_0 \dots$$

Wir haben $f^n(x) = x_0 + n \cdot c$, was im Allgemeinen divergiert.

•

$$f(x) = x^2$$

Dieses Beispiel ist interessant, da das Verhalten wesentlich vom gewählten Startwert x_0 abhängt.

- Ist $-1 < x_0 < 1$, dann erhalten wir bei Iteration eine monoton fallende Zahlenfolge mit

$$\lim_{n \rightarrow \infty} x_n = 0.$$

Ferner fällt auf, dass ein negativer Startwert nach der ersten Iteration „ins Positive hüpf“.

- Gilt $|x_0| > 1$, so divergiert die entsprechende Folge.

Übung 2.1.



Betrachte für

$$f(x) = x^2$$

die beiden Fälle $x_0 = 0$ und $x_0 = 1$. In welchen Zusammenhang könnten die Begriffe Repeller und Attraktor hier gebracht werden?

Übung 2.2.



Finde Fixpunkte der Iteration

$$f(x) = x^2 - 2$$

und entscheide anschliessend, ob es sich um anziehende oder abstossende Fixpunkte handelt.

Beispiel 1. Möchte man die Fixpunkte der Iteration

$$f(x) = x^2$$

finden, so löst man einfach die entsprechende Bedingung,

$$f(x) = x,$$

nach x . In der Tat hier also $x^2 - x = 0$, woraus unmittelbar die Lösungen $x_1 = 0$ und $x_2 = 1$ abgelesen werden können.

Übung 2.3.



Wie könnte man nun entscheiden, ob ein Fixpunkt *anziehend* oder *abstossend* ist?

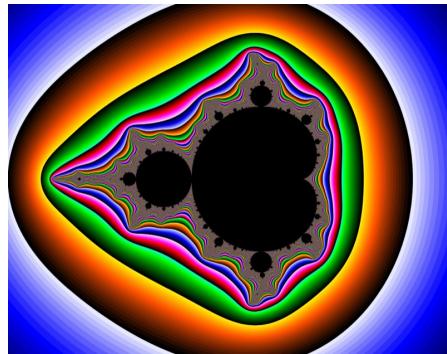


Abbildung 3: The Famous Mandelbrot Set

2.2. Fixpunkte

Evergreens bei der Analyse von Abbildungen sind die Suche nach Fixpunkten oder Fixgeraden.

Definition 2.2: Fixpunkt

in Punkt x^* heisst Fixpunkt (der Periode 1) von f , wenn $f(x^*) = x^*$ gilt.

Bemerkung. Ist x^* ein Fixpunkt der Periode 1 von f , dann gilt natürlich auch $f^n(x^*) = x^*$ für jedes $n \in \mathbb{N}$. Ferner: Seien $n \in \mathbb{N}$ und x^* ein Fixpunkt der Periode $k \in \mathbb{N}$ mit $n \bmod k \equiv 0$, dann ist x^* auch ein Fixpunkt der Periode n .

Beispiele.

- $f(x) = \sqrt{x}$ hat die Fixpunkte 0 und 1.

- $g(x) = 2x + 1$ hat den Fixpunkt $x = -1$.

- $h(x) = x - 2$ hat keinen Fixpunkt.

Definition 2.3: Fixpunkt der Periode k

in Punkt x^* heisst Fixpunkt der Periode k von f , wenn

$$f^k(x^*) = x^*$$

gilt.

Beispiel 2. Wir betrachten

$$f(x) = -x + 1.$$

Dann können wir leicht nachrechnen, dass $x = 0.25$ ein Fixpunkt der Periode 2 von f ist.

Übung 2.4.

Gibt es noch weitere Fixpunkte?

Bleibt noch die Frage, wie man Fixpunkte höherer Ordnung findet? Dazu illustrativ ein

Beispiel 3. Wir betrachten

$$f(x) = 2x - 1.$$

Einen Fixpunkt der Periode 1 finden wir rasch via die Bedingung $f(x) = x$, also

$$2x - 1 = x$$

was unmittelbar $x = 1$ liefert.

Um nun Fixpunkte der Periode 2 zu finden berechnen wir

$$f^2(x) = 2(2x - 1) - 1 = 4x - 3$$

und erhalten

$$4x - 3 = x.$$

Wir sind enttäuscht, weil $x = 1$ bereits wieder Lösung ist. Scheinbar gibt es hier keine reinen Fixpunkte der Periode 2. „Keine reinen“ deshalb, weil ja ein Fixpunkt erster Periode ebenfalls Fixpunkt n -ter Periode ist.

Wir vermuten anhand der obigen Rechnung, dass eine affine Funktion keine echten Fixpunkte höherer Periode hat.

Übung 2.5.

Beweise obige Vermutung.

Die Vermutung war also nicht ganz korrekt: Fixpunkte der Periode 2 liegen auf den Senkrechten zur Winkelhalbierenden.

Nach diesem Exkurs wollen wir ein paar Begriffe festlegen.

Definition 2.4: Attraktor

in Fixpunkt x heisst **anziehend**, wenn für alle Punkte \tilde{x} in einer Umgebung von x gilt, dass

$$\lim_{n \rightarrow \infty} f^n(y) = x.$$

Man nennt x auch etwa **Attraktor**.

Beispiel 4. Beispielsweise ist für

$$f(x) = \sqrt{x}$$

der Punkt $x = 1$ ein Attraktor.

Übung 2.6.

Starte mit den Werten 3 und 0.2 und betrachte deren Verhalten unter

$$\lim_{n \rightarrow \infty} \sqrt[n]{x}.$$

Benutze dabei vielleicht Abbildung 4 auf Seite 15.

Wie könnte man allgemein zeigen, dass 1 ein Attraktor für \sqrt{x} ist?



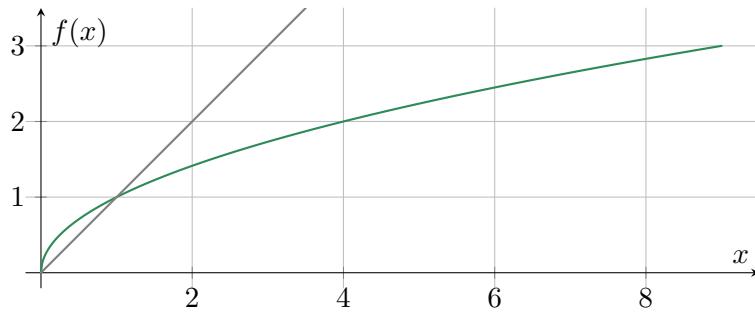


Abbildung 4: Graph der Wurzelfunktion

Definition 2.5: Repeller

in Fixpunkt x heisst **abstossend**, wenn für alle Punkte $\tilde{x} \neq x$ in einer Umgebung von x gilt, dass

$$\lim_{n \rightarrow \infty} f^n(\tilde{x}) \neq x.$$

Man nennt x auch **Repeller**.

Beispiel 5. Für $f(x) = x^2$ ist der Fixpunkt $x = 1$ abstossend.

Für einige ausgewählte Übungen sind weiter unten ab Seite 17 vorgezeichnete Graphen auf dem Silbertablett serviert.

Übung 2.7.

Betrachte die Startwerte $x = 2$ bzw. $x = 0.5$ und zeige allgemein, dass 1 ein Repellor für x^2 ist.

Satz 2.1: Attraktorbedingung

Ist für einen Fixpunkt x , dass $|f'(x)| > 1$, dann ist x ein Repellor. Ist $|f'(x)| < 1$, so handelt es sich um einen Attraktor.

Beweis. Übung, falls man zum Beispiel Lipschitz-Stetigkeit und den Mittelwertsatz googlen möchte.

Sei f Lipschitz-stetig. Der Abstand zwischen x_{n+1} und x^* ist $|x_{n+1} - x^*| = |f(x_n) - f(x^*)| \leq L|x_n - x^*|$. Nach Iteration hat man $|f(f(x_n)) - x^*| \leq L|f(x_{n+1}) - x^*| \leq L^2|x_n - x^*|$. Also ist $|x_k - x^*| \leq L^k|x_0 - x^*|$. Ist $|L| < 1$ dann gilt $L^k \rightarrow 0$ für $k \rightarrow \infty$. Ist f nicht Lipschitz-stetig, so kann man den Mittelwertsatz anwenden, wenn f differenzierbar und stetig ist. Es ist $x_{k+1} - x^* = f(x_k) - f(x^*) = f'(\zeta) \cdot (x_k - x^*)$ mit $\zeta \in (x_k, x^*)$. Also betragsmäßig $|f(x_k) - f(x^*)| = |f'(\zeta)| \cdot |x_k - x^*|$, was für $|f'(\zeta)| < 1$ den Abstand pro Iterationsschritt verkürzt. \square

Übung 2.8.

Gib jeweils ein Beispiel einer Funktion mit

- (a) einem Attraktor bei $x = 3$.
- (b) einem Repellor bei $x = -2$.
- (c) zwei Fixpunkten, bei $x = 1$ bzw. $x = 4$.
- (d) einem Fixpunkt der Periode 2 in $x = 7$, für den $f(7) = 19$ ist.

Übung 2.9.

Welche Aussagen kannst du über die Fixpunkte folgender Funktionen machen? Bestimme gegebenenfalls das „Einzugsgebiet“, falls es sich um einen Attraktor handelt.

- (a) $\cos(x)$
- (b) $2 \sin(x)$
- (c) $0.5x^2 + 0.5$

Übung 2.10.

Untersuche die Funktion

$$f(x) = -x^2 + 1$$

auf Fixpunkte.

Übung 2.11.

Untersuche die Standardparabel

$$f(x) = x^2$$

auf Fixpunkte.

Beispiel 6. Wie sieht es nun für eine verschobene Parabel

$$f(x) = x^2 - 2$$

aus? Wir haben

$$x^2 - x - 2 = 0$$

mit den Lösungen $x_1 = -1$ und $x_2 = 2$.

Für Fixpunkte der Periode zwei muss

$$x^4 - 4x^2 - x + 2 = 0$$

gelten. Um nun die echten 2-periodischen Fixpunkte zu erhalten, dividieren wir die Fixpunkte der Periode 1 aus:

$$(x^4 - 4x^2 - x + 2) \div (x^2 - x - 2) = x^2 + x - 1$$

und setzen gleich 0. Die Lösung dieser schönen Gleichung kennen wir. Die Fixpunkte 2-ter Periode sind also

$$x = \frac{-1 \pm \sqrt{5}}{2}.$$

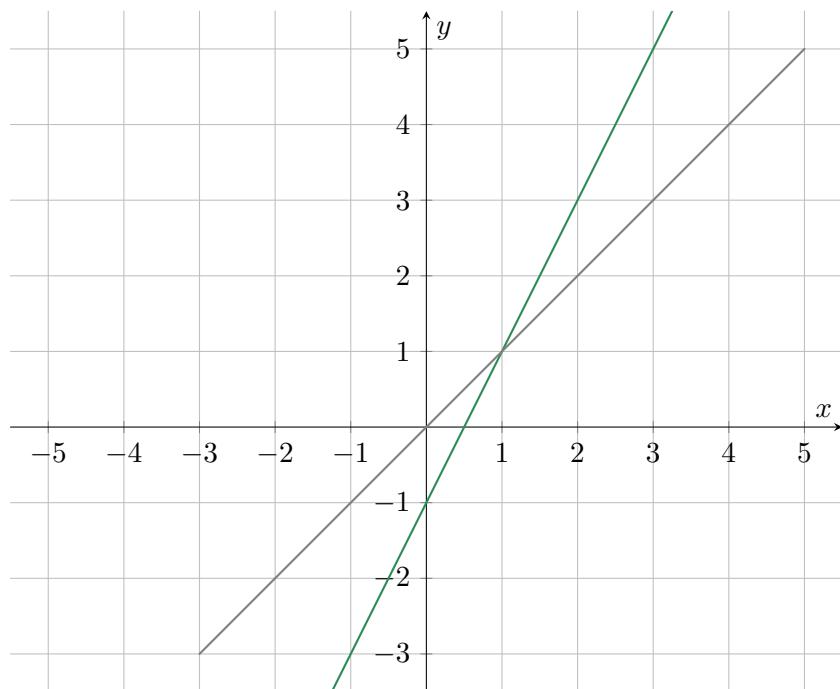


Abbildung 5: Graph der Geraden $2x - 1$

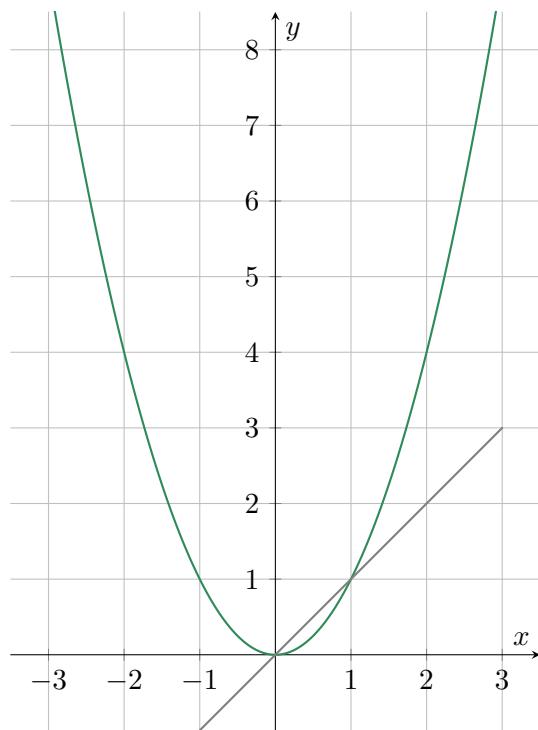


Abbildung 6: Graph der Quadratfunktion

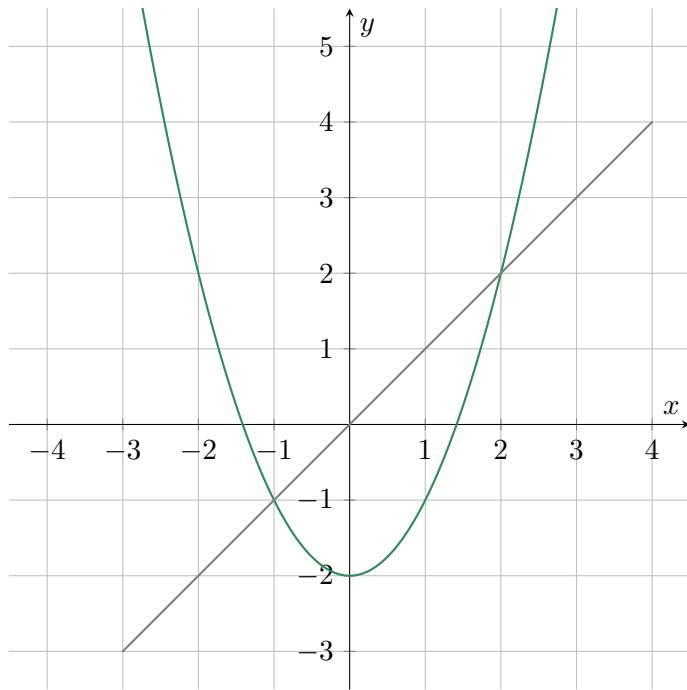


Abbildung 7: Graph von $x^2 - 2$

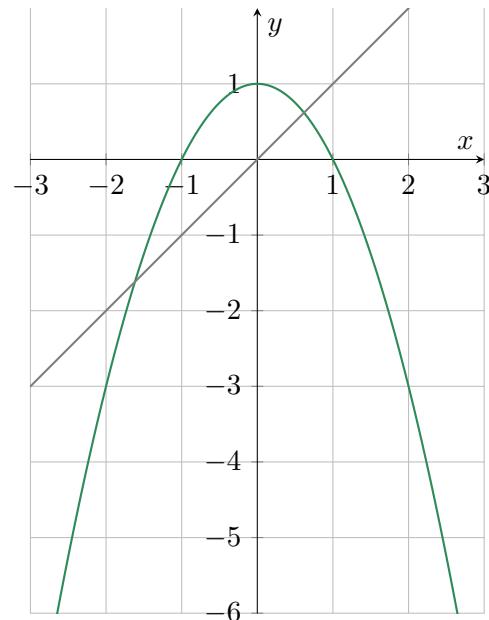


Abbildung 8: Graph von $-x^2 + 1$

2.3. Notizen zu den Übungen

Notizen zu Übung 2.1. Für $k \in \mathbb{N}$ ist $f^k(0) = 0$ und $f^k(1) = 1$, die beiden Werte sind also Fixpunkte von f .

Notizen zu Übung 2.2. Mit der Fixpunktbedingung ergibt sich

$$\begin{aligned} x^2 - 2 &= x \\ x^2 - x - 2 &= 0 \\ x_{1,2} &= \frac{1 \pm 3}{2} \end{aligned}$$

also sind die Fixpunkte 2 und -1 . Ferner ist ein Fixpunkt x_p anziehend, falls $|f'(x_p)| < 1$. Hier ist $f'(x) = 2x$ und damit ist keiner der beiden anziehend.

Notizen zu Übung 2.3. Die Attraktorbedingung für einen Fixpunkt x^* einer Funktion f ist $|f'(x^*)| < 1$.

Beweis. Sei f Lipschitz-stetig. Der Abstand zwischen x_{n+1} und x^* ist $|x_{n+1} - x^*| = |f(x_n) - f(x^*)| \leq L|x_n - x^*|$. Nach Iteration hat man $|f(f(x_n)) - x^*| \leq L|f(x_{n+1}) - x^*| \leq L^2|x_n - x^*|$. Also ist $|x_k - x^*| \leq L^k|x_0 - x^*|$. Ist $|L| < 1$ dann gilt $L^k \rightarrow 0$ für $k \rightarrow \infty$. Ist f nicht Lipschitz-stetig, so kann man den Mittelwertsatz anwenden, wenn f differenzierbar und stetig ist. Es ist $x_{k+1} - x^* = f(x_k) - f(x^*) = f'(\zeta) \cdot (x_k - x^*)$ mit $\zeta \in (x_k, x^*)$. Also betragsmäßig $|f(x_k) - f(x^*)| = |f'(\zeta)| \cdot |x_k - x^*|$, was für $|f'(\zeta)| < 1$ den Abstand pro Iterationsschritt verkürzt. \square

Notizen zu Übung 2.4. Der Fixpunkt der Periode 1 ist $\frac{1}{2}$. Für einen Fixpunkt der Periode 2 gilt: $-(-x+1)+1 \stackrel{!}{=} x$, also $x = x$. Das heisst, es gibt unendlich viele Fixpunkte der Periode 2 für f ; jedes $x \in \mathbb{R}$.

Notizen zu Übung 2.5. Wir rechnen alle Fixpunkte nach.

$$\begin{aligned} mx + q &\stackrel{!}{=} x \\ x(m - 1) &= -q \\ x &= \frac{q}{1 - m} \end{aligned}$$

falls $m \neq 1$; für $m = 1$ ist $q = 0$ und man hat mit $f(x) = x$ eine Fixpunktgerade. Für 2-Zyklen ist

$$\begin{aligned} m(mx + q) + q &\stackrel{!}{=} x \\ x(1 - m^2) &= q(m + 1) \\ x &= \frac{q(m + 1)}{1 - m^2} \\ x &= \frac{q}{1 - m} \end{aligned}$$

wobei $m \neq \pm 1$ galt. Dies ist der Fixpunkt der Periode 1 und wir betrachten daher den Fall $m = -1$ separat. Daraus folgt unter der Fixpunktbedingung sofort $x = x$, so dass jeder Punkt x für $f(x) = -x + q$ ein 2-Zyklus ist.

Für 3-Zyklen:

$$\begin{aligned} m(m(mx + q) + q) + q &= x \\ m(m^2x + mq + q) + q &= x \\ q(m^2 + m + 1) &= x(1 - m^3) \\ x &= q \frac{m^2 + m + 1}{1 - m^3} \end{aligned}$$

Berechne $(m^3 - 1) \div (m^2 + m + 1)$ mit Polynomdivision und sieh dann, dass $x = q \frac{m^2 + m + 1}{1 - m^3} = \frac{q}{1-m}$. Das heisst, es gibt keine weiteren Fixpunkte.

Notizen zu Übung 2.6. Man wählt eine kleine Umgebung von $1, \pm\varepsilon$, und beobachtet das Geschehen. Einmal iteriert ist $f(1+\varepsilon) = \sqrt{1-\varepsilon}$ und k Mal iteriert ergibt $\sqrt[2k]{1+\varepsilon} = (1+\varepsilon)^{\frac{1}{2k}}$, was für $k \rightarrow \infty$ gegen den Grenzwert $(1+\varepsilon)^0 = 1$ konvergiert.

Notizen zu Übung 2.7. Orbits: $\{2, 4, 16, 256, \dots\}$ und $\{0.5, 0.25, 0.125, \dots\}$. Nun betrachten wir einen Punkt in der Nähe von 1 wobei $\varepsilon > 0$, $1 + \varepsilon$. $f(1 + \varepsilon) = (1 + \varepsilon)^2 = 1 + 2\varepsilon + \varepsilon^2$. Dieser Wert wird pro Iteration zunehmen, also ist 1 abstossend nach oben. Allerdings ist $f(1 - \varepsilon) = (1 - \varepsilon)^2 = 1 - 2\varepsilon + \varepsilon^2$, das gegen 0 tendiert.

Notizen zu Übung 2.8.

- a) Ich nehme eine Funktion mit Ableitung an der Stelle 3 betragsmäßig kleiner 1. Zum Beispiel $f(x) = \frac{2}{3}x + 1$, denn $f'(x) = \frac{2}{3}$. $f(3) = 3$ ist Fixpunkt und $|f'(3)| = \frac{2}{3}$.
- b) Wiederum bietet sich eine lineare Funktion mit Steigung grösser 1 an: $g(x) = 2x + 2$ erfüllt die Anforderungen.
- c) Die beiden Werte sind Lösung der quadratischen Gleichung $0 = (x - 1)(x - 4) = x^2 - 5x + 4$ die wir als Fixpunktbedingung umformulieren: $x^2 - 4x + 4 = x$. Also erfüllt $h(x) = (x - 2)^2$ die Bedingung.
- d) Aufgrund der vorhergehenden Ausführungen wissen wir, dass für jede lineare Funktion mit Steigung -1 jedes $x \in \mathbb{R}$ ein 2-Zyklus ist. Für $k(x) = -x + 26$ ist auch $k(7) = 19$ erfüllt.

Notizen zu Übung 2.9.

- a) $\cos(x) = x$ ist analytisch nicht lösbar. Man kann beispielsweise gucken, wo die Graphen sich schneiden und Werte approximieren. Eine approximative Lösung mit einem Taylorpolynom wäre auch denkbar.
- b) Dasselbe gilt wie in a) gilt hier auch.

c)

$$\begin{aligned} 0.5x^2 + 0.5 &\stackrel{!}{=} x \\ x^2 - 2x + 1 &= 0 \\ (x - 1)^2 &= 0 \\ x &= 1 \end{aligned}$$

Die Bedingung liefert $|x| = 1$ und damit ist 1 sicher kein Attraktor.

Notizen zu Übung 2.10. Aus der Fixpunktbedingung $-x^2 + 1 \stackrel{!}{=} x$ folgt $0 = x^2 + x - 1$ deren Lösung die goldenen Schnitte φ und $-\Phi$ sind. Wegen $f'(\varphi) = 2\varphi + 1 > 1$ ist φ instabil und aus $f'(-\Phi) = -2\Phi + 1 < -1$ erkennen wir, dass $-\Phi$ auch Repeller ist. Die Fixpunkte der Periode 2 geben etwas mehr zu tun.

$$f(f(x)) = -(-x^2 + 1)^2 + 1 = -x^4 + 2x^2 \stackrel{!}{=} x.$$

Darin stecken natürlich auch die Fixpunkte der Periode 1 (gegeben durch $x^2 + x - 1 = 0$), die wir mit Hilfe einer Polynomdivision eliminieren, um die Polynomgleichung zu erhalten, welche die reinen Fixpunkte der Periode 2 liefert.

$$(-x^4 + 2x^2 - x) \div (x^2 + x - 1) = x^2 + x$$

und damit $x_3 = 0$ und $x_4 = 1$. Es ist $(f(f(0)))' = -4 \cdot 0^3 + 4 \cdot 0 = 0$ und $(f(f(1)))' = 0$, also sind beide Attraktoren.

Notizen zu Übung 2.11. Aus der Fixpunktbedingung folgt unmittelbar $x^2 - x \stackrel{!}{=} 0$ und damit $x_1 = 0, x_2 = 1$. 0 ist Attraktor, 1 Repeller.

Für die Periode 2 Fixpunkte betrachten wir $x^4 - x \stackrel{!}{=} 0$, also $x^3 = 1$ und haben immer noch die beiden gleichen Fixpunkte.

3. Ausblick



Nach diesem theoretischen Teil geht es nun an einige konkrete Beispiele. Zum Einstieg in die diskrete Modellierung eignet sich meines Erachtens das Walmodell aus Anhang A. Weiter folgen etwas detaillierte Modelle (wie das während der CoV19-Pandemie bekannt gewordenen SIR-Modell, dass Sie bei mir unter Masern finden), wobei man immer den Merksatz von GEORGE BOX im Hinterkopf haben sollte:

All models are wrong, but some are usefull.

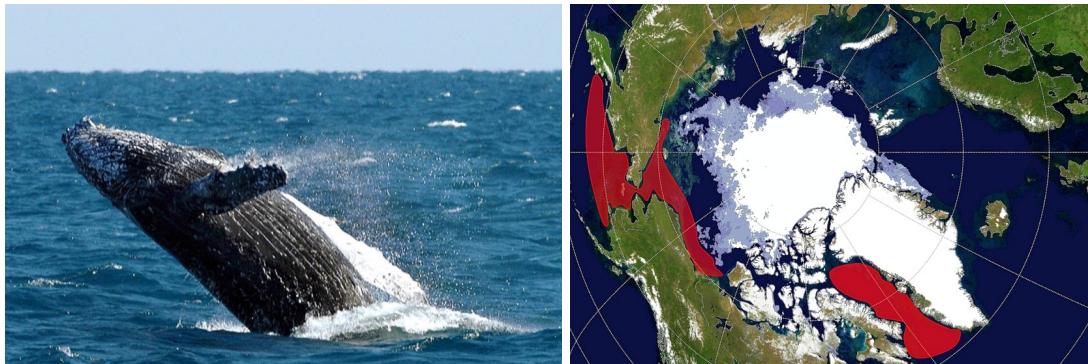


Anschliessend an dieses Skript zeige ich mit Hilfe von Vereinfachungen ausgehend vom Walbeispiel über die logistische Funktion bis zur Mandelbrotmenge einen Zusammenhang von *deterministischem Chaos*. Eine schöne Reise, die man sich nicht entgehen lassen sollte. Dazu existiert aber meinerseits noch keine Skriptvariante, jedoch habe ich mit interaktiven Jupyter Notebooks diese schönen Einsichten illustriert. Wenn man nicht bei mir das Schwerpunkt fach PAM oder Ergänzungsfach AM besucht, trotzdem aber reingucken möchte, dann empfehle ich als Einstieg das Video aus dem QR-Code als Startpunkt. Ich versuche auch sukzessive Teile als Anhänge zu diesem Skript zu generieren. Der zweite, blaue QR-Code startet ein für fortgeschrittene AM-ler empfehlenswertes Repetitionsmodul — das Burger-King-Problem — mit vielen Recaps und Anwendungen des Stoffs.

A. Rettet die Wale

A.1. Situation

Eine Population von ca. 1200 Grönlandwalen (bowhead whale, *Balaena mysticetus*) lebt in einem abgegrenzten Lebensraum des nördlichen Eismeeres (Nordpolarmeer). Da die Wale keine natürlichen Feinde haben und im bezeichneten Lebensraum ein reichhaltiges Nahrungsangebot vorfinden, werden sie sich zunächst exponentiell vermehren. Durch die Zunahme der Anzahl der Wale sinkt aber das Nahrungsangebot, da das als Nahrung dienende Plankton nicht in unbeschränktem Masse zur Verfügung steht. Die Wale können sich nicht mehr mit dem anfänglichen Zuwachsfaktor vermehren. Das Eismeer bietet nur eine begrenzte Zahl G von Grönlandwalen Lebensraum. Diese Zahl liegt nach vorsichtigen Schätzungen bei 20000 Tieren.



A.2. Das Modell

Bei Annäherung an die Sättigungsgrenze G wird der Zuwachs der Wale abnehmen. Das Wachstum der Grönlandwalpopulation kann demnach im Anfangszustand — wenn der Bestand noch weit von der Sättigungsgrenze entfernt ist — als exponentielles Wachstum und im fortgeschrittenen Stadium — wenn sich der Bestand der Sättigungsgrenze nähert — als begrenztes Wachstum beschrieben werden. Die Vermehrung kann somit — unter den genannten Rahmenbedingungen — als logistisches Wachstum modelliert werden

$$W_{n+1} = W_n + r \cdot (G - W_n) \cdot W_n,$$

wobei W_n die Anzahl der Wale im Jahr n , $G = 20000$ die Sättigungsgrenze und $W_0 = 1200$ die Anfangspopulation bezeichnet.

Übung A.1.

Stelle exponentielles und beschränktes Wachstum als Folge/Iteration dar. Plausibilisiere anschliessend das logistische Wachstum als Kombination der beiden.



A.3. Die Analyse

Naturschützer beobachten die Tiere und ermitteln, dass die Anzahl der Tiere im ersten Beobachtungsjahr um $p = 8\%$ wächst.

Übung A.2.

Stelle auf der Grundlage des iterativen logistischen Wachstumsmodells die Entwicklung der Grönlandwalpopulation für einen Zeitraum von 100 Jahren dar. p ist aus dem Wachstumsfaktor des ersten Beobachtungsjahres zu berechnen. Stelle den Populationsumfang in Abhängigkeit der Jahre graphisch dar.

Übung A.3.

Welchen Einfluss haben der Anfangsbestand und der Wachstumsfaktor im ersten Beobachtungsjahr auf die Entwicklung der Walpopulation? Wann ist der Zuwachs am grössten?

Übung A.4.

Das bisherige Modell berücksichtigt noch nicht den Walfang, der jedoch durch ein Walfangabkommen reglementiert werden soll.

Vorschlag 1 für ein Walfangabkommen gestattet, einen festen Prozentsatz des gegenwärtigen Bestandes der Walpopulation pro Jahr abzufischen.

Erstelle ein iteratives Wachstumsmodell, das eine Abfangquote von $q = 1\%$ einbezieht. Erstelle eine Tabelle und einen Graphen für die Entwicklung der Walpopulation für 100 Jahre unter Einfluss der Abfangquote $\alpha = 1\%$.

Experimentiere mit der Abfangquote α . Für welche Werte von α wächst die Walpopulation weiterhin, bei welchem Wert bleibt sie konstant und wann nimmt die Walpopulation ab und stirbt allmählich aus?

Übung A.5.

Ein alternativer Vorschlag sieht ein Walfangabkommen vor, dass pro Jahr eine feste (absolute) Fangzahl erlaubt.

Erstelle wiederum auf der Grundlage der Eingangsdaten ein iteratives Wachstumsmodell, das die konstante Fangzahl von $F = 70$ einbezieht und setze dies ebenfalls in Tabelle und Graphik um.

Experimentiere mit der Fangzahl F . Für welche Werte von F wächst die Population weiterhin, wann bleibt die Population konstant und wann stirbt der Bestand aus?

Andere und weitere Infos liefert zum Beispiel die Website *International Whaling Commission*.

A.4. Notizen zu den Walen

Notizen zu Übung A.1. Beispielsweise so, in Varianten für das exponentielle Wachstum und das logistische.

```
import matplotlib.pyplot as plt

# Rekursive Funktion zur Berechnung des exponentiellen Wachstums
def exponential_growth(n, rate, current_value=1):
    if n == 0:
        return current_value
    else:
        return exponential_growth(n-1, rate, current_value * rate)

# Parameter
n = 10 # Anzahl der Perioden
rate = 1.5 # Wachstumsrate

# Werte berechnen
values = [exponential_growth(i, rate) for i in range(n+1)]

# Werte plotten
plt.plot(range(n+1), values, marker='o')
plt.title('Exponentielles Wachstum')
plt.xlabel('Perioden')
plt.ylabel('Wert')
plt.grid(True)
plt.show()

# Funktion zur Berechnung des logistischen Wachstums mit einem Parameter p
def logistic_growth(n, G, x0, p):
    values = [x0]
    for i in range(n):
        x_next = values[-1] + p * (G - values[-1])
        values.append(x_next)
    return values

# Parameter
n = 10 # Anzahl der Perioden
G = 100 # Kapazitaetsgrenze
x0 = 10 # Anfangswert
p = 0.1 # Wachstumskoeffizient

# Werte berechnen
```

A. Rettet die Wale

```

values = logistic_growth(n, G, x0, p)

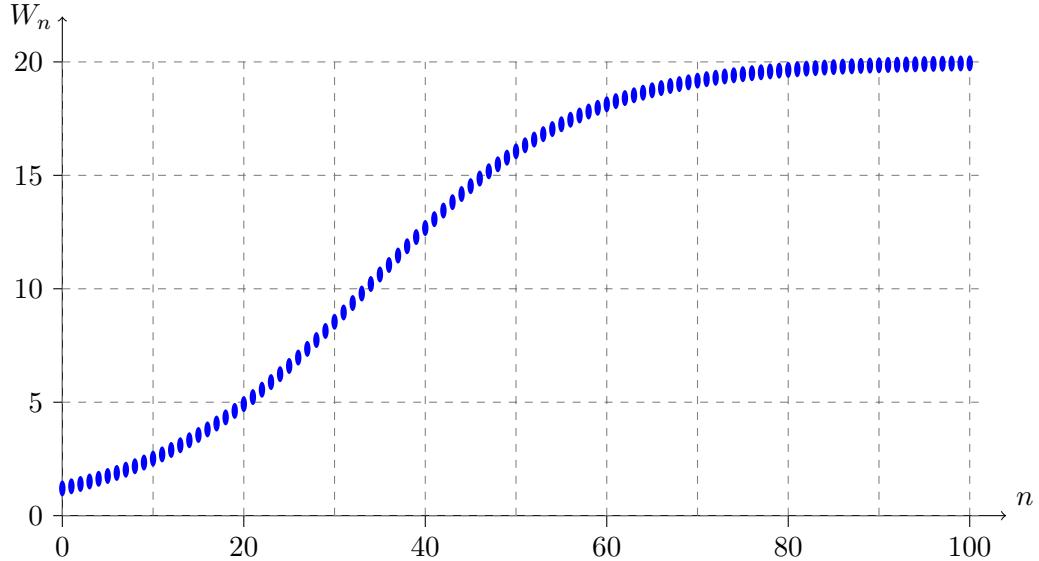
# Werte plotten
plt.plot(range(n+1), values, marker='o')
plt.title('Logistisches Wachstum mit Parameter p')
plt.xlabel('Perioden')
plt.ylabel('Wert')
plt.grid(True)
plt.show()

```

Notizen zu Übung A.2. Mit der Wachstumsbeobachtung berechnen wir

$$\begin{aligned}
 W_0 \cdot 1.08 &= W_0 + pW_0(G - W_0) \\
 0.08W_0 &= pW_0(G - W_0) \\
 p &= \frac{0.08}{G - W_0} \approx 4.2553 \cdot 10^{-6}
 \end{aligned}$$

Der Plot sieht damit singgemäß so aus:



Notizen zu Übung A.3. Bezeichne ω den Wachstumsfaktor im ersten Jahr. Es ist

$$p(\omega, W_0) = \frac{\omega}{G - W_0}$$

Also ist p proportional zu ω und wächst auch mit zunehmendem Anfangsbestand W_0 . Der Zuwachs der Iteration $W_{k+1} = W_k + pW_k(G - W_k)$ ist am größten, wenn $pW_k(G - W_k) = -pW_k^2 + pGW_k$ maximal ist. Da dies eine nach unten offene Parabel ist, suchen

wir den Scheitelpunkt und erhalten $W_k = -\frac{pG}{2p} = \frac{G}{2}$. Also wenn die Population 10 000 beträgt.

Notizen zu Übung ??. Eine Fangquote α liesse sich so modellieren:

$$W_{k+1} = (1 - \alpha)W_k + p(1 - \alpha)W_k(G - (1 - \alpha)W_k)$$

Betrachte, ohne den offensichtlichen Fixpunkt bei 0:

$$\begin{aligned} W_k &= (1 - \alpha)W_k + p(1 - \alpha)W_k(G - (1 - \alpha)W_k) \\ 1 &= (1 + \alpha)(1 + pG) - pa^2W_k \\ pa^2W_k &= (1 + \alpha)(1 + pG) - 1 \\ W_k &= \frac{(1 + \alpha)(1 + pG) - 1}{(1 - \alpha)^2p} \end{aligned}$$

Den Fixpunkte finde ich bei $W_k = \frac{(1-\alpha)(1+pG)-1}{(1-\alpha)^2p} \approx 17773$

Notizen zu Übung A.5.

$$W_{k+1} = (W_k - 70) + p(W_k - 70)(G - (W_k - 70))$$

Sei f die Fangquote, und wir rechnen den Fixpunkt:

$$\begin{aligned} W_k &= W_k - f + p(W_k - f)(G - (W_k - f)) \\ 0 &= -f + p(GW_k - W_k^2 + fW_k - fG + fW_k - f^2) \\ 0 &= -pW_k^2 + p(G + 2f)W_k - (f + p(fG + f^2)) \\ W_k &= \frac{-p(G + 2f) \pm \sqrt{p^2(G + 2f)^2 - 4p(f + p(fG + f^2))}}{-2p} \\ &= \frac{p(G + 2f) \mp \sqrt{p^2(G + 2f)^2 - 4p(f + p(fG + f^2))}}{2p} \end{aligned}$$

Eingesetzt für $f = 70$ liefert dies ca. 19 199.

Ärzte Zeitung, 01.06.2005

Starker Anstieg bei Masern

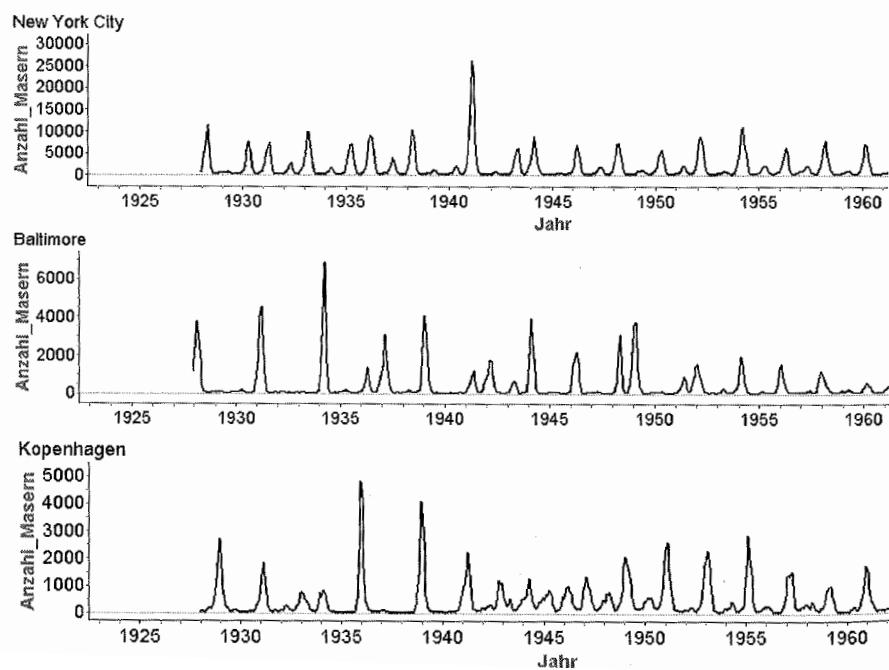
Bislang 383 Erkrankungen - vor allem in Hessen

BERLIN (eb). Die Zahl der Masernerkrankungen hat in Deutschland im Vergleich zum Vorjahr erheblich zugenommen. Das meldet das Robert-Koch-Institut in Berlin in seinem aktuellen Epidemiologischen Bulletin.

Bis zur 18. Kalenderwoche wurden demnach 383 Masernkranke gemeldet; 2004 waren es zur selben Zeit nur 51. Der Anstieg hängt mit den Masernausbrüchen in Hessen zusammen. Dort sind seit Jahresbeginn 250 Erkrankungen gemeldet worden; ein 14jähriges Mädchen war an den Folgen gestorben (wie berichtet).

Die Ausbrüche begannen bereits Ende vorigen Jahres in Offenbach und Frankfurt / Main; seit dem 24. März sind 13 Erkrankte in Wiesbaden gemeldet worden. Ein Drittel war älter als 14 Jahre.

Abb. 4.11. Pressebericht zur Masernepidemie



B. Starker Anstieg bei Masern

B.1. Gekoppelte Populationen

In vielen interessanten Situationen wirken mehrere Populationen aufeinander ein. Die Entwicklung von zwei oder mehreren Populationen ist gekoppelt. In manchen Situationen werden durch biologische oder physikalische Prozesse aus den Angehörigen der einen Populationen Mitglieder einer anderen Population: Junge werden alt, Alte sterben; Gesunde werden krank, Kranke werden gesund; aus Schülern werden Studenten, aus Studenten Berufstätige, bevor aus diesen Rentner werden. In anderen Situationen, die wir betrachten werden, koexistieren mehrere Populationen und ihr Zusammenwirken kann beiden Populationen nützen, beiden schaden oder nur einer Population zum Nutzen sein.



Wir modellieren ein Problem aus der Epidemiologie, bei dem aus Mitgliedern der Population der Gesunden in Folge Ansteckung Kranke werden, bevor diese wiederum gesunden und danach immun werden.

B.2. Masernepidemien

Die Krankheit Masern ist eine durch das Masernvirus hervorgerufene, hoch ansteckende Infektionskrankheit, die vor allem Kinder betrifft. Neben den typischen roten Hautflecken (Masern-Exanthem) ruft die Erkrankung Fieber und einen erheblich geschwächten Allgemeinzustand hervor.

Es können außerdem in manchen Fällen lebensbedrohliche Komplikationen wie Lungen- und Hirnentzündungen auftreten. Man weiß aus Erfahrung, dass es etwa alle zwei Jahre zu regelrechten Masernepidemien kommt. Bemerkenswert an der Epidemie ist, dass in den Zeiten zwischen dem massiven Auftreten die Krankheit fast völlig zum Abklingen zu kommen scheint. Ähnliche Verläufe findet man auch in anderen Ländern, wobei Masernepidemien in Entwicklungsländern in höherer Frequenz und Intensität auftreten. Kann Mathematik helfen, die hier beobachteten Phänomene zu erklären?

B.3. Ein Modell

Betrachten wir ein einzelnes Kind, das noch nicht mit Masern infiziert ist. Solch ein Kind wollen wir **empfänglich** nennen. Sofort nach der Infektion gibt es eine latente Periode von einer Woche (die Inkubationszeit), in der das Kind keine anderen Kinder anstecken kann, und in dem es auch noch keine Krankheitsanzeichen hat. Das Virus hat sich noch nicht hinreichend vermehrt. Nach dieser Woche ist das Kind **infektiös** und kann andere Kinder anstecken. Diese Zeit dauert ebenfalls eine Woche. Nach dieser Zeit erscheinen die typischen roten Pusteln und das Kind erholt sich wieder. Dann kann dieses Kind nicht mehr angesteckt werden und ist immun gegen das Masernvirus. Wir haben somit die drei Populationen:

B. Starker Anstieg bei Masern



Zur Modellierung des Krankheitsverlaufes in der Bevölkerung müssen einige zum Teil stark vereinfachende Annahmen getroffen werden.

Wir gehen von einer Stadt aus, in der folgende Annahmen gelten:

- Sowohl die Inkubationszeit als auch die infektiöse Zeit dauert genau eine Woche.
- Ansteckung soll nur an Wochenenden stattfinden. Damit bleibt die Anzahl von empfänglichen und infektiösen Kindern die ganze Woche über konstant.
- In jeder Woche gibt es eine konstante Anzahl B von Geburten. Wir betrachten eine Stadt mit wöchentlich $B = 120$ Geburten.
- Jedes infektiöse Kind steckt einen festen Bruchteil aller empfänglichen Kinder an. Wir gehen davon aus, dass pro Woche ein infektiöses Kind ein weiteres Kind pro Woche ansteckt. Aus dem Zahlenmaterial konnte man daraus den Wert $f = 0.3 \cdot 10^{-4}$ für den konstanten Bruchteil ermitteln.
- Zu Beginn gibt es 20 infektiöse und 30 000 empfängliche Kinder in dieser Stadt.

Da es jeweils eine Woche dauert, bis ein Kind infektiös wird bzw. sich die Pusteln zeigen, liegt unserem zu erstellendem Modell eine Zeitskala mit Schritten von einer Länge von einer Woche zugrunde. Wir führen folgende Größen ein

$$I_n = \text{Anzahl der infektiösen Kinder in der } n\text{-ten Woche}$$

$$E_n = \text{Anzahl der empfänglichen Kinder in der } n\text{-ten Woche}$$

$$M_n = \text{Zahl der immunen Kinder in der } n\text{-ten Woche}$$

Ein Zahlenbeispiel

Wir betrachten ein kleines Zahlenbeispiel, um ein „Gespür“ für das Modell zu kriegen. Nehmen wir einmal an, es gäbe in der 3. Woche 50 infektiöse, 10 000 empfängliche und 30 immune Kinder. Wenn jedes infektiöse Kind genau drei Kinder ansteckt und pro Woche 200 Kinder geboren werden, so gibt es in der vierten Woche 150 infektiöse, 80 immune und 10 050 empfängliche Kinder.

B.4. Allgemeine Modellierung

Allgemein überlegen wir uns, wie viele empfängliche Kinder ein infektiöses Kind in der $(n + 1)$ -ten Woche ansteckt. Gemäß den oben formulierten Annahmen gilt es E_n

empfängliche Kinder. Daher lautet die Antwort für ein infektiöses Kind $f \cdot E_n$. Nun gibt es aber I_n viele infektiöse Kinder. Diese zusammen stecken $f \cdot I_n \cdot E_n$ Kinder an. Wie viele Kinder sind in der $(n+1)$ -ten Woche empfänglich? Nun, das sind gerade die Empfänglichen aus der Vorwoche vermindert um die Zahl derer, die sich infiziert haben. Hinzu kommen aber noch die Neugeborenen. Hingegen ergibt sich die Zahl der immunen Kinder der $(n+1)$ -Woche, indem man zu den immunen Kindern der Vorwoche noch die infektiösen Kinder der Vorwoche hinzuzählt, da sie inzwischen gesundet sind. Zusammen ergibt sich somit

$$\begin{aligned} I_{n+1} &= f \cdot E_n \cdot I_n \\ E_{n+1} &= E_n - f \cdot E_n \cdot I_n + B \\ M_{n+1} &= M_n + I_n \end{aligned}$$

mit den Anfangsbedingungen $I_0 = 20$, $E_0 = 30000$, $M_0 = 0$. In unserem Fall ist das zu lösende System also

$$\begin{aligned} I_{n+1} &= 0.3 \cdot 10^{-4} \cdot E_n \cdot I_n \\ E_{n+1} &= E_n - 0.3 \cdot 10^{-4} \cdot E_n \cdot I_n + 120 \end{aligned}$$

Wir sind mit den Daten unserer Stadt gestartet, in der es pro Woche 120 Geburten gibt. Nun wollen wir den Einfluss der Geburtenrate studieren und legen dazu die Rate $B = 360$ für eine vergleichbare Stadt in einem Entwicklungsland zugrunde, in dem die Geburtenrate weitaus höher ist. Wie ändert sich dadurch der Verlauf der Anzahl der infektiösen Kinder?

Übung B.1.

Simuliere das obige Szenario mit beiden Geburtenraten für 1000 Zeitenheiten. Vergleiche die Graphiken einerseits mit dem Artikel am Anfang und kommentiere andererseits den Einfluss der Geburtenrate auf den zeitlichen Verlauf der Krankheit.

Wir erkennen in einer Simulation dieselbe Struktur, die sich auch schon in den realen Daten im Zeitungsartikel gezeigt haben. Auf Zeiten des fast völligen Verschwindens von Masern folgen in regelmässigen Abständen epidemieartige Ausbrüche der Krankheit. Man stellt ebenfalls fest, dass der zunächst harmlos erscheinende Parameter B in der Simulation als sehr einflussreiche Grösse! Die Massernepidemien in der Vergleichsstadt aus dem Entwicklungsland sind unvergleichbar heftiger als in der Stadt, mit der wir unsere Überlegungen begonnen haben. Das gilt nicht — wie eine stereotype Wahrnehmung von Entwicklungsländern vielleicht zunächst vermuten lässt — aufgrund mangelnder Hygiene oder eines desolaten Gesundheitssystems, sondern lediglich aufgrund der höheren Geburtenrate!

B.5. Notizen zu Masern

Notizen zu Übung B.1.

```
import numpy as np
import matplotlib.pyplot as plt

# Anfangsbedingungen
I_0 = 20
E_0 = 30000
M_0 = 0

# Parameter
beta = 0.3 * 10**-4
iterations = 1000

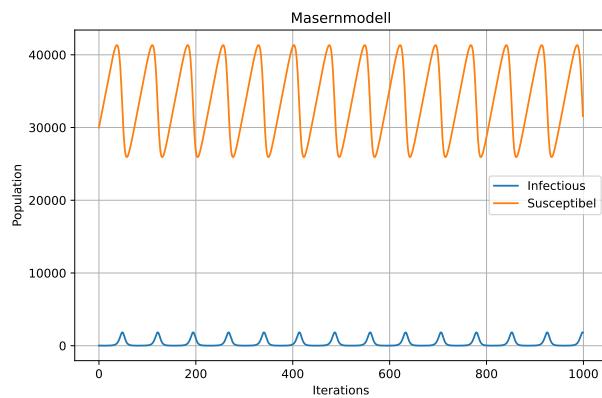
# Arrays zur Speicherung der Werte
I = np.zeros(iterations)
E = np.zeros(iterations)

# Anfangswerte setzen
I[0] = I_0
E[0] = E_0

# Iteration des Modells
for n in range(iterations - 1):
    I[n + 1] = beta * E[n] * I[n]
    E[n + 1] = E[n] - beta * E[n] * I[n] + 120

# Plot der Ergebnisse
plt.figure(figsize=(10, 5))
plt.plot(I, label='Infektioes (I)')
plt.plot(E, label='Empfaenglich (E)')
plt.xlabel('Iterations')
plt.ylabel('Population')
plt.title('Epidemiemodell')
plt.legend()
plt.grid(True)
plt.show()
```

Nun kann man für 120 einen Parameter B setzen und die Simulation für verschiedene Geburtenraten laufen lassen.



Die, für mich erstaunliche, Erkenntnis ist, dass die Peakhöhe allein auf die Änderung der Geburtenrate reagiert.

C. Populationen in Wechselwirkung



Wir untersuchen jetzt die Entwicklung zweier wechselwirkender Populationen (x_n) und (y_n). Dabei werden wir die folgenden drei Hauptfälle unterscheiden

- Symbiose (win-win)
- Räuber-Räuber (lose-lose)
- Räuber-Beute (win-lose)

Gekoppelte Populationen führen auf Systeme von Differenzengleichungen bzw. Differenzialgleichungen.



C.1. Populationen in Symbiose

Das wechselseitig fördernde Zusammenleben zweier Populationen wird im einfachsten Fall durch ein Gleichungssystem der folgenden Form beschrieben:

$$\begin{aligned} x_{n+1} &= x_n + a \cdot y_n \\ y_{n+1} &= y_n + b \cdot x_n \end{aligned}$$

Hierin sind a und b positive Konstanten, die die gegenseitige Beeinflussung beschreiben. Durch Elimination von (y_n) kann man aus dem System eine einzelne Differenzengleichung 2. Ordnung für die Folge der (x_n) gewinnen. Man erhält

$$\begin{aligned} x_{n+1} &= 2x_n + (ab - 1)x_{n-1} \\ y_{n+1} &= 2y_n + (ab - 1)y_{n-1} \end{aligned}$$

Dies aber sind zwei Lucas-Gleichungen, die von $x_n = q^n$ mit $q_{1,2} = 1 \pm \sqrt{ab}$ gelöst werden. Mittels $x_n = rq_1^n + sq_2^n$ werden dann $r, s \in \mathbb{R}$ bestimmt, so dass vorgegebene Anfangsbedingungen erfüllt sind.

Übung C.1.

Erstelle eine Tabelle für $x_0 = 80$, $y_0 = 100$, $a = 0.0625$ und $b = 0.04$ für $n = 10$ Zeiteinheiten. Plotte die beiden Populationen in ein Koordinatensystem für die ersten 40 Zeiteinheiten.

C.2. Notizen zu Symbiose**Notizen zu Übung C.1.**

```
import numpy as np
import matplotlib.pyplot as plt

# Anfangsbedingungen
x_0 = 80
y_0 = 100
a = 0.0625
b = 0.04
n = 10

# Arrays zur Speicherung der Werte
x = np.zeros(n)
y = np.zeros(n)

# Anfangswerte setzen
x[0] = x_0
y[0] = y_0

# Iteration des Modells
for i in range(1, n - 1):
    x[i + 1] = 2 * x[i] + (a * b - 1) * x[i - 1]
    y[i + 1] = 2 * y[i] + (a * b - 1) * y[i - 1]

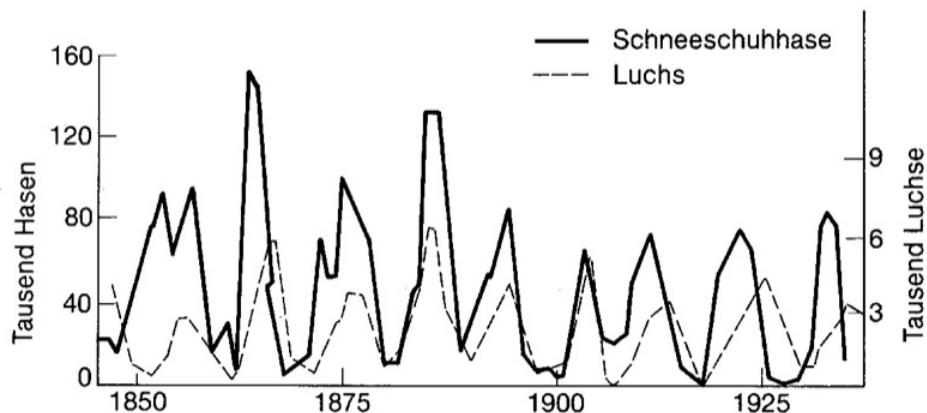
# Plot der Ergebnisse
plt.figure(figsize=(10, 5))
plt.plot(x, label='x')
plt.plot(y, label='y')
plt.xlabel('Iterations')
plt.ylabel('Population')
plt.title('Lotka-Volterra Modell')
plt.legend()
plt.grid(True)
plt.show()
```

Nun muss man die Parameter a und b aus Daten bestimmen oder man kann damit spielen.

C.3. Räuber-Beute-Modelle

Die meisten Organismen müssen ihre Nährstoffe von anderen Organismen erhalten. Das Populationswachstum erfolgt daher zwangsläufig auf Kosten und meist zum Nachteil anderer Populationen. Diese Beziehung zwischen Schädigern und Geschädigten wird als **Räuber-Beute-Systeme** bezeichnet. Historisch sind solche Systeme unabhängig voneinander von dem österreichisch-amerikanischen Mathematiker ALFRED JAMES LOTKA 1925 und dem italienischen Mathematiker und Physiker VITO VOLTERRA 1926 untersucht worden, und zwar anhand der Populationen von Schneeschuhhasen und Luchsen in Kanada bzw. Haien und Speisefischen in der italienischen Adria.

Wir betrachten den zeitlichen Verlauf der Populationen von Luchsen und Hasen, gemessen an den Fangaufzeichnungen der Hudson Bay Company, die über 90 Jahre lang geführt wurden.



Danach schwankten der Eingang von Fellen von Luchsen (Räuber) und Schneeschuhhasen (Beute) mit einer Periode von 9.6 Jahren. Wir beobachten das zyklische Ansteigen und Fallen der jeweiligen Populationen, wobei auf eine hohe Zahl Hasen stets eine wachsende Zahl von Luchsen folgt, woraus in der Folgezeit die Zahl der Hasen drastisch fällt, gefolgt von sinkenden Zahlen von Luchsen, bevor beide Populationen wiederum aufs Neue ansteigen. Die Mathematik kann durch geeignete Modellierung ein erhellendes Licht auf die beobachteten Phänomene werfen?

Wir modellieren ein sehr vereinfachtes Ökosystem, in dem nur Schneeschuhhasen und Luchse leben. Wir vereinfachen weiterhin und nehmen an, dass die Hasen unbeschränkte Wiesen und Wälder zur Verfügung haben, in denen sie genügend Nahrung finden, egal wie viele Hasen es gibt. Dann ist das Wachstum der Hasen exponentiell und lässt sich durch die folgende Differenzengleichung beschreiben:

$$x_{n+1} = x_n + a \cdot x_n$$

für ein $a > 0$.

Nun leben in unserem Ökosystem aber auch Luchse, die sich von Hasen ernähren. Gäbe es keine Hasen, dann verhungerten die Luchse und stürben mehr und mehr aus, d.h.

$$y_{n+1} = y_n - c \cdot y_n$$

mit $c > 0$.

Zum Glück der Luchse sind sie aber nicht vom Aussterben bedroht, solange es Hasen gibt, die sie fressen können. Wie viele Hasen frisst ein einzelner Luchs in einer Zeiteinheit? Nun, in einer ersten Näherung ist es plausibel anzunehmen, dass dies proportional zur Anzahl der Hasen ist, d.h. $b x_n$. Jetzt gibt es aber nicht nur einen Luchs, sondern die Population zum Zeitpunkt n besteht aus y_n Luchsen. Diese fressen dann zusammen $b \cdot x_n \cdot y_n$ Hasen. Ebenso ist der Zuwachs der Luchse proportional zur Anzahl der Begegnungen zwischen den y_n Luchsen und den x_n Hasen. Daher verbessern wir unser Modell zu

$$\begin{aligned} x_{n+1} &= x_n(1 + a) - b \cdot x_n \cdot y_n \\ y_{n+1} &= y_n(1 - c) + d \cdot x_n \cdot y_n \end{aligned}$$

Übung C.2.

Erstelle eine Tabelle für den Verlauf der beiden Populationen, basierend auf der Parameterwahl $x_0 = 150$, $y_0 = 50$, $a = 0.0832$, $b = 0.00207$, $c = 0.06$ und $d = 0.00049$ für 600 Zeiteinheiten.

Die periodische Struktur wie auch die Phasenverschiebung, die ja die Populationsentwicklung der wirklichen Hasen und Luchse charakterisierten, finden wir auch in diesem Modell wieder. Des Weiteren ist bemerkenswert, dass sich die Schwingungen nicht exakt wiederholen, sondern sich gegenseitig aufzuschaukeln scheinen. Dies wird insbesondere in einem Phasendiagramm deutlich. Charakteristisch ist das expandierende Verhalten: Die Populationen scheinen sich spiralförmig immer weiter aufzuschaukeln. Dieses Verhalten hängt mit den hier speziell gewählten Parametern a , b , c und d zusammen. Bei anderer Wahl der Parameter kann sich die Spirale im Phasendiagramm auch nach innen zusammenziehen. In diesem Fall spricht man von einem Attraktor, der einem stabilen Gleichgewicht entspricht.

Übung C.3.

Wie lässt sich ein Gleichgewichtszustand in einem Räuber-Beute-Modell ermitteln? Bestätige den Gleichgewichtszustand (122 | 40) für die Parameter aus obiger Übung.

C.4. Notizen zu Lotka-Voltera

Notizen zu Übung C.2.

```
import numpy as np
import matplotlib.pyplot as plt

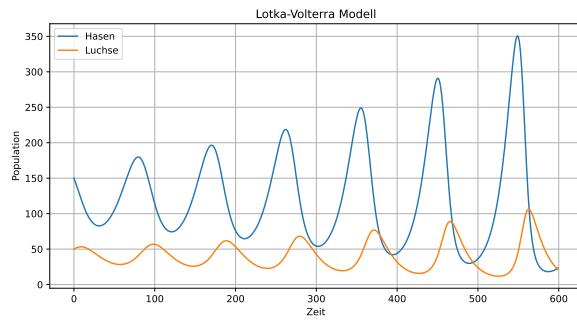
# Anfangsbedingungen
x_0 = 150
y_0 = 50
a = 0.0832
b = 0.00207
c = 0.06
d = 0.00049
time_units = 600

# Arrays zur Speicherung der Werte
x = np.zeros(time_units)
y = np.zeros(time_units)

# Anfangswerte setzen
x[0] = x_0
y[0] = y_0

# Iteration des Modells
for t in range(time_units - 1):
    x[t + 1] = x[t] * (1 + a) - b * x[t] * y[t]
    y[t + 1] = y[t] * (1 - c) + d * x[t] * y[t]

# Plot der Ergebnisse
plt.figure(figsize=(10, 5))
plt.plot(x, label='x')
plt.plot(y, label='y')
plt.xlabel('Time Units')
plt.ylabel('Population')
plt.title('Lotka-Volterra Modell')
plt.legend()
plt.grid(True)
plt.show()
```



Notizen zu Übung C.3. Die Fixpunktbedingung ist

$$\begin{aligned}x &= x(1 + a) - b \cdot x \cdot y \\y &= y(1 - c) + d \cdot x \cdot y\end{aligned}$$

Und es folgt sofort

$$\begin{aligned}0 &= ax - b \cdot x \cdot y \\0 &= -cy + d \cdot x \cdot y\end{aligned}$$

$$\begin{aligned}y &= \frac{a}{b} \\x &= \frac{c}{d}\end{aligned}$$

D. SIR-Modell

GF Math SIR Modell

June 16, 2020

1 Mathematische Modellierung von Epidemien — am Beispiel des SIR Modells

1.1 Mathematische Modelle

Aus aktuellem Anlass habe ich mich dazu entschlossen — etwas spät, aber nach dem Motto: “lieber spät als nie...” — im Grundlagenfach eine kleine Sequenz zu *Modellierung und Differenzialgleichungen* zu halten, welche sonst dem SF oder EF Angewandte Mathematik vorenthalten ist. Ich werde nicht alles Mathematische im Detail erklären, jedoch ab und zu für die Fortgeschrittenen kleine Hinweise einstreuen, damit man bei Interesse weiss, wo man ansetzen könnte. Diese Hinweise kann man aber getrost auch überhören, wenn man möchte.

1.1.1 Was ist ein Modell?

Modell Wikipedia :

Ein Modell ist ein vereinfachtes Abbild der Wirklichkeit. (Wikipedia)

Bei der mathematischen Modellierung wird aus der **Realität** durch Modellbildung ein **Modell** erstellt, das mit mathematischen Methoden ausgewertet werden kann. Das Ergebnis der Berechnungen im Modell muss dann wiederum interpretiert und auf die Wirklichkeit/Realität abgebildet werden.

1.1.2 Was kann ein Modell?

All models are wrong, but some are useful. (George Box)

Ein Modell stimmt *nie* mit der Wirklichkeit überein. Das wird auch nicht verlangt, wäre auch unmöglich, und es gibt viele Fehlerquellen, die gemeinhin nicht in der Mathematik liegen, sondern natürlichen Ursprungs sind (zB fehlerhafte Daten, oder verfälschte Messungen, oder...). Wieso nimmt man dann trotzdem ab und an ein Modell zur Hand? Man weiss aus Erfahrung, dass gewisse, simple Modelle die Realität ziemlich gut beschreiben können. Und es ist mit diesen Modellen möglich, zuverlässige Prognosen zu stellen.

1.1.3 Was ist ein SIR Modell?

Bei mathematischer Modellierung geht es darum, eine schematische, häufig grob vereinfachte, Abstraktion eines Sachverhalts vorzunehmen. Im Beispiel von Epidemien, um die es heute gehen soll, muss eine Gruppe von Menschen in unterschiedliche Klassen aufgeteilt werden. Eine einfache Aufteilung geht beispielsweise so, dass man die Bevölkerung in *Infizierte* , *Gesunde* , und *Genesene*

einteilt. Jedes Individuum gehört genau zu einer Gruppe. Im Englischen spricht man von **Susceptibles**, **Infected** und **Recovered**; kurz eben SIR. Im einfachsten Fall geht man davon aus, dass die Gesamtheit konstant bleibt und dass einmal Infected, die genesen, immun werden. Schematisch sieht dies so aus:

```
[1]: %%latex
\begin{center}
%\fbox{
\includgraphics[trim=2.5cm 22cm 2.5cm 3.5cm, clip,
width=\linewidth]{sirmodel.pdf}
%\end{center}
```



Vermutlich neu für die meisten von euch ist nun, dass es hier nicht um eine Momentaufnahme geht, sondern dass wir *Funktionen als Lösungen* eines dynamischen Systems suchen. Wir möchten zum Beispiel voraussagen können, wie viele zu einem Zeitpunkt t infiziert sein werden, oder vielleicht wie lange es dauert, bis alle infizierten immun sind. Mathematisch heisst das, dass wir Funktionen S , I und R in Abhängigkeit der Zeit t suchen, die uns solche Fragen beantworten können.

- $S(t)$: Anzahl der Empfänglichen zum Zeitpunkt t .
- $I(t)$: Anzahl der Infektiösen zum Zeitpunkt t .
- $R(t)$: Anzahl der Genesenen zum Zeitpunkt t .

1.1.4 Wie finden wir diese Funktionen?

Man sieht rasch, dass es schwierig ist, die Funktionen zu erraten. Jedoch: Wir können etwas über die zeitliche Änderung dieser Funktionen sagen.

- Die Anzahl Personen, welche an einem Tag gesunden ist proportional zur Anzahl der infizierten:

$$\dot{R}(t) = \beta \cdot I(t)$$

Wenn es im Schnitt k Tage dauert, bis man wieder fit ist, so hat β die Größenordnung $\frac{1}{k}$.

- Die Anzahl der Personen, die an einem Tag infiziert werden ist sowohl proportional zur Anzahl Infektiösen als auch zur Anzahl der Empfänglichen:

$$\dot{S}(t) = -\alpha \cdot S(t) \cdot I(t)$$

Den Proportionalitätsfaktor α kann man nicht so leicht abschätzen wie β . α hängt unter anderem sicherlich von der Art der Krankheit und von der Anzahl der Kontakte zwischen den Individuen ab.

- Nun ist klar, wie sich nach unserem Modell die Gruppe der Infizierten ändert:

$$\dot{I}(t) = \alpha \cdot S(t) \cdot I(t) - \beta \cdot I(t)$$

1.1.5 Der Übergang vom diskreten zum kontinuierlichen Modell

```
[3]: %%latex
\begin{center}
%\fbox{
\includegraphics[trim=2.5cm 2.5cm 2.5cm 2.5cm, clip, width=\linewidth]{herleitungdgl.pdf}
\includegraphics[page=2, trim=2.5cm 12cm 2.5cm 2.5cm, clip, width=\linewidth]{herleitungdgl.pdf}    %
\end{center}
```

HERLEITUNG / MOTIVATION

S seien die Empfänglichen. Die zeitliche Änderung dieser Populationsgruppe sei proportional zur Größe von $S(t)$. Je mehr Empfängliche da sind, desto mehr können auch angesteckt werden. Ferner aber auch, so nehmen wir an, proportional zu den Infektösen $I(t)$ zum Zeitpunkt t ; Man könnte auch argumentieren proportional zur Anzahl Begegnungen/Kontakte $S(t) \cdot I(t)$. Dabei nimmt S ab. Wir schreiben für eine kleine Zeitspanne Δt :

$$S(t + \Delta t) = S(t) - \alpha \cdot S(t) \cdot I(t) \cdot \Delta t$$

Bemerkung: Es ist auch illustrativ — zum Beispiel wenn man programmieren möchte —, wenn man obigen Sachverhalt in einer Art "Folgen & Reihen"-Schreibweise notiert:

$$S_{k+1} = S_k - \alpha S_k I_k$$

Für die Infektösen ist demnach

$$I(t + \Delta t) = I(t) + \alpha \cdot S(t) \cdot I(t) \cdot \Delta t$$

Jetzt nimmt man zusätzlich an, dass ein Kranke im Mittel in $\frac{1}{\beta}$ Tagen sich erholt, dass also in einer Zeitspanne Δt im Mittel $\beta \cdot I(t) \cdot \Delta t$ gesunden. Somit haben wir

$$I(t + \Delta t) = I(t) + \alpha S(t) I(t) \Delta t - \beta I(t) \Delta t$$

Damit haben wir noch die Recovered-Änderung mit

$$R(t + \Delta t) = R(t) + \beta I(t) \Delta t$$

Das Modell wird umso geschmeidiger/genauer, je kleinere Zeitabschnitte betrachtet werden. Daher stellen wir nach den Änderungsraten um und lassen $\Delta t \rightarrow 0$ gehen. Es folgt unmittelbar

$$\frac{S(t+\Delta t) - S(t)}{\Delta t} = -\alpha S(t) I(t)$$

$$\frac{I(t+\Delta t) - I(t)}{\Delta t} = \alpha S(t) I(t) - \beta I(t)$$

$$\frac{R(t+\Delta t) - R(t)}{\Delta t} = \beta I(t)$$

und für $\Delta t \rightarrow 0$:

$$\dot{S} = -\alpha S I$$

$$\dot{I} = \alpha S I - \beta I$$

$$\dot{R} = \beta I$$

1.1.6 Methode

Wir haben es mit einem diskreten Modell zu tun, dass wir mit kontinuierlichen Methoden angehen werden. Damit wird es "geschmeidiger", kann aber auch nicht plausible Werte liefern wie Fliesskomazahlen. Oft ist es in der Praxis so, dass man die Lösungen numerisch bestimmen muss, dass man also sowieso kleine Zeitschritte einbaut und nicht analytisch die Lösung findet. Uns ist aber ohnehin klar, dass es beispielsweise 14.2857 Gesunde nicht geben kann. Aber die Methode des Grenzwertübergangs liefert uns präzisere Daten.

Schliesslich wird jetzt noch einmal dieses Differentialgleichungssystem schlank im Überblick projiziert:

$$\dot{S} = -\alpha S I \tag{1}$$

$$\dot{I} = \alpha S I - \beta I \tag{2}$$

$$\dot{R} = \beta I \tag{3}$$

1.2 Plausibilität unseres SIR Modells

1.2.1 Einheitencheck

Stehe t wie üblich für die Zeit. Betrachten wir das Differentialgleichungssystem

$$\dot{S} = -\alpha SI \quad (4)$$

$$\dot{I} = \alpha SI - \beta I \quad (5)$$

$$\dot{R} = \beta I \quad (6)$$

so sehen wir, dass β die Einheit $[\frac{1}{t}]$ haben muss, da ja die linken Seiten die Einheit $[\frac{\text{Individuen}}{t}]$ haben. Etwas Ähnliches haben wir uns vorher schon überlegt. Analog findet man, dass α die Einheit $[\frac{1}{\text{Individuen}\cdot t}]$ hat.

1.2.2 Addieren der Gleichungen des Systems

Wir erhalten $(S + I + R)' = 0$. Das heisst, die Population ist konstant, wie wir angenommen haben.

1.2.3 Abschätzung der Startwerte und qualitative Verläufe

Ganz zu Beginn des Ausbruchs der Epidemie lässt sich unser System grob so formulieren:

$$\dot{S} = -\alpha SI \quad (7)$$

$$\dot{I} = \alpha SI \quad (8)$$

Es kann ja noch gar niemand genesen sein, wenn die Krankheit beispielsweise “neu” ist. Das heisst wir haben $R = 0$ und damit $\dot{R} = 0$. Ist zudem — und dies ist in der Praxis eigentlich immer gegeben — S zu Beginn viel grösser als I , so kann S für kleine t als konstant betrachtet werden. Ist nun S annähernd konstant, dann ist $\dot{S} \approx 0$ und daher haben wir:

$$\dot{I} = \gamma I \quad (9)$$

wobei, $\alpha S =: \gamma$.

Was für eine Funktion erfüllt diese Gleichung? Genau, das ist

$$I(t) = I_0 \cdot e^{\gamma t}.$$

Deshalb haben wir zu Beginn **exponentielles Wachstum**.

Ignoriert man R , wenn zum Beispiel der Genesungsprozess verhältnismässig lange dauert oder man nicht immun wird, dann haben wir

$$\dot{S} = -\alpha SI \quad (10)$$

$$\dot{I} = \alpha SI \quad (11)$$

Da wir annehmen, dass die Gesamtpopulation konstant bleibt, gilt $S + I = 1$ für alle Zeiten t , also $S = (1 - I)$. Somit ergibt sich, da wir eigentlich in diesem Fall nur eine Gleichung brauchen,

$$\dot{I} = \alpha(1 - I)I \quad (12)$$

was **logistisches Wachstum** bedeutet. Übrigens ist diese Gleichung analytisch lösbar, indem man die [Variablen trennt](#) und anschliessend mit Hilfe einer [Partialbruchzerlegung](#) einfache Integrale bereit stellt. Wir werden das im AM erledigen. Die Neugierigen können ja bereits Stift und Papier zur Hand nehmen und selber rechnen.

```
[3]: %%latex
\begin{center}
%\fbox{
\includegraphics[trim=2.5cm 2.5cm 2.5cm 6cm, clip, width=\linewidth]{sirmodel.pdf}
\end{center}
```

Wachstumsverhalten (zu Beginn)

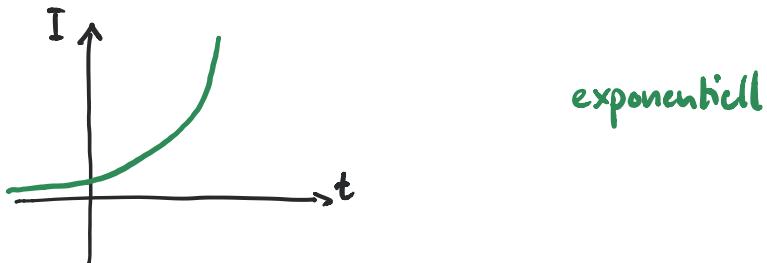
$$\dot{I} = \gamma \cdot I$$

Die Änderung von I ist proportional zu I selbst.

Welche Funktion erfüllt diese Bedingung?

$$\rightarrow I(t) = I_0 \cdot e^{\gamma t}$$

Qualitativ:

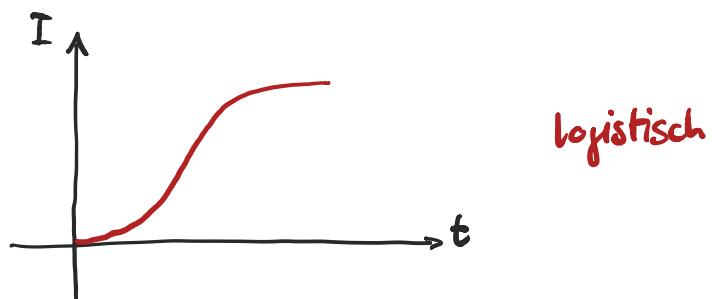


exponentiell

Eher auch zu Beginn, oder bei langer Genesungsphase $\rightarrow R \approx 0$
Grob ist dann $S+I = 100\%$, d.h. $S = 1-I$ und daher

$$\dot{I} = \alpha S I = \alpha I (1-I)$$

Qualitativ:



logistisch

1.3 Das SIR Modell für unser Beispiel — jetzt quantitativ

Ich habe die Gleichungen im nächsten Kapitel in Geogebra bereit gestellt. Dort wird durch das Programm dynamisch der Verlauf berechnet und geplottet. Bei den Schiebereglern kann man die Parameter α und β ändern. Es sei noch vermerkt, dass wir eigentlich nur die ersten beiden Gleichungen brauchen, die dritte Größe (z.B. R) ergibt sich jeweils aus der konstanten Populationsgröße.

Nun folgen diverse statische Plots mit Python, die interessante Details zeigen.

```
[4]: %matplotlib inline
%config InlineBackend.figure_format = 'pdf'
import matplotlib
matplotlib.rcParams["text.usetex"] = True
import matplotlib.pyplot as plt
import numpy as np
from numpy import zeros, linspace
from scipy.integrate import odeint
```

```
[5]: plt.style.use('seaborn')

# Populationsgrösse
N = 40_000
# Anzahl Infektiose zu Beginn, Anzahl Immune zu Beginn
I0, R0 = 1, 0
# Der Rest ist empfänglich
S0 = N - I0 - R0
# Ansteckungsrate und Genesungsrate (pro Person)
alpha, beta = 1.0, 1./20
# linspace (in Tage)
t = np.linspace(0, 65, 300)

# SIR Modell
def deriv(y, t, N, alpha, beta):
    S, I, R = y
    dSdt = -alpha/N * S * I
    dIdt = alpha/N * S * I - beta * I
    dRdt = beta * I
    return dSdt, dIdt, dRdt

# Anfangsbedingungen
y0 = S0, I0, R0
# Rekursive Integration über die Zeitspanne
ret = odeint(deriv, y0, t, args=(N, alpha, beta))
S, I, R = ret.T

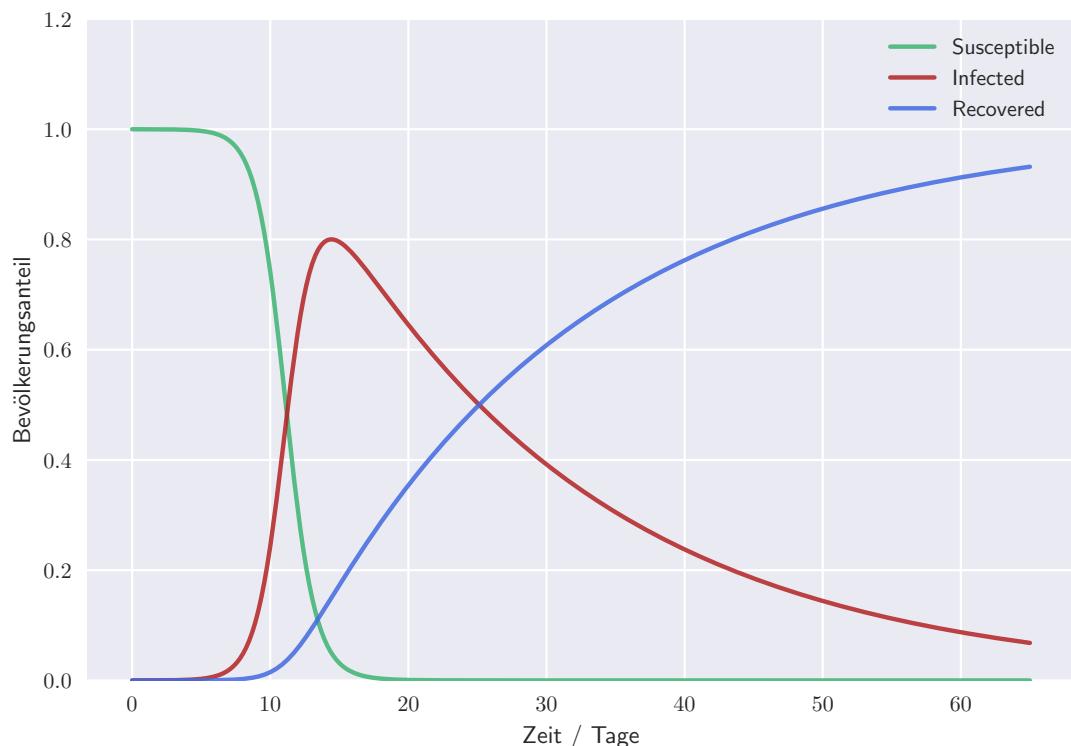
# Plotting S(t), I(t) and R(t)
fig = plt.figure(facecolor='w')
ax = fig.add_subplot(111, frame_on=True, axisbelow=True)
ax.plot(t, S/N, color="#3CB371", alpha=0.85, lw=2, label='Susceptible')
ax.plot(t, I/N, color="#B22222", alpha=0.85, lw=2, label='Infected')
ax.plot(t, R/N, color="#4169E1", alpha=0.85, lw=2, label='Recovered')
ax.set_xlabel(r'Zeit / Tage')
ax.set_ylabel(r'Bevölkerungsanteil')
ax.set_ylim(0,1.2)
```

```

fig.suptitle(r'SIR Modell mit $\alpha=1.0$', fontsize=20)
legend = ax.legend()
legend.get_frame().set_alpha(0.85)
plt.show()

```

SIR Modell mit $\alpha = 1.0$



```

[6]: plt.style.use('seaborn')

# Populationsgrösse
N = 40_000
# Anzahl Infektiose zu Beginn
I0, R0 = 1, 0
# Der Rest ist empfänglich
S0 = N - I0 - R0
# Ansteckungsrate und Genesungsrate (pro Person)
alpha, beta = 0.1, 1./20
# linspace (in Tage)
t = np.linspace(0, 365, 365)

# SIR Modell
def deriv(y, t, N, alpha, beta):

```

```

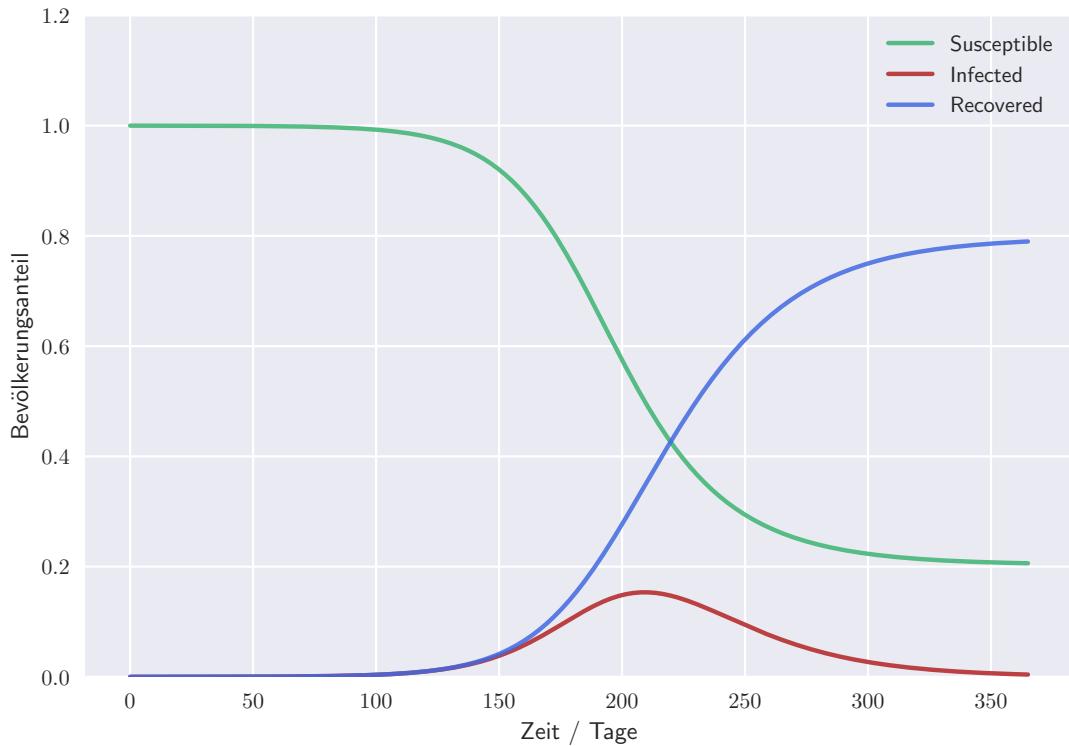
S, I, R = y
dSdt = -alpha/N * S * I
dIdt = alpha/N * S * I - beta * I
dRdt = beta * I
return dSdt, dIdt, dRdt

# Anfangsbedingungen
y0 = S0, I0, R0
# Rekursive Integration über die Zeitspanne
ret = odeint(deriv, y0, t, args=(N, alpha, beta))
S, I, R = ret.T

# Plotting S(t), I(t) and R(t)
fig = plt.figure(facecolor='w')
ax = fig.add_subplot(111, frame_on=True, axisbelow=True)
ax.plot(t, S/N, color="#3CB371", alpha=0.85, lw=2, label='Susceptible')
ax.plot(t, I/N, color="#B22222", alpha=0.85, lw=2, label='Infected')
ax.plot(t, R/N, color="#4169E1", alpha=0.85, lw=2, label='Recovered')
ax.set_xlabel(r'Zeit / Tage')
ax.set_ylabel(r'Bevölkerungsanteil')
ax.set_ylim(0,1.2)
fig.suptitle(r'SIR Modell mit $\alpha=0.1$', fontsize=20)
legend = ax.legend()
legend.get_frame().set_alpha(0.95)
plt.show()

```

SIR Modell mit $\alpha = 0.1$



Es ist also möglich, durch Ändern des Parameters α , den zeitlichen Verlauf der Krankheit innerhalb einer Population zu beeinflussen. Im ersten Beispiel ist der Peak der Infected markant und zwischenzeitlich sind mehr als 80% infiziert. Das muss man vermeiden! Im zweiten Beispiel wurde α reduziert auf einen Zentel, was zum Beispiel mit Massnahmen wie *Social Distancing*, *regelmäßig Händewaschen* etc. erreicht werden kann. Bemerkenswert ist auch, dass im letzten Beispiel nicht alle Susceptibles krank werden und die Krankheit ausstirbt. Beachte auch, dass man für den zeitlichen Verlauf aber viel mehr Geduld braucht.

1.3.1 Dynamisches Geogebrafile zum Spielen

```
[7]: from IPython.display import IFrame  
  
IFrame("https://www.geogebra.org/m/zn54bkmb",800,600)  
# Drück den roten Link um zum dynamischen Geogebra sheet geführt zu werden.
```

```
[7]: <IPython.lib.display.IFrame at 0x7f91302f9dd0>
```

1.3.2 Daten aus der Schweiz (Wikipedia)

Auf [dieser Site](#) sind einige Zahlen/Daten aus der Schweiz zum COVID-19 unter anderem auch graphisch präsentiert. Insbesondere kann hier zum Beispiel der exponentielle und logistische Verlauf gefunden werden.

1.3.3 Parameterschätzung — für den Kanton Bern anhand der März-Zahlen

Man kann nun zum Beispiel I anpassen, indem man die ersten bekannten Messwerte von I nimmt und ein [Curve Fitting](#) durchführt. In den ersten paar Tagen der Epidemie gilt wie oben gesehen näherungsweise $I(t) = I_0 \cdot e^{\gamma \cdot t}$. So erhält man eine Schätzung für γ und kann daraus eine Schätzung für α ermitteln, da zu Beginn $\gamma = \alpha \cdot S_0$.

Hier die Daten für den Kanton Bern im März ([nachgewiesene Infektionen](#)):

Datum im März	Zahl der nachgewiesenen Infektionen
14.	78
16.	123
18.	193
19.	282
20.	377
21.	418
23.	470
24.	532
25.	624

```
[8]: # x axis value list.
date_list = [0,2,4,5,6,7,9,10,11]
# y axis value list.
infections = [78,123,193,282,377,418,470,532,624]

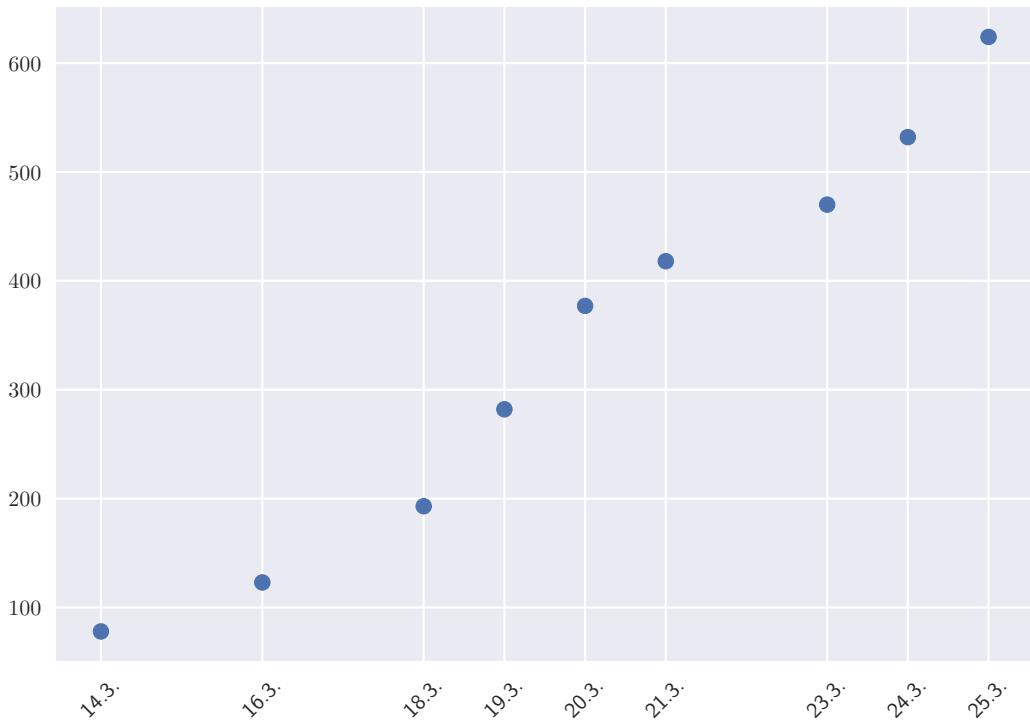
fig = plt.figure(facecolor='w')

plt.scatter(date_list, infections, s=50)
plt.xticks(date_list, ('14.3.', '16.3.', '18.3.', '19.3.', '20.3.', '21.3.', '23.  
→3.', '24.3.', '25.3.'), rotation=45)

fig.suptitle("Anzahl Neuinfektionen im Kanton Bern vom 14. bis 25. März",  
→fontsize=20)

plt.show()
```

Anzahl Neuinfektionen im Kanton Bern vom 14. bis 25. März



Leider hat der Kanton nicht für jeden Tag Daten, so dass ab und zu ein Tag ausgelassen wird. Ich habe ohne Rücksicht auf Verluste einfach diejenigen Werte kumuliert, die ich zur Verfügung hatte. Natürlich "fehlen" dann jeweils nach einem datenlosen Tag diese in der nächsten Aufsummierung. Für das erhobene fitting bedeutet das, dass der erhaltene Wert von γ konservativ ausfällt; also eine milde Schätzung ist.

[WolframAlpha](#) beispielsweise kann so ein Fitting für uns tun:

```
FindFit[{{0,78},{2,201},{4,394},{5,676},{6,1013},{7,1431},{9,1901},{10,2433},{11,3067}},  
78*Exp[c t], {c}, t],
```

wobei ich wie erwähnt die nachgewiesenen Infektionen kumuliert habe.

Es findet für den Ansatz $f(t) = 78 \cdot e^{ct}$ den Wert $c = 0.341643$, mit unserer Notation

$$\gamma = 0.341643.$$

```
[9]: # x axis value list.  
date_list = [0,2,4,5,6,7,9,10,11]  
# y axis value list.  
infections = [78,201,394,676,1013,1431,1901,2433,3067]
```

```

fig = plt.figure(facecolor='w')
plt.scatter(date_list, infections, s=50)

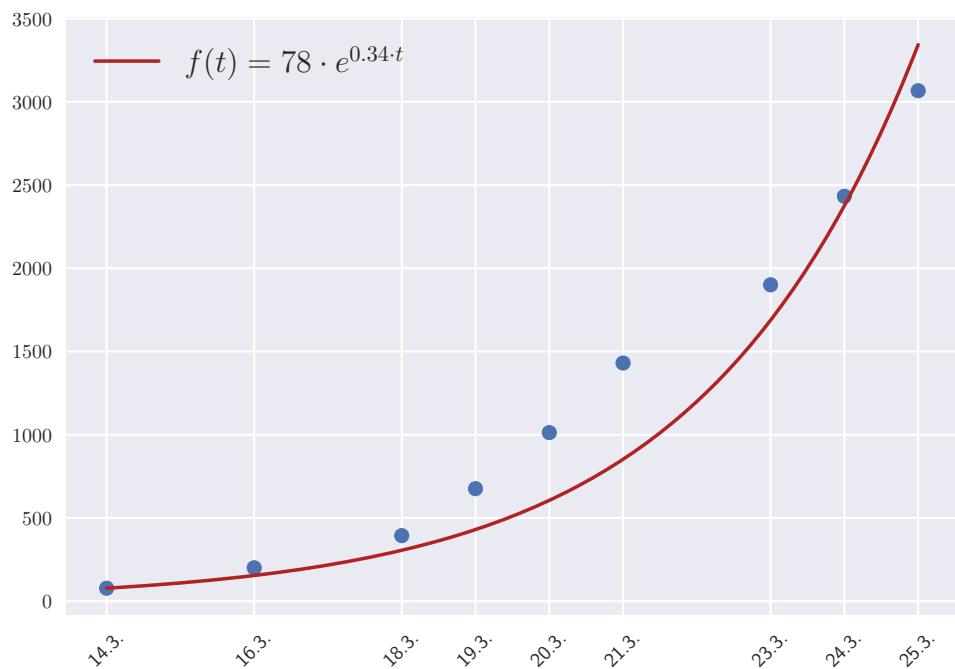
t = linspace(0, 11, 300)
plt.plot(t,78*np.exp(0.341643*t), color="firebrick", label=r"$f(t)=78\cdot e^{0.\underline{3}4\cdot t}$")
plt.xticks(date_list, ('14.3.', '16.3.', '18.3.', '19.3.', '20.3.', '21.3.', '23.\underline{3}.', '24.3.', '25.3.'), rotation=45)

fig.suptitle("Anzahl (kummulierte) Neuinfektionen im Kanton Bern mit Fitting\u21d2\u20ac\u20ac\u20ac", fontsize=20)
plt.legend(fontsize=16)

plt.show()

```

Anzahl (kummulierte) Neuinfektionen im Kanton Bern mit Fitting f



```

[10]: # Populationsgrösse
N = 1_035_000
# Anzahl Infektiose zu Beginn
IO, RO = 78, 0
# Der Rest ist empfänglich
SO = N - IO - RO
# Ansteckungsrate und Genesungsrate (pro Person)

```

```

alpha, beta = 0.341643, 1./20
# linspace (in Tage)
t = np.linspace(0, 120, 120)

# SIR Modell
def deriv(y, t, N, alpha, beta):
    S, I, R = y
    dSdt = -alpha/N * S * I
    dIdt = alpha/N * S * I - beta * I
    dRdt = beta * I
    return dSdt, dIdt, dRdt

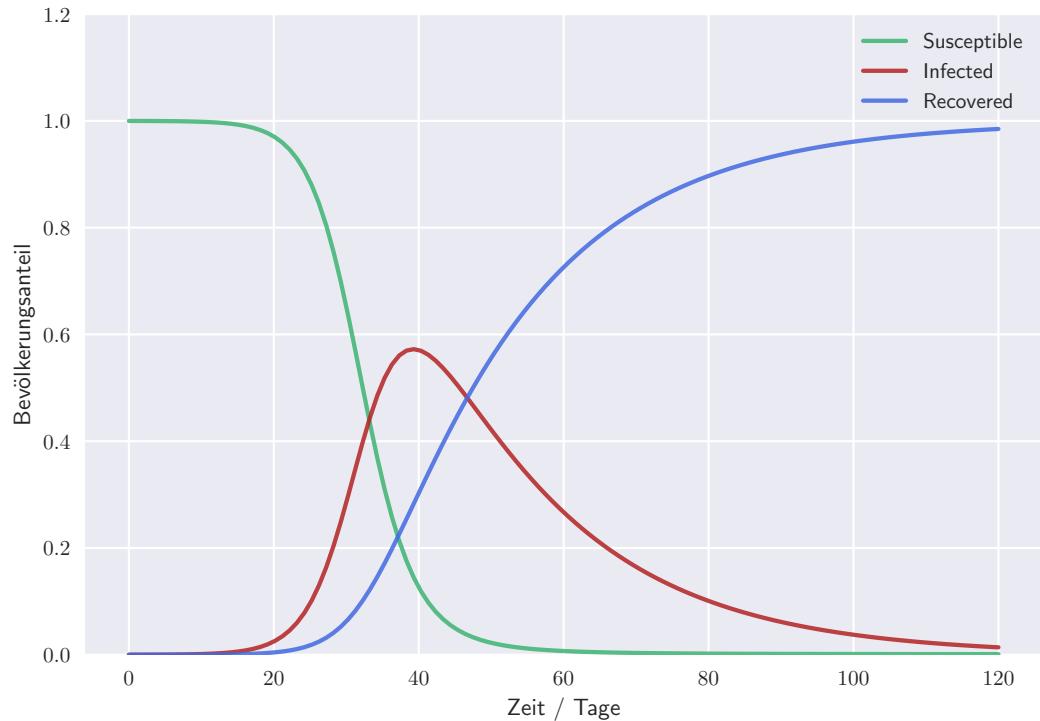
# Anfangsbedingungen
y0 = S0, I0, R0
# Rekursive Integration über die Zeitspanne
ret = odeint(deriv, y0, t, args=(N, alpha, beta))
S, I, R = ret.T

# Plotting S(t), I(t) and R(t)
fig = plt.figure(facecolor='w')
ax = fig.add_subplot(111, frame_on=True, axisbelow=True)
ax.plot(t, S/N, color="#3CB371", alpha=0.85, lw=2, label='Susceptible')
ax.plot(t, I/N, color="#B22222", alpha=0.85, lw=2, label='Infected')
ax.plot(t, R/N, color="#4169E1", alpha=0.85, lw=2, label='Recovered')
ax.set_xlabel(r'Zeit / Tage')
ax.set_ylabel(r'Bevölkerungsanteil')
ax.set_ylim(0,1.2)
fig.suptitle(r'SIR Model für Zahlen aus dem Kanton Bern mit $\alpha=0.' +
            r'\rightarrow 341634$', fontsize=20)
legend = ax.legend()
legend.get_frame().set_alpha(0.85)

plt.show()

```

SIR Model für Zahlen aus dem Kanton Bern mit $\alpha = 0.341634$



Die Daten habe ich aus der Zeit der Schulschliessung. Man kann heute, 31.-Mai 2020, eine *Reproduktionszahl* von $R_0 = 1.22$ für Mitte März nachschlagen, als die Massnahmen bereits in Kraft waren. Vorher war diese Zahl bei etwa 3 anzusetzen und aufgrund des Fittings noch höher. Beachte auch, das das Fitting vermutlich anders ausgefallen wäre, wenn ich die beiden letzten Datenpunkte ausgelassen hätte. Was würde dies für unser Modell für α heissen?

Es ist $\gamma = \alpha \cdot S_0$, also kriegen wir etwas pingelig $\alpha = \frac{\gamma}{S(0)-I(0)} =$

[11]: $0.341643/(1_035_000-78)$

[11]: $3.301147332842475\text{e-}07$

Die Reproduktionszahl betrüge demnach $R_0 = \frac{\alpha}{\beta} \cdot S_0 =$

[12]: $(0.341643/(1_035_000-78))/(1/20)*1_035_000$

[12]: 6.833374978983922

Wollen wir wissen, welchem Wert α in etwa eine Reproduktionszahl von $R_0 = 3$ entspricht, so stellen wir nach α um:

[13]: $3/1_035_000*(1/20)*(1_035_000-78)$

[13]: 0.14998869565217393

Der Wert $\alpha = 0.15$ entspräche also ungefähr einem Wert der Reproduktionszahl von $R_0 = 3$ für unser Modell.

```
[14]: # Populationsgrösse
N = 1
# Anzahl Infektiöse zu Beginn
I0, R0 = 0.001, 0.0
# Der Rest ist empfänglich
S0 = N - I0 - R0
# Ansteckungsrate und Genesungsrate (pro Person)
alpha, beta = 1.5*10**(-1), 1./24
# linspace (in Tage)
t = np.linspace(0, 200, 365)

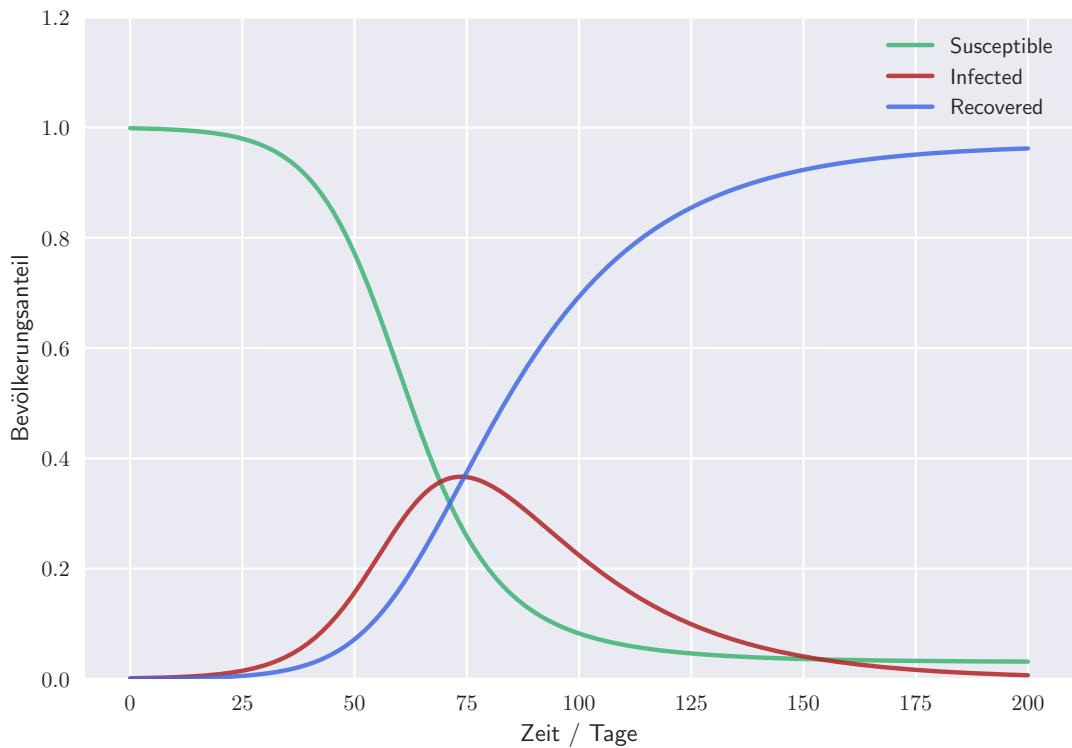
# SIR Modell
def deriv(y, t, N, alpha, beta):
    S, I, R = y
    dSdt = -alpha/N * S * I
    dIdt = alpha/N * S * I - beta * I
    dRdt = beta * I
    return dSdt, dIdt, dRdt

# Anfangsbedingungen
y0 = S0, I0, R0
# Rekursive Integration über die Zeitspanne
ret = odeint(deriv, y0, t, args=(N, alpha, beta))
S, I, R = ret.T

# Plotting S(t), I(t) and R(t)
fig = plt.figure(facecolor='w')
ax = fig.add_subplot(111, frame_on=True, axisbelow=True)
ax.plot(t, S/N, color="#3CB371", alpha=0.85, lw=2, label='Susceptible')
ax.plot(t, I/N, color="#B22222", alpha=0.85, lw=2, label='Infected')
ax.plot(t, R/N, color="#4169E1", alpha=0.85, lw=2, label='Recovered')
ax.set_xlabel(r'Zeit / Tage')
ax.set_ylabel(r'Bevölkerungsanteil')
ax.set_yticks([0, 1, 2])
fig.suptitle(r'SIR Modell mit $\alpha=0.15$', fontsize=20)
legend = ax.legend()
legend.get_frame().set_alpha(0.85)

plt.show()
```

SIR Modell mit $\alpha = 0.15$



Man will natürlich die Reproduktionszahl unter 1 bringen. Dies hiesse für α :

[15]: $1/1_035_000 * (1/20) * (1_035_000 - 78)$

[15]: 0.049996231884057975

also etwa 0.05. Der Verlauf wäre nun für ein R_0 knapp grösser 1.

```
[16]: # Populationsgrösse
N = 1
# Anzahl Infektiose zu Beginn
IO, RO = 0.001, 0.0
# Der Rest ist empfänglich
SO = N - IO - RO
# Ansteckungsrate und Genesungsrate (pro Person)
alpha, beta = 0.7*10**(-1), 1./24
# linspace (in Tage)
t = np.linspace(0, 365, 365)

# SIR Modell
```

```

def deriv(y, t, N, alpha, beta):
    S, I, R = y
    dSdt = -alpha/N * S * I
    dIdt = alpha/N * S * I - beta * I
    dRdt = beta * I
    return dSdt, dIdt, dRdt

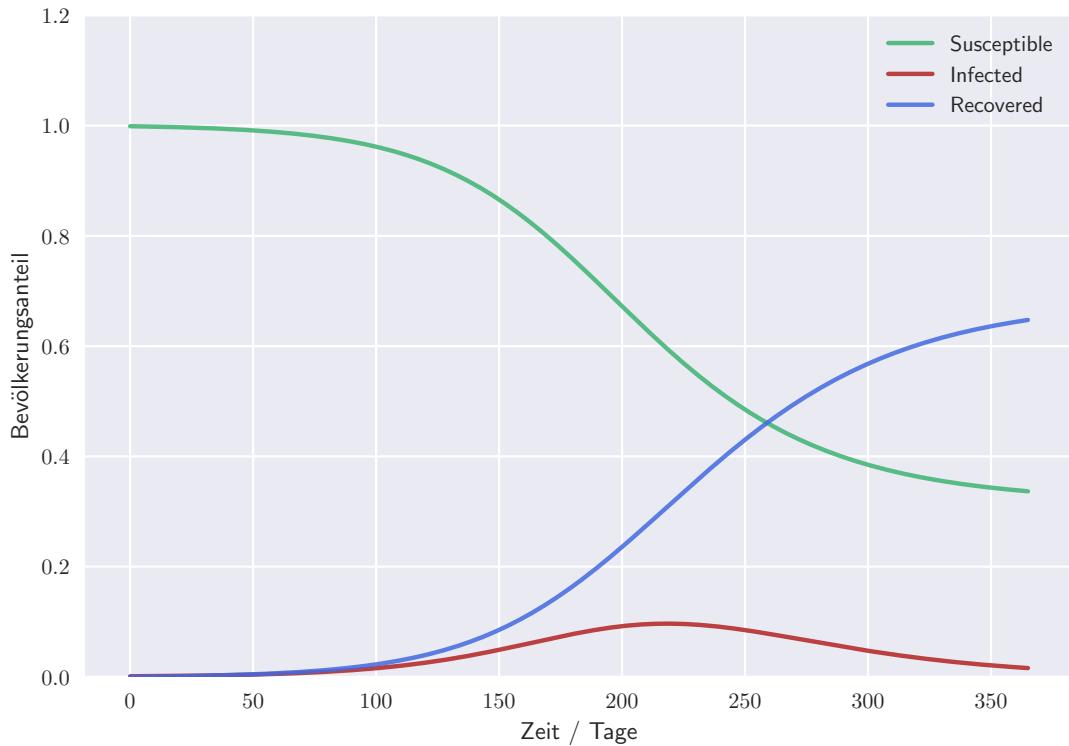
# Anfangsbedingungen
y0 = S0, I0, R0
# Rekursive Integration über die Zeitspanne
ret = odeint(deriv, y0, t, args=(N, alpha, beta))
S, I, R = ret.T

# Plotting S(t), I(t) and R(t)
fig = plt.figure(facecolor='w')
ax = fig.add_subplot(111, frame_on=True, axisbelow=True)
ax.plot(t, S/N, color="#3CB371", alpha=0.85, lw=2, label='Susceptible')
ax.plot(t, I/N, color="#B22222", alpha=0.85, lw=2, label='Infected')
ax.plot(t, R/N, color="#4169E1", alpha=0.85, lw=2, label='Recovered')
ax.set_xlabel(r'Zeit / Tage')
ax.set_ylabel(r'Bevölkerungsanteil')
ax.set_ylim(0,1.2)
fig.suptitle(r'SIR Modell mit $\alpha=0.07$', fontsize=20)
legend = ax.legend()
legend.get_frame().set_alpha(0.95)

plt.show()

```

SIR Modell mit $\alpha = 0.07$



Natürlich kann man jetzt mit schickeren Modellen arbeiten. Wenn ihr Lust habt, so hoffe ich gezeigt zu haben, wie man von diesem Punkt aus hier die Modelle auch verfeinern kann. Stichworte zum Vertiefen sind *SEIR*, *SIRV*, *recursiv SIR*, *zelluläre Automaten* ...

Die "echten" Daten kann man zum Beispiel auf der offiziellen Site [COVID 19 Informationen Kanton Bern](#) einsehen und mit den Modellen vergleichen. Aber auch bei offiziellen Seiten immer gut selber gucken, welche Daten wie verwendet wurden.

Abschliessend noch einige mathematische Bemerkungen zu unserem SIR Modell.

1.4 Mathematische Analyse des SIR Modells

Es folgen noch einige Rechnungen, welche auch die Begriffe [Basisreproduktionszahl](#) und [Herdenimmunität](#) mathematisch greifbar werden lassen.

```
[17]: %%latex
\begin{center}
% \fbox{
\includegraphics[page=1, trim=2.5cm 2.5cm 2.5cm 2.5cm, clip,
width=\linewidth]{reproduktionszahl.pdf}
```

```
\includegraphics[page=2, trim=2.5cm 9cm 2.5cm 2.5cm, clip, ↵
width=\linewidth]{reproduktionszahl.pdf}%
\end{center}
```

Analyse des SIR Modells

zur Basisreproduktionszahl R_0 :

Wir setzen $\gamma := \frac{\alpha}{\beta}$ (Ansteckungsrate eines Infektösen über seine gesamte "Karriere")

$R_0 := \gamma \cdot S(0)$ (Anzahl Menschen, die ein Infektöser (maximal) ansteckt.)

Es ist $\dot{I} = \alpha IS - \beta I \leq \alpha IS(0) - \beta I$, da $S(t) \leq S(0) \quad \forall t \in R_0^+$.

und $\dot{I} \leq \alpha IS(0) - \beta I = \alpha I \frac{R_0}{\gamma} - \beta I = \beta IR_0 - \beta I = \beta I(R_0 - 1)$

Wegen $\beta, I > 0$ nimmt I nur ab, falls $R_0 < 1$. In diesem Fall wird es nicht zu einer Epidemie kommen.

Wann ist dies der Fall?

$$R_0 = \gamma \cdot S(0) < 1$$

$$N_0 - I(0) - R(0) = S(0) < \frac{1}{\gamma} \quad \Leftrightarrow \quad R(0) > N_0 - I(0) - \frac{1}{\gamma}$$

D.h. kann man γ nicht ändern, so kann man $R(0)$ zum Beispiel durch Impfung erhöhen. Typische Werte liegen dann rasch im Bereich von 50% oder mehr. (\rightarrow Herdenimmunität)

Wie viele Menschen sind maximal zur selben Zeit infiziert?

$$\Rightarrow \dot{I} = \alpha IS - \beta I = (\alpha S - \beta) \cdot I \stackrel{!}{=} 0$$

$$\Leftrightarrow S = \frac{\beta}{\alpha} = \underline{\underline{\frac{1}{\gamma}}}$$

$$\text{Betrachte } I(S) \text{ und: } \frac{dI}{dS} = \frac{\frac{dI}{dt}}{\frac{dS}{dt}} = \frac{\alpha IS - \beta I}{-\alpha IS} = \frac{1}{\gamma S} - 1$$

$$\text{Integration liefert : } I(S) = -S + \frac{1}{g} \log(S) + C$$

und die Integrationskonstante erhalten wir durch die Anfangsbedingung

$$I(0) = I(S(0)) \stackrel{!}{=} -S(0) + \frac{1}{g} \log(S(0)) + C$$

$$\Leftrightarrow I(S) = -S + \frac{1}{g} \log(S) + I(0) + S(0) - \frac{1}{g} \log(S(0))$$

$$\text{Somit ist : } I_{\max} = I\left(\frac{1}{S}\right) = -\frac{1}{S} + \frac{1}{g} \log\left(\frac{1}{S}\right) + I(0) + S(0) - \frac{1}{g} \log(S(0))$$

$$=: \underline{\underline{I_{\max}(g)}}$$

Wie viele Menschen werden sich insgesamt anstecken ?

Dann wird $I(t^*) \stackrel{!}{=} 0$ sein. Setze $S^* = S(t^*)$.

$$\text{Es ist } 0 \stackrel{!}{=} I(S^*) = -S^* + \frac{1}{g} \log(S^*) + I(0) + S(0) - \frac{1}{g} \log(S(0))$$

$$\text{und mit Lambert-W-funktion folgt : } S^* = -\frac{1}{g} W_0(-g S(0) \cdot e^{-g N_0})$$

$$\text{Für R folgt : } R(t^*) = N_0 - S^* = N_0 + \frac{1}{g} W_0(-g S(0) \cdot e^{-g N_0})$$

Lambert'sche W-Funktion

E. Integrierter Pflanzenschutz am Weihnachtsbaum

E.1. Die Situation

Ein Weihnachtsmann stellt zu seinem Entsetzen fest, dass alle Nordmannstannen seiner Weihnachtsbaumkolonie von Sitkäläusen befallen sind. Soll er nun zu Pestiziden greifen oder den lieben Marienkäferchen, den natürlichen Feinden der Sitkäläuse, die Arbeit überlassen?



Diese Frage lässt sich an Hand eines Räuber-Beute-Modells beantworten, im vorliegenden Fall angewandt auf das Beutetier Sitkalaus und auf ihren natürlichen Feind, den Marienkäfer.

Gäbe es für die Sitkäläuse keine natürlichen Feinde, so würden sie sich — stark vereinfacht — exponentiell vermehren. Gäbe es für die Marienkäfer keine Beutetiere, so würden sie mangels Futter zugrunde gehen. Treffen nun Sitkäläuse und Marienkäfer aufeinander, geschieht — wer hätte es gedacht? — Folgendes: Die Zahl der Sitkäläuse verringert sich und die Marienkäfer bleiben wohlgenährt am Leben.

E.2. Das Modell

Wir bezeichnen mit x_n die Anzahl der Sitkäläuse und mit y_n die Anzahl der Marienkäfer zur Zeit n . Wenn es für die Sitkäläuse keine natürlichen Feinde gäbe, so würde sich ihre Populationsentwicklung mit der Gleichung

$$x_{n+1} = x_n + \alpha x_n$$

mit einem Parameter $\alpha > 0$ berechnen lassen.

Das Zusammentreffen der Sitkäläuse mit den Marienkäfer, das sich als Produkt von x_n und y_n darstellt, lässt die Sitkalauspopulation weiter schrumpfen, also

$$x_{n+1} = x_n + \alpha \cdot x_n - \beta \cdot x_n \cdot y_n$$

mit einem Parameter $\beta > 0$.

Betrachtet man die Marienkäfer isoliert, so kann man feststellen, dass sie mangels Futter nach demselben Gesetz sterben, wie die Sitkaläuse sich vermehren. In einer Formel ausgedrückt sieht das so aus:

$$y_{n+1} = y_n - \gamma y_n$$

mit einem Parameter $\gamma > 0$.

Anders ist es, wenn sie auf die Sitkaläuse treffen und somit Nahrung haben:

$$y_{n+1} = y_n - \gamma \cdot y_n + \delta \cdot x_n \cdot y_n$$

mit einem Parameter $\delta > 0$.

Beim Einsatz von Pestiziden verringert sich sowohl das Wachstum der Läuse als auch das Wachstum der Marienkäfer. Bei beiden Differenzengleichungen kommt daher der Giftterm $-\varepsilon x_n$ bei den Läusen und $-\varepsilon y_n$ bei den Marienkäfern hinzu mit dem selben Giftfaktor $\varepsilon > 0$. Wir nehmen also an, dass die Marienkäfer in der gleichen Weise unter dem Pestizid leiden wie die Sitkaläuse.

E.3. Die Analyse

Übung E.1.

Stelle zum Text ein passendes Differenzengleichungsmodell für das Räuber-Beute Modell (hier Sitkaläuse versus Marienkäfer) mit und ohne Pestizideinsatz auf. Wähle als Parameter

Anfangswert Läuse	$x_0 = 500$
Anfangswert Marienkäfer	$y_0 = 100$
Vermehrung Läuse	$\alpha = 0.04$
Verminderung Läuse	$\beta = 0.0002$
Schrumpfen Marienkäfer	$\gamma = 0.0035$
Vermehrung Marienkäfer	$\delta = 0.0001$
Pestizid-Wirkung	$\varepsilon = 0.025$

Übung E.2.

Erstelle eine Tabelle und zeichne einen Graph für die Entwicklung der Zahl der Käfer ohne Pestizide, Läuse ohne Pestizide sowie der Käfer und Läuse mit Pestizideinsatz über 200 Zeiteinheiten. Berechne die durchschnittliche Grösse der vier Populationen. Experimentiere und variiere den Parameter ε .

E.4. Notizen zum Weihnachtsbaum

Notizen zu Übung E.1. Ähnlich wie bei Lotka-Voltera notieren wir

$$\begin{aligned}x_{n+1} &= x_n(1 + \alpha - \varepsilon) - \beta \cdot x_n \cdot y_n \cdot (1 - \varepsilon)^2 \\y_{n+1} &= y_n(1 - \gamma - \varepsilon) + \delta \cdot x_n \cdot y_n \cdot (1 - \varepsilon)^2\end{aligned}$$

Notizen zu Übung E.2. Beispielsweise wie folgt:

```
import matplotlib.pyplot as plt

# Anfangswerte
x0 = 500
y0 = 100
alpha = 0.04
beta = 0.0002
gamma = 0.0035
delta = 0.0001
epsilon = 0.025

# Anzahl der Iterationen
iterations = 200

# Listen zum Speichern der Werte von x und y
x_values = [x0]
y_values = [y0]

# Iterieren und die Werte von x und y berechnen
for n in range(iterations):
    x_n = x_values[-1]
    y_n = y_values[-1]

    x_next = x_n * (1 + alpha - epsilon)
                - beta * x_n * y_n * (1 - epsilon)**2
    y_next = y_n * (1 - gamma - epsilon)
                + delta * x_n * y_n * (1 - epsilon)**2

    x_values.append(x_next)
    y_values.append(y_next)

# Ergebnisse plotten
plt.plot(x_values, label='x')
plt.plot(y_values, label='y')
plt.xlabel('Iteration')
plt.ylabel('Werte')
```

```
plt.title('Modell von x und y ueber Iterationen')
plt.legend()
plt.show()
```

F. Von den Walen zur logistischen Gleichung

WalezuLogistik

November 23, 2020

1 Von den Walen zur logistischen Gleichung

1.1 Herleitung

1.1.1 Recap des Walpopulationsmodells

Wir haben für die Entwicklung der Walpopulation ein Modell vom Character

$$W_{k+1} = W_k + \rho W_k (K - W_k)$$

wobei W_k die Anzahl Wale zum Zeitpunkt k , $\rho > 0$ einen Konstante, die unter anderem vom Wachstum der Population abhängig ist und K die Kapazitätsgrenze der Populationsgrösse bezeichnen. Auf gym math gibts das [Video zur Walpopulation](#) auch zum Nachschauen.

Man kann nachrechnen, dass 0 und K Fixpunkte dieses Iterationsmodells sind und für gewisse, vernünftige ρ und K der Fixpunkt 0 instabil und der Fixpunkt K stabil ist.

Übung : Rechne nach.

Experimentiert man ein bisl, kann man beim Modell gucken, was passiert, wenn man seine Grenzen auslotet. Wir setzen das Modell noch mal auf. Dazu bestimmen wir zuerst den Parameter ρ zu einem vermuteten Wachstum von $\tilde{q} = 8\%$ jährlich, Startwert $W_0 = 1\,200$ und Kapazitätsgrenze $K = 20\,000$.

```
[1]: import sympy as sy

W0 = 1200 # starting value
K = 20_000 # capacity
q = 1 + 0.08 # growth factor
rho = sy.symbols('rho') # set r als sympy symbol

eq_rho = sy.Eq(q*W0, W0 + rho*(K-W0)*W0) # set equation
sol_rho = sy.solve(eq_rho) # solve for rho
num_rho = sol_rho[0] # get float
print(r'rho = ' + str(num_rho))
```

$\text{rho} = 4.25531914893617 \cdot 10^{-6}$

```
[2]: num_rho
```

```
[2]: 4.25531914893617 · 10-6
```

```
[3]: %%latex  
\normalfont
```

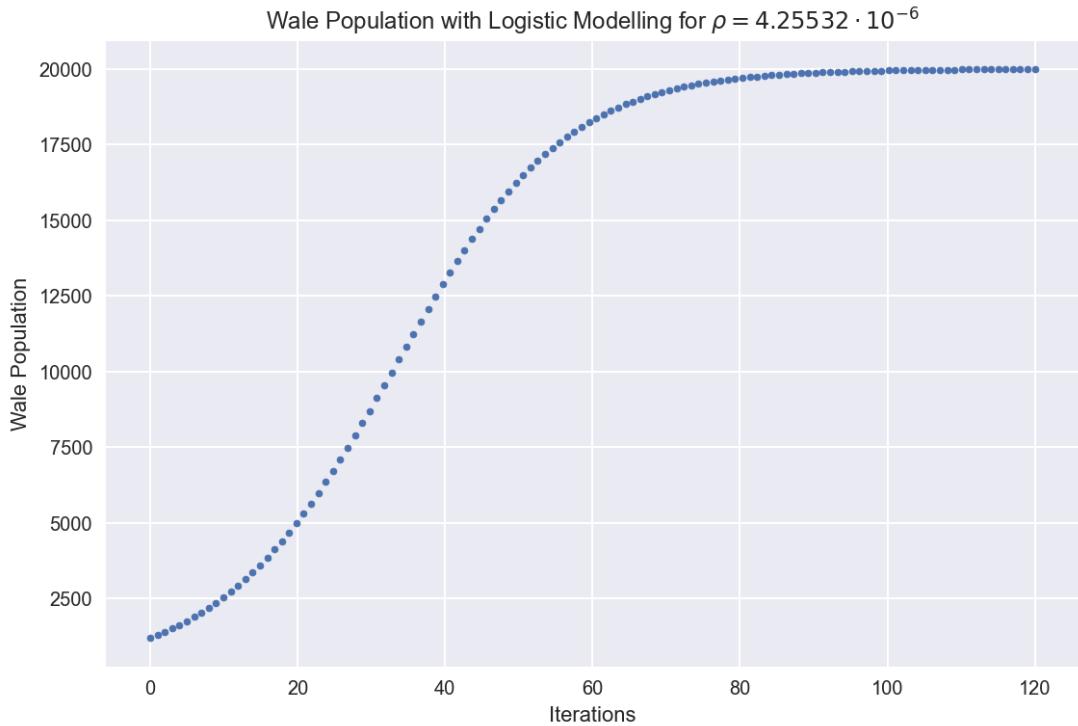
Nun iterieren wir mit den oben genannten Werten, nehmen aber diesmal einen Scatterplot, der die einzelnen Punkte nicht verbindet. Ihr seht im Folgenden wieso. Setzen wir zuerst die Iterationsgleichung als Funktion an:

```
[4]: def walepopfree(W0=1200, K=20_000, rho=4.25532*10**(-6), n=100):  
    # initialize iteration list  
    walepop = []  
    walepop.append(W0)  
    for k in range(n+1):  
        walepop.append(walepop[k] + rho * (K - walepop[k]) * walepop[k])  
    return walepop
```

Und plotten damit eine generierte Prognose mit 120 Einträgen.

```
[5]: walepop=walepopfree(1200, 20_000, numrho, 120)
```

```
[6]: %matplotlib inline  
#%matplotlib notebook  
%config InlineBackend.figure_format = 'retina'  
  
import matplotlib.pyplot as plt  
from numpy import linspace  
  
plt.style.use('seaborn')  
  
x = linspace(0, 120, 122)  
plt.scatter(x, walepop, marker='.')  
plt.xlabel('Iterations')  
plt.ylabel('Wale Population')  
plt.title(r'Wale Population with Logistic Modelling for $\rho=4.25532\cdot 10^{-6}$')  
  
plt.tight_layout()  
  
plt.show()  
#plt.savefig('plot.svg')
```



Mit dem oben aus dem Kontext berechneten ρ ergibt sich was wir erwarten: eine logistische Wachstumskurve mit Startwert 1200 und Kapazitätsgrenze K . Jetzt aber variieren statt wie im vorangegangenen Modul die Startwerte W_0 oder die Fangzahlen/-quoten den Parameter ρ ; und staunen:

```
[7]: walepop1=walepopfree(1200, 20_000, 0.980*10**(-4), 120)
      walepop2=walepopfree(1200, 20_000, 1.270*10**(-4), 120)
      walepop3=walepopfree(1200, 20_000, 1.278*10**(-4), 120)
      walepop4=walepopfree(1200, 20_000, 1.335*10**(-4), 120)
```

```
[8]: fig, axs = plt.subplots(2, 2)
axs[0, 0].plot(walepop1, color='mediumseagreen', label='empfänglich')
axs[0, 0].set_title(r'$\rho=0.980 \cdot 10^{-4}$')
axs[0, 1].plot(walepop2, color='firebrick')
axs[0, 1].set_title(r'$\rho=1.270 \cdot 10^{-4}$')
axs[1, 0].plot(walepop3, color='darkgoldenrod')
axs[1, 0].set_title(r'$\rho=1.278 \cdot 10^{-4}$')
axs[1, 1].plot(walepop4, color='darkviolet')
axs[1, 1].set_title(r'$\rho=1.335 \cdot 10^{-4}$')

fig.suptitle(r'Logistic Model for Different Values of $\rho$')
#l1, = ax1.plot(t, E, color='mediumseagreen', label='empfänglich')
```

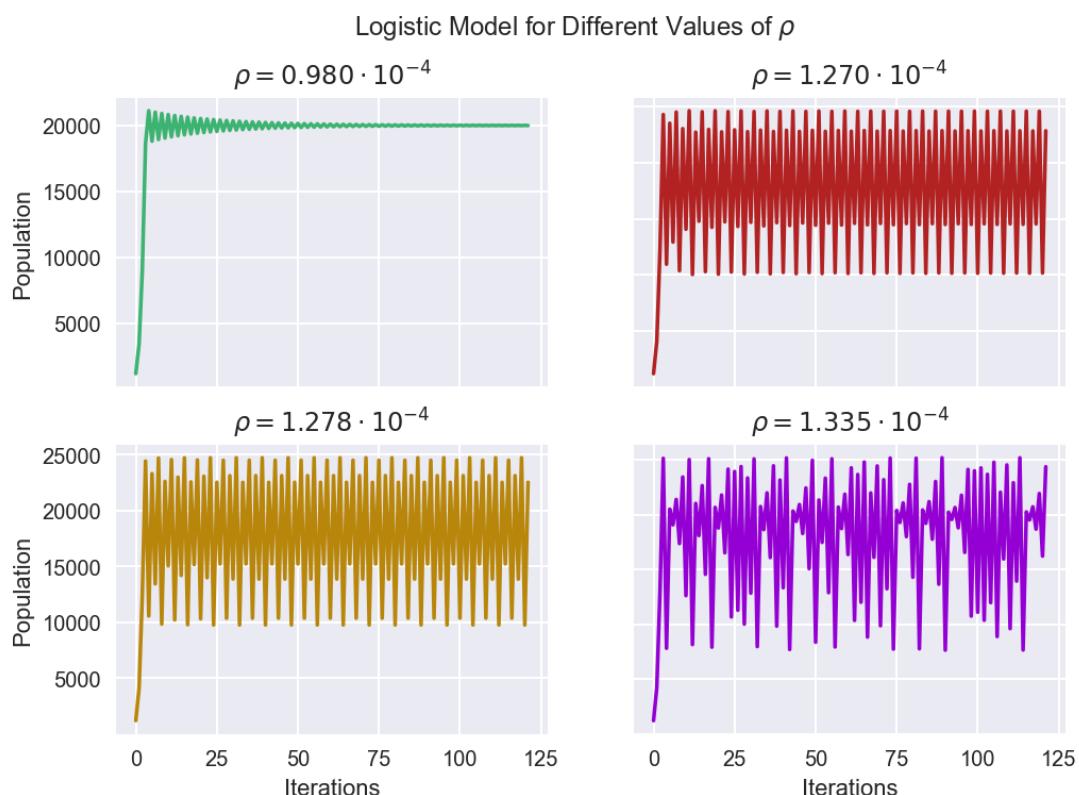
```

#fig.legend()
#plt.xlabel('Tage')
#plt.ylabel('Kinder')
#plt.show()

for ax in axs.flat:
    ax.set(xlabel='Iterations', ylabel='Population')

# Hide x labels and tick labels for top plots and y ticks for right plots.
for ax in axs.flat:
    ax.label_outer()

```



Hier sieht man bloss, dass die Punkte mit fort laufender Iteration ‘nervös’ hin und her springen. Gut ersichtlich ist der zeitliche Verlauf. Um aber die ‘Punkteverteilung’ besser zu sehen, stellen wir auf Scatterplot um, welcher die Punkte nicht verbindet. In der Tat handelt es sich ja auch um einzelne, berechnete Punkte W_k , und nicht um stetige Funktionen.

```

[9]: fig, axs = plt.subplots(2, 2)
axs[0, 0].scatter(x, walepop1, marker='.', color='mediumseagreen', label='empfänglich')
axs[0, 0].set_title(r'Hopping and Stabilizing')

```

```

    axs[0, 1].scatter(x, walepop2, marker='.', color='firebrick')
    axs[0, 1].set_title(r'4-Cycle')
    axs[1, 0].scatter(x, walepop3, marker='.', color='darkgoldenrod')
    axs[1, 0].set_title(r'8-Cycle')
    axs[1, 1].scatter(x, walepop4, marker='.', color='darkviolet')
    axs[1, 1].set_title(r'Chaos')

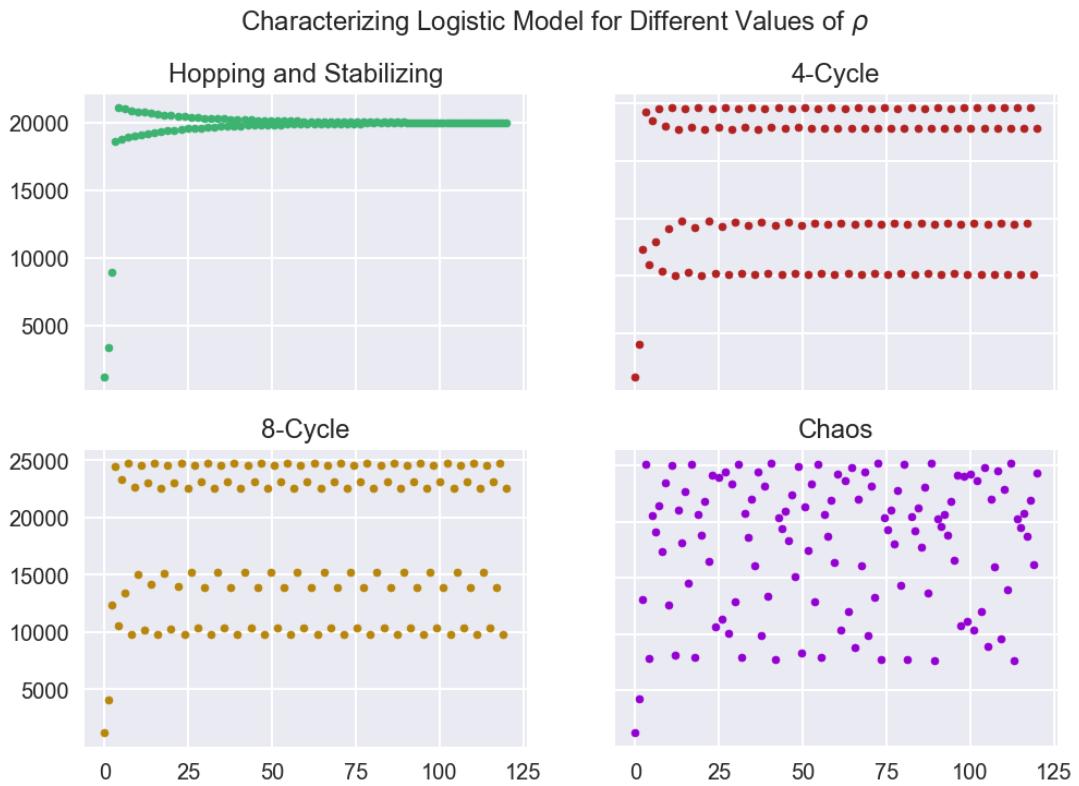
fig.suptitle(r'Characterizing Logistic Model for Different Values of $\rho$')
#l1, = ax1.plot(t, E, color='mediumseagreen', label='empfänglich')

#fig.legend()
#plt.xlabel('Tage')
#plt.ylabel('Kinder')
#plt.show()

#for ax in axs.flat:
#    ax.set(xlabel='Iterations', ylabel='Population')

# Hide x labels and tick labels for top plots and y ticks for right plots.
for ax in axs.flat:
    ax.label_outer()

```



1.2 Analyse

1.2.1 Funktionenschreibweise

Wir wollen für weitere Analysen nun weg von der iterativen Schreibweise hin zur Funktionenschreibweise. Dazu wird die Iteration linear skaliert und normiert, so dass nur noch ein Parameter übrig bleibt.

Übung : Gehe von der Iteration der Wale

$$W_{k+1} = W_k + \rho W_k(K - W_k)$$

aus.

- a) Zeige, dass daraus $W_{k+1} = (1 + \rho K)W_k - \rho W_k^2$ folgt.
- b) Setze $r := 1 + \rho K$ und $x_k := \frac{\rho}{r}W_k$ und zeige, dass daraus die mit dem Parameter r normierte Iterationsgleichung

$$x_{k+1} = r \cdot x_k(1 - x_k)$$

folgt.

In diesem [Video](#), Wale zu logistische Funktion, wird der Inhalt obiger Übung noch einmal erläutert.

1.2.2 Die logistische Funktion

Wir arbeiten also jetzt mit der zugehörigen Funktion

$$f_r(x) = rx(1 - x),$$

welche ich **logistische Funktion** nenne. Diese quadratische Funktion ist grundsätzlich auf ganz \mathbb{R} definiert. Jetzt wird sich das Geschehen eine Weile um diese Funktion f_r drehen.

Übung : Berechne die beiden Nullstellen von f_r und beachte, dass diese unabhängig vom Parameter r sind.

Damit die Funktion “handlicher” wird, schränken wir erstmal den Definitionsbereich \mathbb{D} der freien Variablen x ein, was aber de facto wegen der Isomorphie $\mathbb{R} \simeq (0, 1)$ keiner Einschränkung im mathematischen Sinn gleich kommt. Wählen wir also $\mathbb{D} := [0, 1]$. Nun möchten wir die Wertemenge ebenfalls $\mathbb{W} = [0, 1]$ haben.

Übung : Ein *Isomorphismus* ist vereinfacht gesagt eine bijektive Abbildung zwischen zwei Mengen. Findest du einen Isomorphismus $\varphi : \mathbb{R} \rightarrow (0, 1)$? Kannst du auch die Inversfunktion φ^{-1} angeben?

Übung : Wieso wollen wir $\mathbb{W} = [0, 1]$?

Übung : Für welche Werte von r gilt sicher $\mathbb{W} = [0, 1]$?

Wiederum ist es an der Zeit für etwas Action. Lassen wir einige Orbits zu ausgewählten r s plotten. Wir installieren die logistische Funktion f_r :

```
[10]: def fr(x,r):
        return r*x*(1-x)
```

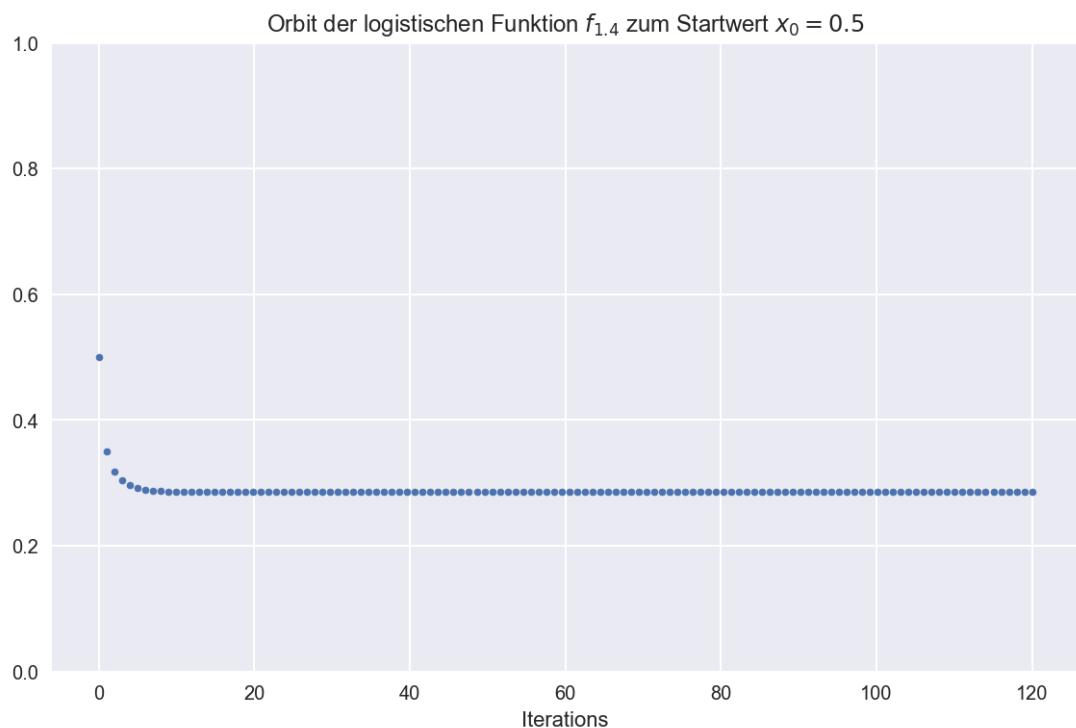
und schreiben eine Funktion, die uns für die logistische Funktion f_r mit Startwert x_0 einen Orbit generiert.

```
[11]: def frorbit(x0=0.5, r=1.4, n=120):
    orbit = []
    orbit.append(x0)
    for k in range(n+1):
        orbit.append(fr(orbit[k],r))
    return orbit
```

```
[12]: orb = frorbit()

plt.scatter(x, orb, marker='.')
plt.xlabel('Iterations')
plt.title(r'Orbit der logistischen Funktion $f_{1.4}$ zum Startwert $x_0=0.5$')

plt.ylim(0,1)
plt.tight_layout()
plt.show()
```



Da die Funktion mit $\mathbb{D} = [0, 1]$ für $0 < r < 4$ uneingeschränkt iteriert werden kann, schauen wir uns ein paar Beispiele an. Unten sind die Plots für $r = 1.7, 2.8, 3.3, 3.7$. Man vergleiche mit dem Walmodell.

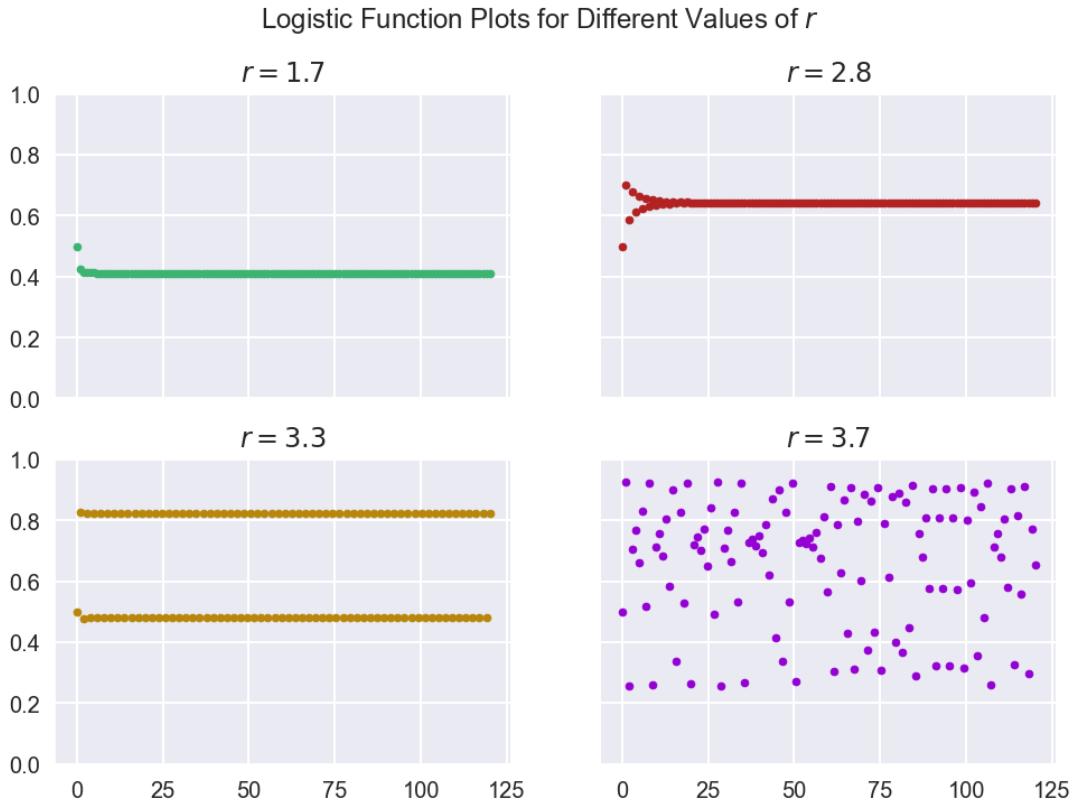
```
[13]: orbit1 = frorbit(0.5, 1.7, 120)
orbit2 = frorbit(0.5, 2.8, 120)
orbit3 = frorbit(0.5, 3.3, 120)
orbit4 = frorbit(0.5, 3.7, 120)
```

```
[14]: fig, axs = plt.subplots(2, 2)
axs[0, 0].scatter(x, orbit1, marker='.', color='mediumseagreen', label='empfänglich')
axs[0, 0].set_title(r'$r=1.7$')
axs[0, 1].scatter(x, orbit2, marker='.', color='firebrick')
axs[0, 1].set_title(r'$r=2.8$')
axs[1, 0].scatter(x, orbit3, marker='.', color='darkgoldenrod')
axs[1, 0].set_title(r'$r=3.3$')
axs[1, 1].scatter(x, orbit4, marker='.', color='darkviolet')
axs[1, 1].set_title(r'$r=3.7$')

fig.suptitle(r'Logistic Function Plots for Different Values of $r$')

axs[0, 0].set_ylim([0, 1])
axs[0, 1].set_ylim([0, 1])
axs[1, 0].set_ylim([0, 1])
axs[1, 1].set_ylim([0, 1])

for ax in axs.flat:
    ax.label_outer()
```



Übung : Welchen r -Werten entsprächen obige Plots den im Walmodell gezeigten Plots?

1.2.3 Fixpunkte von f_r

Übung : Bestimme die Fixpunkte der logistischen Funktion

$$f_r(x) = rx(1 - x).$$

Wir interessieren uns in Kürze über Attraktivität dieser; vorerst noch etwas Geduld.

1.2.4 Werte von r

0 < r < 1 Ich verwende die Funktionen- und die Iterationsschreibweise grad passend für meine Argumentationslinie, welche aber in jedem Fall mit beiden Ansätzen vollzogen werden kann. Dieser Fall, wenn $0 < r < 1$ ist, lässt sich bequem handhaben. Man kann zudem $0 \leq r < 1$ nehmen, da die Fixpunktbedingung über diesen Bereich gelten. Die Iterationsfolge konvergiert für jeden Startwert x_0 , da man x_k nach oben abschätzen kann. Betrachte:

$$x_k = rx_{k-1}(1 - x_{k-1}) < rx_{k-1} < r^2x_{k-2} < \cdots < r^k x_0$$

was wegen $0 \leq r < 1$ klar gegen 0 geht:

$$\lim_{k \rightarrow \infty} r^k x_0 = 0.$$

Also konvergiert die Iterationsfolge für diese r für jeden Startwert x_0 gegen den Fixpunkt 0.

Übung : Bestimme die Bereiche für r , für welche die beiden Fixpunkte (nota bene 1. Ordnung) attraktiv sind.

1< r < 3 Man könnte jetzt unmittelbar weiterstöbern. Ich möchte aber kurz einen Einschub vorziehen, den wir eigentlich bereits zu Beginn absolviert haben. Also entweder als Recap oder jetzt zum Ersten...

1.2.5 Fixpunktlemma

Es gilt:

Seien $f : \mathbb{D} \rightarrow \mathbb{W}$ mit $\mathbb{W} \subset \mathbb{D} \subset \mathbb{R}$, sowie $x_0 \in \mathbb{D}$ und $x_k = f(x_{k-1}) \quad \forall k \in \mathbb{N}$. Wenn die Iterationsfolge $\langle x_k \rangle$ gegen x_p konvergiert, dann ist x_p ein Fixpunkt von f .

Der Beweis ist nahezu trivial, wie es auch der Satz erscheint. Konvergiere $\langle x_k \rangle$ gegen x_p , d.h. $\lim_{k \rightarrow \infty} x_k = x_p$, dann folgt

$$x_p = \lim_{k \rightarrow \infty} x_k = \lim_{k \rightarrow \infty} f(x_{k-1}) = f(x_p).$$

So trivial der Satz scheint, jedoch gilt die Umkehrung im Allgemeinen nicht!

Übung : Finde ein Gegenbeispiel, in dem die Umkehrung nicht gilt.

Da wir uns aber für die Umkehrung — oder zumindest eine abgeschwächte Form davon — interessieren, sind wir etwas vorsichtig. Wir formulieren folgenden Satz.

Sei $I \subset \mathbb{D}$ ein Intervall und f habe in I genau einen Fixpunkt x_p . Ferner gebe es ein M mit $0 < M < 1$ für das $|f'(x)| < M \quad \forall x \in I$.

Dann gilt für jede Iteration mit Startwert $x_0 \in I$:

Die Folge $\langle x_k \rangle$ konvergiert und der Grenzwert ist ihr Fixpunkt x_p .

Der Beweis wurde bereits zu Beginn der Iterationen einmal illustriert.

Übung : Recap des Beweises. Er setzt sich aus dem Fixpunktsatz von Hahn-Banach mit Hilfe des Mittelwertsatzes zusammen.

Wenn wir uns für Startwerte x_0 interessieren, welche anziehend sind, für welchen Bereich lassen wir dann Startwerte zu, wenn wir den obigen Satz berücksichtigen?

Übung : Zeige, dass wir die Startwerte im Bereich $(\frac{r-1}{2r}, \frac{r+1}{2r})$ haben wollen.

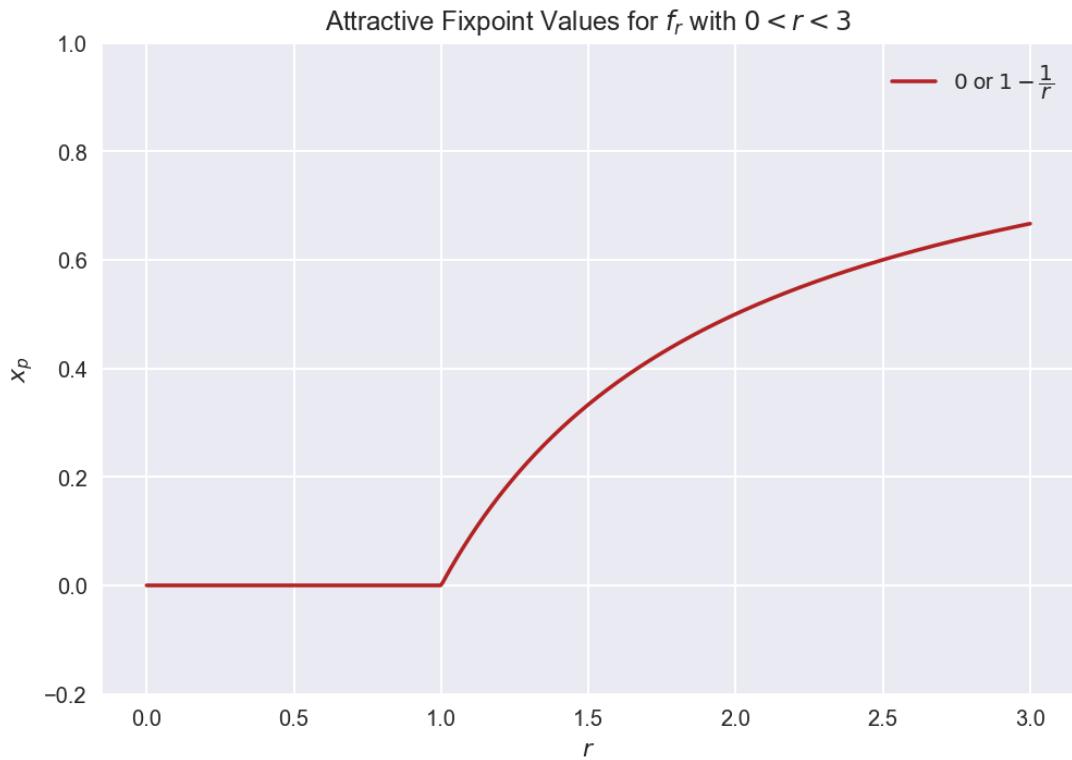
1.2.6 Attraktorwerte von f_r

Übung : Bestätige, dass der Graph der attraktiven Fixpunkte der logistischen Funktion f_r für die Parameterwerte $0 < r < 3$ wie folgt aussehen muss:

```
[15]: import numpy as np

xl = np.linspace(0, 3, 500)
y = np.piecewise(xl, [xl < 1, xl >= 1], [lambda xl: 0, lambda xl: 1-1/xl])

fig = plt.figure()
plt.plot(xl,y, color='firebrick', label=r'$0$ or $1-\frac{1}{r}$')
plt.ylim(-0.2,1)
plt.title(r"Attractive Fixpoint Values for $f_r$ with $0 < r < 3$")
plt.legend()
plt.xlabel(r'$r$')
plt.ylabel(r'$x_p$')
plt.show()
```



Wir haben also noch durch obigen Satz so was wie ein “sicheres” Attraktorintervall

$$I_r := \left(\frac{1}{2} - \frac{1}{2r}, \frac{1}{2} + \frac{1}{2r} \right)$$

erhalten.

Betrachtet man für $0 < r < 1$ den attraktiven Fixpunkt $x_p = 0$, so konvergiert wegen $[0, 1] \subset I_r$ die Iterationsfolge für jeden Startwert x_0 gegen 0. Die Population stirbt immer aus.

```
[16]: orbit5 = frorbit(0.1, 0.3, 120)
orbit6 = frorbit(0.3, 0.5, 120)
orbit7 = frorbit(0.5, 0.7, 120)
orbit8 = frorbit(0.9, 0.7, 120)
```

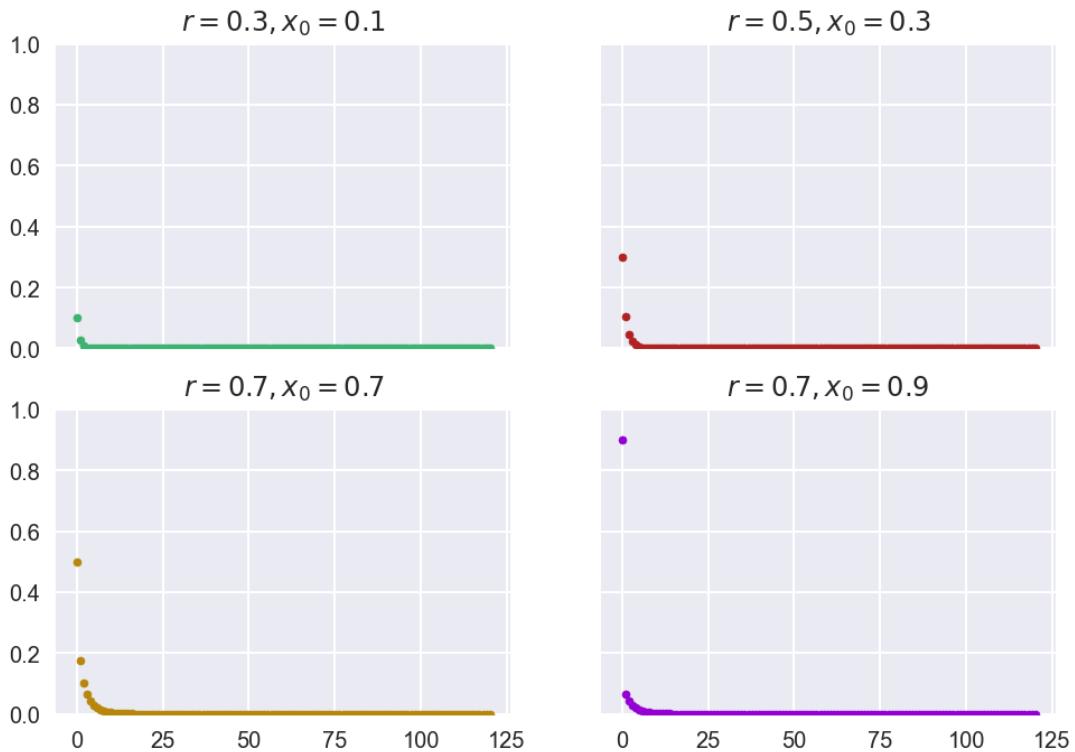
```
[17]: fig, axs = plt.subplots(2, 2)
axs[0, 0].scatter(x, orbit5, marker='.', color='mediumseagreen', label='empfänglich')
axs[0, 0].set_title(r'$r=0.3, x_0=0.1$')
axs[0, 1].scatter(x, orbit6, marker='.', color='firebrick')
axs[0, 1].set_title(r'$r=0.5, x_0=0.3$')
axs[1, 0].scatter(x, orbit7, marker='.', color='darkgoldenrod')
axs[1, 0].set_title(r'$r=0.7, x_0=0.7$')
axs[1, 1].scatter(x, orbit8, marker='.', color='darkviolet')
axs[1, 1].set_title(r'$r=0.7, x_0=0.9$')

fig.suptitle(r'Logistic Function Plots for $0 < r < 1$ and Different Starting Values $x_0$')

axs[0, 0].set_ylim([0, 1])
axs[0, 1].set_ylim([0, 1])
axs[1, 0].set_ylim([0, 1])
axs[1, 1].set_ylim([0, 1])

for ax in axs.flat:
    ax.label_outer()
```

Logistic Function Plots for $0 < r < 1$ and Different Starting Values x_0



Erstaunlich, wie rasch diese Folgen jeweils gegen 0 gehen!

Interessanter wird es jetzt für $1 < r < 3$, da dort der Wert des Fixpunktes für jedes r via $x_p = 1 - \frac{1}{r}$ berechnet wird; also individuell ist. Ich beziehe mich auf das Beispiel $r = 1.4$ weiter oben, von dem wir bereits einen Plot zum Startwert 0.5 haben. Für diesen Fall finden wir also den Fixpunkt bei

```
[18]: from IPython.display import Latex, display, Math
```

```
fpvalue = 1-1/1.4
Latex("$x_p = %.5e" % fpvalue)
```

```
[18]: xp = 2.85714e-01
```

```
[19]: %%latex
\normalfont
```

Für das attraktive Startwertintervall rechnen wir

```
[20]: rvalue = 1.4
dummy = 1/(2*rvalue)
starting = 1/2 - dummy
ending = 1/2 + dummy
```

```

print(rf'I_n = (' + starting + ', ' + ending + ')')

I_n = ( 0.14285714285714285 , 0.8571428571428572 )

[21]: from sympy import *
display(Math("I_n = (" + latex(starting) + "," + latex(ending) + ")"))

I_n = (0.14285714285714285, 0.8571428571428572)

[22]: %%latex
\normalfont

[23]: orbit9 = frorbit(0.1, 1.4, 120)
orbit10 = frorbit(0.5, 1.4, 120)
orbit11 = frorbit(0.7, 1.4, 120)
orbit12 = frorbit(0.9, 1.4, 120)

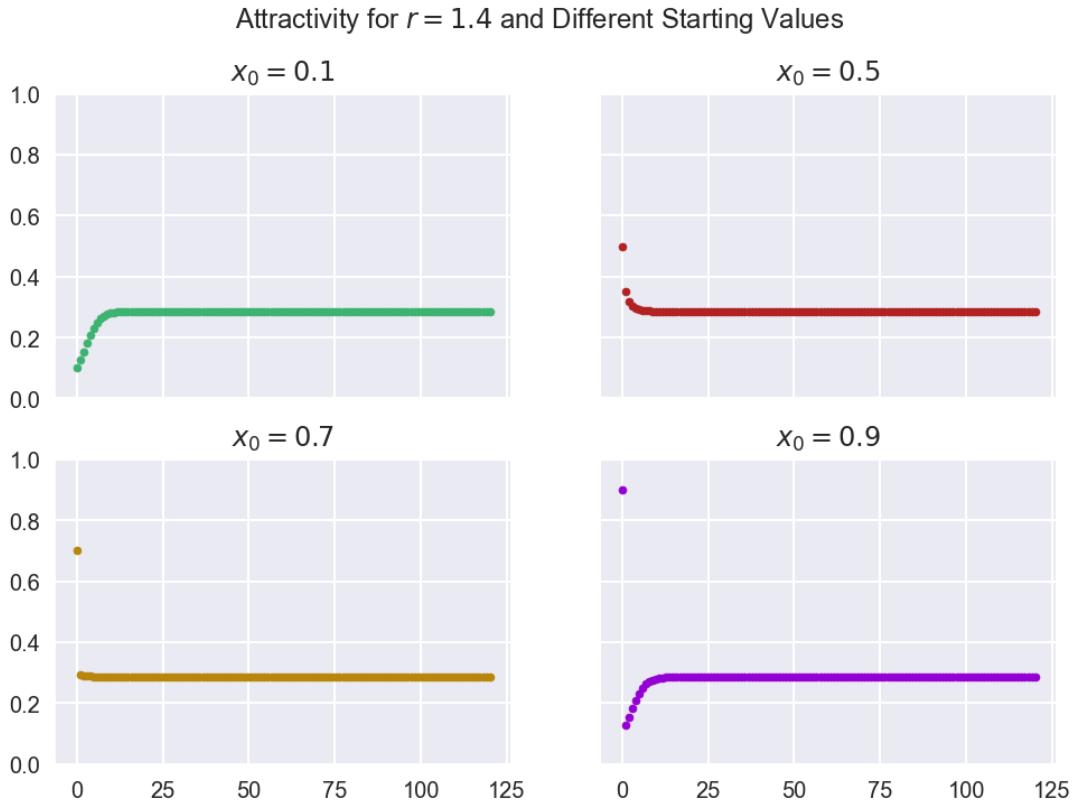
[24]: fig, axs = plt.subplots(2, 2)
axs[0, 0].scatter(x, orbit9, marker='.', color='mediumseagreen', label='empfänglich')
axs[0, 0].set_title(r'$x_0=0.1$')
axs[0, 1].scatter(x, orbit10, marker='.', color='firebrick')
axs[0, 1].set_title(r'$x_0=0.5$')
axs[1, 0].scatter(x, orbit11, marker='.', color='darkgoldenrod')
axs[1, 0].set_title(r'$x_0=0.7$')
axs[1, 1].scatter(x, orbit12, marker='.', color='darkviolet')
axs[1, 1].set_title(r'$x_0=0.9$')

fig.suptitle('Attractivity for $r=1.4$ and Different Starting Values')

axs[0, 0].set_ylim([0, 1])
axs[0, 1].set_ylim([0, 1])
axs[1, 0].set_ylim([0, 1])
axs[1, 1].set_ylim([0, 1])

for ax in axs.flat:
    ax.label_outer()

```



Bemerke: Zweimal sind wir ausserhalb des sicheren Attraktorbereichs gestartet, trotzdem sind wir beim Fixpunkt gelandet. Das bedeutet, dass sich vielleicht dieser Attraktorbereich noch weiter fassen lässt.

1.2.7 Was passiert für $r > 3$?

Klar ist, dass für $r > 3$ die beiden Fixpunkte nicht mehr attraktiv sind; sie sind aber natürlich nach wie vor "da". Wenn man oben die Walplots anschaut, so legt $\rho = 0.98 \cdot 10^{-4}$ den Verdacht nahe, dass Fixpunkte zweiter Ordnung existieren könnten. Daher lösen wir folgende

Übung : Suche Fixpunkte zweiter Ordnung. Verwende Polynomdivision, um die Fixpunkte erster Ordnung aus der Bedingung "rauszudividieren" und so die "reinen" Fixpunkte der Ordnung 2 zu erhalten.

Wir kriegen also nebst $x_{p1} = 0$ und $x_{p2} = 1 - \frac{1}{r}$ nun zusätzlich

$$x_{p3,p4} = \frac{(r+1) \pm \sqrt{(r+1)(r-3)}}{2r}$$

was zusätzlich die Frage aufwirft

Übung : Gibt es einen Bereich für r , in dem diese Fixpunkte der Ordnung 2 stabil sind? Vermutlich ja, wenn man die folgenden Plots anschaut. Dabei habe ich mit den Startwerten x_0 gespielt.

Im Video Fixpunkte von f_r der Periode 2 wird alles erläutert.

```
[25]: orbit13 = frorbit(0.3, 3.1, 120)
orbit14 = frorbit(1-1/3.2, 3.2, 120)
orbit15 = frorbit(1-1/3.29, 3.3, 120)
orbit16 = frorbit(0.3, 3.5, 120)

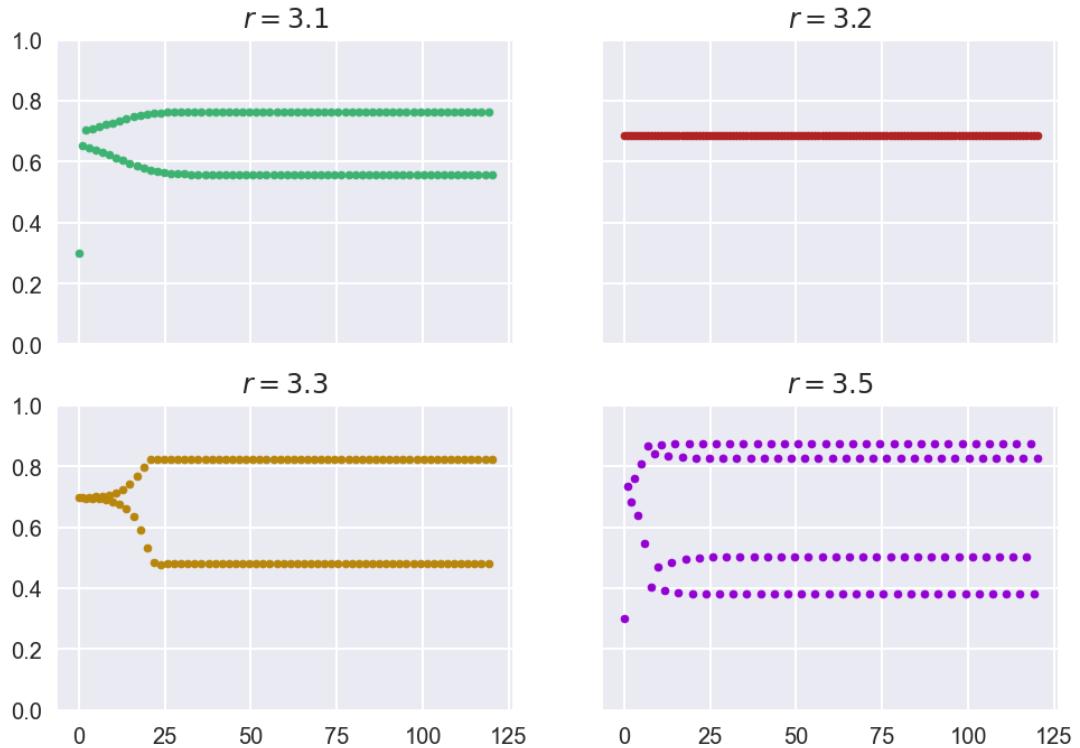
fig, axs = plt.subplots(2, 2)
axs[0, 0].scatter(x, orbit13, marker='.', color='mediumseagreen', ↴
    label='empfänglich')
axs[0, 0].set_title(r'$r=3.1$')
axs[0, 1].scatter(x, orbit14, marker='.', color='firebrick')
axs[0, 1].set_title(r'$r=3.2$')
axs[1, 0].scatter(x, orbit15, marker='.', color='darkgoldenrod')
axs[1, 0].set_title(r'$r=3.3$')
axs[1, 1].scatter(x, orbit16, marker='.', color='darkviolet')
axs[1, 1].set_title(r'$r=3.5$')

fig.suptitle('Fixpoints of 2nd Order and Playing around with Starting Values')

axs[0, 0].set_ylim([0, 1])
axs[0, 1].set_ylim([0, 1])
axs[1, 0].set_ylim([0, 1])
axs[1, 1].set_ylim([0, 1])

for ax in axs.flat:
    ax.label_outer()
```

Fixpoints of 2nd Order and Playing around with Starting Values



Übung : Rechne die Fixpunkt-Werte nach.

Übung : Finde den Attraktivitätsbereich für r für die Fixpunkte zweiter Ordnung.

Weitere Fixpunkte sind nicht mehr lustig zum Rechnen. Man stellt rasch fest, dass der Aufwand von Hand höhere Fixpunkte auszurechnen ins Astronomische steigt.

Ich möchte vor der Fortsetzung noch ein schönes Resultat von Nagashima/Baba aus dem Jahr 1999 vorstellen. Dazu leite ich wieder durch Übungen...

Übung : Zeige: Seien x_1 und x_2 zwei Fixpunkte der Ordnung 2 von f mit $f(x_1) = x_2$ und f gutmütig. Dann gilt für die Ableitung $(f(f(x_1)))' = (f(f(x_2)))'$.

Übung : Mit einer Induktion zeige man nun, dass analog zu oben dieselbe Aussage für Fixpunkte der Periode k gilt. Das heisst: > Alle Fixpunkte eines k Orbits haben dieselbe Steigung, werden also alle gleichzeitig stabil oder instabil.

Übung : Zeige: Wendet man obiges Ergebnis auf die logistische Funktion für die beiden Fixpunkte der Ordnung 2 an, so folgt Attraktivität für $3 < r < 1 + \sqrt{6}$.

Von den Walen zur logistischen Funktion

G. Von der logistischen Gleichung zur Mandelbrotmenge

FP2zuMandelbrot

June 16, 2020

1 Sensibilität von f_r gegenüber den Startwerten

Zuerst gleisen wir wiederum die [logistische Funktion](#) auf und rekapitulieren das Wichtigste.

```
[1]: from numba import jit # just in time compiler
```

```
@jit
def fr(x,r):
    return r*x*(1-x)
```

```
[2]: @jit
def frorbit(x0=0.5, r=1.4, n=120):
    orbit = []
    orbit.append(x0)
    for k in range(n+1):
        orbit.append(fr(orbit[k],r))
    return orbit
```

1.1 Attraktorwerte von f_r

```
[3]: %matplotlib inline
#%matplotlib notebook
%config InlineBackend.figure_format = 'pdf'

import numpy as np
import matplotlib
matplotlib.rcParams['text.usetex'] = True
import matplotlib.pyplot as plt
from numpy import linspace

plt.style.use('seaborn')

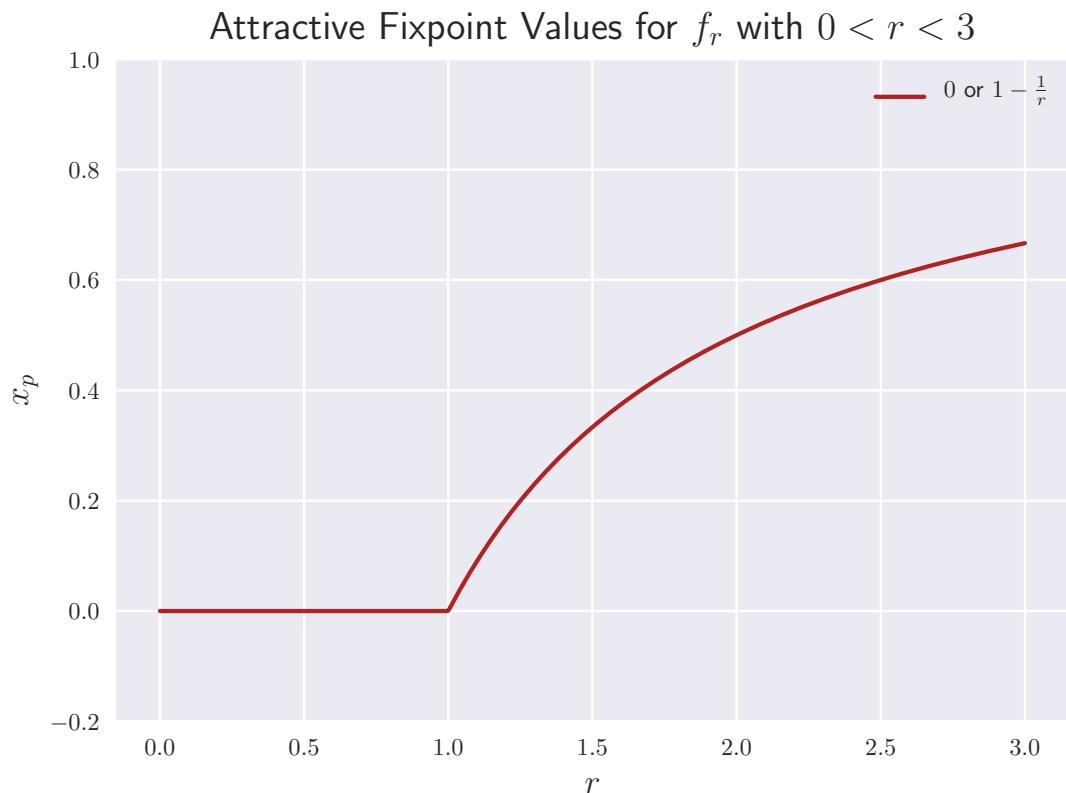
x1 = np.linspace(0, 3, 500)
y = np.piecewise(x1, [x1 < 1, x1 >= 1], [lambda x1: 0, lambda x1: 1-1/x1])

fig = plt.figure()
fig.set_size_inches(7, 5, forward=True)
```

```

plt.plot(xl,y, color='firebrick', label=r'$0$ or $1-\frac{1}{r}$')
plt.ylim(-0.2,1)
plt.title(r'Attractive Fixpoint Values for $f_r$ with $0 < r < 3$', fontsize=16)
plt.legend()
plt.xlabel(r'$r$', fontsize=14)
plt.ylabel(r'$x_p$', fontsize=14)
plt.show()

```



Wir haben also noch durch obigen Satz so was wie ein “sicheres” Attraktorintervall

$$I_r := \left(\frac{1}{2} - \frac{1}{2r}, \frac{1}{2} + \frac{1}{2r} \right)$$

erhalten. Den Attraktorbereich der Fixpunkte der Periode 2 kann man sich auf Wunsch noch mal in Youtube auf gym math zu Gemüte führen: [Attraktoren von \$f_r\$.](#)

[4]:

```

rvalue = 1.4
dummy = 1/(2*rvalue)
starting = 1/2 - dummy
ending = 1/2 + dummy

```

```
print(rf'I_n = (' + starting + ', ' + ending + ')')
fixpoints_ord2_range=1+6**0.5
```

```
I_n = ( 0.14285714285714285 , 0.8571428571428572 )
```

```
[5]: from sympy import *
from IPython.display import Math
display(Math("I_n = (" + latex(starting) + ", " + latex(ending) + ")"))
display(Math(r"1+\sqrt{6} = " + latex(fixpoints_ord2_range)))
```

$$I_n = (0.14285714285714285, 0.8571428571428572)$$

$$1 + \sqrt{6} = 3.449489742783178$$

1.1.1 Was passiert für $r > 3$?

Klar ist, dass für $r > 3$ die beiden Fixpunkte nicht mehr attraktiv sind; sie sind aber natürlich nach wie vor “da”.

Wir kriegen also nebst $x_{p1} = 0$ und $x_{p2} = 1 - \frac{1}{r}$ nun zusätzlich

$$x_{p3,p4} = \frac{(r+1) \pm \sqrt{(r+1)(r-3)}}{2r}$$

und haben bereits gesehen, dass diese Fixpunkte für $3 < r < 3.45$ anziehend sind. Bemerkenswert ist nun, dass die weiteren Verzweigungen (**Bifurkationen**) in immer kürzer werdenden Abständen stattfinden. Und, dass diese verschiedenen oberen Grenzen der Attraktivität der Ordnung k , r_k , selbst einen Grenzwert haben. Es ist:

$$r_\infty := \lim_{k \rightarrow \infty} r_k = 3.569945672 \dots$$

Für $r > r_\infty$ hat man chaotisches Verhalten. Aber selbst in diesen Bereichen gibt es immer wieder “Fenster” der Ordnung. Zudem kann der Verlauf eines Orbits für fixes r äußerst empfindlich gegenüber Anfangsbedingungen sein.

“Leicht” verschiedene Startwerte Im Folgenden soll die Empfindlichkeit des zeitlichen Verlaufs der logistischen Funktion f_r gegenüber nur minim unterschiedlichen Startwerten illustriert werden. Für gewisse r wird nämlich das Verhalten der Iteration so zu sagen “chaotisch”. Man spricht in diesem Kontext auch etwa vom **Schmetterlingseffekt**. Natürlich findet man diesen Effekt nicht für alle $0 < r < 4$, wie wir bereits wissen.

Übung : Google “Schmetterlingseffekt”

Dabei stößt man auch auf dynamische Probleme, welche analytisch anspruchsvoll sind: zum Beispiel das **Doppelpendel**



Betrachte Orbits für $r = 4$:

```
[6]: orbit1 = frorbit(0.1, 4.0, 80)
orbit2 = frorbit(0.1-10**(-15), 4.0, 80)
orbit3 = frorbit(0.1, 4.0, 80)
orbit4 = frorbit(0.1+10**(-17), 4.0, 80)

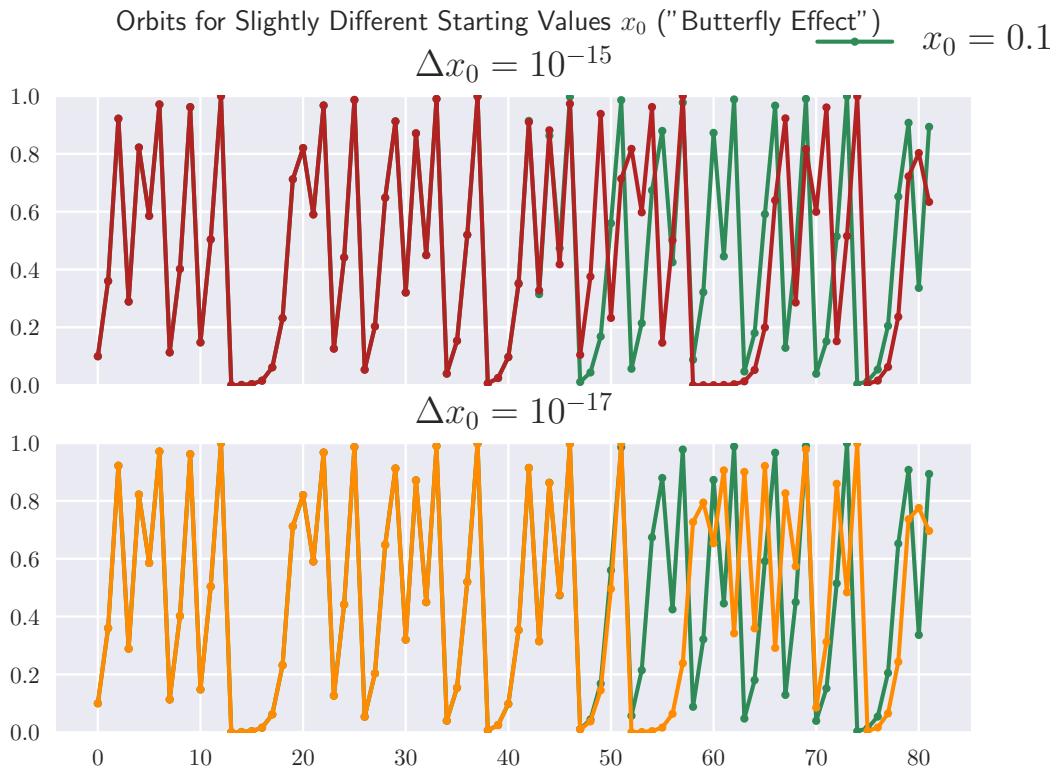
fig, axs = plt.subplots(2)
fig.set_size_inches(7, 5, forward=True)

axs[0].plot(orbit1, marker='.', color='seagreen', label=r'$x_0=0.1$')
axs[0].plot(orbit2, marker='.', color='firebrick')
axs[0].set_title(r'$\Delta x_0=10^{-15}$', fontsize=16)
axs[1].plot(orbit3, marker='.', color='seagreen')
axs[1].plot(orbit4, marker='.', color='darkorange')
axs[1].set_title(r'$\Delta x_0=10^{-17}$', fontsize=16)

fig.suptitle(r'Orbits for Slightly Different Starting Values $x_0$ ("Butterfly Effect")', fontsize=12)

axs[0].set_ylim([0, 1])
axs[1].set_ylim([0, 1])
fig.legend(fontsize=16)

for ax in axs.flat:
    ax.label_outer()
```



```
[7]: from matplotlib import rc
from numba import jit # just in time compiler

#@jit
def plot_cobweb(f, r, x0, nmax=40):

    # Plotte  $y = f_r(x)$  und  $y = x$  für  $0 \leq x \leq 1$ , illustriere für den Startwert  $x_0$ 
    # die Iteration  $x_{k+1} = f_r(x_k)$ .

    x = np.linspace(0, 1, 500)
    fig = plt.figure()
    fig.set_size_inches(7, 7, forward=True)
    fig.suptitle('Graphische Iteration der Logistischen Funktion', fontsize=20)
    ax = fig.add_subplot(111)

    # Plot  $y = f(x)$  and  $y = x$ 
    ax.plot(x, f(x, r), color='#444444', linewidth=2)
    ax.plot(x, x, color='#444444', linestyle='--', linewidth=1.5)

    # Iterate  $x = f_r(x)$  für  $nmax$  steps mit Startwert  $x_0$ .
    px, py = np.empty((2,nmax+1,2))
```

```

px[0], py[0] = x0, 0
for n in range(1, nmax, 2):
    px[n] = px[n-1]
    py[n] = f(px[n-1], r)
    px[n+1] = py[n]
    py[n+1] = py[n]

# Pfad des Orbits plotten
ax.plot(px, py, color='seagreen', linewidth=0.9, alpha=0.7)
ax.plot(x0,x0, color='lime', marker='o', zorder=5)
ax.plot(px[nmax],px[nmax], 'ro', zorder=5)

# Beschriften
ax.minorticks_on()
ax.grid(which='minor', alpha=0.5)
ax.grid(which='major', alpha=0.8)
#ax.set_aspect('equal')
ax.set_xlabel(r'$x$', fontsize=16)
ax.set_ylabel(r'$f_r(x)$', fontsize=16)
ax.set_title(r'$x_0 = {:.1}, r = {:.2}$'.format(x0, r), fontsize=20)

# plt.savefig('cobweb_{:.1}_{:.2}.png'.format(x0, r))

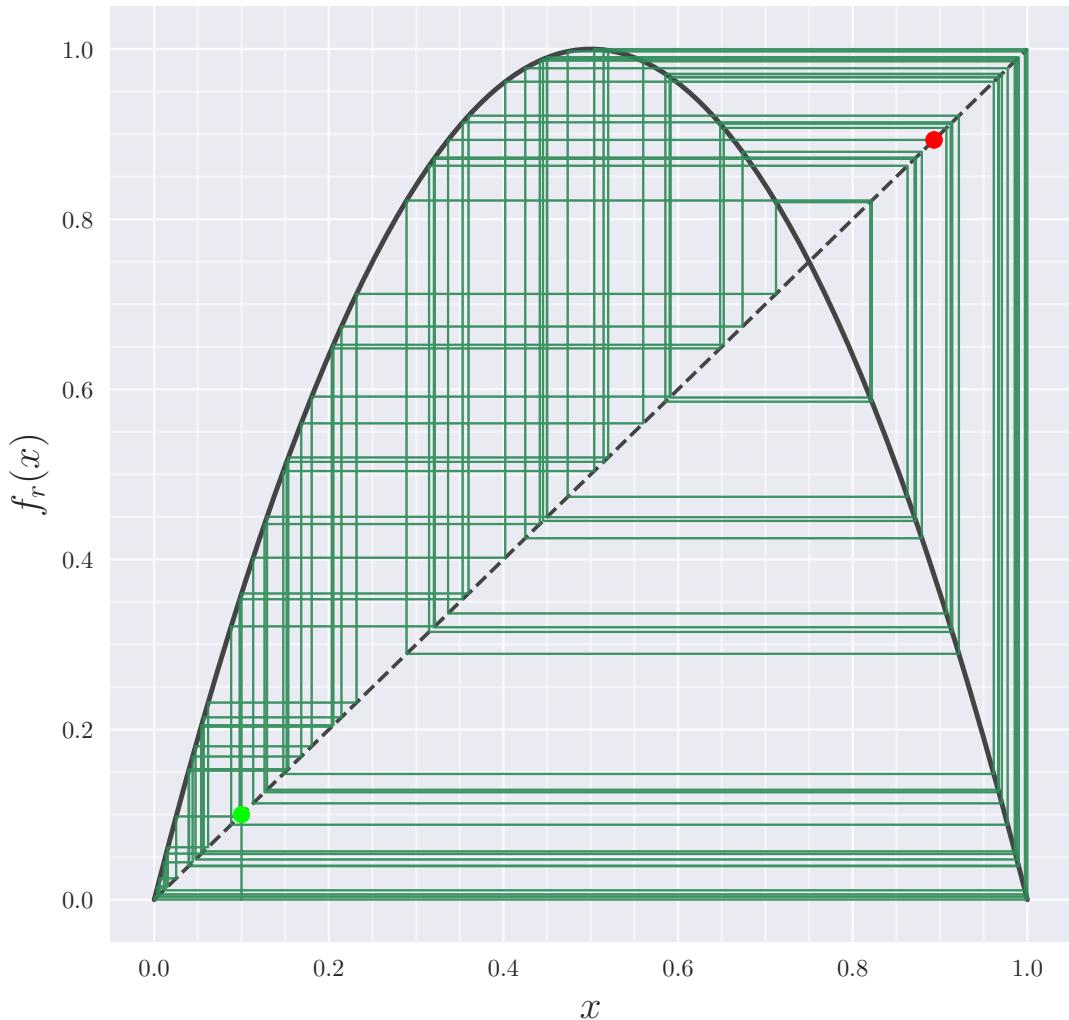
# Logistische Funktion  $f_r(x) = rx(1-x)$ .
func = (lambda x,r: r*x*(1-x))

plot_cobweb(func, 4.0, 0.1, 162)

```

Graphische Iteration der Logistischen Funktion

$$x_0 = 0.1, r = 4.0$$



1.2 Das Feigenbaum-Diagramm

Jetzt möchte ich noch einen Vergleich zwischen den Werten von r und den Fixpunktwerten illustrieren. Wer schon kennt findet das passende Youtube-Video [Feigenbaumdiagramm auf gym math](#). Setzen wir den Plot von oben fort und suchen Fixpunkte höherer Ordnung, so ergibt sich folgendes Bild, das **Feigenbaum-Diagramm** genannt wird:

```
[8]: %load_ext cython
```

[9]: %%cython

```
import numpy as np
cimport numpy as np
cimport cython

@cython.boundscheck(False) # turn off bounds-checking for entire function
@cython.wraparound(False) # turn off negative index wrapping for entire function
def bifurcation(np.int64_t precision=1000, np.int64_t keep=500, np.int64_t num_compute=10000,
                np.float64_t xmin=0, np.float64_t xmax=4, np.float64_t ymin=0, np.float64_t ymax=1):
    """ Acquire bifurcation points for varying mu for logistic map """
    cdef np.ndarray[np.float64_t, ndim=1] mu = np.linspace(xmin, xmax, precision, dtype=np.float64)
    cdef np.float64_t x = 0.5
    cdef np.int64_t i, j, k
    cdef np.ndarray[np.float64_t, ndim=2] points = np.zeros((len(mu) * keep, 2), dtype=np.float64)
    k = 0
    for i in range(len(mu)):
        for j in range(num_compute):
            x = mu[i] * x * (1 - x)
            if j > (num_compute - keep): # we throw away the transient
                points[k, 0] = mu[i]
                points[k, 1] = x
                k += 1
    return points
```

[10]: %%cython

```
import numpy as np
cimport numpy as np

cdef f(np.float64_t mu, np.float64_t x, int n):
    cdef int i
    cdef np.float64_t x0 = x
    for i in range(n):
        x0 = mu * x0 * (1 - x0)
    return x0

def cobweb(np.float64_t mu, int n=1, int num=100, int keep=100, np.float64_t initial = 0.5):
    """ Generate the path for a cobweb diagram """
    cdef np.ndarray[np.float64_t, ndim=2] web = np.zeros((keep, 2))
    cdef np.float64_t x = initial
    cdef np.float64_t y = initial
```

```

cdef int offset = num - keep
cdef int state = 1
if num == keep:
    offset = num - keep + 1
for i in range(1, num):
    if state:
        y = f(mu, x, n)
    else:
        x = y
    state ^= 1
    if i >= offset:
        web[i - offset, 0] = x
        web[i - offset, 1] = y
return web

```

```

[11]: import matplotlib.patches as mpatches

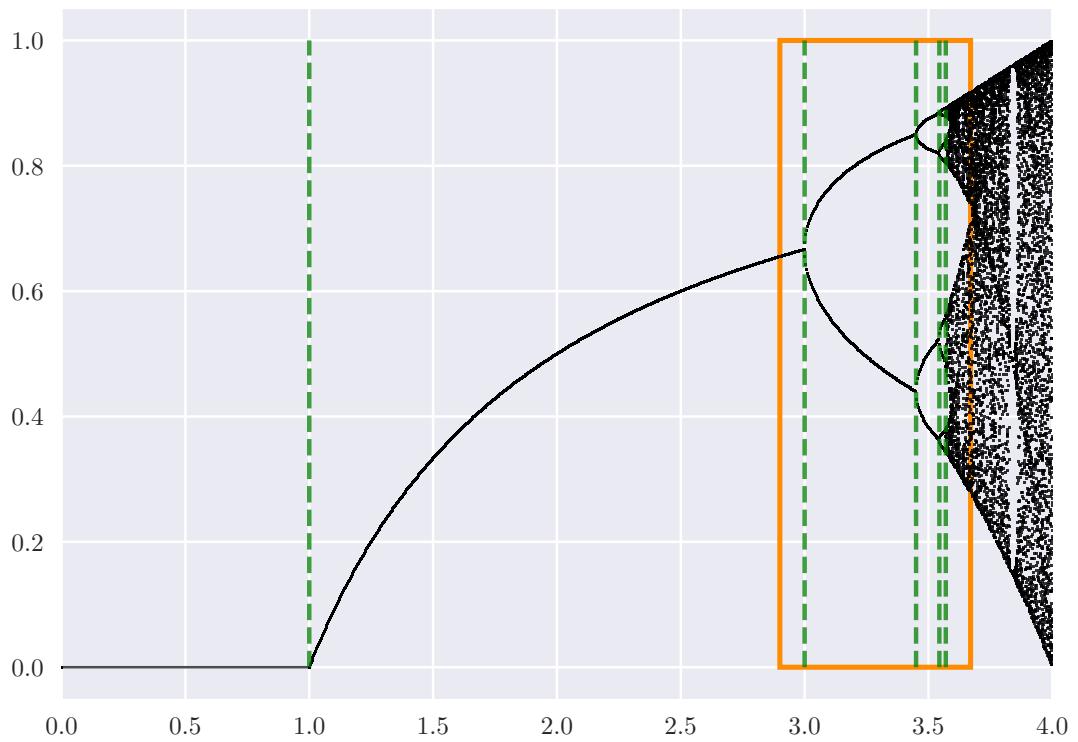
points = bifurcation(xmin=1, xmax=4, precision=800, num_compute=20000, keep=80)
plt.figure(figsize=(7, 5))
plt.plot(points[:, 0], points[:, 1], ',', color='k', alpha=0.8)
plt.plot([0,1],[0,0], color='k', alpha=0.7, linewidth=1)

rectangle = plt.Rectangle((2.9,0.0),3.67-2.9,1.
                           ↪0,linewidth=2,edgecolor='darkorange',facecolor='none')
plt.gca().add_patch(rectangle)

mu_vals = np.array([1, 3, 3.45, 3.544, 3.569945672,])

for mu in mu_vals:
    plt.plot(np.ones(5) * mu, np.linspace(0, 1, 5), color='green', ↪
             ↪linestyle='dashed', alpha=0.75)
plt.xlim(0, 4)
plt.show()

```



Ihr seht hier den kompletten Bereich $0 < r < 4$. Grün habe ich jeweils markante Änderungen im Fixpunktverhalten hervorgehoben. Oben sind das der Übergang vom Attraktor 0 zum Attraktor $1 - \frac{1}{r}$ bei $r = 1$, dann zum 2-Zyklus bei $r = 3$, zum 4-Zyklus bei $r = 1 + \sqrt{6}$ und etwas später zum 8-Zyklus.

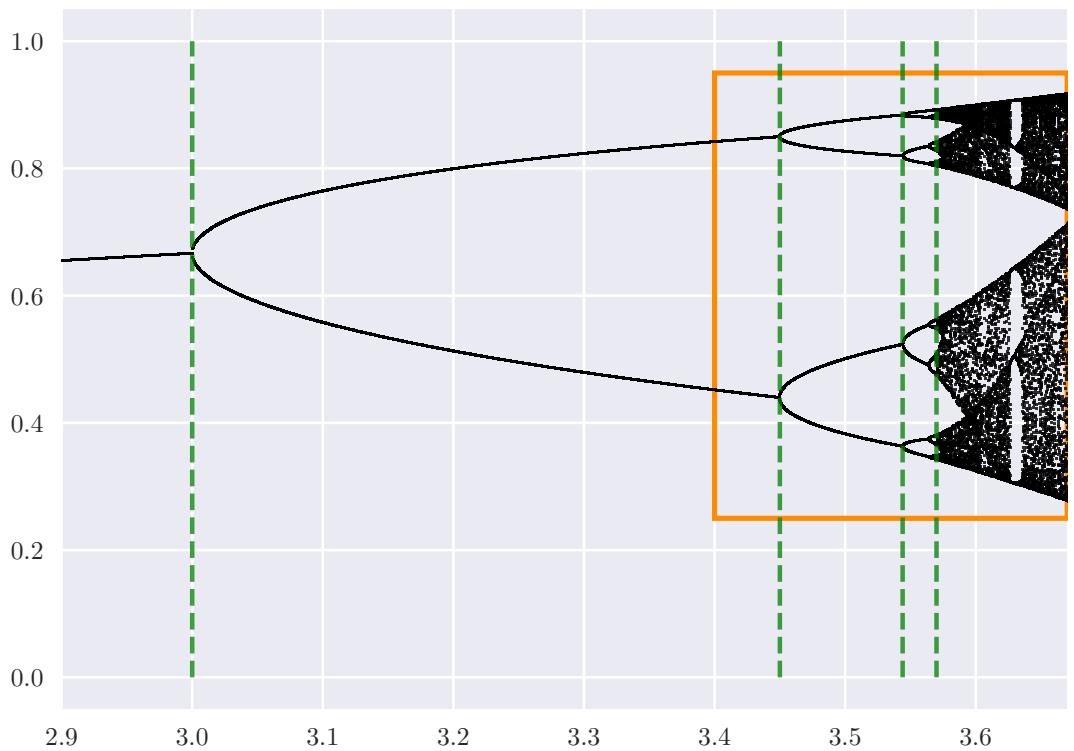
Ferner, falls vom einem zum nächsten Plot ein "Zoom" erfolgt, so hab ich das Fenster in Orange angedeutet. Natürlich kann man sich auch an der Skala orientieren.

```
[12]: points = bifurcation(xmin=1, xmax=4, precision=8000, num_compute=10000, keep=50)
plt.figure(figsize=(7, 5))
plt.plot(points[:, 0], points[:, 1], ',', color='k', alpha=0.8)

mu_vals = np.array([1, 3, 3.45, 3.544, 3.569945672,])

rectangle = plt.Rectangle((3.4, 0.25), 3.67-3.4, 0.95-0.
                           ↪25, linewidth=2, edgecolor='darkorange', facecolor='none')
plt.gca().add_patch(rectangle)

for mu in mu_vals:
    plt.plot(np.ones(5) * mu, np.linspace(0, 1, 5), color='green', ↪
             ↪ linestyle='dashed', alpha=0.75)
plt.xlim(2.9, 3.67)
plt.show()
```

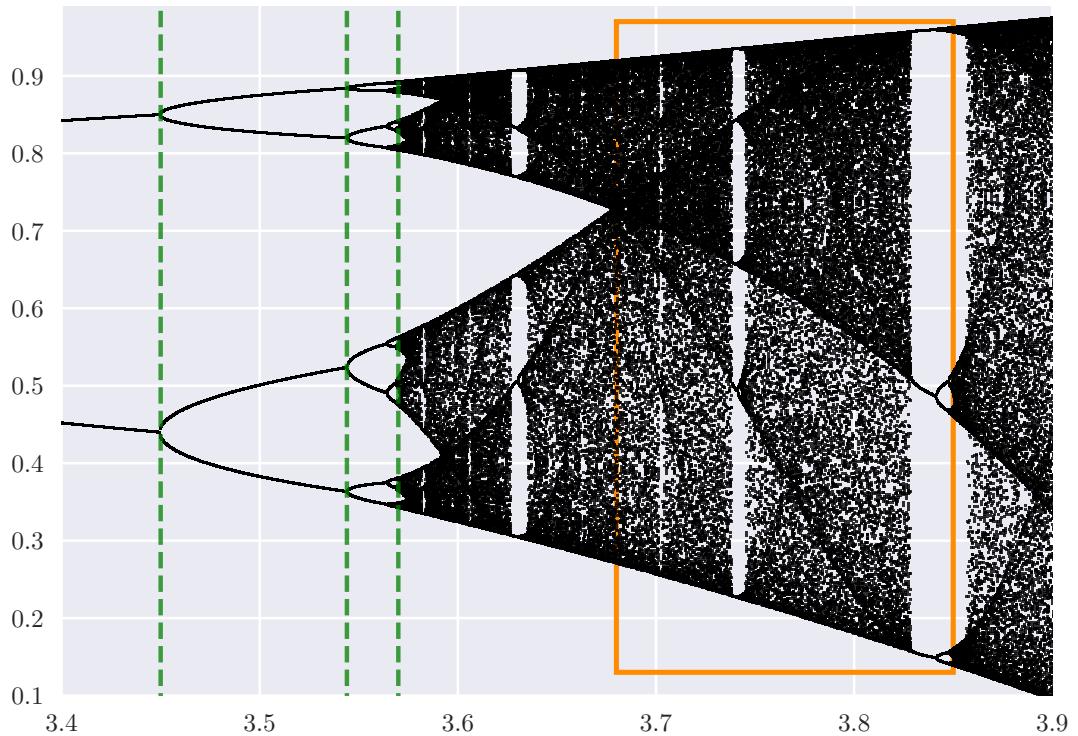


```
[13]: points = bifurcation(xmin=1, xmax=4, precision=10000, num_compute=10000, keep=100)
plt.figure(figsize=(7, 5))
plt.plot(points[:, 0], points[:, 1], 'k', color='k', alpha=0.8)

mu_vals = np.array([1, 3, 3.45, 3.544, 3.569945672,])

rectangle = plt.Rectangle((3.68, 0.13), 3.85 - 3.68, 0.97 - 0.13, linewidth=2, edgecolor='darkorange', facecolor='none')
plt.gca().add_patch(rectangle)

for mu in mu_vals:
    plt.plot(np.ones(5) * mu, np.linspace(0, 1, 5), color='green', linestyle='dashed', alpha=0.75)
plt.xlim(3.4, 3.9)
plt.ylim(0.1, 0.99)
plt.show()
```



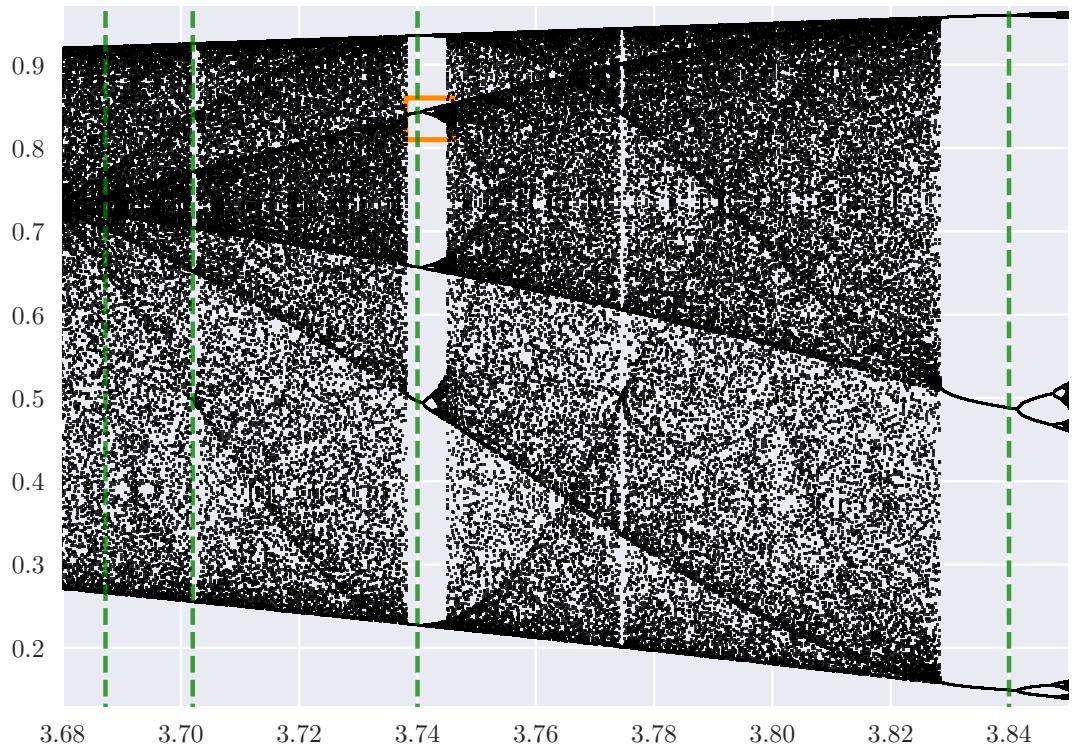
Jetzt setze ich die grünen Marker auf Fenster, in denen man 3-, 5-, 7- und 9-Zyklen beobachten kann.

```
[14]: points = bifurcation(xmin=1, xmax=4, precision=20000, num_compute=20000, ↴
    ↴keep=100)
plt.figure(figsize=(7, 5))
plt.plot(points[:, 0], points[:, 1], ',', color='k', alpha=0.8)

rectangle = plt.Rectangle((3.738,0.81),3.746-3.738,0.86-0.
    ↴81,linewidth=2,edgecolor='darkorange',facecolor='none')
plt.gca().add_patch(rectangle)

mu_vals = np.array([3.84, 3.74, 3.702, 3.68725])

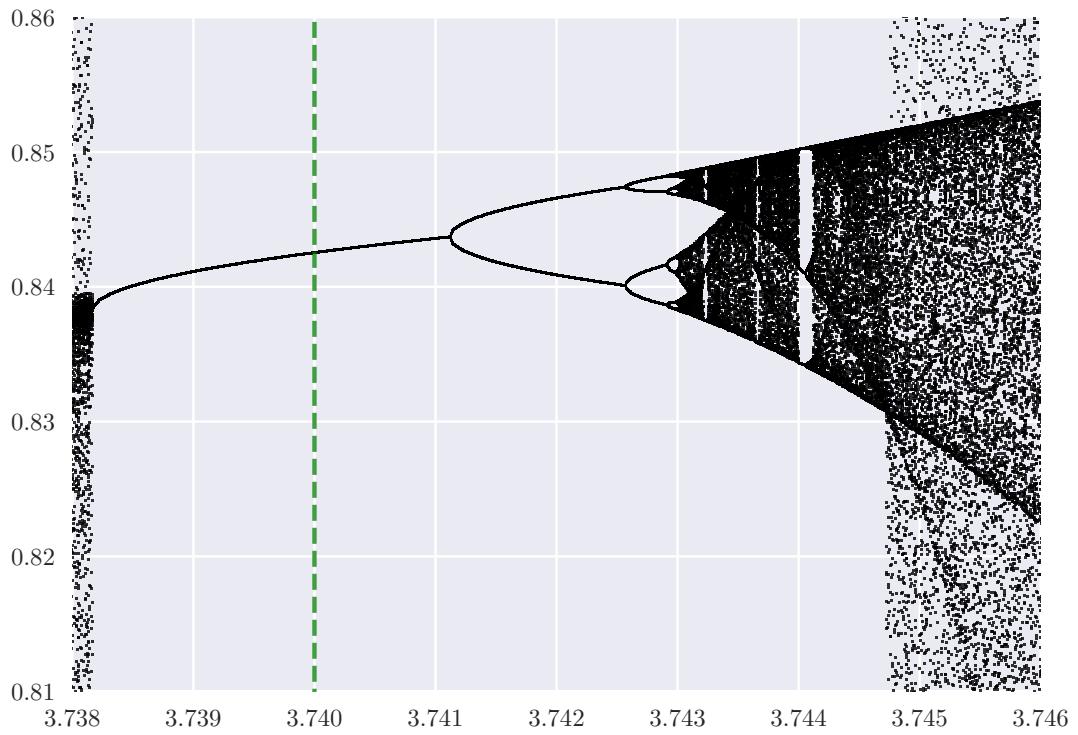
for mu in mu_vals:
    plt.plot(np.ones(5) * mu, np.linspace(0, 1, 5), color='green', ↴
        ↴linestyle='dashed', alpha=0.75)
plt.xlim(3.68, 3.85)
plt.ylim(0.13, 0.97)
plt.show()
```



```
[15]: points = bifurcation(xmin=1, xmax=4, precision=400000, num_compute=20000, ↴
    ↴keep=400)
plt.figure(figsize=(7, 5))
plt.plot(points[:, 0], points[:, 1], 'k', color='k', alpha=0.8)

mu_vals = np.array([3.84, 3.74, 3.702, 3.68725])

for mu in mu_vals:
    plt.plot(np.ones(5) * mu, np.linspace(0, 1, 5), color='green', ↴
    ↴linestyle='dashed', alpha=0.75)
plt.xlim(3.738, 3.746)
plt.ylim(0.81, 0.86)
plt.show()
```



Phänomenal! Da haben wir eine kleine Kopie — zumindest siehts so aus — des “grossen” Feigenbaum-Diagramms! Man spricht in diesem Zusammenhang etwa auch von **Selbstähnlichkeit**. Dazu mehr im nächsten Kapitel Section 2.

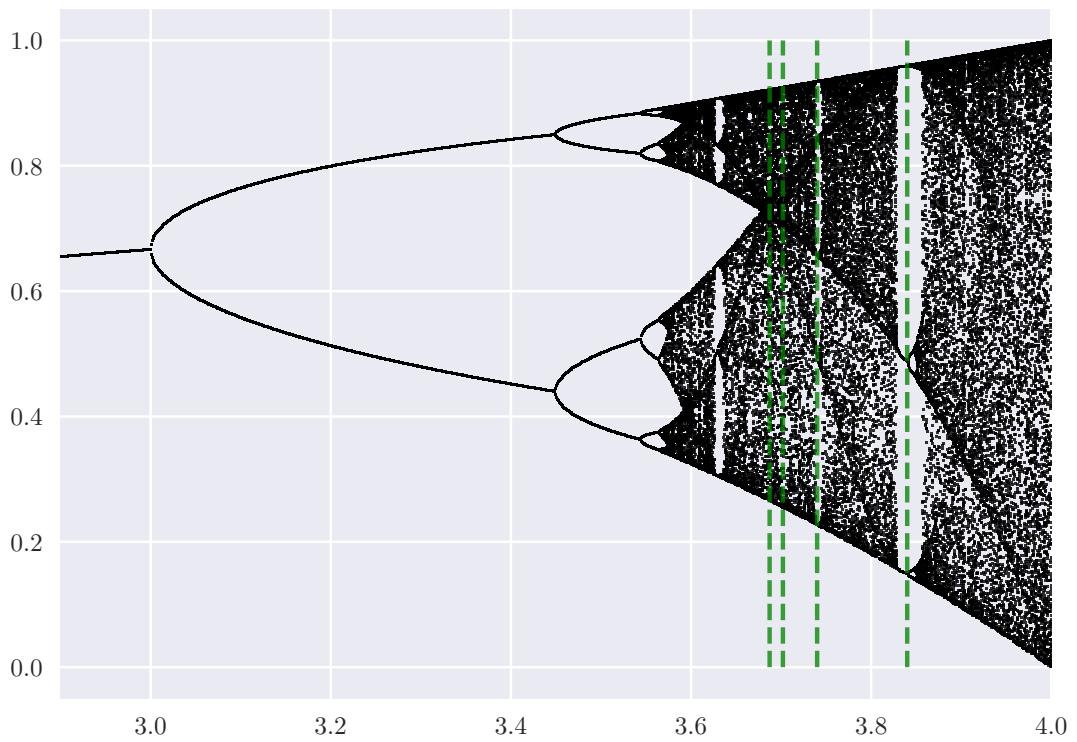
1.2.1 Ungerade Zyklen

Zu Beginn hatte man den Eindruck, es würden nur Zyklen der Ordnung 2^k existieren. Es gibt aber wie oben gesehen auch ungerade Zyklen, zum Beispiel der Ordnung 3,5,7 oder 9. Die sind im Folgenden im Überblick mit einem Cobweb Diagramm illustriert.

```
[16]: points = bifurcation(xmin=1, xmax=4, precision=3000, num_compute=20000, ↴
    ↴keep=100)
plt.figure(figsize=(7, 5))
plt.plot(points[:, 0], points[:, 1], 'k', color='k', alpha=0.8)

mu_vals = np.array([3.84, 3.74, 3.702, 3.68725])

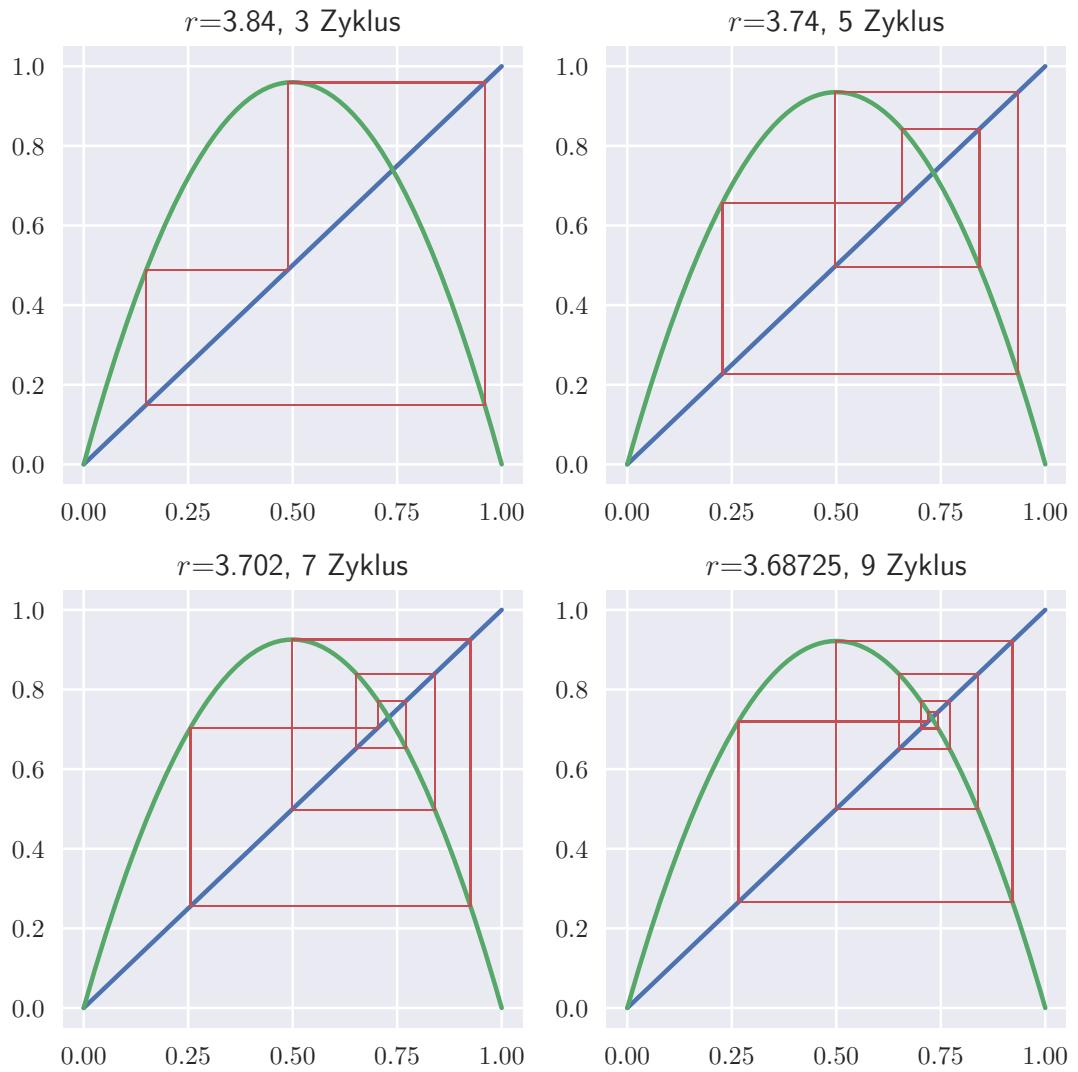
for mu in mu_vals:
    plt.plot(np.ones(4) * mu, np.linspace(0, 1, 4), color='green', ↴
        ↴linestyle='dashed', alpha=0.75)
plt.xlim(2.9, 4)
plt.show()
```



```
[17]: x = np.linspace(0, 1, 5000)
n = 1

def f(x, mu, n):
    x1 = x
    for i in range(n):
        x1 = mu * x1 * (1 - x1)
    return x1

fig, axarr = plt.subplots(2, 2, figsize=(6, 6))
for index, i, j in [(i, int(i / 2), i % 2) for i in range(4)]:
    axarr[i, j].plot(x, x)
    axarr[i, j].plot(x, f(x, mu_vals[index], n))
    web = cobweb(mu_vals[index], n=n, num=5000, keep=1000, initial=0.8)
    axarr[i, j].plot(web[:, 0], web[:, 1], linewidth=0.5)
    axarr[i, j].set_title(r'$r$={}, {} Zyklus'.format(mu_vals[index], (index + 1) * 2 + 1))
plt.tight_layout()
plt.show()
```



2 Fraktale und die Mandelbrotmenge

Ein **Fraktal** ist ...

Aus Wikipedia:

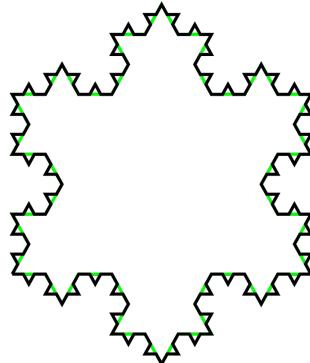
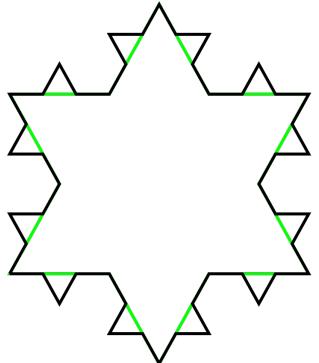
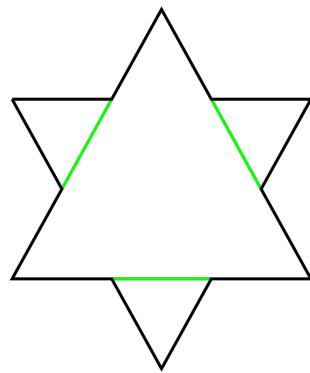
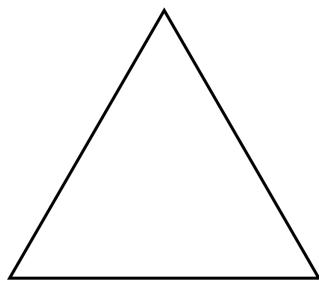
Fraktal ist ein vom Mathematiker Benoît Mandelbrot 1975 geprägter Begriff (lateinisch *fractus* „gebrochen“, von lateinisch *frangere*, (in Stücke zer-),brechen), der bestimmte natürliche oder künstliche Gebilde oder geometrische Muster bezeichnet.

Diese Gebilde oder Muster besitzen im Allgemeinen keine ganzzahlige Hausdorff-Dimension, sondern eine gebrochene – daher der Name – und weisen zudem einen hohen Grad von Skaleninvarianz bzw. Selbstähnlichkeit auf. Das ist beispielsweise der Fall, wenn ein Objekt aus mehreren verkleinerten Kopien seiner selbst besteht. Geometrische

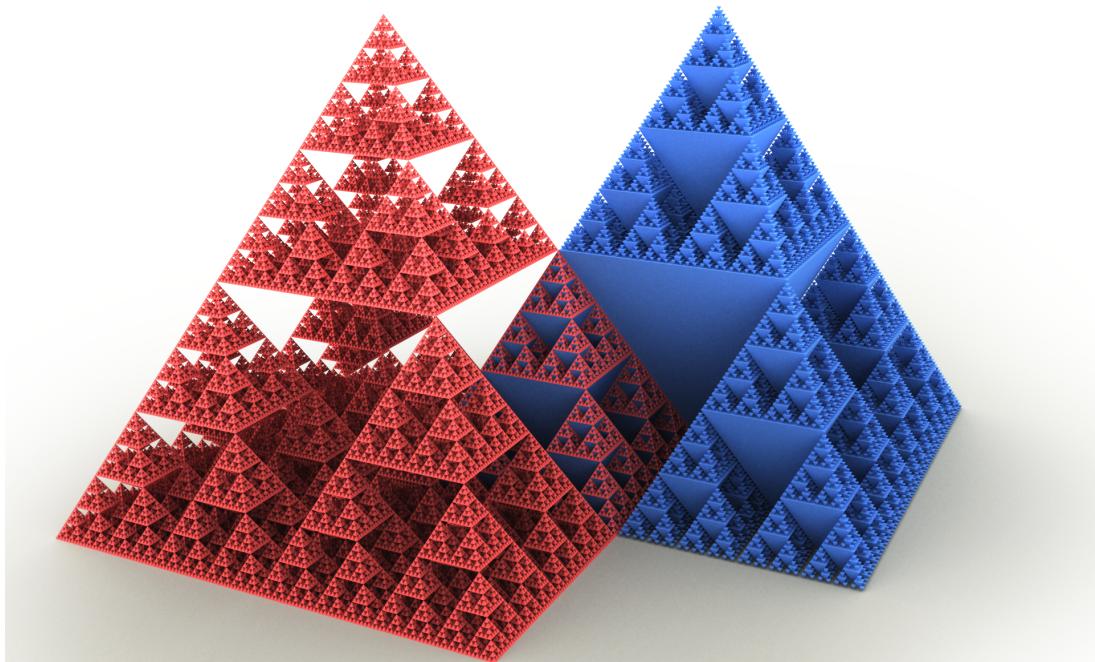
Objekte dieser Art unterscheiden sich in wesentlichen Aspekten von gewöhnlichen glatten Figuren.

Zuerst möchte ich einige Bilder zeigen, bevor wir uns dann sehnlichst auf die Mandelbrotmenge stürzen.

2.1 Die Koch'sche Schneeflocke



2.2 Die Sierpinski Pyramide



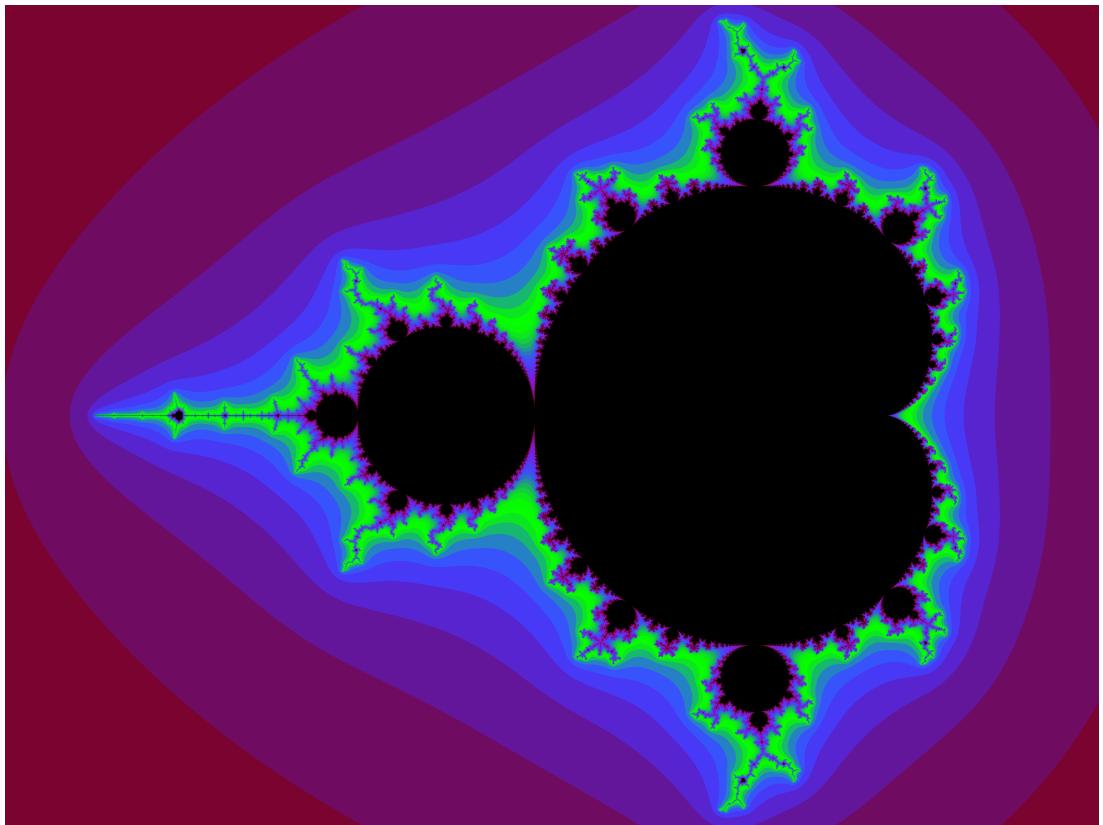
2.3 Romanesco



2.4 Farn



2.5 Mandelbrot Menge



H. Der Kreis schliesst sich

Mandelbrot zu Wale

June 16, 2020

1 Das Apfelmännchen

1.1 Definition

Wir zeichnen zuerst das Apfelmännchen, auch **Mandelbrotmenge** genannt. Wir definieren die Mandelbrotmenge salopp als

Die Menge aller Punkte $c \in \mathbb{C}$, so dass die komplexe Abbildung

$$f(z) = z^2 + c$$

mit Startwert $z_0 = 0$ nicht divergiert.

1.1.1 Beispielorbits

Bevor wir uns überlegen, wie man diese Menge berechnet, wollen wir ihre Form anschauen. Wir berechnen ein paar Orbits für bestimmte Startwerte c .

```
[1]: %matplotlib inline
#%matplotlib notebook
%config InlineBackend.figure_format = 'pdf'

from numba import jit # just in time compiler: @jit

@jit
def mf(z,c) :
    return z*z+c
```

```
[2]: @jit
def make_mfOrbit(c,anzahl_iterationen) :
    cOrbit = [0]
    for k in range(anzahl_iterationen):
        cOrbit.append(mf(cOrbit[k],c))
    return cOrbit
```

```
[3]: make_mfOrbit(1,8)
```

```
[3]: [0, 1, 2, 5, 26, 677, 458330, 210066388901, 3275921592928522330]
```

```
[4]: make_mfOrbit(0.1,5)
[4]: [0.0, 0.1, 0.1100000000000001, 0.1121, 0.11256641, 0.1126711966602881]

[5]: make_mfOrbit(complex(0,1),10)
[5]: [0j, 1j, (-1+1j), -1j, (-1+1j), -1j, (-1+1j), -1j, (-1+1j), -1j, (-1+1j)]

[6]: make_mfOrbit(complex(0,-1),10)
[6]: [0j, -1j, (-1-1j), 1j, (-1-1j), 1j, (-1-1j), 1j, (-1-1j), 1j, (-1-1j)]

[7]: make_mfOrbit(complex(1,1),6)
[7]: [0j, (1+1j), (1+3j), (-7+7j), (1-97j), (-9407-193j), (88454401+3631103j)]
```

1.2 Mandelbrot Plots

Jetzt wollen wir die Mandelbrotmenge plotten...

```
[8]: # Hier zählen wir, ob bzw. nach wie vielen Iterationen klar ist, dass der Orbit
      ↴divergiert.

@jit
def iterationen_Bis_Divergent(c=complex(1,0), maxiter=100):
    z = complex(0,0)
    for iteration in range(maxiter):
        z = (z*z) + c

        if abs(z) > 4:
            break
            pass
    pass
    return iteration
```



```
[9]: import numpy as np
# import matplotlib
#matplotlib.rcParams['text.usetex'] = True # set LaTeX Fonts for Plots
import matplotlib.pyplot as plt

# plt.style.use('default')
# plt.style.use('seaborn')
plt.style.use('classic')
# plt.style.use('bmh')

# create atlas, plot mandelbrot set, display set
def mandelbrot(x_fromto=[-2.25, 0.75], y_fromto=[-1.5, -1.5+3], ↴
               maxiter_perpoint=100, resolution=1000):
```

```

# location and size of the atlas rectangle
realAxis = np.linspace(x_fromto[0], x_fromto[1], resolution)
imaginaryAxis = np.linspace(y_fromto[1], y_fromto[0], resolution)
realAxisLen = len(realAxis)
imaginaryAxisLen = len(imaginaryAxis)

# 2-D array to represent mandelbrot atlas
atlas = np.empty((realAxisLen, imaginaryAxisLen))

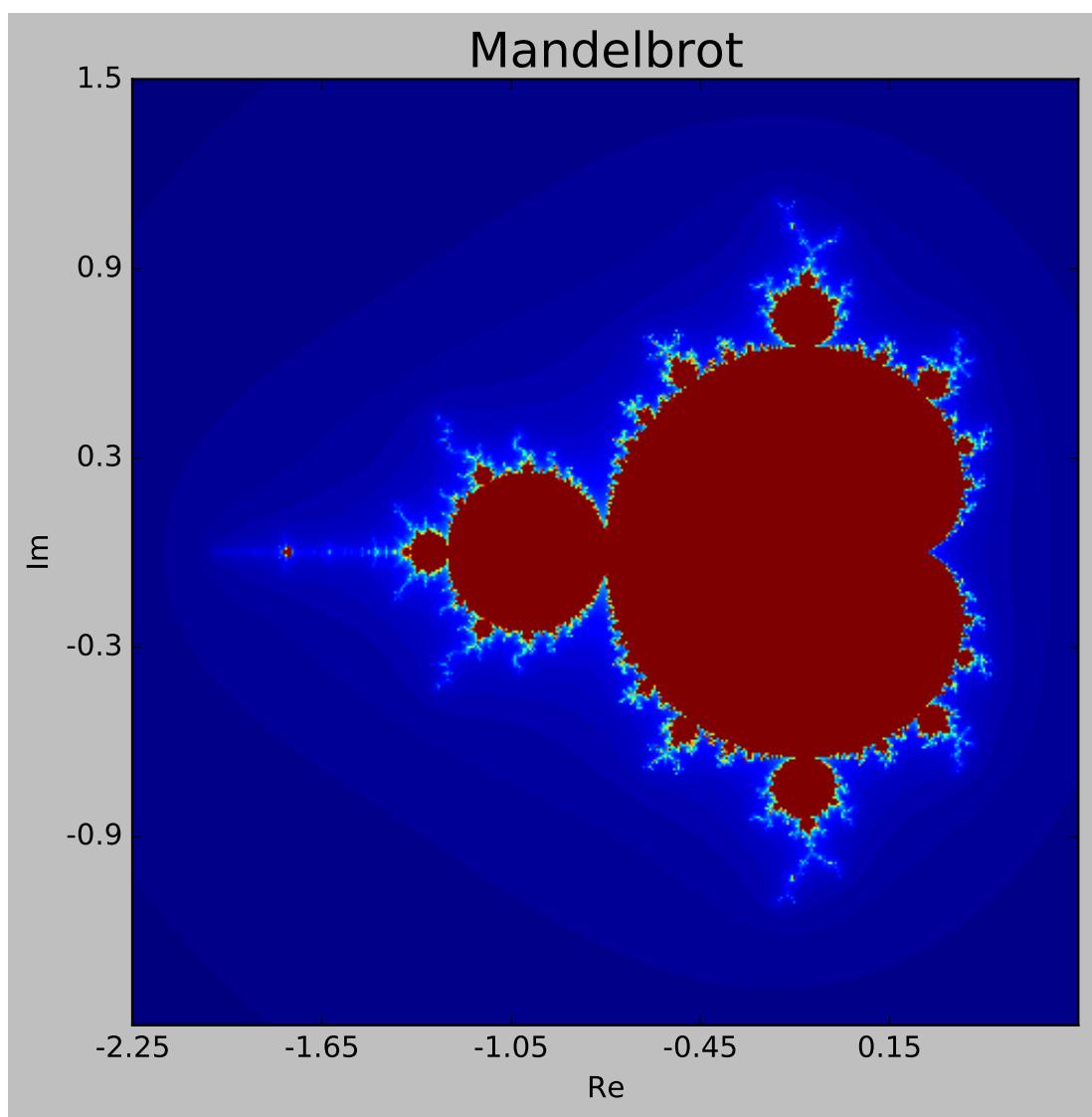
# color each point in the atlas depending on the iteration count
for ix in range(realAxisLen):
    for iy in range(imaginaryAxisLen):
        cx = realAxis[ix]
        cy = imaginaryAxis[iy]
        c = complex(cx, cy)

        atlas[ix, iy] = iterationen_Bis_Divergent(c, maxiter_perpoint)
    pass
pass

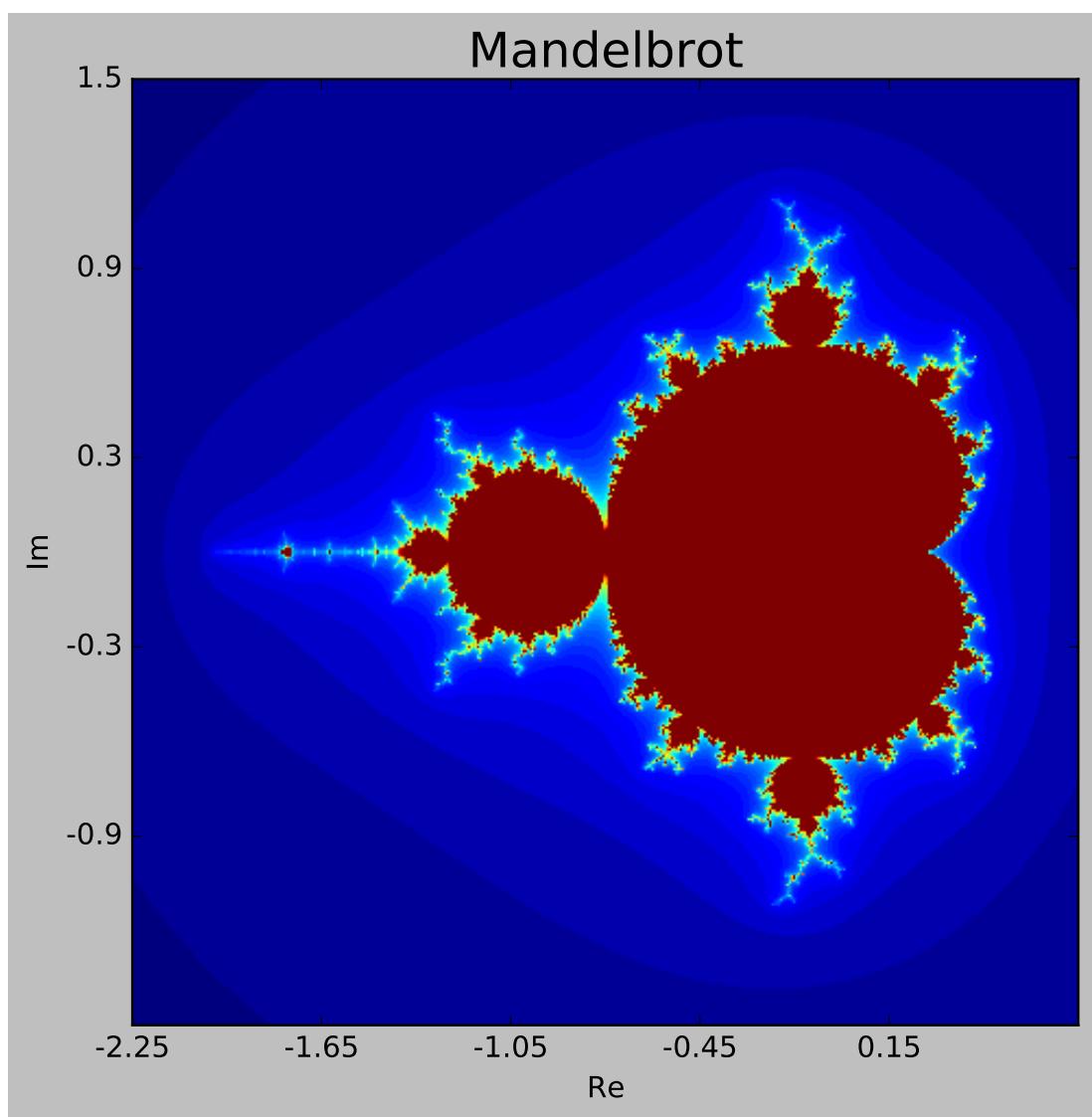
# plot and display mandelbrot set
fig = plt.figure()
fig.set_size_inches(7, 7, forward=True)
ax = fig.add_subplot(111)
#ax = fig.add_axes([0.1, 0.1, 0.8, 0.8]) # main axes
ax.set_xlabel('Re')
ax.set_ylabel('Im')
ax.set_title('Mandelbrot', fontsize=20)
ax.set_xticks([0,1*resolution/5,2*resolution/5,3*resolution/5,4*resolution/
←5,resolution])
    ax.set_xticklabels([x_fromto[0],_
←round(x_fromto[0]+(x_fromto[1]-(x_fromto[0]))*1/5, 5),_
←round(x_fromto[0]+(x_fromto[1]-(x_fromto[0]))*2/5, 5),_
←round(x_fromto[0]+(x_fromto[1]-(x_fromto[0]))*3/5, 5),_
←round(x_fromto[0]+(x_fromto[1]-(x_fromto[0]))*4/5, 5), x_fromto[1]])
    ax.set_yticks([0,1*resolution/5,2*resolution/5,3*resolution/5,4*resolution/
←5,resolution])
    ax.set_yticklabels([y_fromto[1],_
←round(y_fromto[1]+(y_fromto[0]-(y_fromto[1]))*1/5, 5),_
←round(y_fromto[1]+(y_fromto[0]-(y_fromto[1]))*2/5, 5),_
←round(y_fromto[1]+(y_fromto[0]-(y_fromto[1]))*3/5, 5),_
←round(y_fromto[1]+(y_fromto[0]-(y_fromto[1]))*4/5, 5), y_fromto[0]])
plt.imshow(atlas.T, interpolation="nearest")
plt.show()

```

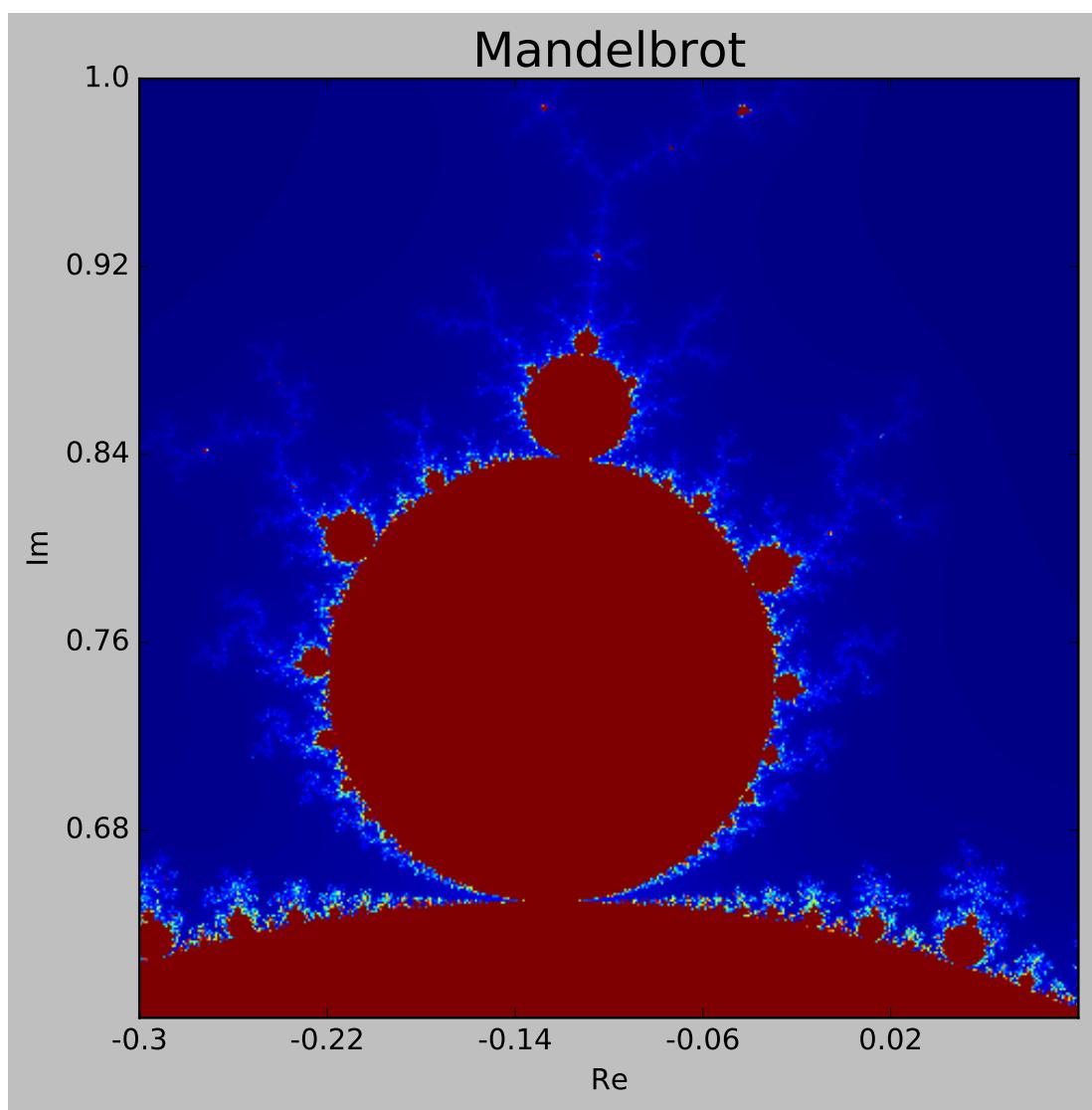
[10]: mandelbrot()



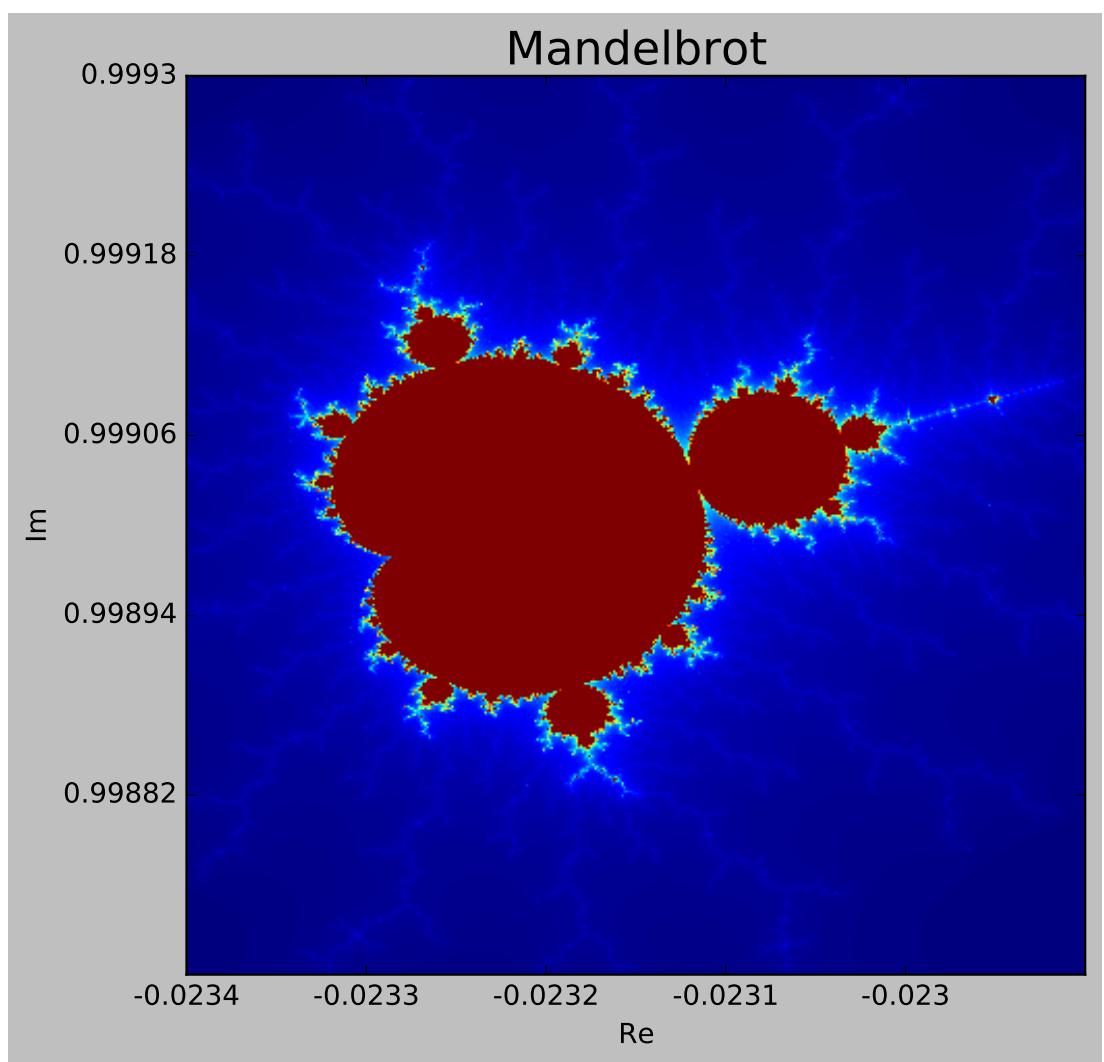
```
[11]: mandelbrot([-2.25, 0.75], [-1.5, -1.5+3], 50, 5120)
```



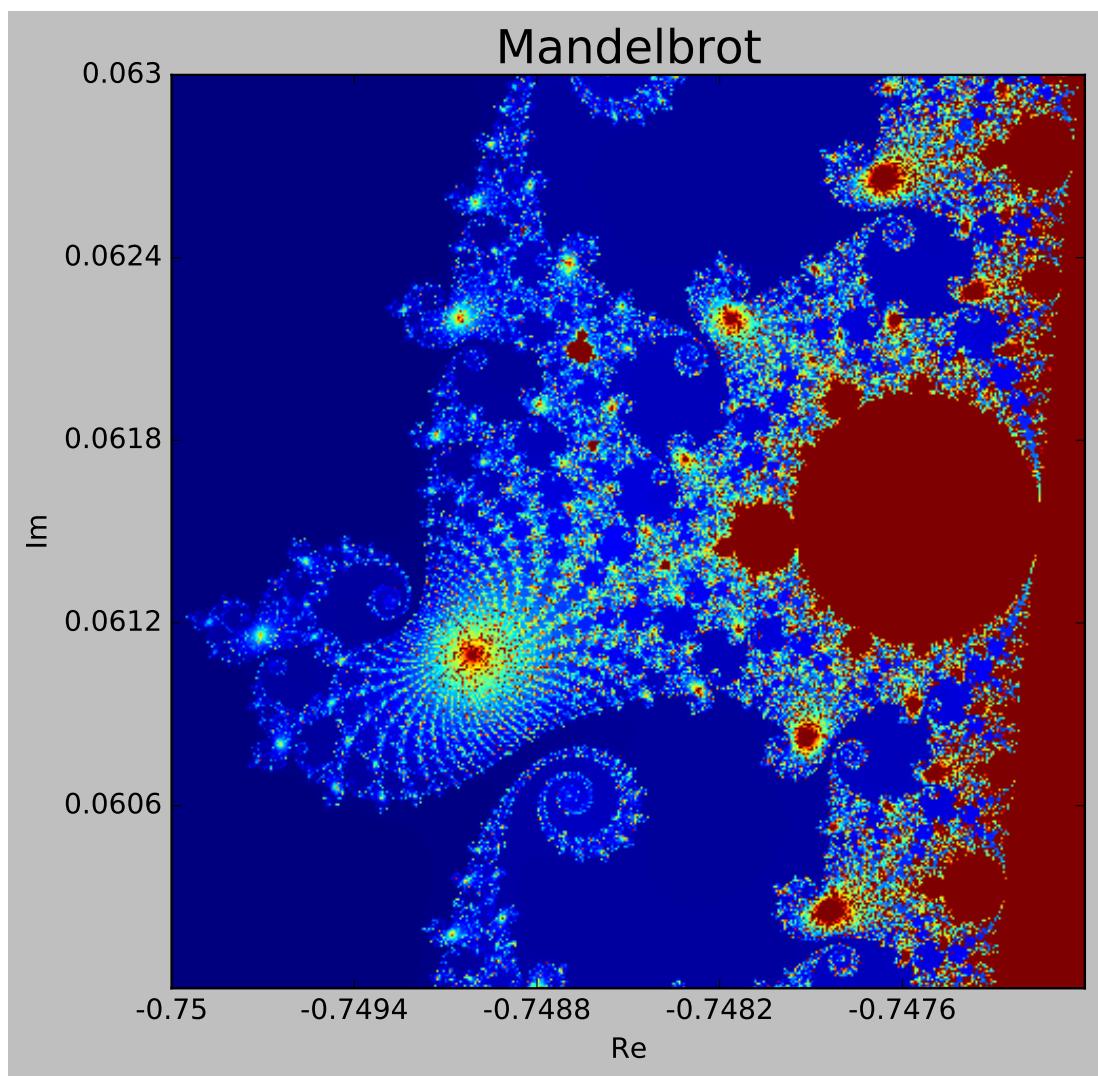
```
[12]: mandelbrot([-0.3, 0.1], [0.6, 1.0], 600, 5120)
```



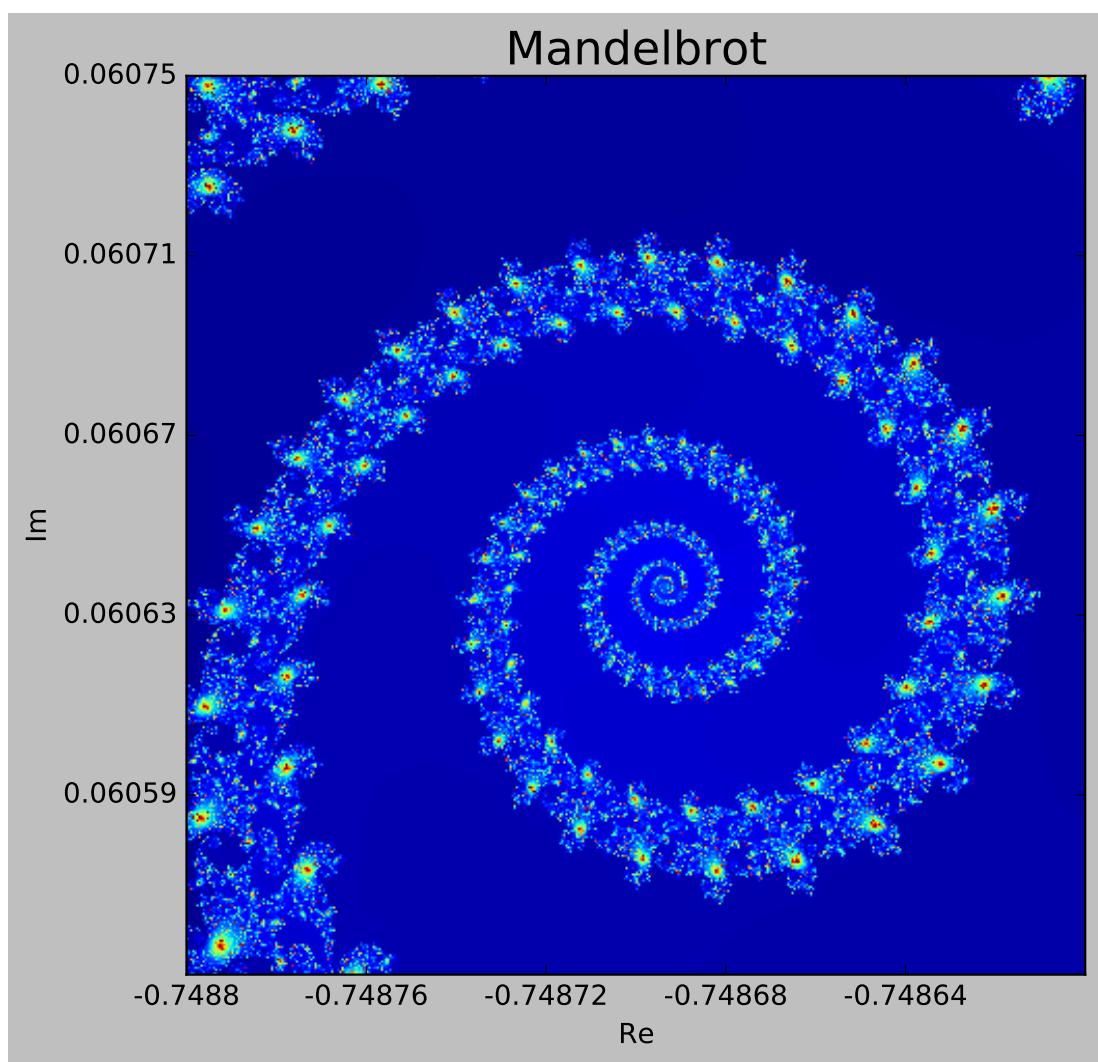
```
[13]: mandelbrot([-0.0234, -0.0229], [0.9987, 0.9993], 600, 5120)
```



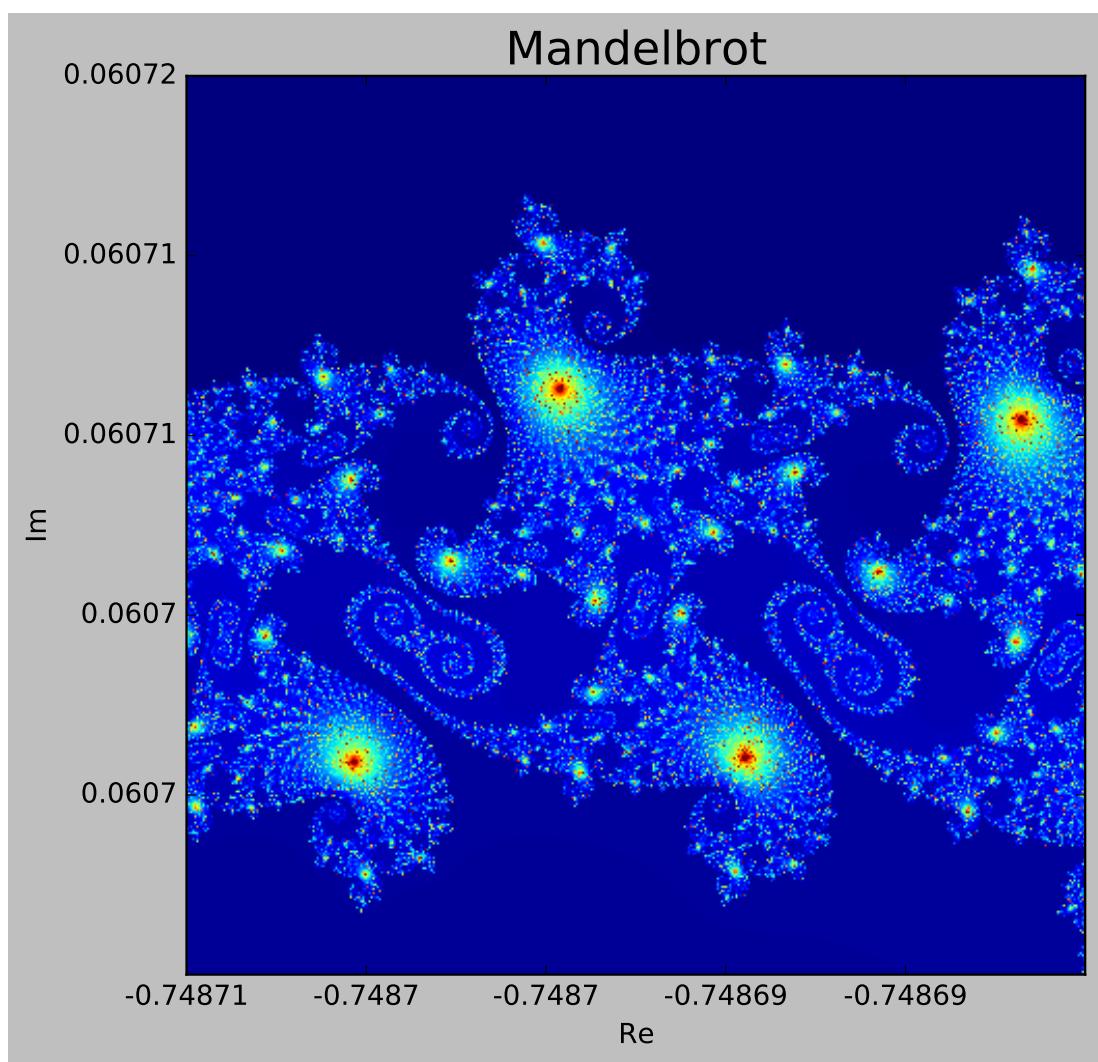
```
[14]: mandelbrot([-0.75, -0.747], [0.06, 0.063], 2000, 5120)
```



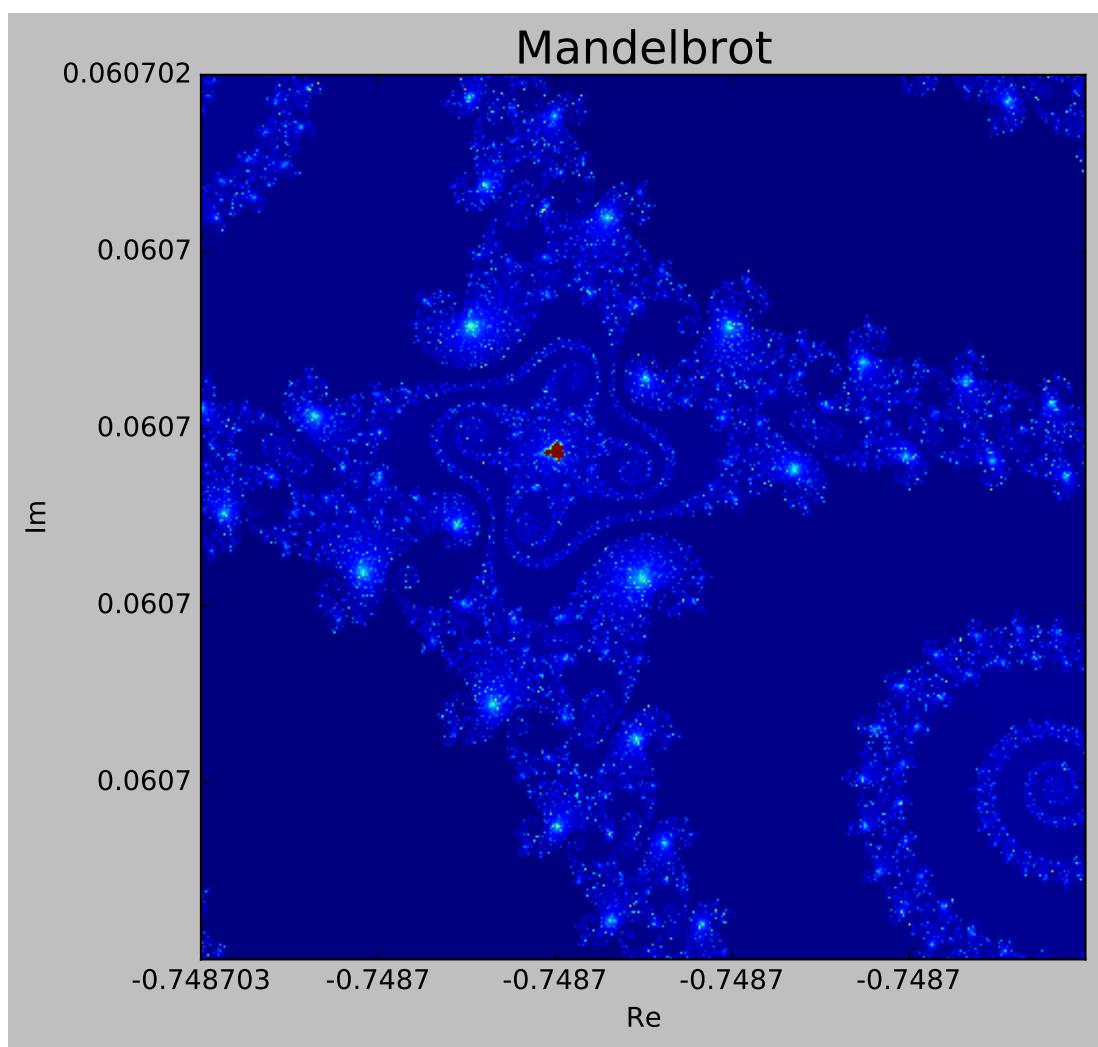
```
[15]: mandelbrot([-0.7488, -0.7486], [0.06055, 0.06075], 2000, 5120)
```



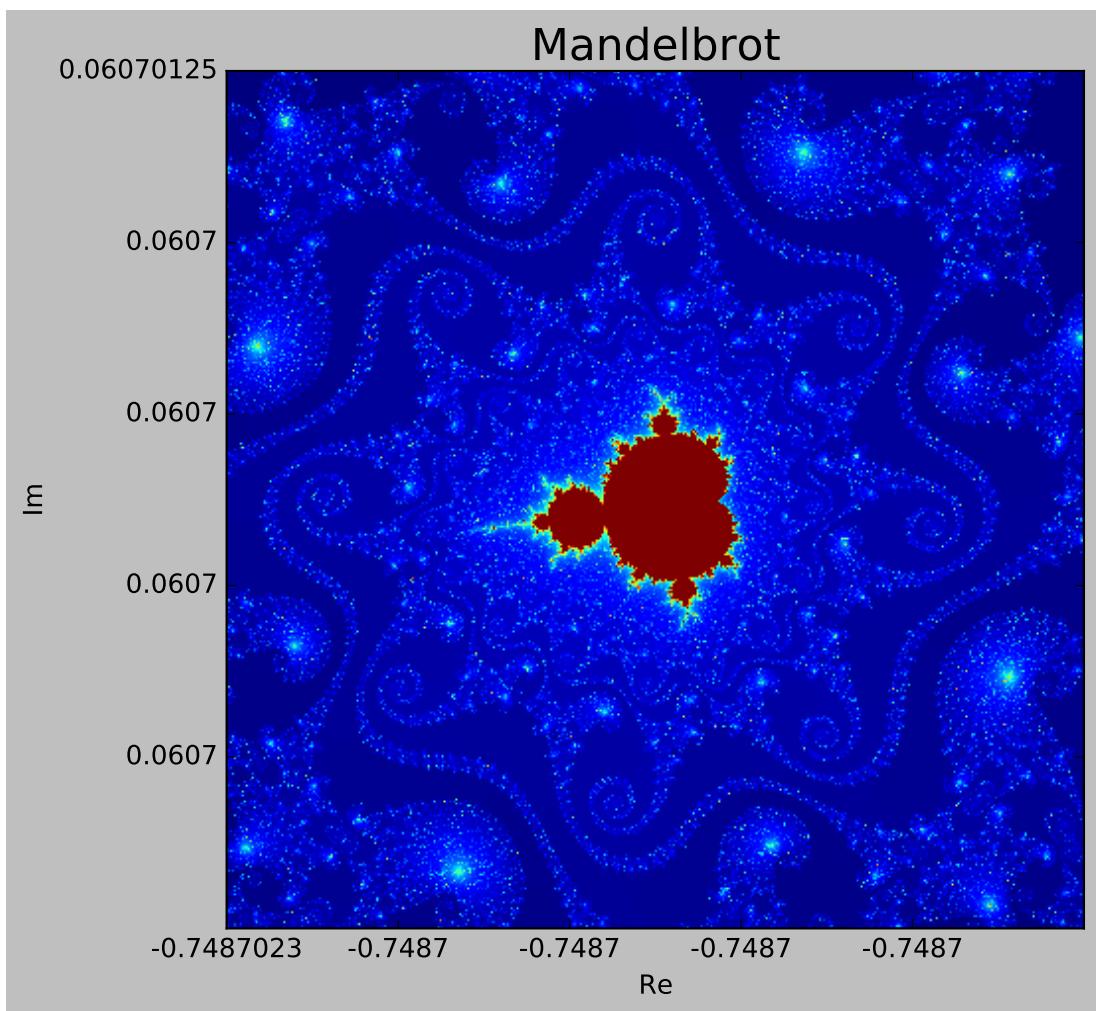
```
[16]: mandelbrot([-0.74871, -0.74868], [0.06069, 0.06072], 2500, 5120)
```



```
[17]: mandelbrot([-0.748703, -0.748701], [0.0607, 0.060702], 6000, 5120)
```



```
[18]: mandelbrot([-0.7487023, -0.7487021], [0.06070105, 0.06070125], 6000, 5120)
```



Übung : Man guckt nach, an welcher Stelle hier jeweils “reingezoomt” wird.

1.3 Wie lässt man solche Bilder berechnen?

Grundsätzlich guckt man, ob für einen Wert $c \in \mathbb{C}$ der Orbit von $f(z) = z^2 + c$ mit Startwert $z = 0$ divergiert. Falls nein, dann wird er Punkt rot eingefärbt. Falls aber ja, dann wird der Punkt mit einer Farbe eingefärbt, die vom Index des Iterationsschrittes abhängt, an dem die Folge als divergent erklärt wird. Stellt sich also die Frage, wann dass eine Folge als divergent eingestuft werden darf?

Wer oben das `mandelbrot`Programm überflogen hat, dem ist aufgefallen, dass als Abbruchbedingung für die Iteration eines Startwerts die Bedingung $|z_k| > 2$ verwendet wurde. Man kann nämlich zeigen, dass, falls für einen Wert z_k in der Itartionsfolge $|z_k| > 2$ gilt, der Orbit sicher divergiert. Diesen Fact macht man sich in der Programmierung zu Nutze.

Ist für ein $k \in \mathbb{N}$ der Betrag $|z_k| > 2$, so divergiert die Folge $\langle z_k \rangle$ definiert durch $z_{k+1} = z_k^2 + c$ mit fixem $c \in \mathbb{C}$ und Startwert $z_0 = 0$.

Beweis : Für $f(z) = z^2 + c$ mit $z_0 = 0$ haben wir

$$z_0 = 0, z_1 = c, z_2 = c^2 + c, \dots$$

Ferner ist

$$\frac{|f(z)|}{|z|} = \frac{|z^2 + c|}{|z|} \geq \frac{|z|^2 - |c|}{|z|} = |z| - \frac{|c|}{|z|} > |z| - 1 > 1$$

wobei die Dreiecksungleichung verwendet wurde. Das heisst wir haben ein Wachstum pro Zeitschritt mit Faktor grösser 1.

Ist nun $|c| < 2$ und für ein k $|z_k| > 2$, dann erfüllt $z = z_k$ die Voraussetzung und die Folge wird divergieren. Ist $|c| > 2$, dann ist $|c^2 + c| \geq |c|^2 - |c| = |c| \cdot |c - 1| > |c| > 2$. Also erfüllt $z = z_2$ die Voraussetzung.

Der [kommentierte Beweis zur Divergenz im Apfelmännchen von gym math](#) gibts unter eben dem Link.

1.4 Was hat das Apfelmännchen mit den Walen zu tun?

Betrachte die Iteration $z_{k+1} = z_k^2 + c$ und setze eine affine Transformation $z_k = ax_k + b$ mit $a, b, x_k \in \mathbb{R}$. Es ist

$$ax_{k+1} + b = (ax_k + b)^2 + c = a^2 x_k^2 + 2abx_k + b^2 + c$$

Es folgt

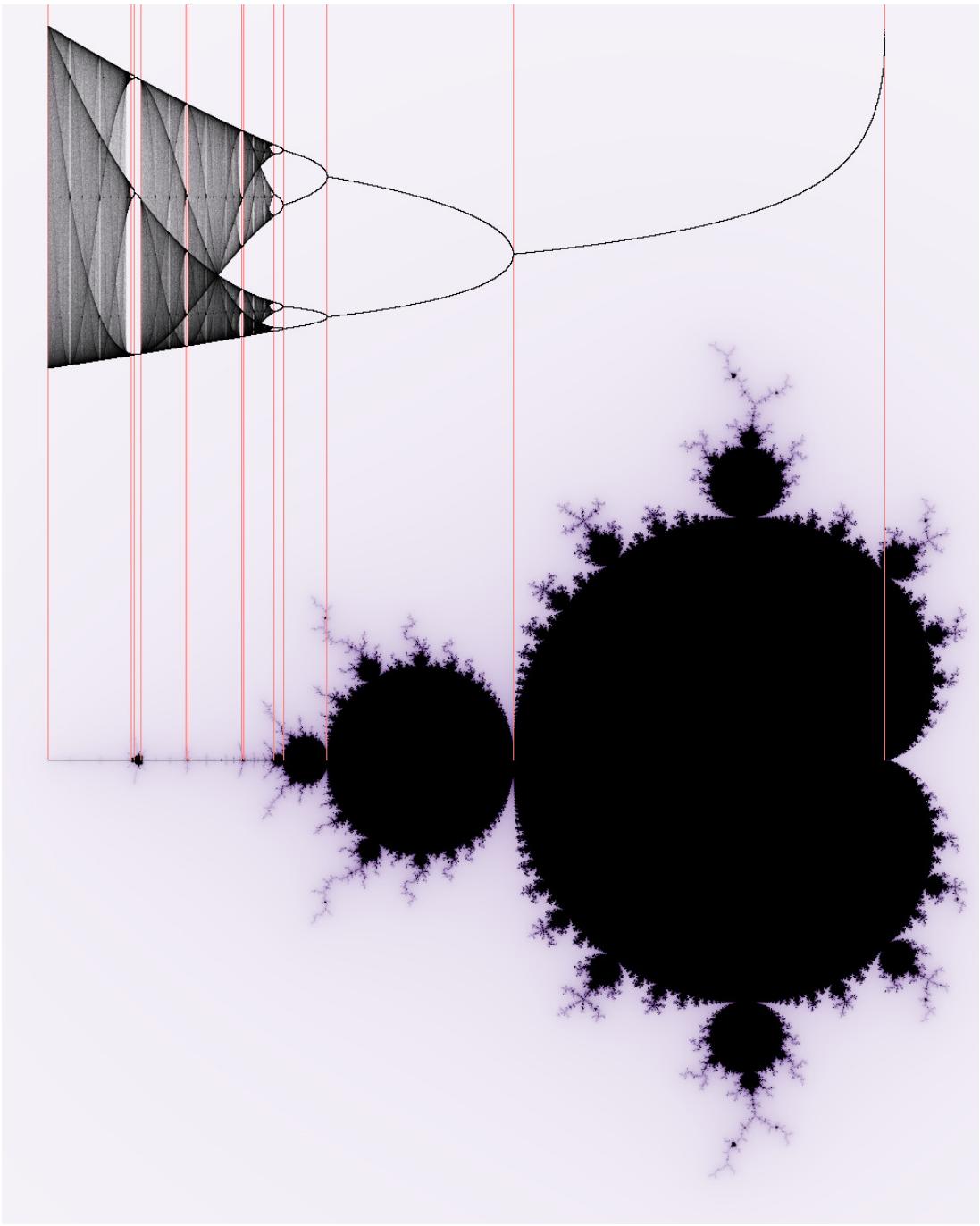
$$x_{k+1} = 2bx_k(1 + \frac{a}{2b}x_k) + \frac{b^2 - b + c}{a}$$

Übung : Zeige obige Folgerung. Falls es nicht klappt, dann kannst du dir den [Zusammenhang zwischen Mandelbrot und Wale](#) anschauen.

Übung : Begründe, dass ein Vergleich mit der logistischen Funktion $f_r(x)$ heisst: $b = \frac{r}{2}$ und $a = -r$. Ferner zeige man, dass der interessante Bereich $1 < r < 4$ den Werten $-2 < c < \frac{1}{4}$ entspricht.

Übung : Bestimme die c -Bereiche der Fixpunktregionen $1 < r < 3$ und $3 < r < 1 + \sqrt{6}$.

Abschliessend begreife man folgendes Bild !



1.5 Enjoy! Don't Freak out!

Übung : Jetzt kommt eine gemütliche Pflichtaufgabe. Höre zum Video Mandelbrot Zoom, in dem eine Stelle in der komplexen Ebene vergrössert wird, den sinnigen Titel "Functionality" dazu. Mindestens bis Minute 7.5 den Zoom angucken. Dem Video voraus schicken möchte ich, dass die Dimension des Universums im Verhältnis zu einem Atomkern in etwa $10^{40} \div 1$ ist. In Längen gedacht heisst dies also für Zahlen, welche sich erst an der 40-sten Stelle nach dem Komma unterscheiden, dass das Universum um einen Atomdurchmesser verschoben wurde. Erinnere dich an diesen Fact,

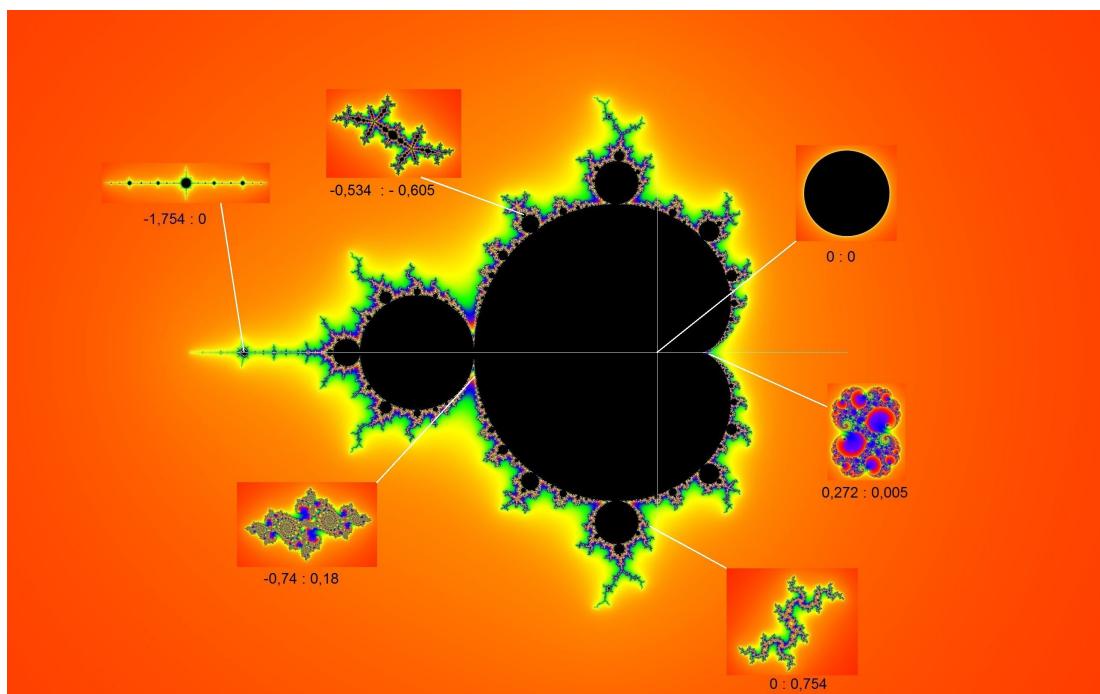
wenn jeweils die Größenordnung im Video eingeblendet wird.

So, los gehts! Zuerst den Sound einrichten...

- 1) Functionality 2 von DJ Dimsa



- 2) Mandelbrot Zoom



Abbildungsverzeichnis

1.	Sierpinski Dreieck, Struktur im Zufall	6
2.	Pascal'sches Sierpinski Dreieck	7
3.	The Famous Mandelbrot Set	13
4.	Graph der Wurzelfunktion	15
5.	Graph der Geraden $2x - 1$	17
6.	Graph der Quadratfunktion	17
7.	Graph von $x^2 - 2$	18
8.	Graph von $-x^2 + 1$	18