

# **Fraktale und Chaos**

## **Eine praktische Einführung**

von

Dietmar Grätzer

Kahl am Main      Dezember 2002

Copyright © 2002 by Dietmar Grätzer

PDF-Konvertierung 04.2003 von Thomas Antoni - <http://www.qbasic.de>

# Übersicht

<b>Fraktale und Chaos</b> .....	1
Übersicht .....	2
1. Vorbemerkung .....	4
2. Iterationen im Reellen .....	5
2.1 Die wiederholte Quadrierung einer Zahl .....	5
2.2 Umkehriteration .....	10
2.3 Eine erweiterte Iteration – Einsatz von Experimentierprogrammen .....	12
2.4 Periodische Punkte – Die Iteration $x_{n+1} = x_n^2 - 1$ .....	16
2.5 Chaos – Die Iteration $x_{n+1} = x_n^2 - 2$ .....	21
2.6 Allgemeine Betrachtung der Iteration $x_{n+1} = x_n^2 + a$ .....	24
2.7 Das Feigenbaum–Diagramm .....	32
2.8 Der 3er–Zyklus im Feigenbaum–Diagramm .....	34
3. Komplexe Zahlen .....	39
3.1 Einführende Überlegungen .....	39
3.2 Gaußsche Zahlenebene .....	40
3.3 Addition .....	41
3.4 Multiplikation .....	41
3.5 Betrag einer komplexen Zahl .....	42
3.6 Wurzel aus einer komplexen Zahl .....	43
4. Iterationen im Komplexen .....	46
4.1 Ein Einführungsbeispiel – Die Iteration $z_{n+1} = z_n^2$ .....	46
4.2 Die allgemeine Iteration $z_{n+1} = z_n^2 + c$ .....	51
4.3 Darstellung der Julia–Menge durch Umkehriteration .....	58
4.4 Einkreisung der ausgefüllten Julia–Menge .....	61
4.5 Fixpunkte, Zyklen und Einzugsbereiche .....	66
4.6 Die Mandelbrot–Menge .....	72
4.7 Einige morphologische Betrachtungen .....	78

5. Iterationen in den Quaternionen .....	82
6. Anhang .....	85
6.1 QuickBasic – Programme .....	85
6.2 Koordinaten – Transformation .....	114
6.3 Konvergenzsatz .....	115
6.4 Berechnung der Feigenbaum–Konstanten .....	117
6.5 Internet–Adressen .....	120
6.6 Literaturhinweise .....	121

# 1. Vorbemerkung

Mit der rasanten Entwicklung der PC-Technologie ist es heute möglich geworden, auch komplizierte mathematische Berechnungen quasi "zu Hause" durchzuführen und mathematische Zusammenhänge am Bildschirm sichtbar zu machen.

Besonders bestechend sind Computer-Grafiken der fraktalen Geometrie, die als "Julia-Mengen" oder "Apfelmännchen" in weiten Kreisen bekannt wurden und durch ihre einmalig schönen Strukturen auch den Laien ansprechen und begeistern.

Im Internet werden mittlerweile Programme angeboten, die kostenlos heruntergeladen werden können und mit denen man faszinierende Grafiken produzieren kann.

Um den mathematischen Hintergrund besser verstehen zu können, wurde in der vorliegenden Schrift versucht, ausgehend von einfachen Beispielen, die theoretischen Grundlagen mit elementaren Mitteln zu entwickeln. Dabei steht die quadratische Iteration mit ihrer verblüffend einfachen Formel im Vordergrund. Es ist erstaunlich, wie durch Erweiterung dieser einfachen Formel, beginnend mit den reellen Zahlen über die komplexen Zahlen bis zu den Quaternionen, sich immer neue Aspekte ergeben. Dabei ist es nicht die Absicht, alle Eigenschaften streng mathematisch zu beweisen, es soll vielmehr ein Überblick gegeben und die wesentlichen Gedankengänge sollen aufgezeigt werden.

Die formelmäßigen Ableitungen werden durch einfache Computerprogramme ergänzt, die in der weit verbreiteten Programmiersprache "QBasic" geschrieben sind. Diese Programme erheben keinen Anspruch auf optimale Programmierung, sondern sollen Anregungen geben, eigene Rechenexperimente durchzuführen, indem Parameter geändert oder die Rechenalgorithmen nach eigenen Interessen modifiziert und ausgebaut werden können.

Damit soll dieses sehr reizvolle und alles andere als "trockene" Spezialgebiet der Mathematik veranschaulicht und einem interessierten Kreis auf praktische Weise nähergebracht werden.

## 2. Iterationen im Reellen

### 2.1 Die wiederholte Quadrierung einer Zahl

Ausgangspunkt für unsere Betrachtungen ist ein einfaches Rechenbeispiel, das wir mit einem (wissenschaftlichen) Taschenrechner durchführen:

Wir geben eine Zahl 0.5 ein und drücken mehrmals hintereinander die  $x^2$ -Taste. Wir beobachten, daß das angezeigte Ergebnis immer kleiner wird und offenbar gegen 0 strebt.

Als nächstes geben wir die Zahl 1.2 ein und drücken wieder mehrmals die  $x^2$ -Taste. Das Ergebnis wird immer größer und scheint gegen Unendlich ( $\infty$ ) zu streben.

Geben wir 0 ein, dann zeigt sich, daß bei wiederholtem Drücken der  $x^2$ -Taste dieser Wert sich nicht ändert.

Zum gleichen Verhalten kommen wir, wenn wir 1 eingeben. Dieser Wert ändert sich nicht.

Die Ergebnisse unserer Rechnungen stellen wir in einer Tabelle zusammen:

$x_0$	0	0.5	1	1.2
$x_1$	0	0.25	1	1.44
$x_2$	0	0.0625	1	2.073
$x_3$	0	0.0039	1	4.299
$x_4$	0	0.000015	1	18.48
* *	↓	↓	↓	↓
	0	0	1	$\infty$
	Fixpunkt anziehend	Grenzpunkt	Fixpunkt abstoßend Scheidepunkt	Grenzpunkt

Damit sind wir schon in der Lage, einige wesentliche Ergebnisse festzuhalten und wichtige Begriffe einzuführen:

- 1) Wenn man in einem Rechenvorgang das Ergebnis mit diesem Rechenvorgang immer wieder wiederholt, dann spricht man von einer Iteration.

- 2) Der Anfangswert  $x_0$ , mit dem man die Iteration beginnt, heißt Startwert.

- 3) Bei der Iteration entsteht eine Folge von Zahlen:

$$x_0, x_1, x_2, x_3, \dots, x_n, x_{n+1}, \dots = (x_n) = O(x_0),$$

die man als Bahn oder als Orbit des Punktes  $x_0$  bezeichnet.

- 4) In unserem Beispiel ist:

$x_0$  Startwert

$$x_1 = x_0^2$$

$$x_2 = x_1^2 = x_0^4$$

$$x_3 = x_2^2 = x_0^8$$

\*

\*

\*

Allgemein berechnet sich der Nachfolger  $x_{n+1}$  aus dem Vorgänger  $x_n$  nach der Gleichung:

$$x_{n+1} = x_n^2 ; \quad n = 0, 1, 2, 3, \dots$$

Diese Gleichung heißt Iterationsgleichung.

- 5) Speziell in unserem Beispiel haben wir immer wieder das Quadrat einer Zahl berechnet.

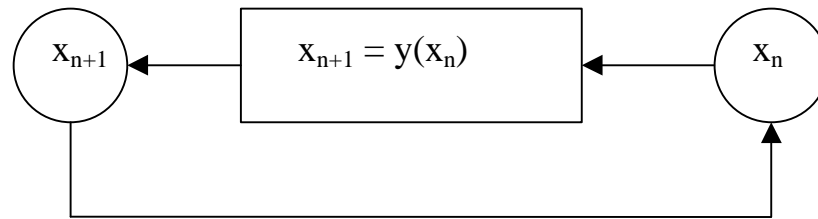
Die Funktion  $y(x) = x^2$

heißt Iterationsfunktion. Wir können daher auch für unsere Iterationsgleichung schreiben:

$$x_{n+1} = y(x_n)$$

allgemeine Iterationsgleichung

Die Iteration kann veranschaulicht werden durch folgenden Rückkopplungsprozeß:



- 6) Wir haben beobachtet, daß sich bei der Iteration die Zahlen 0 und 1 nicht ändern. Diese Zahlen  $x_1^* = 0$  und  $x_2^* = 1$  heißen daher Fixpunkte der Iteration.
- 7) Für die Fixpunkte ergibt sich nach der Iteration der gleiche Zahlenwert. Es ist also in unserem Fall:

$$x = x^2 \quad \text{oder} \quad \boxed{x^2 - x = 0}$$

Diese Gleichung heißt Fixpunktgleichung. Die Lösungen der Fixpunktgleichung ergeben die Fixpunkte:

$$x(x - 1) = 0 \quad \Rightarrow \quad \boxed{\begin{array}{l} x_1^* = 0 \\ x_2^* = 1 \end{array}}$$

Allgemein hat die Fixpunktgleichung mit der Iterationsfunktion  $y(x)$  die Form:

$$\boxed{y(x) - x = 0} \quad \underline{\text{allgemeine Fixpunktgleichung}}$$

- 8) Zur Charakterisierung der Fixpunkte  $x^*$  legt man fest (siehe hierzu auch den Konvergenzsatz im Anhang 6.3):

$x^*$  ist anziehend, wenn  $|y'(x^*)| < 1$

$x^*$  ist abstoßend, wenn  $|y'(x^*)| > 1$

$x^*$  ist indifferent, wenn  $|y'(x^*)| = 1$

In diesem Zusammenhang spricht man auch vom Attraktionsverhalten der Fixpunkte.

Erläuterungen:

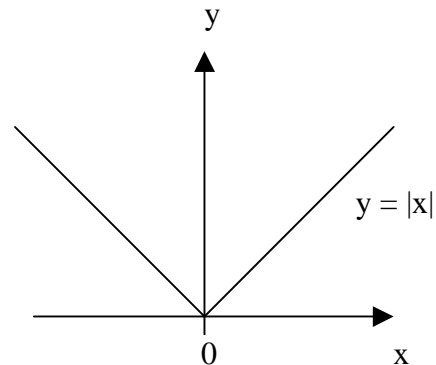
a)  $y'(x)$  ist die erste Ableitung (Tangentensteigung) von  $y(x)$ :

$$\frac{\Delta y(x)}{\Delta x} \rightarrow y'(x) \quad \text{für} \quad \Delta x \rightarrow 0$$

b) Der Betrag einer Zahl ist definiert durch:

$$|x| = \begin{cases} x & \text{für } x > 0 \\ -x & \text{für } x < 0 \\ 0 & \text{für } x = 0 \end{cases}$$

$$|x| < 1 \Leftrightarrow -1 < x < +1$$



9) In unserem Fall ist:

$$y(x) = x^2; \quad y'(x) = 2 \cdot x$$

$$y'(x_1^*) = 2 \cdot 0 = 0 \Rightarrow x_1^* = 0 \quad \text{ist anziehender Fixpunkt}$$

$$y'(x_2^*) = 2 \cdot 1 = 2 \Rightarrow x_2^* = 1 \quad \text{ist abstoßender Fixpunkt,}$$

in Übereinstimmung mit unserem Rechenexperiment. Der anziehende Fixpunkt  $x_1^* = 0$  zieht quasi seine Umgebung an, wovon man sich überzeugen kann, wenn man die Iteration mit verschiedenen Startwerten  $x_0$  aus der näheren Umgebung von  $x_1^*$  beginnt.

10) Der Einzugsbereich  $A(x_1^*)$  für den anziehenden Fixpunkt  $x_1^* = 0$  ist gegeben durch:

$$A(x_1^*) = \{x_0 \mid (x_n) \rightarrow x_1^* \text{ für } n \rightarrow \infty\}$$

$$= \{x_0 \mid |x_0| < 1\}$$

Diese formale Schreibweise wird gelesen als: "Der Einzugsbereich von  $x_1^*$  ist die Menge aller  $x_0$ , für die gilt:  $|x_0| < 1$ ."

Faßt man den Punkt  $\infty$  ebenfalls als anziehenden Fixpunkt auf, dann kann man formal schreiben:

$$A(\infty) = \{x_0 \mid |x_0| > 1\}$$



Für das Verhalten der Iteration gilt somit:

$$(x_n) \rightarrow \begin{cases} 0 & \text{für } |x_0| < 1 \text{ und } n \rightarrow \infty \\ 1 & \text{für } |x_0| = 1 \text{ und } n \rightarrow \infty \\ \infty & \text{für } |x_0| > 1 \text{ und } n \rightarrow \infty \end{cases}$$

- 11) Der abstoßende Fixpunkt  $x_2^* = 1$  ist als Scheidepunkt anzusehen, an dem die beiden Einzugsbereiche  $A(x_1^*)$  und  $A(\infty)$  aneinanderstoßen.

Zum Beispiel unterscheiden sich die beiden Startwerte  $x_0 = 0.9999$  und  $x_0 = 1.0001$  nur um 0.0002. Trotzdem ist das Grenzverhalten der Iteration für beide Fälle sehr unterschiedlich.

## 2.2 Umkehriteration

Wenn wir von der Iteration  $x_{n+1} = x_n^2 = y(x_n)$  mit dem Startwert  $x_0$  ausgehen, dann erhalten wir die Zahlenfolge:

$$x_0, x_1, x_2, x_3, \dots, x_{n-1}, x_n, x_{n+1}, \dots$$

Es stellt sich nun die Aufgabe, bei vorgegebenem  $x_n$  den Vorgänger  $x_{n-1}$  (das Urbild von  $x_n$ ) zu ermitteln.

Für  $x_n$  gilt:  $x_n = x_{n-1}^2$

Daraus ergibt sich:  $x_{n-1} = \pm \sqrt{x_n}$

Da wir mit reellen Zahlen rechnen, betrachten wir nur das "+"-Zeichen und schreiben:

$$x_{n+1} = + \sqrt{x_n} = y^{-1}(x_n)$$

Diese Gleichung stellt die Umkehriteration zu unserer Ausgangsiteration  $x_{n+1} = y(x_n)$  dar mit folgenden Eigenschaften:

1) Die Fixpunktgleichung ist identisch mit der der Ausgangsiteration:

$$x^2 - x = 0$$

mit den gleichen Fixpunkten:

$$x_1^* = 0 \quad \text{und} \quad x_2^* = 1$$

2) Für das Attraktionsverhalten folgt:

$$y^{-1}(x) = \sqrt{x} = x^{\frac{1}{2}}$$

$$y^{-1}'(x) = \frac{1}{2} x^{-\frac{1}{2}} = \frac{1}{2\sqrt{x}}$$

$$y^{-1}'(x_1^*) \rightarrow \infty \quad \Rightarrow \quad x_1^* = 0 \quad \text{ist abstoßender Fixpunkt}$$

$$y^{-1}'(x_2^*) = \frac{1}{2} \quad \Rightarrow \quad x_2^* = 1 \quad \text{ist anziehender Fixpunkt}$$

Der Punkt  $\infty$  kann in diesem Fall als abstoßender Fixpunkt angesehen werden.

3) Damit liegt folgende Schlußfolgerung nahe:

Bei der Umkehriteration  $x_{n+1} = y^{-1}(x_n)$  kehrt sich das Attraktionverhalten der Fixpunkte gegenüber der Ausgangsiteration  $x_{n+1} = y(x_n)$  um.

Das Einzugsgebiet des anziehenden Fixpunktes  $x_2^*$  ist somit:

$$A(x_2^*) = \{x_0 \mid 0 < x_0 < \infty\}$$

Diese Ergebnisse können durch Rechenbeispiele mit dem Taschenrechner bestätigt werden, indem wir die Funktion  $\sqrt{x}$  wiederholt anwenden. Beispiele zeigt die folgende Tabelle:

$x_0$	0.1	1	100
$x_1$	0.316	1	10
$x_2$	0.562	1	3.162
$x_3$	0.749	1	1.778
$x_4$	0.865	1	1.333
$x_5$	0.930	1	1.154
*	↓	↓	↓
*			
	1	1	1

## 2.3 Eine erweiterte Iteration – Einsatz von Experimentierprogrammen

Bisher haben wir den rein quadratischen Iterator  $y(x) = x^2$  betrachtet. Als nächsten Schritt fügen wir noch einen konstanten Wert hinzu und untersuchen den Iterator  $y(x) = x^2 - \frac{1}{2}$ .

Als Iterationsgleichung ergibt sich dann:

$$x_{n+1} = x_n^2 - \frac{1}{2}$$

Die Fixpunktgleichung lautet:

$$x = x^2 - \frac{1}{2} ; \quad x^2 - x - \frac{1}{2} = 0$$

Die allgemeine quadratische Gleichung  $x^2 + p \cdot x + q = 0$

wird gelöst durch:  $x_{1,2} = -\frac{p}{2} \pm \sqrt{\frac{p^2}{4} - q}$

Damit ergeben sich in unserem Fall als Lösungen die beiden Fixpunkte:

$$x_{1,2}^* = \frac{1}{2} \pm \sqrt{\frac{1}{4} + \frac{1}{2}}$$

$$x_1^* = \frac{1}{2}(1 + \sqrt{3}) \approx 1,366$$

$$x_2^* = \frac{1}{2}(1 - \sqrt{3}) \approx -0,366$$

Für das Attraktionsverhalten ergibt sich:

$$y(x) = x^2 - \frac{1}{2} ; \quad y'(x) = 2x$$

$$y'(x_1^*) \approx 2,732 \quad \Rightarrow \quad x_1^* \text{ ist abstoßender Fixpunkt}$$

$$y'(x_2^*) \approx -0,732 \quad \Rightarrow \quad x_2^* \text{ ist anziehender Fixpunkt}$$

Der Einzugsbereich des anziehenden Fixpunktes  $x_2^*$  ist:

$$A(x_2^*) = \{x_0 \mid |x_0| < \frac{1}{2}(1 + \sqrt{3})\}$$

Für diesen Fall wollen wir nun einige Experimentierprogramme einsetzen, mit denen wir die theoretischen Ergebnisse numerisch veranschaulichen können.

### Programm FOLGE1

Mit diesem ersten, einfachen Programm kann die Bahn einer Iteration zahlenmäßig verfolgt werden.

Eingabewert ist der Parameter  $a = -0.5$ . Als Startwert wählen wir einen Wert, der etwas kleiner ist als der Wert des abstoßenden Fixpunktes  $x_1^*$ , nämlich  $x_0 = 1.36$ . Die maximale Anzahl von Iterationen wählen wir zu  $n_{\max} = 20$ .

Ergebnis Ausdruck:

n	x(n)
0	1.36
1	1.3496
2	1.32142
3	1.246152
4	1.052894
5	.6085858
6	-.1296234
7	-.4831978
8	-.2665199
9	-.4289671
10	-.3159872
11	-.4001521
12	-.3398783
13	-.3844827
14	-.352173
15	-.3759741
16	-.3586434
17	-.3713749
18	-.3620807
19	-.3688976
20	-.3639146

Beliebige Taste zum Fortsetzen drücken

Wir sehen, wie sich die Iterationsfolge langsam dem anziehenden Fixpunkt  $x_2^*$  annähert.

Mit dem Programm können wir aber noch einige weitere Experimente durchführen:

Die Erhöhung der Anzahl von Iterationen auf  $n_{\max} = 500$  zeigt, wie der Grenzwert  $x_2^*$  schon sehr genau angenähert und ein Endzustand der Iteration, bedingt durch die Rechengenauigkeit, erreicht wird.

Bei der Eingabe von  $x_0 = 1.4$  und  $n_{\max} = 8$  sieht man, wie die Iteration schnell divergiert und gegen  $\infty$  strebt.

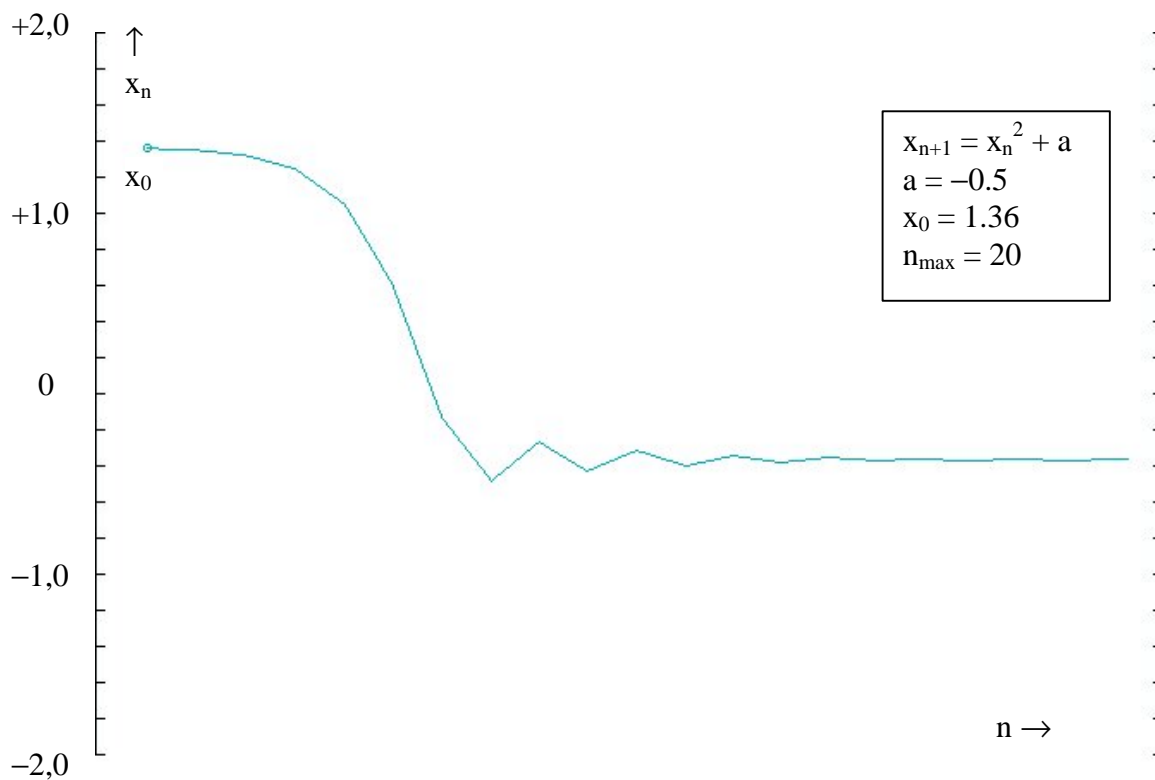
### Programm ORBIT1

Dieses Programm dient dazu, die Bahn (den Orbit) des Anfangspunktes  $x_0$  für die Iteration grafisch darzustellen.

Eingabewerte sind bezogen auf unseren Fall zum Beispiel: Parameter  $a = -0.5$ ,

Startwert  $x_0 = 1.36$  und maximale Anzahl von Iterationen  $n_{\max} = 20$ .

Es ergibt sich folgender Ergebnisausdruck:



Beliebige Taste zum Fortsetzen drücken

## Programm GRAFIT1

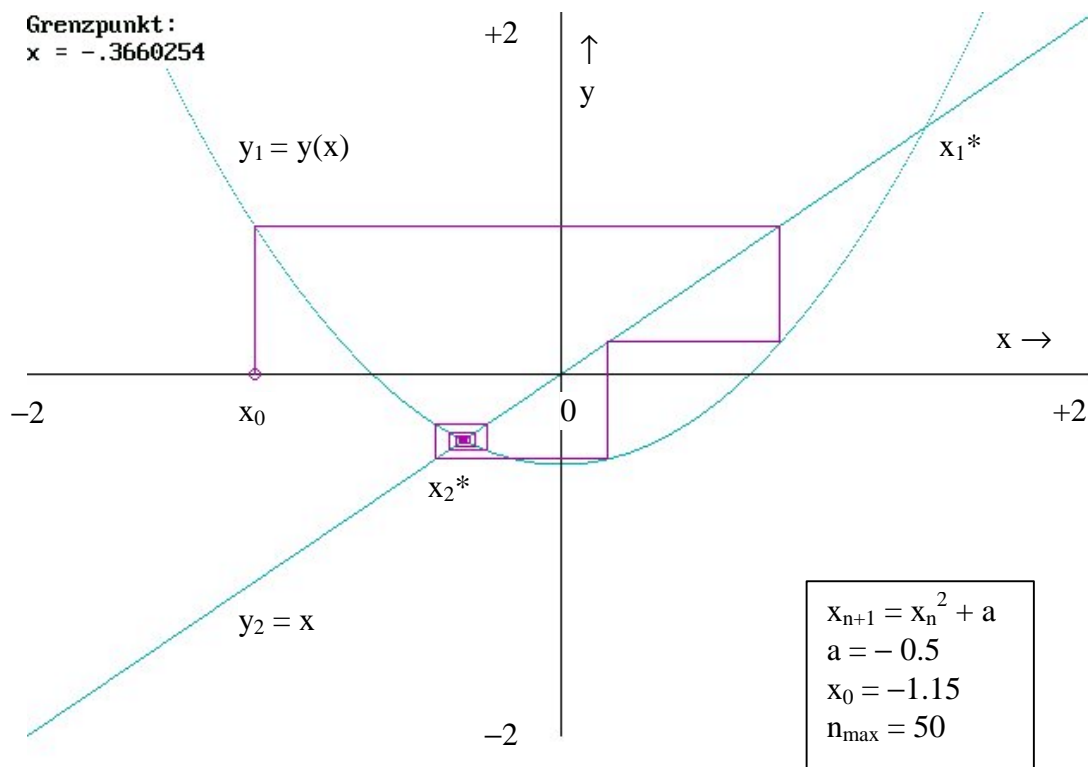
Mit diesem Programm kann eine grafische Iteration durchgeführt werden. Dabei werden ausgehend von der allgemeinen Fixpunktgleichung  $x = y(x)$  zunächst die beiden Funktionen  $y_1 = y(x)$  und  $y_2 = x$  (Winkelhalbierende) dargestellt.

Die Schnittpunkte der Funktionen  $y_1$  und  $y_2$  sind die Lösungen der Fixpunktgleichung und damit die Fixpunkte.

Dann wird der Linienzug:

$x_0 \rightarrow y(x_0) \rightarrow x_1 = y(x_0) \rightarrow y(x_1) \rightarrow x_2 = y(x_1) \rightarrow y(x_2) \rightarrow x_3 = y(x_2) \rightarrow \dots$   
eingezeichnet.

Für unseren Fall als Beispiel ergibt sich für die Eingabewerte  $a = -0.5$ ,  $x_0 = -1.15$  und  $n_{\max} = 50$  der folgende Ergebnisausdruck:



Beliebige Taste zum Fortsetzen drücken

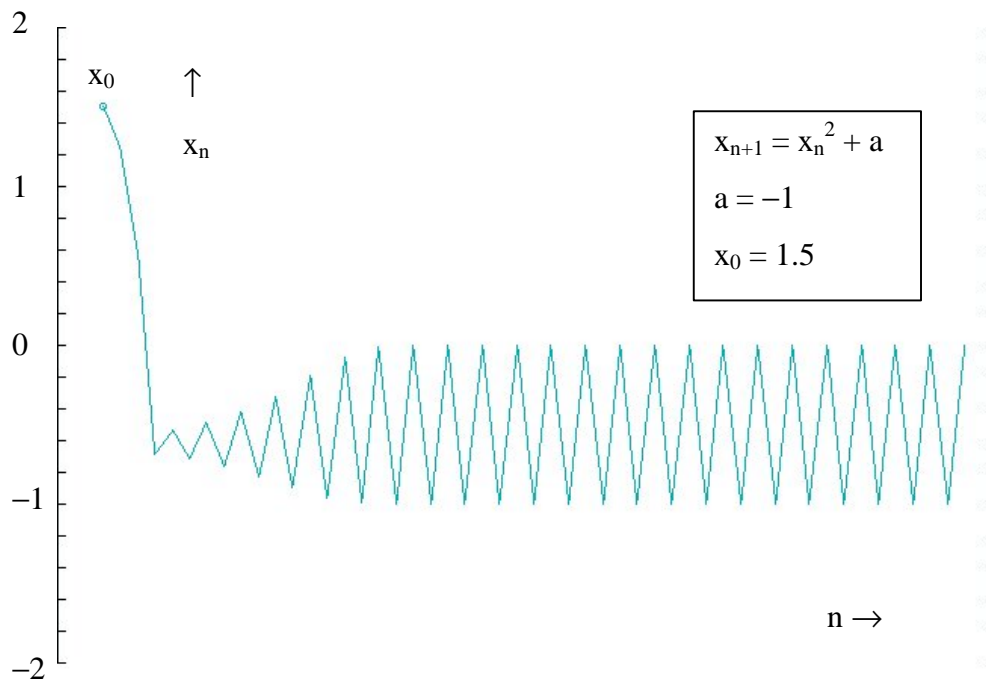
## 2.4 Periodische Punkte – Die Iteration $x_{n+1} = x_n^2 - 1$

Bisher haben wir die Iteration  $x_{n+1} = x_n^2 + a$  für die Parameterwerte  $a = 0$  und  $a = -0.5$  untersucht. Was passiert, wenn wir den Parameterwert  $a$  weiter verkleinern ? Wir wählen als nächstes Beispiel  $a = -1$ , untersuchen somit die Iteration

$$x_{n+1} = x_n^2 - 1$$

Das Programm FOLGE1 zeigt für  $a = -1$ ;  $x_0 = 1.5$  und  $n_{\max} = 25$  ein seltsames Verhalten der Iteration: Die Zahlenfolge nähert sich abwechselnd zwei Grenzwerten, nämlich 0 und  $-1$ .

Mit dem Programm ORBIT1 können wir ähnliches beobachten: Die Bahn der Iteration springt nach dem Einschwingen zwischen 2 Werten hin und her. Einen Ausdruck zeigt die folgende Abbildung.



Beliebige Taste zum Fortsetzen drücken



Es entsteht ein sogenannter Zweier-Zyklus mit den Zyklus-Elementen 0 und -1. Wir schreiben den Zyklus in der Form:

$$Z^* = \{0, -1\}$$

Zunächst untersuchen wir die Iteration nach der uns schon bekannten Weise.

Die Fixpunktgleichung  $x^2 - x - 1 = 0$  liefert die beiden Fixpunkte

$$x_{1,2} = \frac{1}{2} \pm \sqrt{\frac{1}{4} + 1}$$

$$x_1 = \frac{1}{2}(1 + \sqrt{5}) \approx 1,618$$

$$x_2 = \frac{1}{2}(1 - \sqrt{5}) \approx -0,618$$

Für das Attraktionsverhalten ergibt sich:

$$y(x) = x^2 - 1; \quad y'(x) = 2x$$

$$y'(x_1^*) \approx 3,236 \quad \Rightarrow \quad x_1^* \text{ ist abstoßend}$$

$$y'(x_2^*) \approx -1,236 \quad \Rightarrow \quad x_2^* \text{ ist abstoßend}$$

Wir erhalten also 2 abstoßende Fixpunkte  $x_1^*$  und  $x_2^*$ .

Um das seltsame Verhalten aufzuklären, müssen wir einige weitere Begriffe einführen.

Wir gehen von der allgemeinen Iterationsgleichung für die Iterationsfunktion  $y(x)$  aus und schreiben:

$$x_{n+1} = y(x_n) = y^1(x_n)$$

$y^1(x)$  nennen wir nun die Erste Iterierte.

Wird die Iteration fortgesetzt, dann erhalten wir eine Verkettung der Iterationsfunktion. Wir schreiben:

$$x_{n+2} = y^1 \circ y^1(x_n) = y^1(y^1(x_n)) = y^2(x_n)$$

und nennen  $y^2(x)$  die Zweite Iterierte von  $y(x)$ .

Eine weitere Iteration ergibt:

$$x_{n+3} = y^1(y^2(x_n)) = y^3(x_n)$$

$y^3(x)$  nennen wir die Dritte Iterierte von  $y(x)$  u.s.w.

Insbesondere gilt dann:

$$x_n = y^n(x_0)$$

Wir untersuchen nun die Zweite Iterierte mit der Fixpunktgleichung

$$y^2(x) - x = 0$$

Explizit errechnet sich in unserem Fall:

$$y^1(x) = x^2 - 1$$

$$y^2(x) = (x^2 - 1)^2 - 1 = x^4 - 2x^2$$

Somit ergibt sich für die Fixpunktgleichung eine Gleichung 4. Grades, die gelöst werden muß:

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

Um diese Gleichung zu lösen, halten wir zunächst fest:

Fixpunkte der Ersten Iterierten sind auch Fixpunkte der Zweiten Iterierten.

Denn:

Ist  $x^*$  Fixpunkt der Ersten Iterierten, dann gilt:  $y^1(x^*) = x^*$ .

Es folgt:  $y^2(x^*) = y^1(y^1(x^*)) = y^1(x^*) = x^*$

D.h.:  $x^*$  ist Fixpunkt der Zweiten Iterierten.

Mittels Polynomdivision ergibt sich:

$$(x^4 - 2x^2 - x) : [(x - x_1^*) (x - x_2^*)] =$$

$$= (x^4 - 2x^2 - x) : (x^2 - x - 1) = x^2 + x$$

$$\begin{array}{r} x^4 - x^3 - x^2 \\ \hline \end{array}$$

$$\begin{array}{r} x^3 - x^2 - x \\ \hline \end{array}$$

$$0$$

Für die Fixpunktgleichung gilt somit folgende Linearfaktorzerlegung:

$$x^4 - 2x^2 - x = (x^2 - x - 1) (x^2 + x) = 0$$

$$= (x - x_1^*) (x - x_2^*) x (x + 1) = 0$$

Daraus folgen als Fixpunkte für die Zweite Iterierte:

$$x_1^* = \frac{1}{2}(1 + \sqrt{5})$$

$$x_2^* = \frac{1}{2}(1 - \sqrt{5})$$

$$x_3^* = 0$$

$$x_4^* = -1$$

Attraktionsverhalten:

$$y^2(x) = x^4 - 2x^2; \quad y^2'(x) = 4x^3 - 4x$$

$$y^2'(x_1^*) \approx 10,472 \quad \Rightarrow \quad x_1^* \text{ ist abstoßend}$$

$$y^2'(x_2^*) \approx 1,527 \quad \Rightarrow \quad x_2^* \text{ ist abstoßend}$$

$$y^2'(x_3^*) = 0 \quad \Rightarrow \quad x_3^* \text{ ist anziehend}$$

$$y^2'(x_4^*) = 0 \quad \Rightarrow \quad x_4^* \text{ ist anziehend}$$

Wir erhalten somit die beiden anziehenden Fixpunkte  $x_3^*$  und  $x_4^*$ , die wir auch als Attraktoren bezeichnen. Sie bilden den anziehenden 2er-Zyklus  $Z^* = \{x_3^*, x_4^*\}$ .

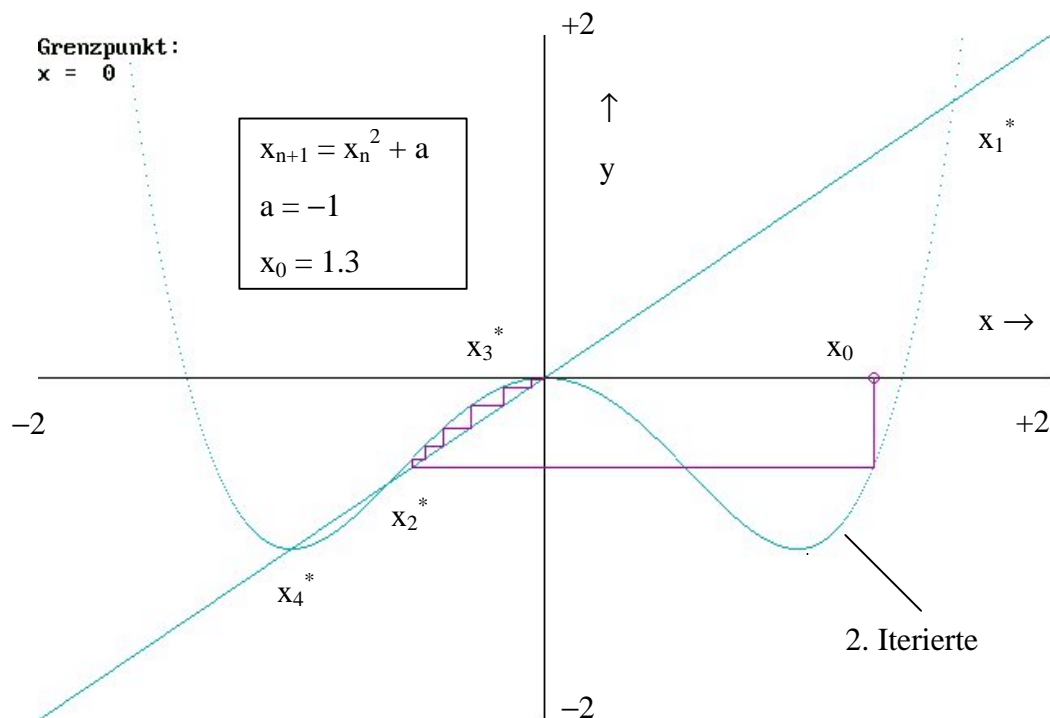
Zu Veranschaulichung sehen wir uns die grafische Iteration mit dem Programm GRAFIT1 an.

Als Funktion setzen wir die Zweite Iterierte ein:

```

funktion:                                'Festlegung der Iterationsfunktion
  FOR j = 1 TO 2
    x = x ^ 2 + a
  NEXT j
  y = x
RETURN

```



Beliebige Taste zum Fortsetzen drücken

## Allgemeine Anmerkungen:

- 1) Für die Punkte  $x_3^*$  und  $x_4^*$  ergeben sich folgende Iterationsfolgen:

$$x_3^*; y^1(x_3^*) \neq x_3^*; y^2(x_3^*) = x_3^*; y^3(x_3^*) = y^1(x_3^*); \dots\dots$$

$$x_4^*; y^1(x_4^*) \neq x_4^*; y^2(x_4^*) = x_4^*; y^3(x_4^*) = y^1(x_4^*); \dots\dots$$

Sie bilden den anziehenden 2er-Zyklus.

- 2) Für die Elemente des anziehenden 2er-Zyklus gilt:

$$Z^* = \{x_3^*, x_4^*\} = \{x_3^*, y^1(x_3^*)\}; y^1(x_4^*) = x_3^*$$

$$= \{x_4^*, y^1(x_4^*)\} = \{x_4^*, x_3^*\}$$

- 3)  $x_3^*$  und  $x_4^*$  sind periodische Punkte der Funktion  $y(x)$ .

- 4)  $x_3^*$  und  $x_4^*$  sind Fixpunkte von  $y^2(x)$  aber nicht von  $y^1(x)$ .

- 5)  $x_3^*$  und  $x_4^*$  sind Punkte der Periode 2.

- 6) Die Fixpunkte  $x_1^*$  und  $x_2^*$  stellen Scheidepunkte dar.

- 7) Der Einzugsbereich (Attraktionsbereich) für den anziehenden Zyklus  $Z^*$  ist gegeben durch:

$$A(Z^*) = \{x_0 \mid |x_0| < \frac{1}{2}(1 + \sqrt{5})\}$$

- 8) Diese Eigenschaften können verallgemeinert werden, wie wir später bei 3er-Zyklen, 4er-Zyklen u.s.w. noch sehen werden.

Allgemein kann man für einen Zyklus der Periode m schreiben:

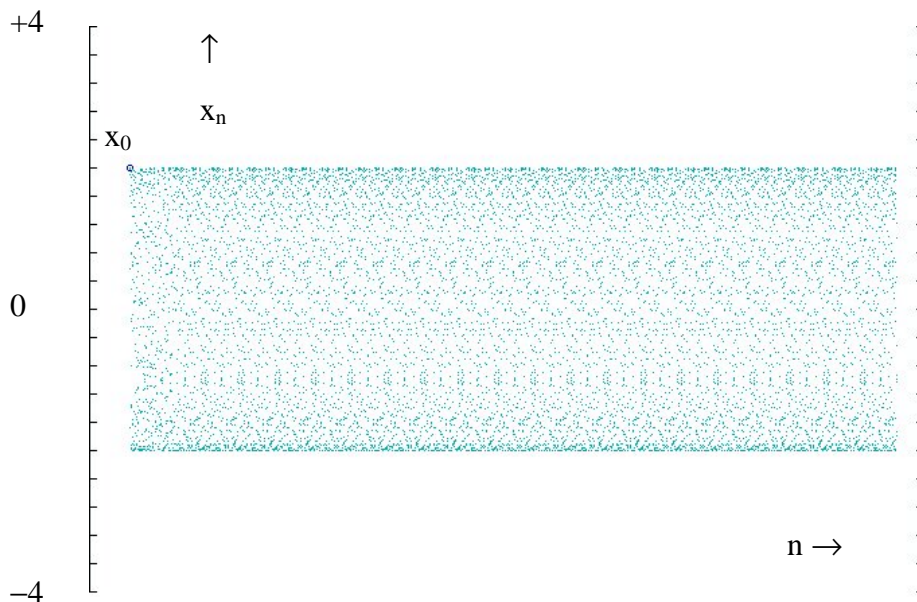
$$Z^* = \{x^*, y^1(x^*) \neq x^*, y^2(x^*) \neq x^*, \dots\dots, y^{m-1}(x^*) \neq x^*\}; y^m(x^*) = x^*$$

## 2.5 Chaos - Die Iteration $x_{n+1} = x_n^2 - 2$

Als nächstes lernen wir eine Iteration kennen, die sich völlig anders verhält als die bisher betrachteten Beispiele. Wir untersuchen die Iteration

$$x_{n+1} = x_n^2 - 2$$

Als ersten Test lassen wir das Programm ORBIT1 ablaufen. Wir nehmen jedoch eine Änderung des Programms vor, indem wir den LINE-Modus durch den PSET-Modus ersetzen und somit eine Punktfolge aufzeichnen lassen. Als Eingabewerte legen wir fest: Parameter  $a = -2$ ; Startwert  $x_0 = 1.9999$ ; Anzahl der Iterationen  $n_{\max} = 10000$ .



Beliebige Taste zum Fortsetzen drücken

Es ergibt sich ein Bild, wie es typisch ist für "chaotisches Verhalten". Wir erhalten weder Grenzwerte noch Zyklen, sondern die Punkte sind ungeordnet verteilt. Die Werte lassen sich zwar berechnen aber nicht mehr voraussagen.

Weiterhin beobachten wir, daß die Iterationswerte  $x_n$  für alle  $n$  in einem begrenzten Bereich

$$B^* = \{x_n \mid |x_n| < k\}$$

mit einem bestimmten, endlichen  $k$ -Wert bleiben und nicht etwa gegen Unendlich ( $\infty$ ) streben.

Eine Untersuchung der Fixpunktgleichung  $x^2 - x - 2 = 0$  liefert die beiden abstoßenden Fixpunkte

$$x_1^* = 2 \quad \text{und} \quad x_2^* = -1$$

Somit können wir als "Einzugsbereich" von  $B^*$  angeben:

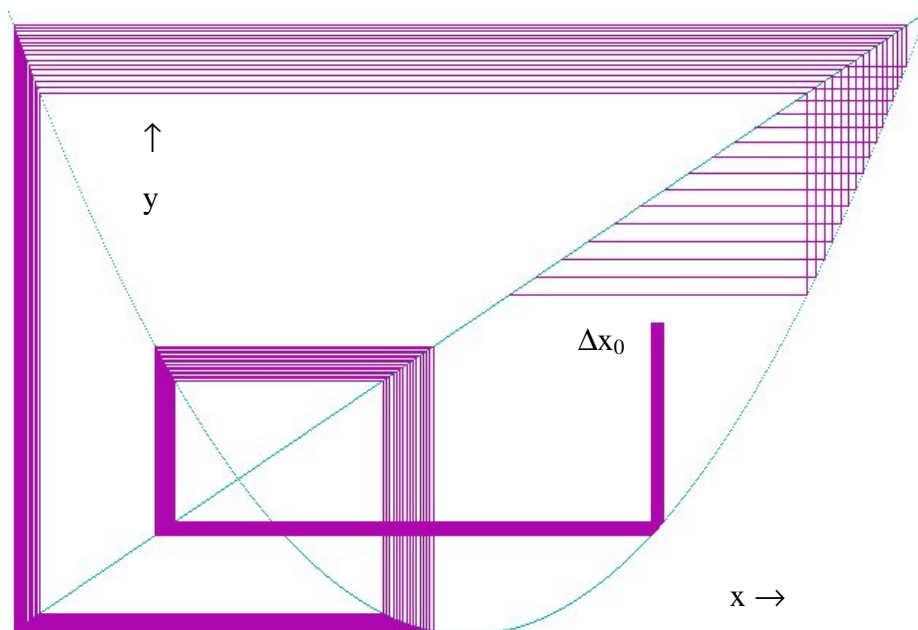
$$A(B^*) = \{x_0 \mid |x_0| < 2\}$$

(Ein Startwert  $x_0 = 2.0001$  ergibt einen "Überlauf" des Programms.)

Für das Verhalten der Iteration können wir also schreiben:

$$(x_n) \xrightarrow{n \rightarrow \infty} \begin{cases} B^* & \text{für } |x_0| < 2 \\ 2 & \text{für } |x_0| = 2 \\ \infty & \text{für } |x_0| > 2 \end{cases}$$

Ein weiteres, wichtiges Indiz für chaotisches Verhalten ist die sogenannte Sensitivität. Man versteht darunter die sensitive Abhängigkeit von den Anfangsbedingungen. Diese Eigenschaft können wir mit dem Programm GRAFIT2 durch grafische Iteration eines Intervalls veranschaulichen. Als Werte geben wir ein:  $a = -2$ ;  $x_0 = 0.8$ ;  $\Delta x_0 = 0.05$  und  $n_{\max} = 5$ .



Beliebige Taste zum Fortsetzen drücken

Wir sehen, wie sich ein kleines Anfangsintervall  $\Delta x_0$  im Laufe von nur wenigen Iterationen gewaltig vergrößert. Nehmen wir einige weitere Iterationsschritte hinzu, dann wird schließlich das ganze  $x$ -Intervall  $[-2, +2]$  überdeckt. Ein typisches Kennzeichen für chaotisches Verhalten. Kleine Änderungen in den Anfangsbedingungen können große Folgen haben. Diese Eigenschaft wird in der Chaos-Theorie als "Schmetterlingseffekt" bezeichnet. Es gibt noch weitere Anzeichen für Chaos, darauf wollen wir hier aber nicht eingehen.

## 2.6 Allgemeine Betrachtung der Iteration $x_{n+1} = x_n^2 + a$

Bisher haben wir die allgemeine Iterationsgleichung

$$x_{n+1} = x_n^2 + a$$

exemplarisch für ausgewählte Parameterwerte  $a$  betrachtet. Dabei haben wir festgestellt, daß das Iterationsverhalten zu unterschiedlichen Ergebnissen führen kann:

- a) Die Iteration konvergiert gegen einen Grenzwert.
- b) Die Iteration nähert sich einem Zyklus mit periodischen Punkten.
- c) Die Iteration ergibt einen begrenzten, chaotischen Bereich.
- d) Die Iteration wächst unbegrenzt und divergiert gegen Unendlich.

Es stellen sich eine Reihe von Fragen: Wie geht das eine Verhalten in das andere über ?

Welche Bereiche des Parameters  $a$  führen zu einem speziellen Verhalten ? Wie kann man das komplizierte Verhalten übersichtlich darstellen ?

Diesen Fragen wollen wir uns im folgenden zuwenden.

Wir betrachten zunächst die Fixpunkte der verschiedenen Iterationen:

### Fixpunkte der Ersten Iterierten

Fixpunktgleichung:  $x^2 - x + a = 0$

$$\begin{aligned} \text{Fixpunkte:} \quad x_{1,2}^* &= \frac{1}{2} \pm \sqrt{\frac{1}{4} - a} \\ &= \frac{1}{2} (1 \pm \sqrt{1 - 4a}) \end{aligned}$$

Attraktionsverhalten:

$$y^1(x) = x^2 + a; \quad y^{1'}(x) = 2x$$

$$y^{1'}(x_1^*) = 1 + \sqrt{1 - 4a}$$

$$y^{1'}(x_2^*) = 1 - \sqrt{1 - 4a}$$



Für verschiedene Parameterwerte  $a$  tragen wir die Zahlenwerte der Ableitungen in eine Tabelle ein:

$a$	$y^1'(x_1^*)$	$y^1'(x_2^*)$	Fixpunkteigenschaften
0,5	–	–	imaginär
0,25	1	1	indifferent / indifferent
0	2	0	abstoßend / anziehend
–0,5	2,73	–0,732	abstoßend / anziehend
–0,75	3	–1	abstoßend / indifferent
–1	3,23	–1,23	abstoßend / abstoßend
–2	4	2	abstoßend / abstoßend

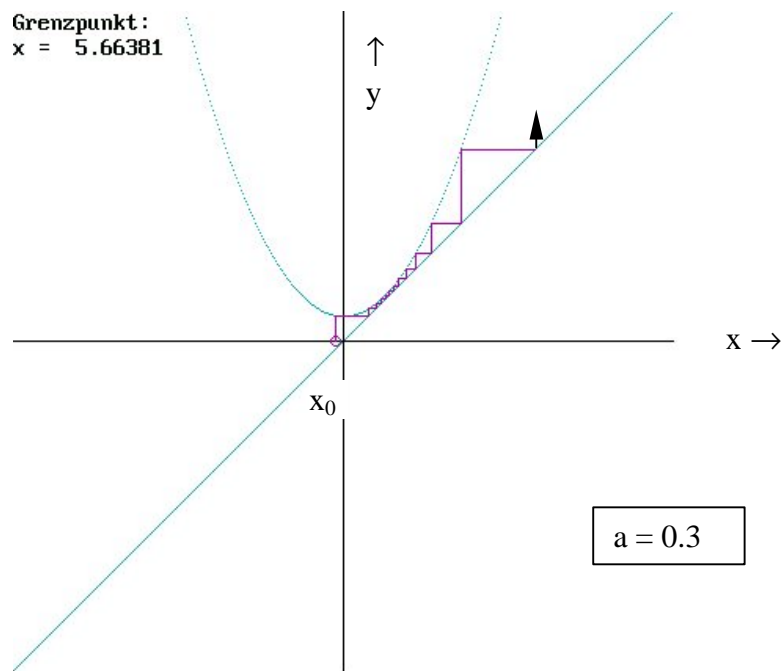
Wir sehen, daß der Fixpunkt  $x_1^*$  für  $a = 0,25$  indifferent ist und dann für kleinere Parameterwerte  $a$  abstoßend bleibt.

Der Fixpunkt  $x_2^*$  ist für  $a = 0,25$  ebenfalls indifferent, wird im Bereich  $-0,75 < a < 0,25$  anziehend und bei  $a = -0,75$  wieder indifferent. Für  $a < -0,75$  verliert der Fixpunkt  $x_2^*$  seine Attraktivität und wird abstoßend.

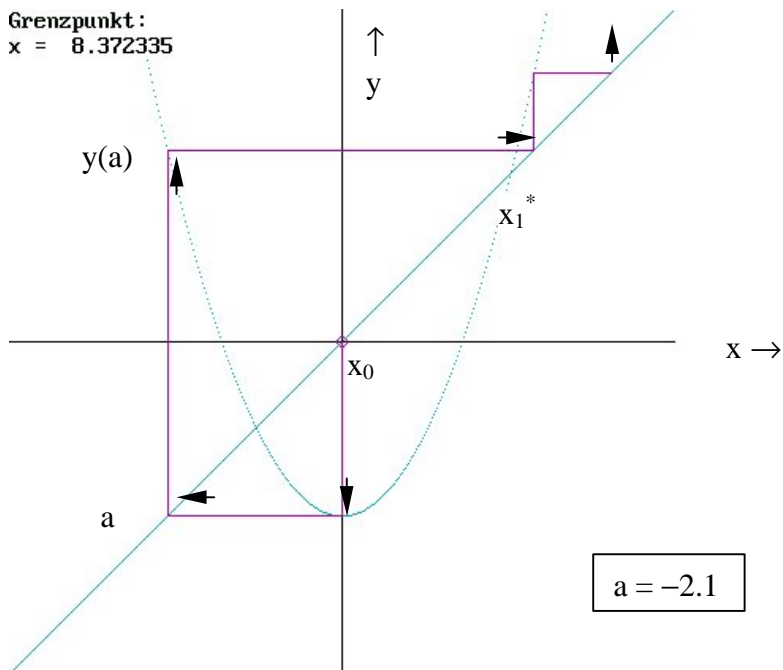
### Abschätzung der Parametergrenzen und des Attraktionsbereichs

Es stellt sich nun die Frage, in welchem Bereich kann der Parameter  $a$  der Iteration  $x_{n+1} = x_n^2 + a$  überhaupt liegen, damit sich konvergente Iterationsfolgen ergeben können ?

Wir betrachten dazu zwei Beispiele der grafischen Iteration, einmal für  $a = 0,3$  und zum anderen für  $a = -2,1$ .



Beliebige Taste zum Fortsetzen drücken



Beliebige Taste zum Fortsetzen drücken

Im ersten Fall beobachten wir, daß die Parabel oberhalb der Winkelhalbierenden liegt und diese nicht schneidet. Es gibt keine konvergierenden Iterationsfolgen. Der Grenzfall liegt dann vor, wenn die Parabel die Winkelhalbierende berührt, d.h. wenn die beiden Schnittpunkte  $x_1^*$  und  $x_2^*$  zusammenfallen. Aus

$$\frac{1}{2}(1 + \sqrt{1 - 4a}) = \frac{1}{2}(1 - \sqrt{1 - 4a})$$

folgt:

$$a = \frac{1}{4}$$

Im zweiten Fall verfolgen wir den Linienzug der Iteration für den Startwert  $x_0 = 0$ :

$$x_0 = 0 \rightarrow y(x_0) = y(0) = a \rightarrow x_1 = y(x_0) = a \rightarrow y(x_1) = y(a) \rightarrow x_2 = y(x_1) = y(a)$$

Andererseits gilt für den Fixpunkt:  $y(x_1^*) = x_1^*$ .

Ist nun  $y(a) > x_1^*$ , dann divergiert jede Iterationsfolge. Der Grenzfall ist gegeben durch:

$$y(a) = x_1^* \quad \text{oder}$$

$$a^2 + a = \frac{1}{2}(1 + \sqrt{1 - 4a})$$

Daraus ergibt sich:

$$4a^3(a + 2) = 0$$

mit der Lösung

$$a = -2$$

D.h.: Für die Iteration  $x_{n+1} = x_n^2 + a$  können sich nur konvergierende Iterationsfolgen ergeben, wenn der Parameter  $a$  im Bereich

$$-2 \leq a \leq \frac{1}{4}$$

liegt.

Für den Attraktionsbereich ergibt sich damit:

$$K = \{x_0 \mid (y^n(x_0)) \text{ beschränkt für } n \rightarrow \infty\}$$

$$A = \{x_0 \mid |x_0| < \frac{1}{2}(1 + \sqrt{1-4a}) \text{ mit } -2 \leq a \leq 0,25\} \subset K$$

Die Menge  $K$  wird, wie sich später zeigen wird, im Komplexen die ausgefüllte Julia-Menge, deren Rand die Julia-Menge ergeben.

### Fixpunkte der Zweiten Iterierten

Iteration:  $y^1(x) = x^2 + a$   
 $y^2(x) = (x^2 + a)^2 + a = x^4 + 2ax^2 + a(a+1)$

Fixpunktgleichung:  $y^2(x) - x = 0$   
 $x^4 + 2ax^2 - x + a(a+1) = 0$

Da  $x_1^*$  und  $x_2^*$  Lösungen dieser Gleichung sind, können wir folgende Polynomdivision durchführen:

$$\begin{array}{r} (x^4 + 2ax^2 - x + a(a+1)) : (x^2 - x + a) = x^2 + x + (a+1) \\ \underline{x^4 - x^3 + ax^2} \quad - \\ x^3 + ax^2 - x + a(a+1) \\ \underline{x^3 - x^2 + ax} \quad - \\ (a+1)x^2 - (a+1)x + a(a+1) \\ \underline{(a+1)x^2 - (a+1)x + a(a+1)} \quad - \\ 0 \end{array}$$

Fixpunkte:  $x_{1,2}^* = \frac{1}{2}(1 \pm \sqrt{1-4a})$

Die Gleichung  $x^2 + x + (a+1) = 0$  wird gelöst durch:

$$\begin{aligned} x_{3,4}^* &= -\frac{1}{2} \pm \sqrt{\frac{1}{4} - (a+1)} \\ &= -\frac{1}{2}(1 \pm \sqrt{1-4(a+1)}) \end{aligned}$$

Attraktionsverhalten:

$$y^2(x) = x^4 + 2ax^2 + a(a + 1)$$

$$y^{2'}(x) = 4x^3 + 4ax$$

Die numerische Berechnung der Werte für die Ableitungen in Abhängigkeit vom Parameter  $a$  ergibt die folgende Tabelle:

$a$	$y^{2'}(x_3^*)$	$y^{2'}(x_4^*)$	Fixpunkteigenschaften
-0,5	–	–	imaginär
-0,75	1	1	indifferent / indifferent
-1	0	0	anziehend / anziehend
-1,25	-1	-1	indifferent / indifferent
-1,5	-2	-2	abstoßend / abstoßend

Für  $a < -0,75$  wurde  $x_2^*$  abstoßend, stattdessen entsteht nun ein anziehender 2er-Zyklus  $Z^* = \{x_3^*, x_4^*\}$ , der bis  $a = -1,25$  erhalten bleibt und dann abstoßend wird. Was passiert dann ?

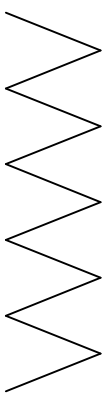
### Bifurkation - Der Weg ins Chaos

Die gestellte Frage wollen wir experimentell mit dem Programm ORBIT1 beantworten. Wir wählen den PSET-Modus und geben ein:  $x_0 = 0$ ;  $n_{\max} = 10000$ .

Durch Eingabe verschiedener Parameterwerte  $a$  können wir beobachten, wie für  $a \leq -0,75$  sich die Bahn der Iteration in 2 Linien aufspaltet und einen 2er-Zyklus ergibt. Bei  $a \leq -1,25$  beobachten wir ein weiteres Aufspalten der Linien. Es zeigt sich ein 4er-Zyklus.

Durch Probieren können wir Verzweigungspunkte  $a_i$  bestimmen, bei denen sich gerade Periodenverdopplungen (Bifurkationen) ergeben, d.h. Punkte  $a_i$ , bei denen die Fixpunkte  $x^*$  von  $y^m(x)$  ein indifferentes Verhalten zeigen ( $|y^{m'}(x^*)| = 1$ ;  $m = 2^i$ ).

Wir erhalten die folgende Tabelle:

Parameterwert		Zyklus
$a_0 = 0,25$		1er-Zyklus
$a_1 = -0,75$		2er-Zyklus
$a_2 = -1,25$		4er-Zyklus
$a_3 \approx -1,3681$		8er-Zyklus
$a_4 \approx -1,3941$		16er-Zyklus
$a_5 \approx -1,3997$		
*		*
*		*
*		*

Dieses Verhalten lässt sich beobachten bis zu einem Grenzwert  $a_\infty \approx -1,4012$ , den man den Feigenbaum-Punkt nennt (Mitchell J. Feigenbaum, geb. 1944, amerikanischer Physiker und Mathematiker). Ab diesem Punkt nimmt für  $a < a_\infty$  die Iteration einen chaotischen Charakter an.

Aus unseren experimentellen Ergebnissen können wir den Wert der Feigenbaum-Konstanten abschätzen:

$$\delta \approx \frac{a_3 - a_4}{a_4 - a_5} = \frac{-1,3681 + 1,3941}{-1,3941 + 1,3997} = 4,64$$

Diese Konstante hat eine fundamentale Bedeutung in der Chaos-Theorie. Sie ist allgemein definiert durch den Grenzwert:

$$\delta = \lim_{n \rightarrow \infty} \frac{a_{n-1} - a_n}{a_n - a_{n+1}}$$

Ein Zahlenwert mit höherer Genauigkeit ist gegeben durch  $\delta \approx 4,6692016$  (zur numerischen Berechnung siehe Anhang 6.3).

Die Definitionsgleichung können wir für große  $n$  folgendermaßen umformen:

$$a_{n+1} \approx \frac{\delta + 1}{\delta} \cdot a_n - \frac{1}{\delta} \cdot a_{n-1} = a_n + (a_n - a_{n-1}) \cdot \frac{1}{\delta}$$

Damit ergibt sich der Feigenbaum-Punkt zu:

$$a_{\infty} = \lim_{n \rightarrow \infty} \left( \frac{\delta+1}{\delta} \cdot a_n - \frac{1}{\delta} \cdot a_{n-1} \right)$$

Diese Gleichung kann dazu benutzt werden, mit dem einfachen Programm FEIGE1 aus unseren experimentellen Ergebnissen, den Feigenbaum-Punkt abzuschätzen.

```
' -----
' Quellprogramm FEIGE1.BAS
' -----
' Abschaetzung des Feigenbaum-Punktes
' -----
' Eingabewerte:
a1 = -1.3941          'Parameterwerte
a2 = -1.3997
' -----

CLS
del = 4.64           'Feigenbaum-Konstante

FOR i = 1 TO 8
    a3 = (del + 1) / del * a2 - 1 / del * a1
    PRINT i, a3
    a1 = a2
    a2 = a3
NEXT i

END
```

Nach bereits 5 Iterationen wird nach der Abschätzung weiterer  $a_i$ -Werte der Grenzwert

$$a_{\infty} \approx -1,401238$$

erreicht.

## 2.7 Das Feigenbaum–Diagramm

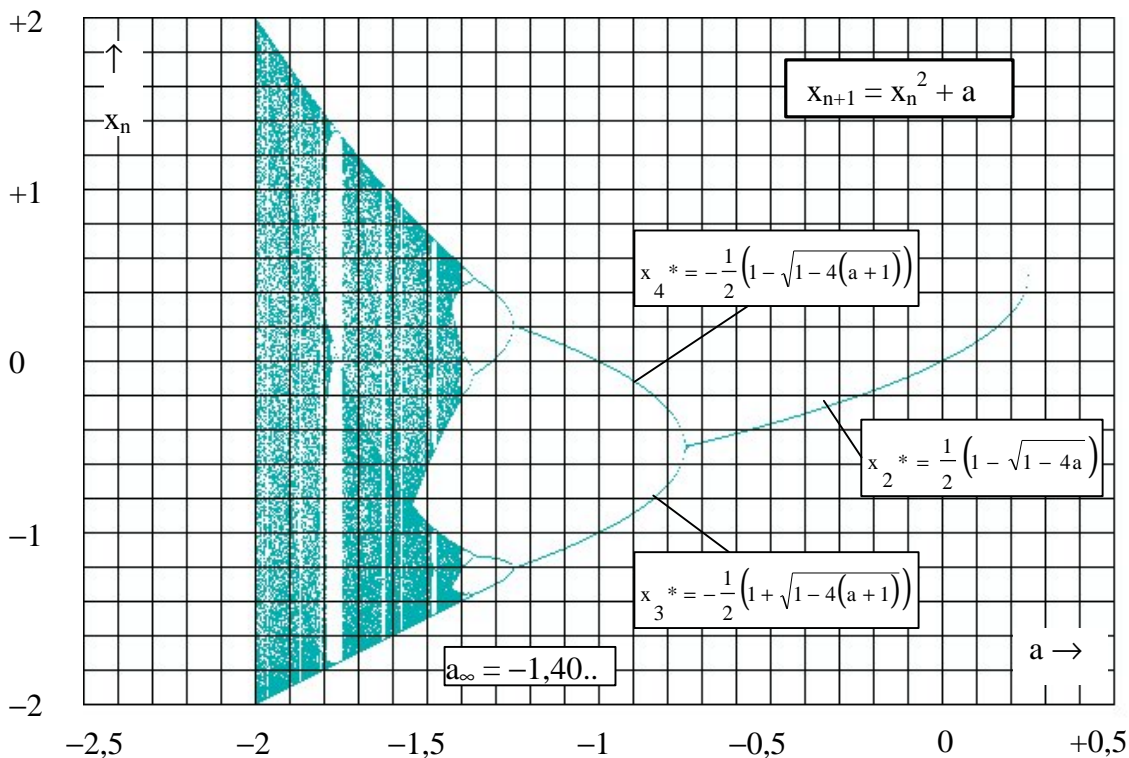
Die bisher erhaltenen Ergebnisse zur Iteration  $x_{n+1} = x_n^2 + a$  lassen sich sehr übersichtlich in einem Diagramm darstellen, das nach den grundlegenden Arbeiten von M. Feigenbaum, Feigenbaum–Diagramm genannt wird. Es ist ein sehr bemerkenswertes Fraktal, das in einem engen Zusammenhang mit der später zu diskutierenden Mandelbrot–Menge steht.

Das Prinzip besteht darin, daß man den Endzustand der Iteration in Abhängigkeit vom Parameter  $a$  grafisch darstellt.

Das Verfahren zur Darstellung kann wie folgt angegeben werden:

1. Startwert  $x_0 = 0$
2. Maximale Anzahl der Iterationen  $n_{\max}$  (z.B.  $n_{\max} = 800$ )
3. Punkte bis  $n = m$ ,  $m < n_{\max}$  außer Acht lassen (z.B.  $m = 400$ )
4. Punkte ab  $n = m$  bis  $n = n_{\max}$  anzeigen (z.B.  $x_{400} \dots x_{800}$ )
5. Iterationsverfahren in Abhängigkeit des Parameters  $a$  wiederholen

Mit dem Programm FEIGE2 erhält man das folgende Diagramm:

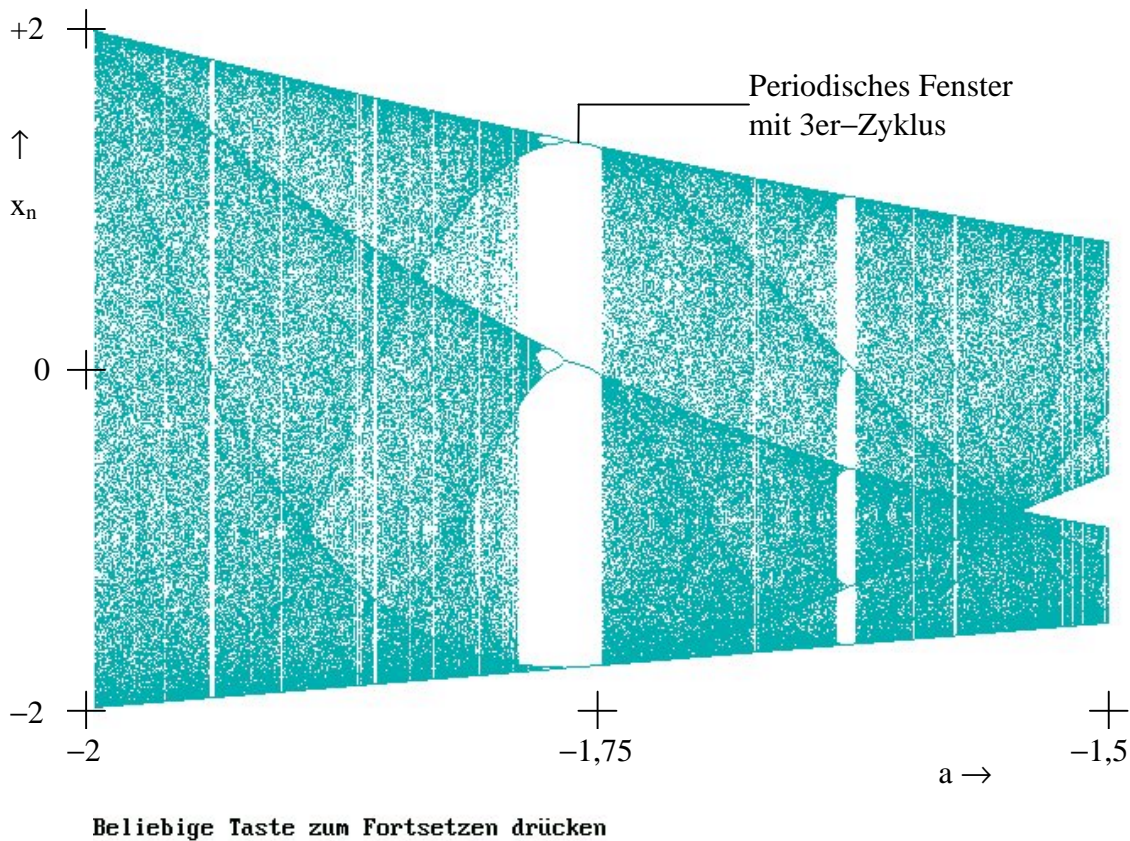


Beliebige Taste zum Fortsetzen drücken



Das Diagramm zeigt, wenn man von rechts nach links fortschreitet, zunächst ab  $a = 0,25$  einen einzelnen Fixpunkt  $x_2^*$ , ab  $a = -0,75$  entsteht eine Gabelung (Bifurkation) mit dem 2er-Zyklus  $\{x_3^*, x_4^*\}$  bis  $a = -1,25$ . Dann folgen weitere Periodenverdopplungen bis der Feigenbaum-Punkt  $a_\infty \approx -1,40$  erreicht ist. Das folgende Chaosgebiet bis  $a = -2$  zeigt eine Bänderstruktur und wird von sogenannten periodischen Fenstern unterbrochen. Für  $a < -2$  ist die Iteration divergent.

Einen interessanten Ausschnitt aus dem Feigenbaum-Diagramm mit dem Fenster der Periode 3 zeigt das folgende Bild.



## 2.8 Der 3er–Zyklus im Feigenbaum–Diagramm

In den vorhergehenden Betrachtungen haben wir schon verschiedene Zyklen kennengelernt, gegen die die Iteration  $x_{n+1} = x_n^2 + a$  strebt. Dabei konnten für den 2er–Zyklus exakte Berechnungsformeln angegeben werden. Bei den Zyklen mit höherer Periodenzahl ist dies nicht mehr möglich und man muß auf numerische Näherungslösungen zurückgreifen.

Um weitere anschauliche und verallgemeinerungsfähige Gesetzmäßigkeiten kennenzulernen, bietet es sich an, den 3er–Zyklus im Chaos–Gebiet des Feigenbaum–Diagramms zu studieren. Aus dem Diagramm entnehmen wir den Wert  $a = -1,76$ , bei dem ein 3er–Zyklus auftritt. Wir untersuchen also die Iteration

$$x_{n+1} = x_n^2 + a \quad \text{für} \quad a = -1,76$$

Nach den Ergebnissen der vorhergehenden Kapitel erhalten wir:

### Fixpunkte der ersten Iteration

$$x_1^* = \frac{1}{2}(1 + \sqrt{1 - 4a}) \approx 1,9177$$

$$x_2^* = \frac{1}{2}(1 - \sqrt{1 - 4a}) \approx -0,9177$$

$$y^1(x) = x^2 + a; \quad y^{1'}(x) = 2x$$

$$y^{1'}(x_1^*) \approx 3,8354 \quad \Rightarrow \quad x_1^* \text{ ist abstoßend}$$

$$y^{1'}(x_2^*) \approx -1,8354 \quad \Rightarrow \quad x_2^* \text{ ist abstoßend}$$

### Fixpunkte der zweiten Iteration

$$x_1^* \approx 1,9177$$

$$x_2^* \approx -0,9177$$

$$x_3^* = -\frac{1}{2}(1 + \sqrt{1 - 4(a+1)}) \approx -1,5049$$

$$x_4^* = -\frac{1}{2}(1 - \sqrt{1 - 4(a+1)}) \approx 0,5049$$

$$y^2'(x) = 4x^3 + 4ax$$

$$y^2'(x_3^*) \approx -3,038 \quad \Rightarrow \quad x_3^* \text{ ist abstoßend}$$

$$y^2'(x_4^*) \approx -3,039 \quad \Rightarrow \quad x_4^* \text{ ist abstoßend}$$

Wir erhalten also einen abstoßenden 2er-Zyklus

$$Z_1^* = \{x_3^*, x_4^*\} = \{x_3^*, y^1(x_3^*) \neq x_3^*\}; y^2(x_3^*) = x_3^*$$

### Fixpunkte der dritten Iteration

Zunächst stellen wir fest:

- a) Ist  $x_1^*$  Fixpunkt von  $y^1(x)$ , dann ist  $x_1^*$  auch Fixpunkt von  $y^2(x)$  und  $y^3(x)$ .

Denn:

$$y^1(x_1^*) = x_1^*$$

$$y^2(x_1^*) = y^1(y^1(x_1^*)) = y^1(x_1^*) = x_1^*$$

$$y^3(x_1^*) = y^1(y^2(x_1^*)) = y^1(x_1^*) = x_1^*$$

(Entsprechendes gilt für  $x_2^*$ )

- b) Die Zyklus-Elemente  $x_3^*$  und  $x_4^*$  des 2er-Zyklus sind keine Fixpunkte von  $y^3(x)$ .

Denn:

$$y^2(x_3^*) = x_3^* \quad \text{und} \quad y^1(x_3^*) \neq x_3^*$$

$$\Rightarrow y^3(x_3^*) = y^1(y^2(x_3^*)) = y^1(x_3^*) \neq x_3^*$$

(entsprechend für  $x_4^*$ )

Zur numerischen Berechnung des 3er-Zyklus benutzen wir das Programm FOLGE1 mit den Eingabewerten:  $a = -1.76$ ;  $x_0 = 0.2$ ;  $n_{\max} = 200$ .

Die Iteration erreicht einen Endzustand mit dem Zyklus

$$Z_2^* = \{0,0238306; -1,759432; 1,335601\}; 0,0238306$$

$$= \{x_5^*; x_6^*; x_7^*\}; x_5^*$$

$$= \{x_5^*; y^1(x_5^*) \neq x_5^*; y^2(x_5^*) \neq x_5^*\}; y^3(x_5^*) = x_5^*$$

Wir weisen nach:

- a) Die Zyklus-Elemente  $x_5^*$ ,  $y^1(x_5^*)$  und  $y^2(x_5^*)$  sind Fixpunkte von  $y^3(x)$ .

Denn:

$$y^3(x_5^*) = x_5^*$$

$$y^3(y^1(x_5^*)) = y^1(y^3(x_5^*)) = y^1(x_5^*)$$

$$y^3(y^2(x_5^*)) = y^2(y^3(x_5^*)) = y^2(x_5^*)$$

- b) Die Fixpunkte  $x_5^*$ ,  $x_6^*$  und  $x_7^*$  von  $y^3(x)$  bilden einen anziehenden 3er-Zyklus.

Denn:

$$y^3(x) = y^1(y^1(y^1(x))); \quad y^1(x) = x^2 + a$$

Nach der Kettenregel:

$$v = w(z(y(x))); \quad \frac{dv}{dx} = \frac{dw}{dz} \cdot \frac{dz}{dy} \cdot \frac{dy}{dx}$$

folgt:

$$y^{3'}(x) = y^{1'}(y^2(x)) \cdot y^{1'}(y^1(x)) \cdot y^{1'}(x)$$

$$= 2y^2(x) \cdot 2y^1(x) \cdot 2x$$

$$= 8(x^4 + 2ax^2 + a(a+1))(x^2 + a)x$$

Mit  $y^3(x_5^*) = x_5^*$  erhalten wir:

$$y^{3'}(x_5^*) = y^{3'}(y^1(x_5^*)) = y^{3'}(y^2(x_5^*))$$

D.h.: Die Ableitungen von  $y^3(x)$  stimmen in den Punkten des 3er-Zyklus überein.

Numerisch ergibt sich:

$$y^{3'}(x_5^*) \approx -0,447 \quad \Rightarrow \quad Z_2^* = \{x_5^*, x_6^*, x_7^*\} \text{ ist anziehend}$$

Die Fixpunktgleichung  $y^3(x) - x = 0$  ist eine Gleichung 8. Grades und besitzt demnach 8 Lösungen, von denen wir bisher 5 gefunden haben. Es ergibt sich die Frage nach den weiteren 3 Lösungen? Dazu setzen wir ein weiteres Rechenprogramm ein.

### Programm FIXPKT1

Mit diesem Programm kann man nach dem Prinzip der Intervallschachtelung die Gleichung  $y^m(x) - x = 0$  numerisch lösen. Dazu wird ein Intervallraster vorgegeben, in dem nach den Lösungen gesucht wird. Wenn keine Fixpunkte im Intervall gefunden werden, dann wird 0 ausgegeben. Wenn das Raster zu grob gewählt wird, dann kann es vorkommen, daß ein Fixpunkt nicht gefunden wird. Das ist dann der Fall, wenn in einem Intervall mehrere Fixpunkte liegen.

Mit den Eingabewerten:  $a = -1.76$ ;  $m = 3$ ;  $x_{\min} = -2$ ;  $x_{\max} = +2$  und der Anzahl der Teilintervalle  $k = 20$ , ergibt sich folgender Ausdruck:

Fixpunktsuche 3 -te Iteration, $a = -1.76$	
Intervall	Fixpunkt
-----	
[-2.000 , -1.800]	0.000000
[-1.800 , -1.600]	-1.759432 → $x_6^*$
[-1.600 , -1.400]	0.000000
[-1.400 , -1.200]	0.000000
[-1.200 , -1.000]	0.000000
[-1.000 , -0.800]	-0.917745 → $x_2^*$
[-0.800 , -0.600]	0.000000
[-0.600 , -0.400]	0.000000
[-0.400 , -0.200]	0.000000
[-0.200 , 0.000]	-0.133203 → $x_9^*$
[ 0.000 , 0.200]	0.023831 → $x_5^*$
[ 0.200 , 0.400]	0.000000
[ 0.400 , 0.600]	0.000000
[ 0.600 , 0.800]	0.000000
[ 0.800 , 1.000]	0.000000
[ 1.000 , 1.200]	0.000000
[ 1.200 , 1.400]	1.275460 → $x_8^*$
[ 1.400 , 1.600]	0.000000
[ 1.600 , 1.800]	0.000000
[ 1.800 , 2.000]	1.917745 → $x_1^*$

Beliebige Taste zum Fortsetzen drücken

Wir finden 2 weitere zusätzliche Fixpunkte:

$$x_8^* \approx 1,275460$$

$$x_9^* \approx -0,133203$$

Mit dem Programm FOLGE1 ergibt sich schließlich der weitere Fixpunkt

$$x_{10}^* \approx -1,742257$$

Diese Punkte bilden den 3er-Zyklus

$$Z_3^* = \{x_8^*, x_9^*, x_{10}^*\}; y^1(x_{10}^*) = x_8^*$$

Dieser Zyklus ist abstoßend wie eine numerische Rechnung zeigt:

$$y^3(x_8^*) = y^3(x_9^*) = y^3(x_{10}^*) \approx 2,368$$

### Weitere Anmerkungen

- 1) Der Einzugsbereich für den anziehenden 3er-Zyklus ist gegeben durch:

$$A(Z_2^*) = \{x_0 \mid |x_0| < 1,9177\}$$

- 2) Wenn zur Überprüfung eines abstoßenden Zyklus das Programm FOLGE1 eingesetzt wird, dann können nur die ersten Iterationsschritte ausgewertet werden, weil sonst wegen der begrenzten Rechengenauigkeit, die Iteration schließlich gegen den anziehenden Zyklus strebt (Sensitivität).
- 3) Für  $a = -1,7763$  ergibt sich eine Periodenverdopplung mit einem anziehenden 6er-Zyklus, wie mit dem Programm ORBIT1 nachgewiesen werden kann.

### 3. Komplexe Zahlen

#### 3.1 Einführende Überlegungen

Bisher haben wir uns damit beschäftigt, die Iteration  $x_{n+1} = x_n^2 + a$  im Bereich der reellen Zahlen zu untersuchen. Reelle Zahlen lassen sich als Punkte auf der Zahlengeraden darstellen. Die Erweiterung der reellen Zahlen durch die komplexen Zahlen führt zu Punkten in einer Ebene und wir erwarten dadurch neue Einsichten. Aber wie kommt man zu den komplexen Zahlen? Wir wollen einen anschaulichen Weg beschreiben und nur die Gesetzmäßigkeiten behandeln, die für das weitere Verständnis notwendig sind.

Wir gehen zunächst von der Gleichung  $x^2 = 4$  aus. Diese Gleichung wird im Reellen gelöst durch die beiden Wurzeln  $x_{1,2} = \pm\sqrt{4} = \pm 2$ , denn es gilt als Probe:  $(+2)(+2) = 4$  und  $(-2)(-2) = 4$ .

Außerdem gilt das Wurzelgesetz  $\sqrt{a \cdot b} = \sqrt{a} \cdot \sqrt{b}$ .

Beispiel:  $\sqrt{4 \cdot 9} = \sqrt{36} = 6$  andererseits  
 $= \sqrt{4} \cdot \sqrt{9} = 2 \cdot 3 = 6$

Wie sieht nun die Lösung der Gleichung  $x^2 = -4$  aus? Offensichtlich gibt es keine reellen Zahlen als Lösungen, die diese Gleichung erfüllen. Formal schreiben wir wie im Reellen:

$$x_{1,2} = \pm\sqrt{-4} = \pm\sqrt{4 \cdot (-1)} = \pm\sqrt{4} \cdot \sqrt{-1} = \pm 2 \cdot \sqrt{-1}$$

Für den Ausdruck  $\sqrt{-1}$  führen wir nun das mathematische Symbol  $i$  ein und nennen es die imaginäre Einheit. Sie ist definiert durch:

$$\boxed{i^2 = -1}$$

Damit erhalten wir als Lösungen für unsere Gleichung  $x^2 = -4$  die beiden Werte:

$$x_{1,2} = \pm 2i$$

Wir machen die Probe:

$$(+2i)(+2i) = 4 i^2 = -4 \quad \text{und}$$

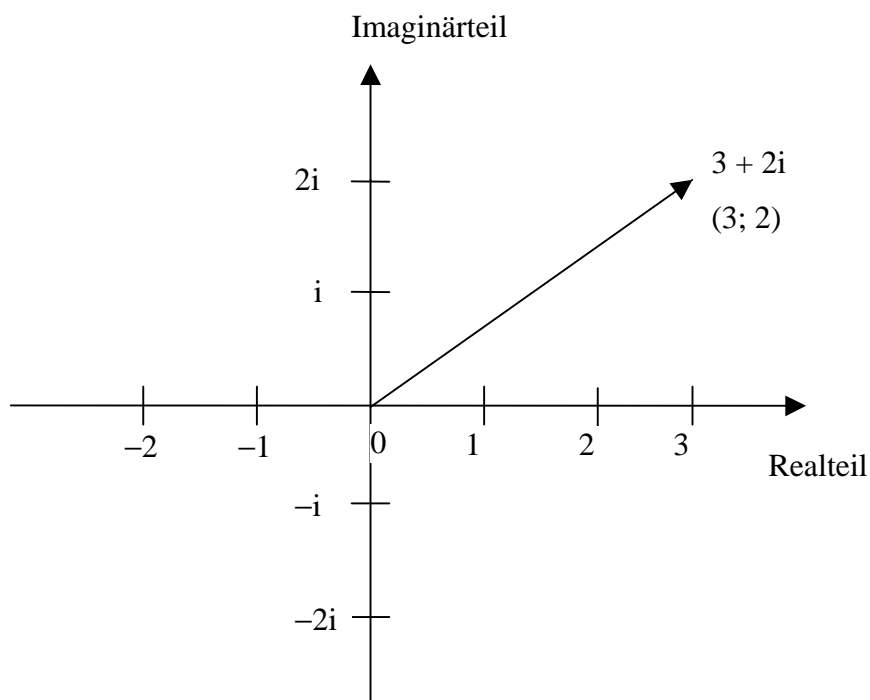
$$(-2i)(-2i) = 4 i^2 = -4$$

Die Zahlen  $2 \cdot i$  oder allgemein  $b \cdot i$ , mit  $b$  reell, nennt man imaginäre Zahlen. Die Zahlen  $z = a + bi$ , mit  $a, b$  reell, nennt man komplexe Zahlen. Dabei ist  $a$  der Realteil und  $b$  der Imaginärteil der komplexen Zahl  $z$ .

Die komplexen Zahlen stellen eine Erweiterung der reellen Zahlen dar, mit denen man ähnlich wie mit den reellen Zahlen rechnen kann und die das Problem der Wurzel aus einer negativen Zahl lösen. Insbesondere ergeben sich die Rechengesetze der reellen Zahlen als Spezialfälle aus den Rechengesetzen für komplexe Zahlen.

### 3.2 Gaußsche Zahlenebene

Da eine komplexe Zahl aus zwei Komponenten besteht, kann man sie nach Carl Friedrich Gauß (1777 – 1855) in einem ebenen Koordinatensystem darstellen. Wir betrachten als Beispiel die Zahl  $z = 3 + 2i$ .





Die Zahl  $z = 3 + 2i$  können wir auch in Komponentenschreibweise als geordnetes Zahlenpaar darstellen:  $z = (3; 2)$ . Für die imaginäre Einheit gilt dann insbesondere die folgende Schreibweise:

$$i = (0; 1)$$

### 3.3 Addition

Welche Rechenregeln gelten nun für komplexe Zahlen ?

Sind die 2 komplexen Zahlen

$$z_1 = a_1 + b_1 i = (a_1; b_1)$$

$$z_2 = a_2 + b_2 i = (a_2; b_2)$$

gegeben, dann ist folgende Definition für die Addition der beiden Zahlen  $z_1$  und  $z_2$  naheliegend:

$$z_1 + z_2 = a_1 + a_2 + (b_1 + b_2) i$$

$$(a_1; b_1) + (a_2; b_2) = (a_1 + a_2; b_1 + b_2)$$

### 3.4 Multiplikation

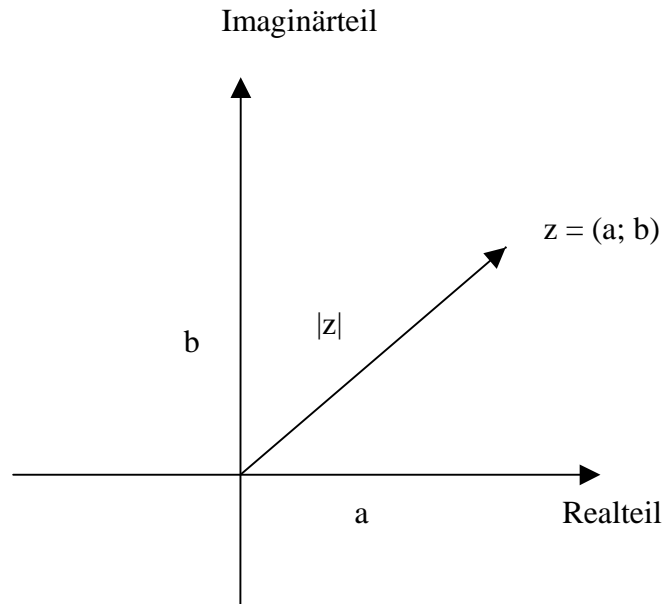
Bei der Definition der Multiplikation zweier komplexer Zahlen lassen wir uns von den Rechengesetzen wie im Reellen leiten, benutzen dabei nur die Tatsache, daß  $i^2 = -1$  ist. Dann ergibt sich:

$$\begin{aligned} z_1 \cdot z_2 &= (a_1 + b_1 i) (a_2 + b_2 i) \\ &= a_1 a_2 + a_1 b_2 i + a_2 b_1 i + b_1 b_2 i^2 \\ &= a_1 a_2 - b_1 b_2 + (a_1 b_2 + a_2 b_1) i \end{aligned}$$

$$(a_1; b_1) \cdot (a_2; b_2) = (a_1 a_2 - b_1 b_2; a_1 b_2 + a_2 b_1)$$

### 3.5 Betrag einer komplexen Zahl

Die "Größe" einer komplexen Zahl wird durch ihren Betrag angegeben. Wir betrachten folgende Darstellung in der Gaußschen Zahlenebene:



Der Betrag einer komplexen Zahl ist definiert als die Länge des Zeigers  $z$ . Nach dem Satz des Pythagoras ist:

$$|z| = \sqrt{a^2 + b^2}$$

### 3.6 Wurzel aus einer komplexen Zahl

Bei unseren weiteren Untersuchungen benötigen wir noch die Wurzel aus einer komplexen Zahl. Wir gehen aus von der Gleichung:

$$\sqrt{p+iq} = u+iv$$

Die Wurzel aus einer komplexen Zahl  $(p+iq)$  wird im allgemeinen wieder eine komplexe Zahl  $(u+iv)$  ergeben, wobei Real- und Imaginärteil  $u$  bzw.  $v$  gesucht sind.

Durch Quadrieren der Gleichung erhalten wir:

$$\begin{aligned} p+iq &= (u+iv)^2 = (u+iv)(u+iv) \\ &= u^2 - v^2 + 2iuv \end{aligned}$$

Der Vergleich von Real- und Imaginärteil ergibt das Gleichungssystem

$$\begin{aligned} p &= u^2 - v^2 \\ q &= 2uv \end{aligned}$$

mit den Unbekannten  $u$  und  $v$ .

Bei der Lösung müssen wir folgende Fälle unterscheiden:  $q = 0$  und  $q \neq 0$ .

1)  $q = 0$

$\Rightarrow u = 0; v \neq 0$  oder  $u \neq 0; v = 0$ . Der Fall  $u = 0$  und  $v = 0$  ist trivial.

a)  $u = 0; v \neq 0$

$$\Rightarrow p = -v^2 < 0$$

(Das Quadrat einer Zahl ist immer positiv, damit ist  $p < 0$ .)

$$\Rightarrow v_{1,2} = \pm\sqrt{-p}$$

b)  $u \neq 0; v = 0$

$$\Rightarrow p = u^2 > 0$$

$$\Rightarrow u_{1,2} = \pm\sqrt{p}$$

2)  $\boxed{q \neq 0}$ 

$$\Rightarrow u \neq 0; v \neq 0$$

Damit können wir schreiben:

$$v = \frac{q}{2u}$$

Eingesetzt in die erste Gleichung des Gleichungssystems ergibt:

$$p = u^2 - \frac{q^2}{4u^2}$$

Durch eine Umformung erhalten wir:

$$u^4 - pu^2 - \frac{q^2}{4} = 0$$

Diese Gleichung 4. Grades wird zunächst gelöst durch:

$$(u^2)_{1,2} = \frac{p}{2} \pm \sqrt{\frac{p^2}{4} + \frac{q^2}{4}} > 0$$

Da das Quadrat größer 0 ist, betrachten wir nur das "+"-Zeichen und erhalten:

$$u^2 = \frac{1}{2} \left( p + \sqrt{p^2 + q^2} \right)$$

$$\Rightarrow u_{1,2} = \pm \sqrt{\frac{1}{2} \left( p + \sqrt{p^2 + q^2} \right)}$$

Zusammenfassung:

$\sqrt{p+iq} = u+iv$	
$q = 0$	
$p \leq 0$	$p > 0$
$u = 0$ $v_{1,2} = \pm \sqrt{-p}$	$u_{1,2} = \pm \sqrt{p}$ $v = 0$
$q \neq 0$	
$u_{1,2} = \pm \sqrt{\frac{1}{2} \left( p + \sqrt{p^2 + q^2} \right)}$  $v = \frac{q}{2u}$	

Beispiel: Berechnung von  $\sqrt{-3}$

$$p = -3; q = 0$$

$$\Rightarrow u = 0; v_{1,2} = \pm\sqrt{3}$$

$$\Rightarrow \sqrt{-3} = \pm i\sqrt{3}$$

Rechenalgorithmus Programm KWURZEL1

```
'-----
'Quellprogramm: KWURZEL1.BAS
'-----
'Berechnung der Wurzel aus einer komplexen
'Zahl
'-----
'Eingabewerte:
p = 1                      'Wurzel (p + iq)
q = 1
'-----

CLS

IF q = 0 THEN
  IF p <= 0 THEN
    u1 = 0: u2 = 0
    v1 = SQR(-p): v2 = -v1
  ELSE
    u1 = SQR(p): u2 = -u1
    v1 = 0: v2 = 0
  END IF
ELSE
  u1 = SQR((p + SQR(p ^ 2 + q ^ 2)) / 2): u2 = -u1
  IF u1 = 0 THEN v1 = SQR(-p) ELSE v1 = q / 2 / u1
  v2 = -v1
END IF

PRINT "WURZEL ("; p; " + "; q; "*i)"
PRINT "Z1 = ("; u1; "; "; v1; ")"
PRINT "Z2 = ("; u2; "; "; v2; ")"

END
```

Ergebnis Ausdruck:

WURZEL ( 1 + 1\*i)

Z1 = ( 1.098684 ; .4550899 )

Z2 = (-1.098684 ; -.4550899 )

## 4. Iterationen im Komplexen

### 4.1 Ein Einführungsbeispiel – Die Iteration $z_{n+1} = z_n^2$

Wir beginnen nun damit, die Iterationen, die wir bisher im Reellen untersucht haben, ins Komplexe zu übertragen. Ein einfaches Studienobjekt, das aber schon wesentliche Zusammenhänge zeigt, ist die einfache Quadrierung einer komplexen Zahl. Die Iterationsgleichung lautet:

$$z_{n+1} = z_n^2$$

Dabei sind nun aber  $z_{n+1}$  und  $z_n$  komplexe Zahlen, die wie folgt geschrieben werden können:

$$z_{n+1} = x_{n+1} + iy_{n+1}$$

$$z_n = x_n + iy_n$$

Einsetzen ergibt:

$$\begin{aligned} x_{n+1} + iy_{n+1} &= (x_n + iy_n)^2 = (x_n + iy_n)(x_n + iy_n) \\ &= x_n^2 - y_n^2 + 2ix_ny_n \end{aligned}$$

Durch Gleichsetzen der Real- und Imaginärteile erhalten wir 2 Iterationsgleichungen für reelle Zahlen:

$$\begin{array}{ll} x_{n+1} = x_n^2 - y_n^2 & \text{(Realteil)} \\ y_{n+1} = 2x_ny_n & \text{(Imaginärteil)} \end{array}$$

Diese beiden Gleichungen beschreiben die Bahn eines Punktes in der komplexen Ebene.

Wie im Reellen erhalten wir aus der Fixpunktgleichung

$$z^2 - z = 0; \quad z(z - 1) = 0$$

die Fixpunkte

$$z_1^* = 0 \quad \text{und} \quad z_2^* = 1$$

Für das Attraktionsverhalten können wir ebenfalls die Ergebnisse aus dem Reellen übernehmen:

$$y(z) = z^2; \quad y'(z) = 2z$$

$$y'(z_1^*) = 0 \quad \Rightarrow \quad z_1^* \text{ ist anziehend}$$

$$y'(z_2^*) = 2 \quad \Rightarrow \quad z_2^* \text{ ist abstoßend}$$

Konkret kann diese Iteration mit dem Programm KFOLGE1 numerisch durchgeführt werden.

Eingabeparameter sind z.B. für den komplexen Startwert  $z_0 = (x_0; y_0)$ :  $x_0 = 0.2$ ;  $y_0 = 0.97$ ;

maximale Anzahl der Iterationen  $n_{\max} = 15$ . Es ergibt sich der folgende Ausdruck:

n	x(n)	y(n)	Betrag von z(n)
0	0.2000000	0.9700000	0.9904040
1	-.9009001	0.3880000	0.9809000
2	0.6610769	-.6990985	0.9621649
3	-.0517160	-.9243157	0.9257613
4	-.8516849	0.0956038	0.8570340
5	0.7162271	-.1628487	0.7345073
6	0.4864616	-.2332732	0.5395010
7	0.1822285	-.2269569	0.2910613
8	-.0183022	-.0827160	0.0847167
9	-.0065070	0.0030278	0.0071769
10	0.0000332	-.0000394	0.0000515
11	-.0000000	-.0000000	0.0000000
12	-.0000000	0.0000000	0.0000000
13	0.0000000	-.0000000	0.0000000
14	0.0000000	0.0000000	0.0000000
15	0.0000000	0.0000000	0.0000000

Beliebige Taste zum Fortsetzen drücken

Wie man sieht, streben die Zahlenfolgen  $x_n, y_n$  mit  $|z_n| = \sqrt{x_n^2 + y_n^2}$  gegen den Fixpunkt  $z_1^* = 0$ .

Das Konvergenzverhalten dieser Iteration kann mit diesem Programm durch weitere Experimente überprüft werden.

Besonders interessant ist die Betrachtung der Einzugsbereiche:

Zunächst ist der Einzugsbereich des anziehenden Fixpunktes  $z_1^* = 0$  gegeben durch:

$$A(z_1^*) = \{z \mid |z| < 1\}$$

Der Einzugsbereich des Punktes  $\infty$  kann angegeben werden durch:

$$A(\infty) = \{z \mid |z| > 1\}$$

Der Grenzbereich, an den diese beiden Bereiche aneinanderstoßen, ist die Menge

$$J = \{z \mid |z| = 1\}$$

Dies ist eine Julia-Menge (benannt nach dem französischen Mathematiker Gaston Julia (1893 – 1978)), die in unserem Fall eine glatte Kurve, nämlich den Einheitskreis, darstellt und nun in der Zahlenebene die Eigenschaft eines Scheidekreises hat. Wir erinnern uns an die Scheidepunkte im Reellen und sehen nun die flächenhafte Erweiterung.

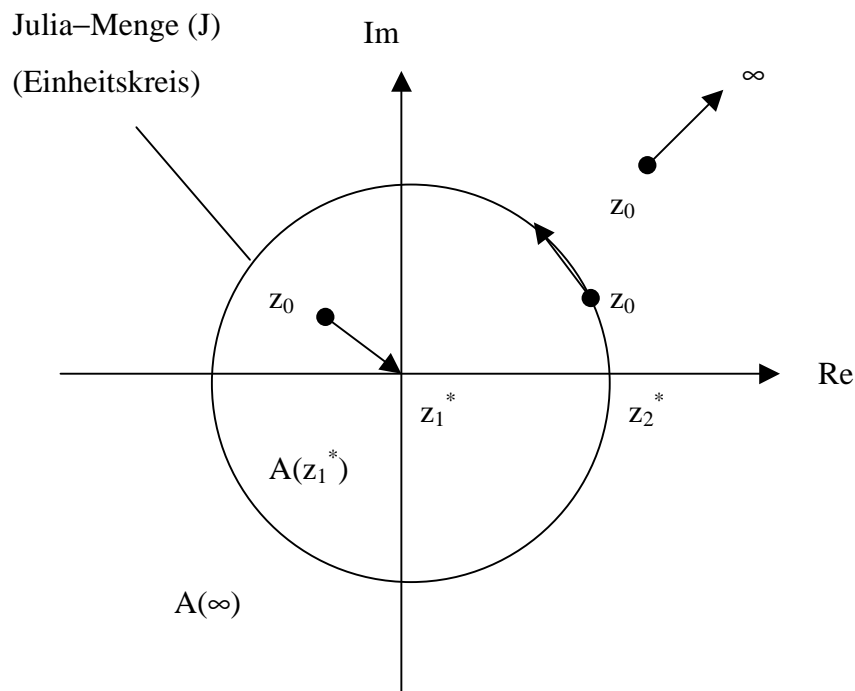
Startpunkte innerhalb der Julia-Menge konvergieren gegen den Fixpunkt  $z_1^*$  und Startpunkte außerhalb der Julia-Menge divergieren gegen Unendlich.

Die Menge

$$K = \{z \mid |z| \leq 1\}$$

nennt man auch die ausgefüllte Julia-Menge. Damit ist die Julia-Menge der Rand der ausgefüllten Julia-Menge:

$$J = \text{Rd}(K)$$





Es ist zusätzlich festzuhalten:

- a) Der anziehende Fixpunkt  $z_1^*$  liegt im Inneren der ausgefüllten Julia-Menge:

$$z_1^* \in \{z \mid |z| < 1\} = A(z_1^*)$$

- b) Der Einzugsbereich des anziehenden Fixpunktes  $A(z_1^*)$  ist identisch mit dem Innern der ausgefüllten Julia-Menge  $K$ , wir schreiben dafür  $\underline{K}$ :

$$A(z_1^*) = \{z \mid |z| < 1\} = \underline{K}$$

- c) Der abstoßende Fixpunkt  $z_2^*$  liegt in der Julia-Menge:

$$z_2^* \in J$$

- d) Punkte in der Julia-Menge werden bei der Iteration wieder in Punkte der Julia-Menge abgebildet. Die Julia-Menge wird somit in sich selbst abgebildet ( $J \rightarrow J$ ). Man sagt, die Julia-Menge ist gegenüber der Iteration invariant.

Den letzten Punkt wollen wir mit unserem Programm KFOLGE1 überprüfen. Als Startwert

$z_0 = (x_0; y_0)$  setzen wir einen Punkt des Einheitskreises mit  $y_0 = \sqrt{1 - x_0^2}$  ein. Dann ergibt sich folgender Ausdruck:

n	x(n)	y(n)	Betrag von z(n)
0	0.2000000	0.9797959	1.0000000
1	-.9200000	0.3919184	0.9999999
2	0.6927999	-.7211298	0.9999999
3	-.0400564	-.9991973	0.9999999
4	-.9967907	0.0800485	0.9999998
5	0.9871839	-.1595833	0.9999995
6	0.9490653	-.3150761	0.9999989
7	0.8014520	-.5980555	0.9999978
8	0.2846549	-.9586255	0.9999956
9	-.8379344	-.5457549	0.9999913
10	0.4042857	0.9146137	0.9999825
11	-.6730712	0.7395304	0.9999651
12	-.0938804	-.9955133	0.9999301
13	-.9822332	0.1869184	0.9998603
14	0.9298436	-.3671950	0.9997206
15	0.7297770	-.6828679	0.9994413

Beliebige Taste zum Fortsetzen drücken

Wir sehen, daß die Iteration des Startpunktes  $z_0$  auf dem Einheitskreis mit  $|z_n| = 1$  herumwandert. Nach einer gewissen Anzahl von Iterationen stimmt das wegen der

Rechengenauigkeit des Computers nicht mehr. Die Sensitivität von den Anfangsbedingungen macht sich bemerkbar. Einen genauen Nachweis kann man erbringen, wenn man die Iteration statt in kartesischen Koordinaten in Polarkoordinaten durchführt. Die Iterationsfolge zeigt dann ein typisch chaotisches Verhalten.

## 4.2 Die allgemeine Iteration $z_{n+1} = z_n^2 + c$

### Iterationsgleichungen

Mit der komplexen Iterationsgleichung

$$z_{n+1} = z_n^2 + c$$

erreichen wir einen gewissen Abschluß, was den allgemeinen Formelausbau betrifft. Die komplexen Größen der Iterationsgleichung sind gegeben durch:

$$z_{n+1} = x_{n+1} + iy_{n+1}$$

$$z_n = x_n + iy_n$$

$$c = a + ib$$

Eingesetzt ergibt:

$$\begin{aligned} x_{n+1} + iy_{n+1} &= (x_n + iy_n)^2 + a + ib \\ &= x_n^2 - y_n^2 + a + i(2x_n y_n + b) \end{aligned}$$

Damit erhält man die allgemeinen Iterationsgleichungen für den Real- und Imaginärteil:

$$\begin{aligned} x_{n+1} &= x_n^2 - y_n^2 + a \\ y_{n+1} &= 2x_n y_n + b \end{aligned}$$

### Fixpunkte der Ersten und Zweiten Iterierten

Mit unseren erarbeiteten Kenntnissen über komplexe Zahlen können wir die folgende Berechnung der Fixpunkte vornehmen. Aus der Fixpunktgleichung

$$z^2 - z + c = 0$$

ergeben sich die Fixpunkte zunächst in der Form:

$$z_{1,2}^* = \frac{1}{2} \left( 1 \pm \sqrt{1 - 4c} \right)$$

Dabei ist  $c = a + ib$  nun komplex. Es ergibt sich damit:

$$z_{1,2}^* = \frac{1}{2} \left( 1 \pm \sqrt{1 - 4a - 4ib} \right)$$

Wir setzen nun:  $p = 1 - 4a$  und  $q = -4b$

und erhalten damit:

$$z_{1,2}^* = \frac{1}{2}(1 \pm \sqrt{p+iq})$$

Die Wurzel aus einer komplexen Zahl haben wir gelöst (siehe 3.6) durch:

$$\sqrt{p+iq} = u_{1,2} + iv_{1,2}$$

Damit folgt:

$$\begin{aligned} z_{1,2}^* &= \frac{1}{2}(1 \pm (u_{1,2} + iv_{1,2})) \\ &= \frac{1}{2}(1 \pm u_{1,2}) \pm \frac{1}{2}iv_{1,2} \end{aligned}$$

Da  $u_2 = -u_1$  und

$v_2 = -v_1$  ist,

folgt schließlich für die Fixpunkte:

$$\begin{aligned} z_1^* &= \frac{1}{2}(1 + u_1) + \frac{1}{2}iv_1 \\ z_2^* &= \frac{1}{2}(1 - u_1) - \frac{1}{2}iv_1 \end{aligned}$$

Entsprechend kann man die Fixpunkte der Zweiten Iterierten explizit berechnen.

$$z_{3,4}^* = -\frac{1}{2}(1 \pm \sqrt{p+iq})$$

mit

$$p = -3 - 4a; \quad q = -4b; \quad \sqrt{p+iq} = u_{1,2} + iv_{1,2}$$

und damit:

$$\begin{aligned} z_3^* &= -\frac{1}{2}(1 + u_1) - \frac{1}{2}iv_1 \\ z_4^* &= -\frac{1}{2}(1 - u_1) + \frac{1}{2}iv_1 \end{aligned}$$

### Definition der Julia- Menge

Bei der vorhergehenden Untersuchung der Iteration  $z_{n+1} = z_n^2$  haben wir festgestellt, daß die Iterationsfolge des Startpunktes  $z_0$  innerhalb der ausgefüllten Julia-Menge gegen den Grenzpunkt  $z^*$  und außerhalb gegen Unendlich strebt.

Wir definieren nun im allgemeinen Fall die ausgefüllte Julia-Menge  $K$  als die Menge aller komplexer Startpunkte  $z_0$ , für die die Iterationsfolge der Funktion  $y(z)$  beschränkt bleibt, und die Julia-Menge  $J$  als deren Rand.

$K = \{z_0 \mid (y^n(z_0)) \text{ beschränkt}\}$ $J = \text{Rd}(K)$
---

Die Menge  $J$  besteht somit aus Punkten, in deren beliebiger Umgebung sowohl Punkte von  $K$  als auch Punkte, die nicht zu  $K$  gehören, enthalten sind.

$$z \in \text{Rd}(K) \Leftrightarrow \forall \varepsilon > 0: \exists x \in U_\varepsilon(z), x \in K \wedge \exists y \in U_\varepsilon(z), y \notin K$$

### Divergenzkreis

Da wir noch nicht wissen, welche Eigenschaften die Julia-Menge im allgemeinen Fall besitzt, versuchen wir zunächst einen Divergenzkreis

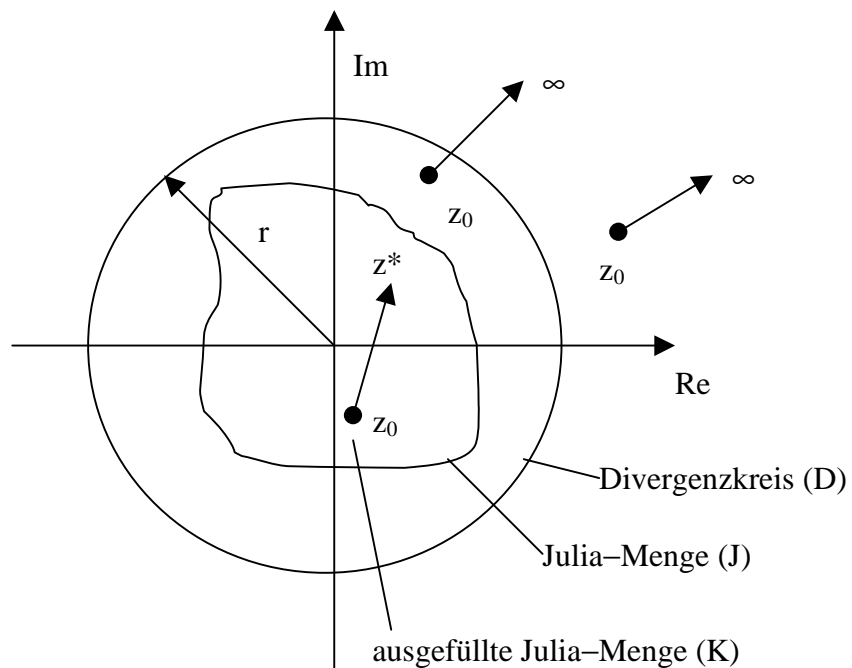
$$D = \{z \mid |z| = r\}$$

mit einem bestimmten Radius  $r$  abzuschätzen mit der folgenden Eigenschaft:

Alle Anfangspunkte  $z_0$  außerhalb des Divergenzkreises sollen sicher gegen Unendlich divergieren.

$$(y^n(z_0)) \xrightarrow{n \rightarrow \infty} \infty \quad \text{für} \quad |z_0| > r$$

Die Zahl  $r$  wird auch Fluchtschranke genannt.



Damit muß die ausgefüllte Julia-Menge  $K$  innerhalb von  $D$  liegen.

$$K \subseteq \{z \mid |z| \leq r\}$$

### Abschätzung der Fluchtschranke

Für die Iterationsfunktion  $y(z) = z^2 + c$  können wir folgende Fluchtschranke  $r_1$  angeben:

$$r_1 = 2 + |c|$$

Denn es gilt:

$$\begin{aligned} |y(z)| &= |z^2 + c| \geq |z|^2 - |c| = \left(|z| - \frac{|c|}{|z|}\right) |z| \geq \\ &\geq (|z| - |c|) |z|, \quad \text{da } |z| \geq 1 \end{aligned}$$

Für  $|z| \geq 2 + |c|$  bzw.  $|z| - |c| \geq 2$  folgt somit:

$$|y(z)| \geq 2|z|$$

Es gilt entsprechend:

$$|y^2(z)| = |y(y(z))| \geq 2|y(z)| \geq 2^2 |z|$$

$$|y^3(z)| = |y(y^2(z))| \geq 2|y^2(z)| \geq 2^3 |z|$$

.....

$$|y^n(z)| = |y(y^{n-1}(z))| \geq 2|y^{n-1}(z)| \geq 2^n |z|$$

Für  $n \rightarrow \infty$  strebt  $2^n$  gegen Unendlich, somit ergibt sich der Grenzwert

$$\lim_{n \rightarrow \infty} y^n(z) = \infty \quad \text{für} \quad |z| \geq 2 + |c|$$

Die Zahl  $r_1 = 2 + |c|$  ist somit eine Fluchtschranke für  $y(z) = z^2 + c$ .  $\diamond$

Es gibt noch weitere Abschätzungen für eine Fluchtschranke, z.B.:

$$r_2 = \frac{1}{2} \left( 1 + \sqrt{1 + 4|c|} \right) \quad \text{oder}$$

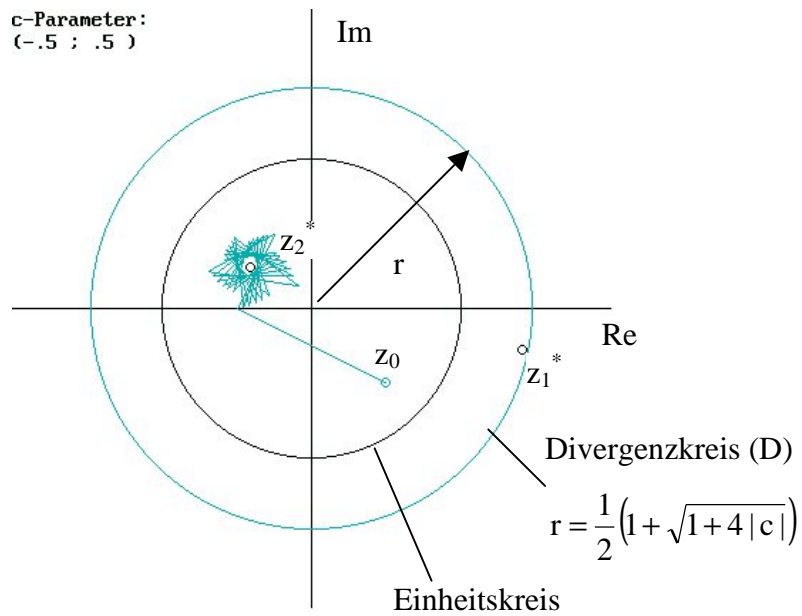
$$r_3 = \text{Max}(|c|, 2)$$

mit  $r_2 \leq r_3 \leq r_1$ .  $r_2$  ist somit die schärfste Abschätzung.

### Programm KORBIT1

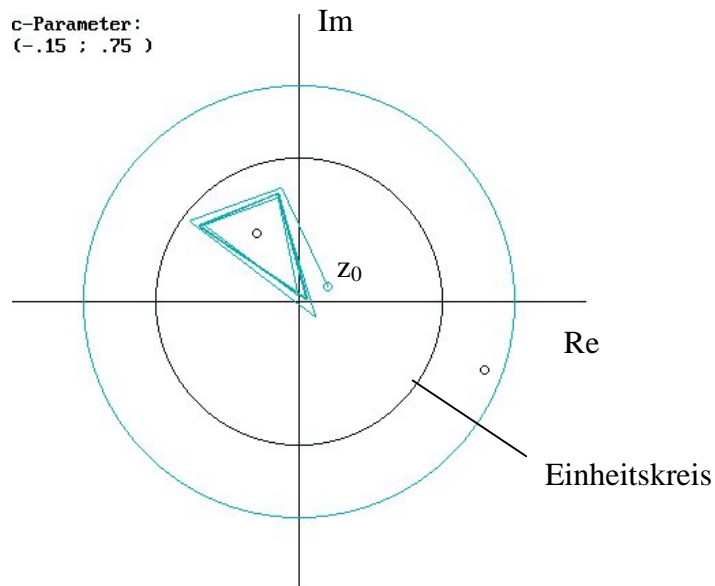
Um das Grenzverhalten der komplexen Iteration zu studieren, können mit diesem Programm die Fixpunkte der ersten Iterierten, der Divergenzkreis und die Bahn einer komplexen Iteration berechnet und grafisch dargestellt werden. Eingabewerte sind z.B.: Parameter  $c = (a, b)$ :  $a = -0.5$ ;  $b = 0.5$ ; Startwert  $z_0 = (x_0, y_0)$ :  $x_0 = 0.5$ ;  $y_0 = -0.5$ ; maximale Anzahl von Iterationen  $n_{\max} = 30$ .

Bei dem folgenden Ausdruck sieht man, wie der anziehende Fixpunkt langsam von der Iteration angenähert wird. Der Anfangspunkt  $z_0 = (0.5; -0.5)$  gehört in diesem Fall also zur ausgefüllten Julia-Menge.



Beliebige Taste zum Fortsetzen drücken

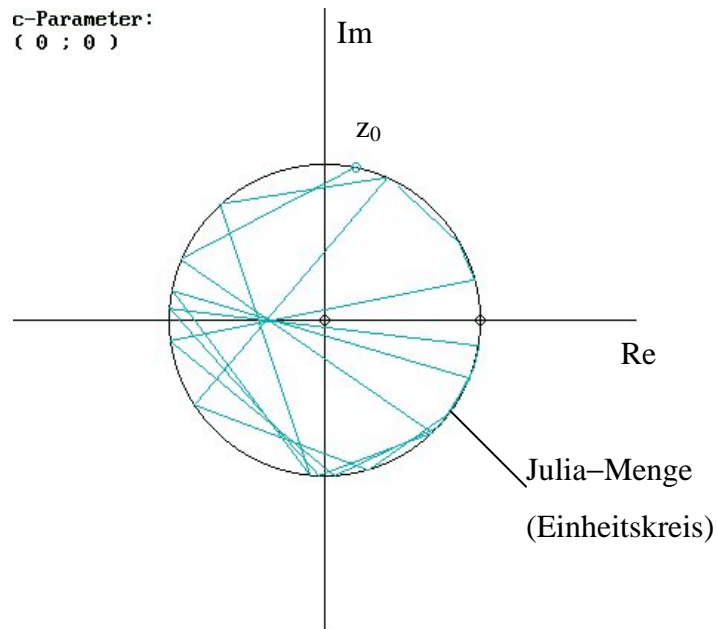
Mit diesem Programm können wir aber auch studieren, wie ein 3-er Zyklus im Komplexen entsteht. Wir geben ein:  $a = -0.15$ ;  $b = 0.75$ ;  $x_0 = 0.2$ ;  $y_0 = 0.1$ ;  $n_{\max} = 100$ . Die Iteration springt in einem Dreieck:



Beliebige Taste zum Fortsetzen drücken



Es besteht auch die Möglichkeit, das Iterationsverhalten der speziellen Iteration  $z_{n+1} = z_n^2$  aus dem vorhergehenden Kapitel zu veranschaulichen. Wir setzen:  $a = 0$ ;  $b = 0$  und wählen einen Punkt aus der Julia-Menge (Einheitskreis):  $x_0 = 0.2$ ;  $y_0 = \text{SQR}(1-x_0^2)$ ;  $n_{\max} = 20$ .



Beliebige Taste zum Fortsetzen drücken

Man kann sehr schön sehen, wie die Iterationspunkte innerhalb der Julia-Menge (Einheitskreis) herumwandern und chaotisch verteilt sind.

### 4.3 Darstellung der Julia–Menge durch Umkehriteration

Bisher wissen wir noch nicht, wie im allgemeinen Fall eine Julia–Menge überhaupt aussieht. Man kann sie zwar definieren, so wie wir es getan haben, aber deren Aussehen übersteigt jede Vorstellungskraft. Wir sind damit in einer Situation wie der Mathematiker Gaston Julia vor etwa 85 Jahren. Erst der Einsatz von Computern mit entsprechenden Grafik–Darstellungen machte es möglich, diese abstrakte Menge zu visualisieren. Das Ergebnis war mehr als überraschend.

Die Grundidee ist dabei die folgende: Am Anfang unserer Betrachtungen (siehe 2.2) haben wir festgestellt, daß sich bei einer Umkehriteration das Attraktionsverhalten umkehrt. Wir erwarten, daß bei der Umkehriteration anziehende Fixpunkte und auch anziehende Zyklen abstoßend werden, insbesondere der Fixpunkt  $\infty$ . Auf der anderen Seite werden abstoßende Fixpunkte und abstoßende periodische Punkte, die in der Julia–Menge liegen, bei der Umkehriteration anziehend. Zusammenfassend können wir festhalten: Die Julia–Menge wird bei der Umkehriteration anziehend.

#### Berechnungsformeln der Umkehriteration

Bei der Umkehriteration zur Iteration  $z_{n+1} = z_n^2 + c$  bestimmen wir den Vorgänger (das Urbild)  $z_n$  zum Iterationspunkt  $z_{n+1}$ . Die Berechnung erfolgt nach der Iterationsgleichung:

$$z_{n+1} = \pm \sqrt{z_n - c}$$

Dabei ist:

$$\begin{aligned} z_{n+1} &= x_{n+1} + iy_{n+1} \\ z_n &= x_n + iy_n \quad \text{und} \\ c &= a + ib \end{aligned}$$

Eingesetzt:

$$\begin{aligned} x_{n+1} + iy_{n+1} &= \pm \sqrt{x_n + iy_n - a - ib} \\ &= \pm \sqrt{x_n - a + i(y_n - b)} \end{aligned}$$

Wir setzen

$$p_n = x_n - a \quad \text{und} \quad q_n = y_n - b$$

und erhalten:

$$x_{n+1} + iy_{n+1} = \pm \sqrt{p_n + iq_n}$$

Die komplexe Wurzel wird gelöst durch (siehe 3.6):

$$\sqrt{p_n + iq_n} = u_{n/1,2} + iv_{n/1,2}$$

Damit ist:

$$x_{n+1} + iy_{n+1} = \pm u_{n/1,2} \pm iv_{n/1,2}$$

Da  $u_{n/2} = -u_{n/1}$  und

$$v_{n/2} = -v_{n/1},$$

folgt:

$$x_{n+1} + iy_{n+1} = \pm u_{n/1} \pm iv_{n/1}$$

und damit die Iterationsgleichungen:

$\begin{aligned} x_{n+1} &= \pm u_{n/1}; & u_n &= f(p_n, q_n) = f(x_n, y_n) \\ y_{n+1} &= \pm v_{n/1}; & v_n &= f(p_n, q_n) = f(x_n, y_n) \end{aligned}$
--

## Verfahren

Das Verfahren zur Darstellung der Julia-Menge kann mit folgenden Schritten angegeben werden:

Aufteilung des Bildschirms in ein Punktraster (z.B. 420 x 420 Pixel)

Festlegung des komplexen Bereichs (dynamische z-Ebene,  $z = (x, y)$ ;

$$\text{z.B.: } -2 \leq x \leq +2, -2 \leq y \leq +2)$$

Durchführung der Iteration  $z_n$  für  $n = 1(1)n_{\max}$  (z.B.  $n_{\max} = 100000$ )

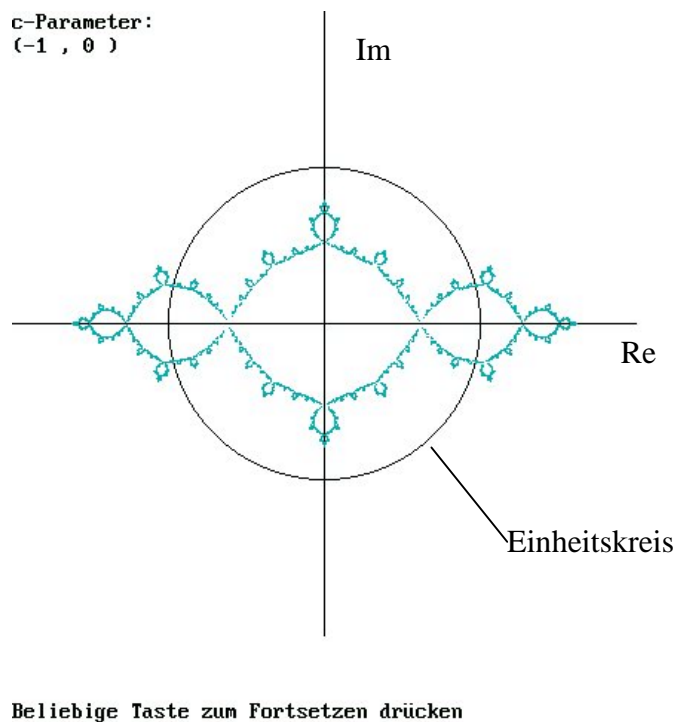
Farbige Darstellung der Gitterpunkte ab einer bestimmten "Einschwingzeit"  $n_0$

(z.B.  $n_0 = 50$ ). Damit wird erreicht, daß ein Punkt nach  $n_0$  Iterationen mit hinreichender Genauigkeit einen Punkt der Julia-Menge darstellt.

Da die Julia-Menge gegenüber der Iteration invariant ist, liegen die folgenden Punkte ebenfalls in der Julia-Menge, wobei der Einfluß der Sensitivität durch die Umkehriteration umgangen wird.

## Programm JULIA1

Mit diesem Programm wird das Verfahren realisiert. Es ergibt sich für die Eingabegrößen  $c = (a, b)$ :  $a = -1$ ;  $b = 0$ ; Startwert  $z_0 = (x_0, y_0)$ :  $x_0 = -0.1$ ;  $y_0 = 0$ ; maximale Anzahl der Iterationen  $n_{\max} = 100000$  das folgende Bild als Beispiel:



Dieses Bild vermittelt einen ersten Eindruck von der fraktalen Struktur der Julia-Menge. Mit dem Programm kann man nun weitere Julia-Mengen für verschiedene Parameter  $c$  berechnen und grafisch darstellen. Speziell erhält man für  $c = 0$  den erwarteten Einheitskreis. Durch Probieren wird man schnell herausfinden, wie vielgestaltig und kompliziert Julia-Mengen sind. Wir werden später noch weitere Algorithmen kennenlernen, mit denen noch mehr Details dargestellt werden können. Dieses Programm soll den ersten, grundsätzlichen Eindruck vermitteln.

Offen bleibt zunächst auch die Frage, für welche  $c$ -Werte sich überhaupt Julia-Mengen und in welcher Gestalt ergeben? Die Antwort erhalten wir durch die später zu behandelnde Mandelbrot-Menge, die ähnlich dem Feigenbaum-Diagramm für reelle Iterationen, Licht in das Verhalten der komplexen Iterationen bringt.

## 4.4 Einkreisung der ausgefüllten Julia–Menge

Wir wollen nun ein weiteres Verfahren kennenlernen, mit dem man Julia–Mengen grafisch darstellen kann. Ausgangspunkt ist die komplexe Iterationsgleichung

$$\begin{aligned} z_{n+1} &= y^1(z_n) = z_n^2 + c \\ &= y^{n+1}(z_0) \end{aligned}$$

mit der Fluchtschranke

$$r = 2 + |c|$$

Wir betrachten nun alle Startpunkte  $z_0$ , deren Betrag kleiner gleich der Fluchtschranke  $r$  ist:

$$|z_0| \leq r,$$

alle anderen Startpunkte streben bei der Iteration gegen Unendlich. Die verbleibenden Punkte fassen wir in einer Menge

$$Q^{(0)} = \{z_0 \mid |z_0| \leq r\}$$

zusammen. Diese Menge stellt in der komplexen Ebene eine Kreisscheibe mit dem Radius  $r$  dar und ist eine erste Näherung für die ausgefüllte Julia–Menge ( $K \subseteq Q^{(0)}$ ).

Führen wir nun eine erste Iteration der verbleibenden Startpunkte  $z_0$  durch, dann werden einige Punkte die Fluchtschranke  $r$  bereits überschreiten und gegen  $\infty$  streben. Für die übrig gebliebenen Punkte ergibt sich die Menge:

$$Q^{(1)} = \{z_0 \mid |z_1| = |y^1(z_0)| \leq r\}$$

Diese Menge  $Q^{(1)}$  liegt innerhalb von  $Q^{(0)}$ :

$$Q^{(1)} \subset Q^{(0)}$$

Weitere Iterationen lassen weitere Punkte ins Unendliche entweichen und es ergeben sich die Mengen:

$$Q^{(2)} = \{z_0 \mid |z_2| = |y^2(z_0)| \leq r\}$$

$$Q^{(3)} = \{z_0 \mid |z_3| = |y^3(z_0)| \leq r\}$$

.....

$$Q^{(n)} = \{z_0 \mid |z_n| = |y^n(z_0)| \leq r\}$$

mit

$$Q^{(n)} \subset Q^{(n-1)} \subset \dots \subset Q^{(2)} \subset Q^{(1)} \subset Q^{(0)}$$

Führen wir einen Grenzübergang durch, dann ergibt sich:

$$\lim_{n \rightarrow \infty} Q^{(n)} = \{z_0 \mid \lim_{n \rightarrow \infty} |z_n| = \lim_{n \rightarrow \infty} |y^n(z_0)| \leq r\}$$

Der Ausdruck  $\lim_{n \rightarrow \infty} |y^n(z_0)| \leq r$  ist per definitionem gleichbedeutend damit, daß die

Iterationsfolge  $(y^n(z_0))$  beschränkt ist. Ein Vergleich mit der Definition der ausgefüllten Julia-Menge

$$K = \{z_0 \mid (y^n(z_0)) \text{ beschränkt}\}$$

zeigt, daß der Grenzwert somit die ausgefüllte Julia-Menge darstellt:

$$\lim_{n \rightarrow \infty} Q^{(n)} = K$$

Mit unserer Mengenkonstruktion nähern wir uns also von außen der Julia-Menge an.

Für den Spezialfall  $c = 0$  kann man diesen Gedankengang explizit nachvollziehen. Es ist:

$$Q^{(0)} = \{z_0 \mid |z_0| \leq 2\}$$

$$Q^{(1)} = \{z_0 \mid |z_0|^2 \leq 2\} = \{z_0 \mid |z_0| \leq 2^{\frac{1}{2}}\}$$

$$Q^{(2)} = \{z_0 \mid |z_0|^4 \leq 2\} = \{z_0 \mid |z_0| \leq 2^{\frac{1}{2^2}}\}$$

$$Q^{(3)} = \{z_0 \mid |z_0|^8 \leq 2\} = \{z_0 \mid |z_0| \leq 2^{\frac{1}{2^3}}\}$$

.....

$$Q^{(n)} = \{z_0 \mid |z_0|^{2^n} \leq 2\} = \{z_0 \mid |z_0| \leq 2^{\frac{1}{2^n}}\}$$

Es ergeben sich also konzentrische, ineinanderliegende Kreisscheiben mit den Radien:

$$2^{\frac{1}{2}}, \quad 2^{\frac{1}{4}}, \quad 2^{\frac{1}{8}}, \quad \dots, \quad 2^{\frac{1}{2^n}}, \quad \dots = \sqrt{2}, \quad \sqrt{\sqrt{2}}, \quad \sqrt{\sqrt{\sqrt{2}}}, \quad \sqrt{\sqrt{\sqrt{\sqrt{2}}}}, \quad \dots$$

Da diese Radienfolge gegen 1 konvergiert, gilt:

$$\lim_{n \rightarrow \infty} Q^{(n)} = \{z_0 \mid |z_0| \leq 1\}$$

Dies ist die ausgefüllte Julia-Menge, die bekanntlich in unserem Fall durch eine Kreisscheibe mit dem Radius 1 dargestellt wird.

## Verfahren

Aufbauend auf diesen Überlegungen können wir ein Verfahren angeben, mit dem die Julia-Menge schrittweise von außen angenähert werden kann.

1. Aufteilung des Bildschirms in ein Punkteraster (z.B. 400 x 400 Pixel)
2. Festlegung des komplexen Bereichs der  $z$ -Ebene (dynamischer Bereich,  $z = (x, y)$ ; z.B.:  $-2 \leq x \leq +2, -2 \leq y \leq +2$ )
3. Festlegung einer Fluchtschranke  $r$  (z.B.  $r = 2 + |c|$ )
4. Wahl einer Anzahl von Iterationen  $n_{\max}$  (z.B.  $n_{\max} = 0, 1, 2, 3, \dots, 7, 100$ )
5. Für jeden Gitterpunkt prüfe, ob für  $n = 1(1)n_{\max}$  die Bedingung  $|z_n| = |y^n(z_0)| \leq r$  erfüllt ist. Wenn das der Fall ist, dann stelle den Punkt  $z_0 = (x_0, y_0)$  als Punkt der angenäherten, ausgefüllten Julia-Menge dar.

## Level- Set- Algorithmus

Nach dem Verfahren ergibt sich ein Algorithmus, dessen Kernstück mit folgendem Programmteil prinzipiell beschrieben werden kann. (Die Umrechnung der mathematischen Koordinaten in Bildschirmkoordinaten wurde nicht berücksichtigt, siehe Anhang 6.2.):

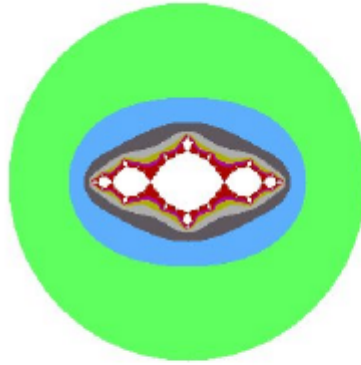
```
'Berechnung der ausgefüllten Julia-Menge durch
'Abtasten der Gitterpunkte mit Test auf Mengenzugehörigkeit
'Vorgabewerte: Parameter  $c = (a, b)$  und Fluchtschranke  $r$ 

FOR x = xmin TO xmax STEP xschritt      'GitterSpalten
  FOR y = ymin TO ymax STEP yschritt    'GitterZeilen
    x1 = x: y1 = y
    FOR n = 1 TO nmax                    'Iterationsschleife
      xx = x1 ^ 2 - y1 ^ 2 + a           'Hilfsgroesse
      y1 = 2 * x1 * y1 + b               'Imaginaerteil
      x1 = xx                             'Realteil
      IF x1 ^ 2 + y1 ^ 2 > r ^ 2 THEN EXIT FOR
      'Punkt gehoert nicht zur ausgefüllten Julia-Menge
    NEXT n
    'Punktausgabe
    IF x1 ^ 2 + y1 ^ 2 <= r ^ 2 THEN PSET (x, y)
    'Punkt gehoert zur angenaeherten, ausgefüllten Julia-
    'Menge
  NEXT y
NEXT x
```

## Programm JULIA2

Mit dem vollständigen Programm erhält man mit den Eingabewerten für  $c = (a, b)$ :  $a = -1$  und  $b = 0$  die folgende Grafik:

c-Parameter:  
(-1 , 0 )

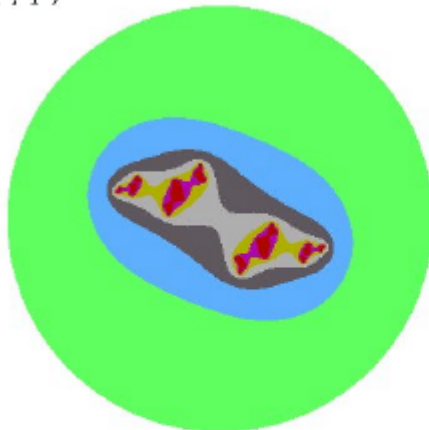


Beliebige Taste zum Fortsetzen drücken

Dies ist ein Beispiel für eine zusammenhängende Julia-Menge.

Mit den Eingabewerten  $a = -1$  und  $b = 1$  erhalten wir ein Bild, bei dem Einschnürungen und nicht zusammenhängene Teilmengen entstehen.

c-Parameter:  
(-1 , 1 )



Beliebige Taste zum Fortsetzen drücken



Wir erhalten damit das wichtige Ergebnis, daß ausgefüllte Julia–Mengen entweder zusammenhängend oder unzusammenhängend sein können.

Mit diesem anschaulichen Ergebnis wird man zu folgender Definition geführt:

Die Menge  $K$  ist unzusammenhängend, wenn sie sich in zwei nichtleere, elementfremde (disjunkte) Teilmengen zerlegen läßt.

Oder in Mengenschreibweise:

$$K \text{ unzusammenhängend} \Leftrightarrow \exists Q_1, Q_2 \neq \emptyset: K \subseteq (Q_1 \cup Q_2) \wedge K \cap Q_1, K \cap Q_2 \neq \emptyset \wedge Q_1 \cap Q_2 = \emptyset$$

## 4.5 Fixpunkte, Zyklen und Einzugsbereiche

Wir haben im vorhergehenden Kapitel die Julia-Menge von außen schrittweise angenähert. Welche Eigenschaften besitzt das Innere der Julia-Menge ? An einem speziellen Beispiel haben wir gesehen (siehe 4.1), daß sich im Innern der ausgefüllten Julia-Menge ein anziehender Fixpunkt befindet. Wir wollen nun experimentell zeigen, daß generell anziehende Fixpunkte und anziehende Zyklen stets im Innern der ausgefüllten Julia-Menge liegen.

Anziehende oder abstoßende Fixpunkte können wir explizit nur bis zur 2. Iterierten berechnen. Bei Zyklen mit höherer Periodenzahl sind wir auf numerische Näherungsverfahren angewiesen. Im folgenden wird ein Algorithmus angegeben, der ähnlich dem Feigenbaum-Diagramm im Reellen den Endzustand der nun komplexen Iteration bestimmt. Das Rechenschema sieht in der Programmiersprache QuickBasic wie folgt aus:

```
'Iteration zur naeherungsweisen Fixpunktberechnung
'Vorgabewerte: Parameter c = (a, b) und Fluchtschranke r

x = 0: y = 0                                'Startwerte
FOR i = 1 TO 1000                            'Iterationsschleife
  xx = x ^ 2 - y ^ 2 + a                    'Iterationsgleichungen
  y = 2 * x * y + b
  x = xx
  IF x ^ 2 + y ^ 2 > r ^ 2 THEN EXIT FOR    'Abbruchbedingung
  sx = (x - xmin) / xschritt                'Koordinaten-
  zy = (ymax - y) / yschritt                'transformationen
  IF i > 900 THEN CIRCLE (sx, zy), 3
  'Fuer den Endzustand Kreispunkte zeichnen
NEXT i
```

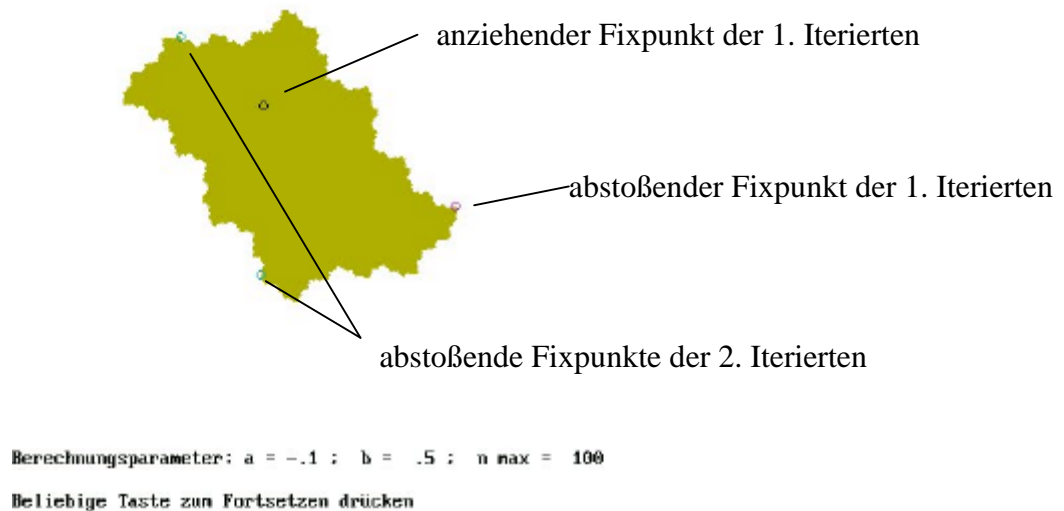
### Programm JULIA3

Mit diesem Programm wird zunächst die Julia-Menge nach dem Level-Set-Algorithmus berechnet, anschließend die explizit angebbaren anziehenden und abstoßenden Fixpunkte der 1. und 2. Iterierten und schließlich nach dem Näherungsverfahren die anziehenden Fixpunkte mit höherer Periodenzahl.

Wir wollen die folgenden drei Fälle beispielhaft studieren:

#### A. Ein anziehender Fixpunkt

Mit den Eingabedaten  $a = -0.1$ ;  $b = 0.5$ ;  $n_{\max} = 100$  ergibt sich folgendes Bild:



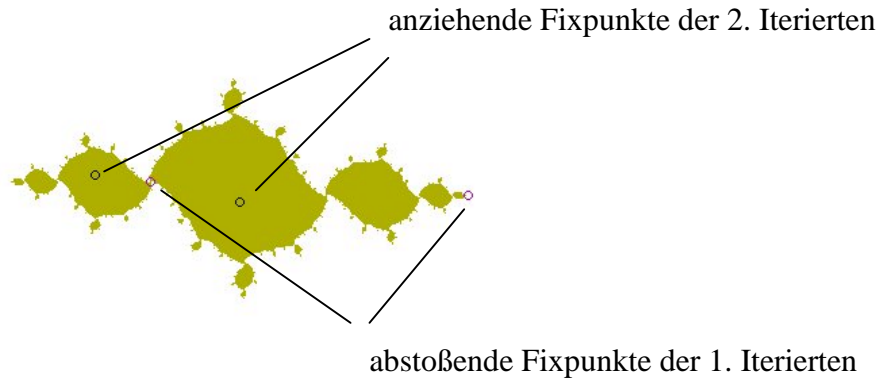
Der anziehende Fixpunkt der 1. Iterierten liegt im Innern der Julia-Menge, die abstoßenden Fixpunkte der 1. und 2. Iterierten auf dem Rand der ausgefüllten Julia-Menge also in der Julia-Menge. Der Einzugsbereich des anziehenden Fixpunktes  $z^*$  ist das Innere der ausgefüllten Julia-Menge  $\underline{K}$ :

$$A_y(z^*) = \underline{K}$$

#### B. Zwei anziehende Fixpunkte (2er-Zyklus)

Eingabedaten:  $a = -1$ ;  $b = 0.1$ ;  $n_{\max} = 100$

Es ergeben sich 2 anziehende Fixpunkte eines 2er-Zyklus im Innern der Julia-Menge. Diese Punkte sind auch Lösungen der Fixpunktgleichung für die 2. Iterierte.



Berechnungsparameter:  $a = -1$  ;  $b = .1$  ;  $n_{\max} = 100$

Beliebige Taste zum Fortsetzen drücken

Ist  $z^*$  der anziehende Fixpunkt von  $y^2(z)$ , dann ist der 2er-Zyklus gegeben durch:

$$Z^* = \{z^*, y(z^*)\}; y^2(z^*) = z^*$$

Für die Einzugsbereiche von  $z^*$  und  $y(z^*)$  schreiben wir:

$$A_{y^2}(z^*) \text{ und } A_{y^2}(y(z^*))$$

Es gilt dann:

$$A_y(Z^*) = A_{y^2}(z^*) \cup A_{y^2}(y(z^*)) = \underline{K}$$

$A_y(Z^*)$  nennt man den Einzugsbereich des anziehenden Zyklus  $Z^*$ . Er ist identisch mit dem Innern der ausgefüllten Julia-Menge  $\underline{K}$ .

Interessant ist die Lage der beiden abstoßenden Fixpunkte der 1. Iterierten, sie liegen in der Julia-Menge. Der eine abstoßende Fixpunkt trennt die beiden Einzugsbereiche.

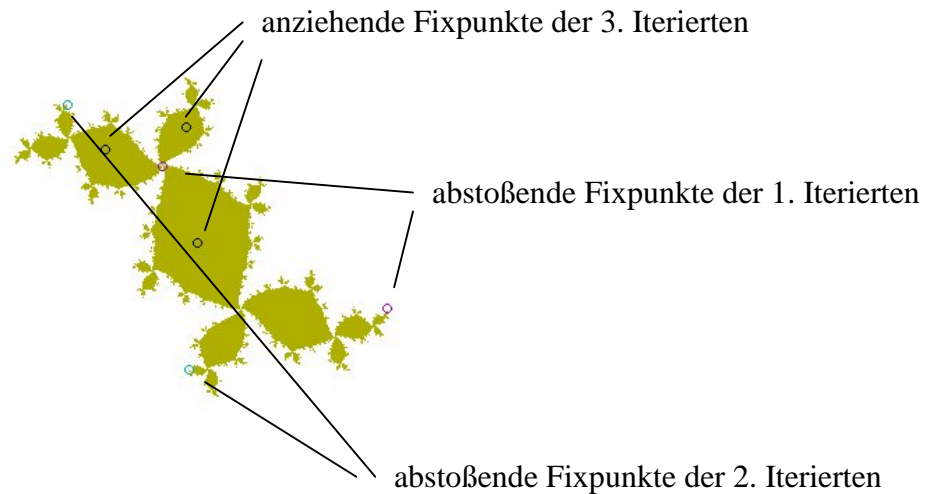
### C. Drei anziehende Fixpunkte (3er-Zyklus)

Eingabedaten:  $a = -0.1$ ;  $b = 0.75$ ;  $n_{\max} = 100$

In diesem Fall erhalten wir einen anziehenden 3er-Zyklus

$$Z^* = \{z^*, y(z^*), y^2(z^*)\}; y^3(z^*) = z^*$$

Die Elemente  $z^*$ ,  $y(z^*)$  und  $y^2(z^*)$  sind anziehende Fixpunkte der 3. Iterierten und ergeben sich nach dem Näherungsverfahren. Sie liegen innerhalb der Julia-Menge.



Berechnungsparameter: a = -.1 ; b = .75 ; n max = 100

Beliebige Taste zum Fortsetzen drücken

Für den Einzugsbereich des Zyklus  $Z^*$  gilt:

$$A_y(Z^*) = A_{y^3}(z^*) \cup A_{y^3}(y(z^*)) \cup A_{y^3}(y^2(z^*)) = \underline{K}$$

Der eine abstoßende Fixpunkt der 1. Iterierten trennt die Einzugsbereiche

Allgemein kann man für den Einzugsbereich eines anziehenden Zyklus der Periode  $m$

$$Z^* = \{z^*, y(z^*), y^2(z^*), \dots, y^{m-1}(z^*)\}; y^m(z^*) = z^*$$

nun schreiben:

$$A_y(Z^*) = \bigcup_{k=0}^{m-1} A_{y^m}(z_k^*) = \underline{K}$$

mit  $z_0^* = z^*, z_1^* = y(z^*), z_2^* = y^2(z^*), \dots, z_{m-1}^* = y^{m-1}(z^*)$

Bezeichnen wir den Einzugsbereich des Punktes  $\infty$  mit  $A_y(\infty)$ , dann ist die Julia-Menge  $J$  sowohl der Rand von  $A_y(\infty)$  als auch von  $A_y(Z^*)$  mit den  $m$  Einzugsbereichen des Zyklus.

$$J = \text{Rd}(A_y(\infty)) = \text{Rd}(A_y(Z^*))$$

Dies ist der Grund für die komplizierte Struktur der Julia-Menge.

### Programm JULIA4

Wir wollen noch ein weiteres Experiment durchführen, indem wir der Frage nachgehen, was bei der Iteration mit Punkten passiert, die der Julia-Menge angehören ?

Bei dem folgenden Programm gehen wir wie folgt vor:

1. Wir berechnen die ausgefüllte Julia-Menge mit dem Level-Set-Algorithmus.
2. Wir berechnen, wie wir es unter 4.3 getan haben, einen Punkt der Julia-Menge mittels Umkehriteration mit dem untenstehenden Algorithmus.
3. Wir benutzen diesen Punkt als Startpunkt für die anschließenden (Vorwärts-) Iterationen.

```
'Berechnung eines Punktes der Julia-Menge
'durch Umkehriteration

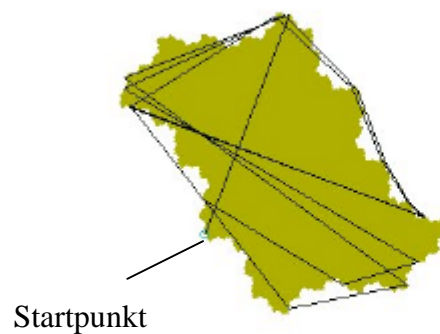
x = .01: y = 0                                'Startwerte
FOR i = 1 TO n0                                'n0 > 50, legt den Punkt in der
                                                'Julia-Menge fest

    'Loesung der Wurzel
    p = x - a: q = y - b
    IF q = 0 THEN
        IF p <= 0 THEN
            u = 0: v = SQR(-p)
        ELSE
            u = SQR(p): v = 0
        END IF
    ELSE
        u = SQR((p + SQR(p ^ 2 + q ^ 2)) / 2)
        IF u = 0 THEN v = SQR(-p) ELSE v = q / 2 / u
    END IF
    x = u: y = v
    'zufaellige Auswahl der beiden Wurzeln
    IF RND > .5 THEN x = -x: y = -y
NEXT i

'Koordinatentransformation
sx = (x - xmin) / xschritt
zy = (ymax - y) / yschritt

'Punktausgabe
CIRCLE (sx, zy), 3, 12
'Zeichne einen Kreispunkt als Punkt der Julia-Menge
```

Der Ergebnisausdruck für  $a = 0$ ;  $b = 0.5$ ; und  $n_0 = 65$  ( $n_0$  legt den Startpunkt in der Julia-Menge für die nachfolgende Iteration fest) zeigt, wie der Startpunkt, der in der Julia-Menge liegt, bei der Iteration chaotisch in dieser Menge herumwandert. Damit wurde die vollständige Invarianz der Julia-Menge ( $J \leftrightarrow J$ ) gegenüber der Iteration auch im allgemeinen Fall anschaulich gemacht.



Berechnungsparameter:  $a = 0$  ;  $b = .5$   
 Beliebige Taste zum Fortsetzen drücken

## 4.6 Die Mandelbrot–Menge

Eines der faszinierendsten Fraktale der modernen Mathematik ist die Mandelbrot–Menge, die nach dem französisch–amerikanischen Mathematiker Benoit B. Mandelbrot (geb. 1924 in Warschau) benannt ist. Sie besitzt nicht nur eine äußerst komplizierte Struktur, sondern erklärt auch auf anschauliche Weise die unterschiedlichen Formen der Julia–Mengen.

Wir wollen einige Gedankengänge aufzeigen, wie man zu dieser Menge kommt. Dazu betrachten wir zunächst die Iteration

$$z_{n+1} = y_c^{n+1}(z_0)$$

für den Startpunkt  $z_0 = 0$ . Dann erhalten wir mit  $z_{n+1} = z_n^2 + c$  die Folge:

$$z_0 = 0$$

$$z_1 = c$$

$$z_2 = c^2 + c$$

$$z_3 = (c^2 + c)^2 + c$$

$$\dots\dots\dots$$

Den Punkt  $z_0 = 0$  nennt man in diesem Fall den kritischen Punkt und die dazugehörige Folge  $(z_n)$  die kritische Bahn.

### Definition der Mandelbrot- Menge

Wir definieren nun die Mandelbrot–Menge  $M$  als die Menge aller Parameterwerte  $c$ , für die die kritische Bahn beschränkt bleibt, also nicht ins Unendliche strebt.

$$M = \{c \mid (y_c^n(0)) \text{ beschränkt}\}$$

Ein Vergleich dieser Definition mit der Definition der ausgefüllten Julia–Menge

$$K = \{z \mid (y_c^n(z)) \text{ beschränkt}\}$$

zeigt die wesentlichen Unterschiede:



Während die Elemente von  $K$  in der  $z$ -Ebene liegen, die wir auch als dynamische Ebene bezeichnen, liegen die Elemente von  $M$  in der  $c$ -Ebene, die auch Parameterebene genannt wird. Zur Berechnung von  $M$  wird die Iteration generell mit  $z_0 = 0$  bzw.  $z_1 = c$  gestartet.

Es gilt nun der wichtige Satz:

Ist der Parameterwert  $c$  ein Element der Mandelbrot-Menge  $M$ , genau dann ist die zugehörige ausgefüllte Julia-Menge zusammenhängend.

$c \in M \iff K \text{ zusammenhängend}$
--

### Beschränktheit

Im Kapitel 4.2 haben wir bei der Iteration  $y_c^n(z)$  eine Fluchtschranke

$$r = \text{Max}(|c|, 2)$$

angegeben. Das bedeutet, daß die Folge  $(y_c^n(z))$  gegen  $\infty$  strebt, wenn  $|z| \geq |c|$  und  $|z| > 2$  ist.

In unserem Fall können wir setzen:

$$z = z_1 = c$$

D.h., wenn die Bedingungen  $|c| \geq |c|$  (trivial !) und  $|c| > 2$  erfüllt sind, dann strebt

$$(y_c^n(c)) \rightarrow \infty$$

und das zugehörige  $c$  gehört nicht zur Mandelbrot-Menge ( $c \notin M$ ).

Damit erhalten wir die folgende Abschätzung für die Mandelbrot-Menge:

$M \subset \{c \mid  c  \leq 2\}$
-----------------------------------

Die Mandelbrot-Menge liegt also in der  $c$ -Ebene in einer Kreisscheibe mit  $|c| \leq 2$ .

## Darstellung der Mandelbrot- Menge

Mit diesen Ergebnissen können wir zur Darstellung der Mandelbrot–Menge ein Verfahren angeben, das ähnlich wie der Level–Set–Algorithmus aufgebaut ist:

1. Aufteilung des Bildschirms in ein Punkteraster (z.B. 600 x 400 Pixel)
2. Festlegung des komplexen Bereichs der  $c$ –Ebene, in der die Mandelbrot–Menge liegt (Parameterbereich; z.B.:  $c = (a, b)$ :  $-2 \leq a \leq +1$ ,  $-1 \leq b \leq +1$ )
3. Festlegung einer maximalen Anzahl von Iterationen  $n_{\max}$  (z.B.  $n_{\max} = 100$ )
4. Für jeden Gitterpunkt prüfe, ob für  $n = 1(1)n_{\max}$  die Bedingung  $|z_n| = |y_c^n(0)| \leq 2$  erfüllt ist. Wenn das der Fall ist, dann stelle den Punkt  $c = (a, b)$  als Punkt der angenäherten Mandelbrot–Menge dar.

## Programm MANDEL1

Mit den Eingabeparametern für den  $c$ –Bereich:

$$a_{\min} = -2; \quad a_{\max} = +1$$

$$b_{\min} = -1;$$

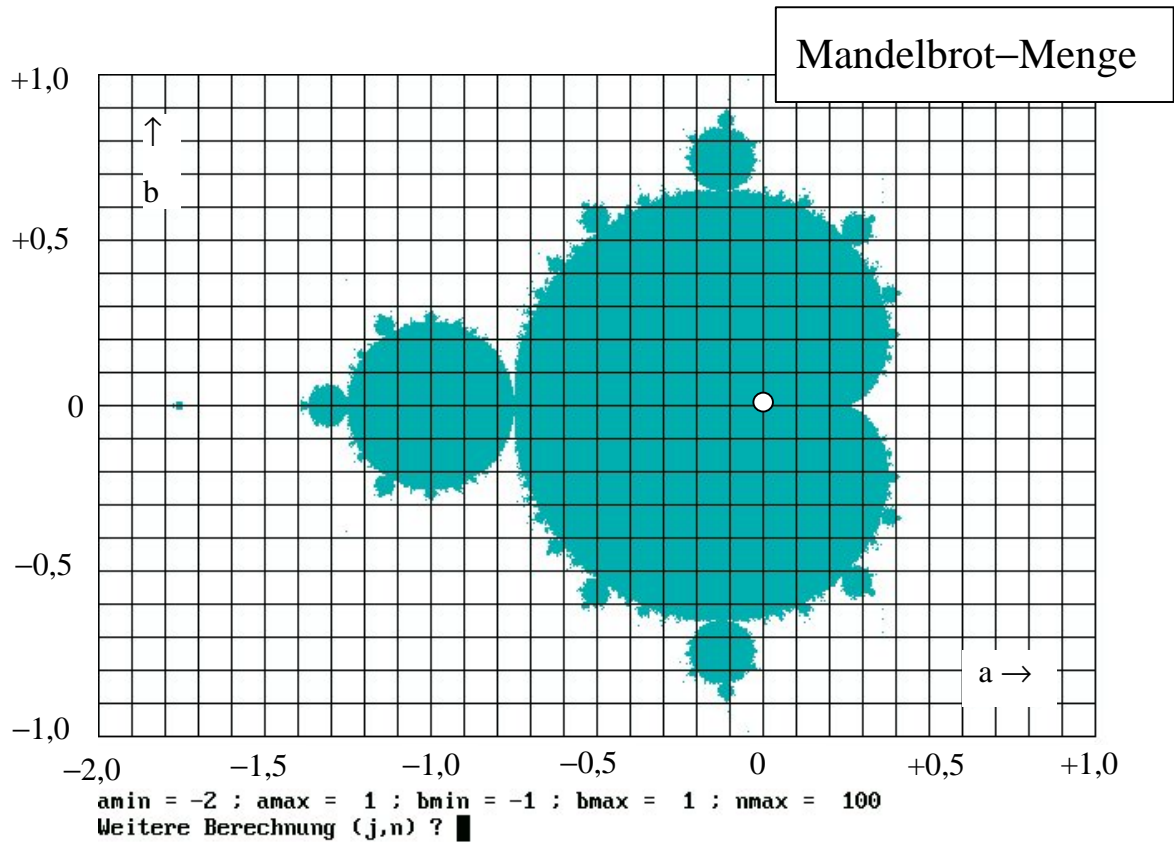
$$\text{und der Anzahl der Iterationen } n_{\max} = 100$$

erhalten wir das berühmte "Apfelmännchen", die Mandelbrot–Menge.

Sie besteht im wesentlichen aus dem zykliden Hauptkörper  $W_0$  im Bereich

$-0,75 \leq a \leq 0,4$  und im Bereich  $-1,25 \leq a \leq -0,75$  aus dem kreisförmigen Nebenkörper  $W_1$ .

Dann schließen sich einzelne Knospen an, die sich weiter fortsetzen und damit den komplexen Rand dieser Menge ergeben.



### **Mandelbrot- Menge und Feigenbaum- Diagramm**

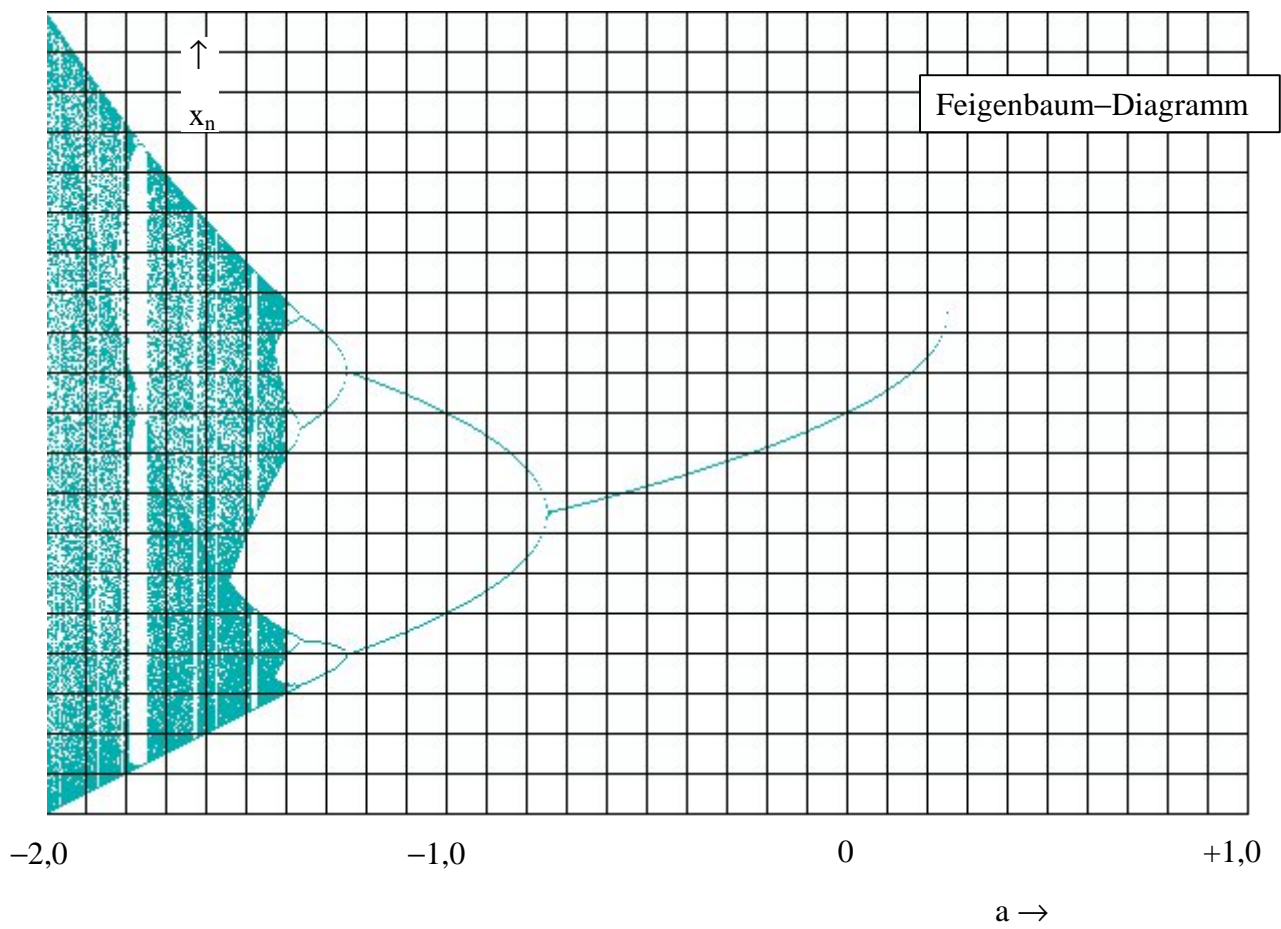
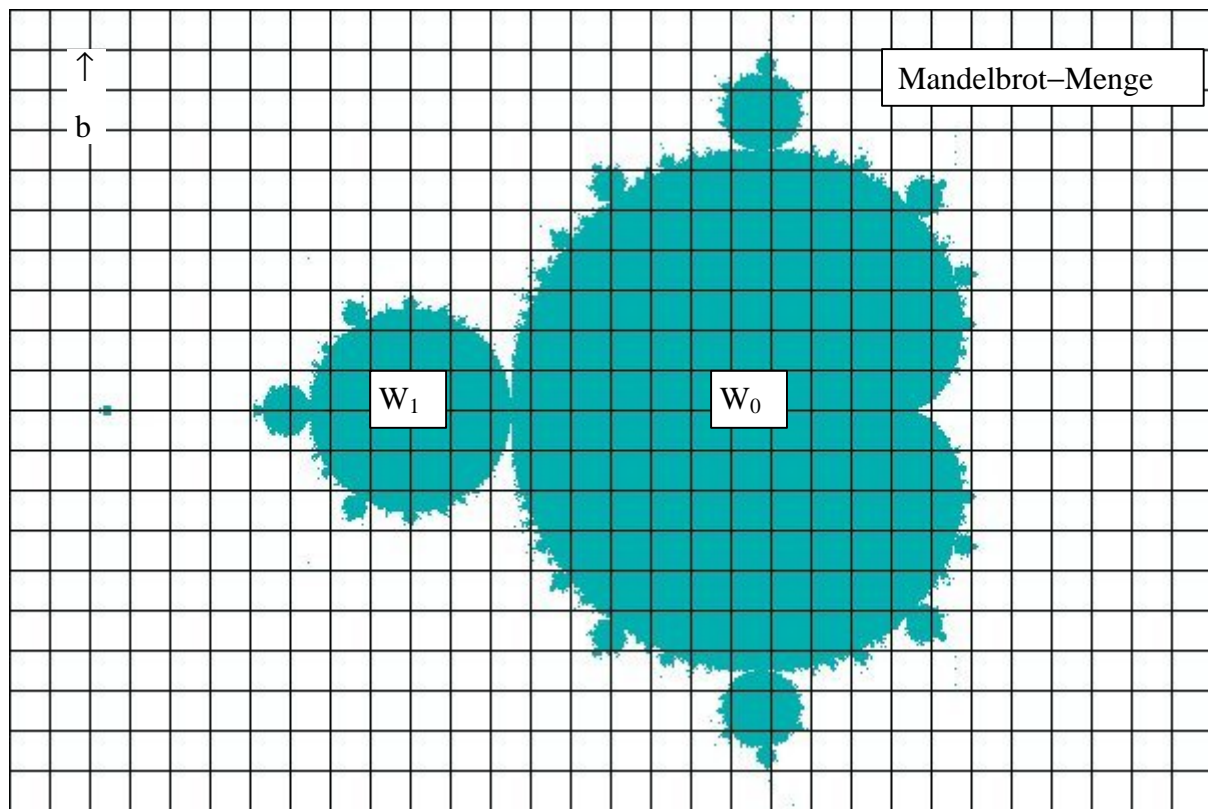
Wir haben schon früher darauf hingewiesen, daß zwischen dem Feigenbaum–Diagramm, das wir im Reellen erhalten haben, und der komplexen Mandelbrot–Menge ein enger Zusammenhang besteht. Das Feigenbaum–Diagramm stellt nämlich den reellen Teil der Mandelbrot–Menge mit den Iterationsergebnissen im Endzustand dar. Auf der folgenden Seite sind diese beiden Diagramme mit dem gemeinsamen Parameter  $a$  untereinandergestellt.

Wenn wir das Feigenbaum–Diagramm von rechts nach links betrachten, dann sehen wir zunächst eine Linie, die einen anziehenden Fixpunkt darstellt. Dies entspricht bei der Mandelbrot–Menge dem zykliden Hauptkörper  $W_0$ .

Dann erfolgt im Feigenbaum–Diagramm eine Aufspaltung in zwei Linien, die den 2er–Zyklus darstellen. In der Mandelbrot–Menge ergibt sich im Komplexen in diesem Bereich der kreisförmige Nebenkörper  $W_1$ .

Wir erkennen, daß der Verdopplungsbaum im Reellen einem System von kleiner werdenden Knospen in der Mandelbrot–Menge entspricht.

Dem periodischen Fenster der Periode 3, das wir im Reellen ausführlich diskutiert haben, können wir nun eine sekundäre Mandelbrot–Menge an der Spitze zuordnen.



## 4.7 Einige morphologische Betrachtungen

Wir sind nun gut vorbereitet, eine Reise in die phantastische Welt der Fraktale anzutreten. Dabei sehen wir nicht nur bizarre Grafiken, sondern verstehen auch etwas den theoretischen Hintergrund. Bei unserer Reise werden wir aber auch an Grenzen stoßen, die immer schwieriger mathematisch zu beschreiben oder überhaupt noch nicht verstanden und erforscht sind.

Als Führer auf unserer Reise dient uns die Mandelbrot–Menge als Landkarte, die für uns einen Kontinent im Ozean mit einer komplizierten Küstenlinie wiedergibt. Die Reise unternehmen wir mit unserem Programm JULIA3, mit dem wir die "Sehenswürdigkeiten" auf dem Bildschirm sichtbar machen können. Als Iterationstiefe wählen wir bei allen Beispielen  $n_{\max} = 100$ .

Und nun zu den Stationen unserer Reise:

<b>c</b>	
<b>a</b>	<b>b</b>
0	0

Startpunkt ist der Nullpunkt im Innern des Hauptkörpers  $W_0$  der Mandelbrot–Menge. Wir erhalten eine Kreisscheibe. Es ist erstaunlich, daß diese einfache Grafik die Keimzelle für die vielgestaltigen Fraktale dieser Familie darstellt.

<b>c</b>	
<b>a</b>	<b>b</b>
−0,3	+0,4

Wenn wir uns im Hauptkörper  $W_0$  der Mandelbrot–Menge bewegen, dann ergeben sich zusammenhängende Julia–Mengen mit einem anziehenden Fixpunkt.

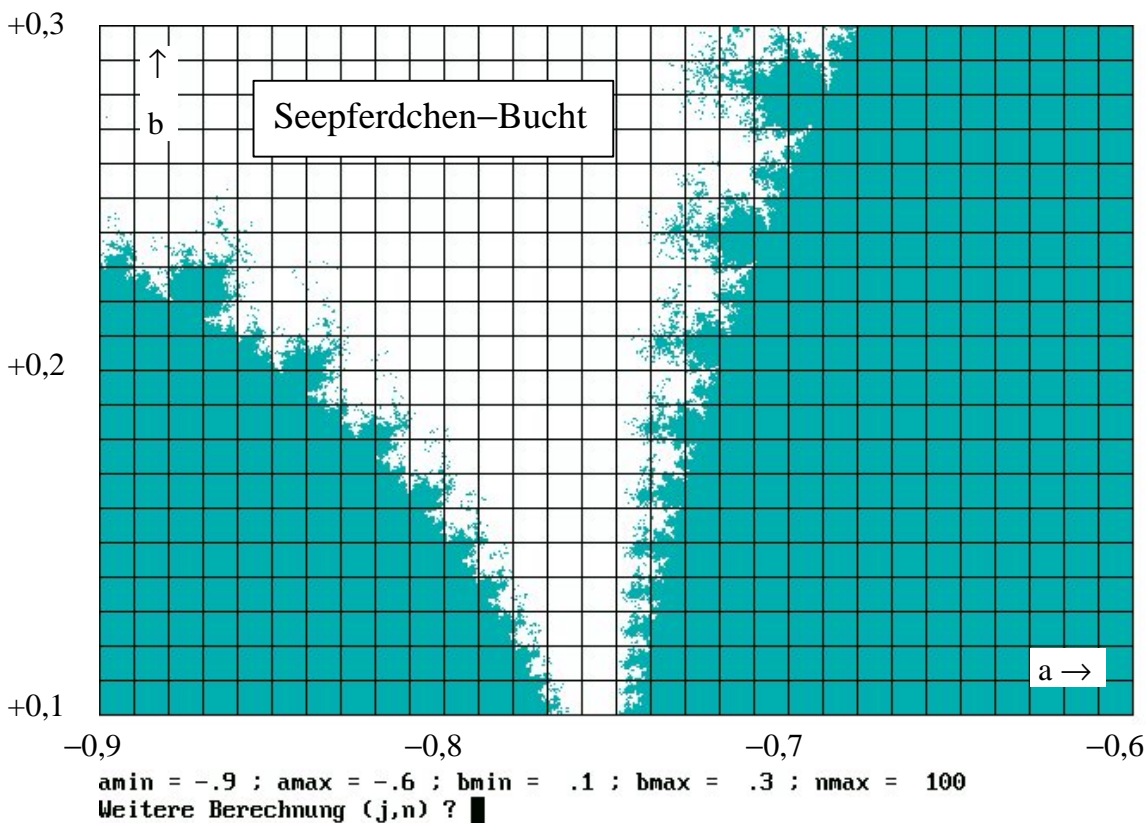
<b>c</b>	
<b>a</b>	<b>b</b>
−1,0	+0,1

Parameterwerte  $c$  aus dem Nebenkörper  $W_1$  ergeben Julia–Mengen mit einer Struktur, die 2 anziehende Fixpunkte enthält, deren Einzugsbereiche durch einen abstoßenden Fixpunkt getrennt werden.

c	
a	b
-0,15	+0,75
+0,27	+0,55
-0,5	+0,55
-0,62	+0,42
-1,15	+0,25

Wir wenden uns nun den Knospen der Mandelbrot-Menge zu. Für die verschiedenen, ausgewählten Parameterwerte  $c$  ergeben sich Strukturen, die an Windmühlen erinnern, in deren "Flügel" die anziehenden Fixpunkte der verschiedenen Zyklen sitzen. Wir beobachten 3er-, 4er-, 5er-, 6er- und 7er-Zyklen. Die "Nabe" wird immer durch einen abstoßenden Fixpunkt dargestellt. Interessant ist der letzte Fall  $c = (-1,15; +0,25)$ , bei dem ein 6er-Zyklus entsteht, der in 2 Gruppen mit je 3 Elementen um einen abstoßenden Fixpunkt angeordnet ist.

Bei der Darstellung weiterer Beispiele wollen wir uns einem Gebiet der Mandelbrot-Menge zuwenden, dessen Küstenregion besonders ansprechende Julia-Mengen liefert. Es ist der Einschnitt zwischen dem Hauptkörper  $W_0$  und dem Nebenkörper  $W_1$ , den wir die "Seepferdchen-Bucht" nennen wollen. Mit dem Programm MANDEL1 erhalten wir folgenden Ausschnitt aus der Mandelbrot-Menge:



Aus diesem Gebiet enthält die folgende Tabelle einige ausgewählte Beispiele.

<b>c</b>		<b>Beschreibung</b>
<b>a</b>	<b>b</b>	
−0,77	+0,11	verschränkte Spirale, 2–armig
−0,746	+0,13	Seepferdchen
−0,716	0,25	offene Spirale, 13–armig
−0,71	0,29	Stern, 11–strahlig

Die Beispiele stellen offenbar unzusammenhängende Julia–Mengen dar, deren  $c$ –Parameter nicht mehr in der Mandelbrot–Menge liegen. Ihre Strukturen haben sich aus den "Windmühlen" der Knospen mit ihren Zyklen entwickelt.

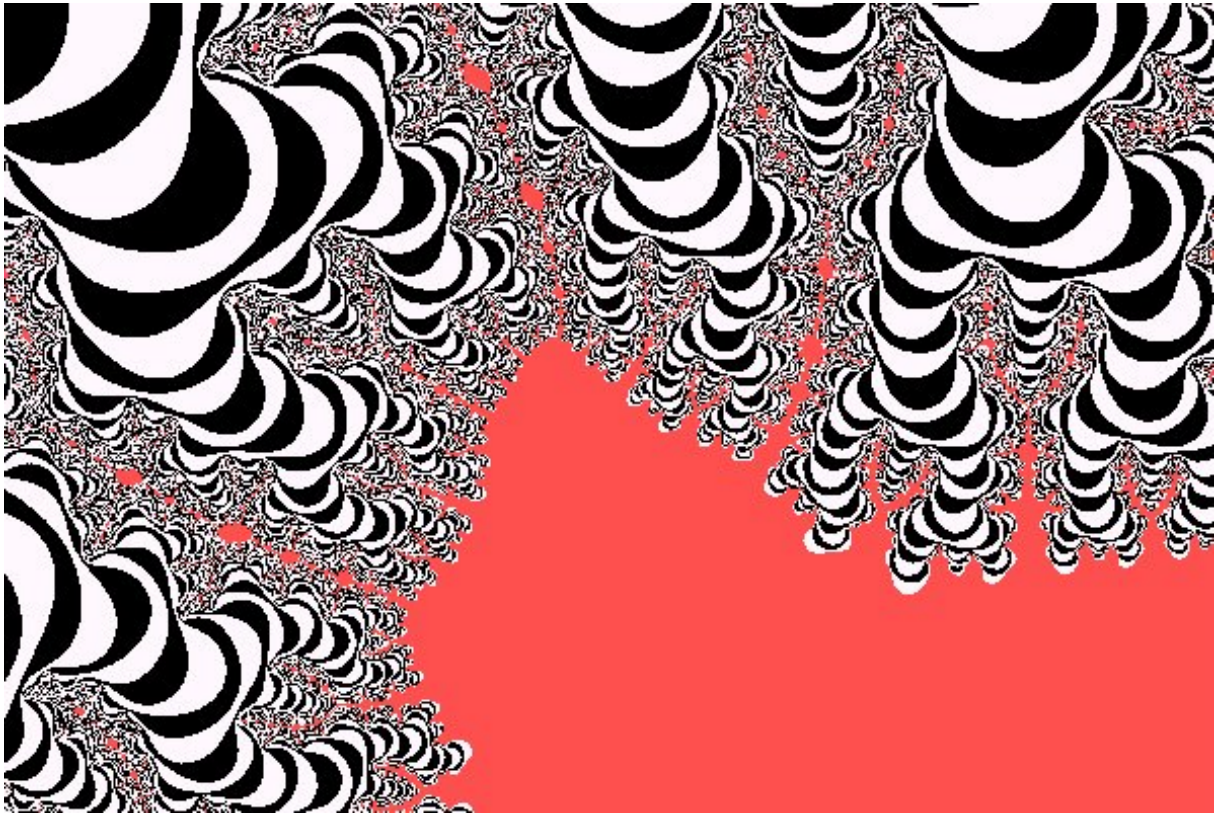
Mit dem Programm MANDEL1 können wir aber auch eine Reise in den "Mikrokosmos" unternehmen. Durch ständige Wahl einer Ausschnittsvergrößerung ergeben sich über das Bild einer sekundären Mandelbrotmenge immer neue Strukturen, die eine gewisse Selbstähnlichkeit zeigen, eine typische Eigenschaft von Fraktalen.

Folgende Bildersequenz kann man ausprobieren:

<b>a<sub>min</sub></b>	<b>a<sub>max</sub></b>	<b>b<sub>min</sub></b>	<b>Vergrößerungs- faktor</b>
−2.0	+1.0	−1.0	1
−0.3	0	0.8	10
−0.12	−0.09	0.92	100
−0.108	−0.105	0.925	1000
−0.1069	−0.1066	0.9256	10000

Eine Reise in die Unendlichkeit !





Und nun sind eigenen Exkursionen keine Grenzen mehr gesetzt. Die Mandelbrot-Menge stellt ein Buch mit unendlich vielen Seiten dar, die man nur aufschlagen muß. Man wird immer wieder neue Strukturen entdecken, die vorher noch kein Mensch gesehen hat.

## 5. Iterationen in den Quaternionen

### Einführung

Wir haben in Abschnitt 3 die Menge der reellen Zahlen  $\mathbf{R}$  durch die komplexen Zahlen zur Menge  $\mathbf{C}$  ergänzt. Als Erweiterung der komplexen Zahlen wurden höhere komplexe Zahlen, die Quaternionen, von W. R. Hamilton (1805 – 1865) eingeführt, die die Zahlenmenge  $\mathbf{H}$  ergeben. Die Quaternionen bilden einen vierdimensionalen Raum mit den Einheiten 1, i, j und k.

Damit läßt sich ein Quaternion q darstellen durch:

$$q = a_0 \cdot 1 + a_1 \cdot i + a_2 \cdot j + a_3 \cdot k; \quad q \in \mathbf{H} \text{ und } a_i \in \mathbf{R}$$

Für die Einheiten werden folgende Beziehungen festgesetzt:

$$\begin{aligned} 1^2 &= 1; \quad i^2 = j^2 = k^2 = -1 \\ i \cdot j &= -j \cdot i = k; \quad j \cdot k = -k \cdot j = i; \quad k \cdot i = -i \cdot k = j \end{aligned}$$

Für Quaternionen gelten ganz ähnliche Rechengesetze wie für komplexe Zahlen. Es gilt aber im allgemeinen nicht mehr das Kommutativgesetz der Multiplikation, d.h. bei der Multiplikation zweier Quaternionen ist die Reihenfolge der Faktoren zu beachten.

$$q_1 \cdot q_2 \neq q_2 \cdot q_1$$

### Iteration mit Quaternionen

Quaternionen können nun dazu benutzt werden, Julia-Mengen dreidimensional darzustellen. Dazu verallgemeinern wir unsere bisherige Iterationsgleichung zu:

$$q_{n+1} = q_n^2 + c$$

In dieser Gleichung setzen wir nun:

$$q_{n+1} = x_{n+1} + y_{n+1} \cdot i + z_{n+1} \cdot j$$

$$q_n = x_n + y_n \cdot i + z_n \cdot j$$

$$c = a + b \cdot i$$

Wir setzen also  $q$  dreidimensional und  $c$  zweidimensional (komplex) an. D.h. wir beschränken uns bei der Iteration auf 3 Dimensionen und setzen die  $k$ -Komponente Null, obwohl Quaternionen eigentlich im 4-dimensionalen Raum definiert sind.

In unsere Iterationsgleichung eingesetzt ergibt sich dann:

$$\begin{aligned} x_{n+1} + y_{n+1} \cdot i + z_{n+1} \cdot j &= \\ &= (x_n + y_n \cdot i + z_n \cdot j) \cdot (x_n + y_n \cdot i + z_n \cdot j) + a + b \cdot i \end{aligned}$$

Unter Beachtung der Beziehungen für die Einheiten, insbesondere deren Reihenfolge bei der Multiplikation, erhalten wir:

$$\begin{aligned} x_{n+1} + y_{n+1} \cdot i + z_{n+1} \cdot j &= \\ &= x_n^2 - y_n^2 - z_n^2 + a + (2x_n y_n + b) \cdot i + 2x_n z_n \cdot j \end{aligned}$$

Der Koeffizientenvergleich der Komponenten führt uns zu den folgenden Iterationsgleichungen:

$x_{n+1} = x_n^2 - y_n^2 - z_n^2 + a$	(x-Komponente)
$y_{n+1} = 2x_n y_n + b$	(y-Komponente)
$z_{n+1} = 2x_n z_n$	(z-Komponente)

Diese Gleichungen beschreiben nun die Bahn der Iteration in einem 3-dimensionalen Raum.

Es ist naheliegend den Betrag eines Quaternions im 3-dimensionalen Raum durch die Beziehung

$ p  = \sqrt{x^2 + y^2 + z^2}$
--------------------------------

festzulegen.

Die Konvergenzbetrachtungen der Iteration in den Quaternionen können wir aus dem Komplexen entsprechend übernehmen.

Als Definition der Julia-Menge wird man nun festsetzen:

$K = \{q_0 \mid (y^n(q_0)) \text{ beschränkt} \}$	(ausgefüllte Julia-Menge)
$J = \text{Oberfläche von } K$	(Julia-Menge)

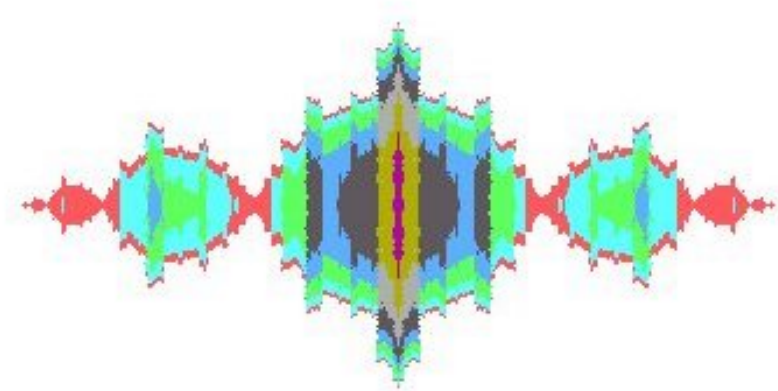
## Programm QUAT1

Bei diesem Programm bauen wir auf den Level-Set-Algorithmus für komplexe Zahlen auf, erweitern ihn nur um eine weitere Komponente  $z$ .

Problematisch ist dabei die Darstellung der dritten Dimension, da unser Bildschirm nur zweidimensional ist. Es gibt Algorithmen zur Darstellung von Computergrafiken, die es ermöglichen, 3-dimensionale Objekte perspektivisch und mit Schattenwirkungen auf dem Bildschirm darzustellen. Wir verzichten auf diese Möglichkeiten, da wir uns nur mit dem Prinzip befassen wollen. Wir stellen daher die dritte  $z$ -Komponente durch Höhenlinien dar. Damit wird zwar kein echtes 3-dimensionales Bild erzeugt, aber ein ungefährender Eindruck von der Kompliziertheit der Julia-Mengen in den Quaternionen vermittelt.

Mit den Eingabedaten  $a = -1$ ;  $b = 0$  und  $n_{\max} = 15$  erhalten wir eine Grafik, die die Rotationssymmetrie der Julia-Menge um die  $x$ -Achse zeigt.

**c-Parameter:**  
**(-1 , 0 )**



## 6. Anhang

### 6.1 QuickBasic – Programme

#### Übersicht über die QuickBasic - Programme

**Programm FOLGE1.BAS**

Einfaches Experimentierprogramm zur quadratischen Iteration  $x_{n+1} = x_n^2 + a$

**Programm ORBIT1.BAS**

Berechnung des Orbits (der Bahn) der Iteration  $x_{n+1} = x_n^2 + a$

**Programm GRAFIT1.BAS**

Grafische Iteration eines Punktes

**Programm GRAFIT2.BAS**

Grafische Iteration eines Intervalls

**Programm FEIGE1.BAS**

Abschätzung des Feigenbaum–Punktes

**Programm FEIGE2.BAS**

Berechnung des Feigenbaum–Diagramms für die Iteration  $x_{n+1} = x_n^2 + a$

**Programm FIXPKT1.BAS**

Numerische Berechnung von Fixpunkten der m–ten Iterierten von  $x_{n+1} = x_n^2 + a$  mittels Intervallschachtelung

**Programm KWURZEL1.BAS**

Berechnung der Wurzel aus einer komplexen Zahl

**Programm KFOLGE1.BAS**

Berechnung der Bahn der komplexen Iteration  $z_{n+1} = z_n^2$

**Programm KFOLGE2.BAS**

Berechnung der Bahn der komplexen Iteration  $z_{n+1} = z_n^2 + c$

**Programm KORBIT1.BAS**

Berechnung und grafische Darstellung des Orbits, der Fixpunkte der ersten Iterierten sowie des Divergenzkreises der komplexen Iteration  $z_{n+1} = z_n^2 + c$

**Programm JULIA1.BAS**

Berechnung der Julia–Menge mit Umkehriteration  $z_{n+1} = \pm \text{SQR}(z_n - c)$

**Programm JULIA2.BAS**

Berechnung der ausgefüllten Julia-Menge für die komplexe Iteration  $z_{n+1} = z_n^2 + c$  nach dem Level-Set-Algorithmus. Die Iterationstiefe wird schrittweise erhöht und damit die Julia-Menge mit Niveauflächen angenähert.

**Programm JULIA3.BAS**

Das Programm berechnet die Julia-Menge für die Iteration  $z_{n+1} = z_n^2 + c$  nach dem Level-Set-Algorithmus, außerdem alle Fixpunkte der 1. und 2. Iterierten mit den expliziten Formeln. Anziehende Fixpunkte mit höherer Periodenzahl werden numerisch angenähert.

**Programm JULIA4.BAS**

Das Programm berechnet die Julia-Menge für die Iteration  $z_{n+1} = z_n^2 + c$  nach dem Level-Set-Algorithmus. Mit Umkehriteration wird ein Punkt in der Julia-Menge angenähert, der anschließend iteriert wird.

**Programm MANDEL1.BAS**

Berechnung der Mandelbrot-Menge der komplexen Iteration  $z_{n+1} = z_n^2 + c$  nach dem Level-Set-Algorithmus. Wahlweise können Niveaulinien dargestellt werden.

**Programm JULIA5.BAS**

Berechnung der Julia-Menge der komplexen Iteration  $z_{n+1} = z_n^2 + c$  mit dem Level-Set-Algorithmus. Wahlweise können Niveaulinien dargestellt werden.

**Programm QUAT1.BAS**

Das Programm berechnet die Julia-Menge mit Quaternionen für die Iteration  $q_{n+1} = q_n^2 + c$  nach dem Level-Set-Algorithmus. Die z-Komponente wird durch Höhenlinien dargestellt.

**Programm FEIGE3.BAS**

Berechnung der Feigenbaum-Konstanten nach dem Newton-Verfahren

```

'-----
'Quellprogramm: FOLGE1.BAS
'-----
'Einfaches Experimentierprogramm zur
'quadratischen Iteration  $x(n+1) = x(n)^2 + a$ 
'-----
'Eingabewerte:
a = -.5                'Parameter a
x = 1.36               'Startwert x0
n = 20                'Anzahl der Iterationen nmax
'-----

CLS
PRINT
PRINT
PRINT 0, x
FOR i = 1 TO n          'Iteration
    x = x ^ 2 + a
    PRINT i, x
NEXT i

LOCATE 1, 1
PRINT "n", "x(n)"
PRINT "-----"

END

```

```

'-----
'Quellprogramm: ORBIT1.BAS
'-----
'Berechnung des Orbits (der Bahn) der
'Iteration  $x(n+1) = x(n)^2 + a$ 
'-----
'Eingabewerte:
a = -.5           'Parameter a
x = 1.36          'Startwert x0
n = 20            'Anzahl der Iterationen nmax
'-----

SCREEN 12          'VGA, 640 x 480 Pixel, 16 Farben
px = 620           'Festlegung der Bildschirmpunkte (Bildschirmbereich)
py = 420

xmin = -2: xmax = 2           'mathematischer Bereich
xschritt = (xmax - xmin) / py 'Schrittweite

zy = (xmax - x) / xschritt     'Transformationsgleichung
CIRCLE (30, zy), 2, 14

FOR sx = 30 + (px - 50) / n TO px - 20 STEP (px - 50) / n 'Iteration
  x = x ^ 2 + a
  IF x > xmax THEN
    PRINT "Iteration ausserhalb des Rechenbereichs"
    EXIT FOR
  END IF
  zy = (xmax - x) / xschritt
  'PSET (sx, zy), 12
  LINE -(sx, zy), 12
NEXT sx

LINE (0, 0)-(0, py)           'Skalierung
FOR j = 0 TO py STEP py / 20
  LINE (0, j)-(5, j)
NEXT j
LINE (px, 0)-(px, py)
FOR j = 0 TO py STEP py / 20
  LINE (px, j)-(px - 5, j)
NEXT j

END

```



```

'-----
'Quellprogramm: GRAFIT1.BAS
'-----
'Grafische Iteration eines Punktes
'-----
'Eingabewerte:
a = -.5           'Parameter a
x0 = -1.15        'Startwert x0
m = 1             'Iterierte m
n = 50            'Anzahl der Iterationen nmax
'-----

SCREEN 12
px = 420: py = 420           'Bildschirmbereich

xmin = -2: xmax = 2         'mathematischer Bereich
ymin = -2: ymax = 2

xschritt = (xmax - xmin) / px 'Schrittweite
yschritt = (ymax - ymin) / py

FOR sx = 0 TO px             'Darstellung der Iterationsfunktion
    x = xmin + sx * xschritt 'y1 = y(x)
    GOSUB funktion
    IF y > ymax THEN GOTO 100
    zy = (ymax - y) / yschritt
    PSET (sx, zy), 12
100 : NEXT sx

FOR sx = 0 TO px             'Darstellung der Winkelhalbierenden
    x = xmin + sx * xschritt 'y2 = x
    y = x
    IF y > ymax THEN EXIT FOR
    zy = (ymax - y) / yschritt
    PSET (sx, zy), 12
NEXT sx

'Achsenkreuz
LINE (0, ymax / yschritt)-((xmax - xmin) / xschritt, ymax / yschritt)
LINE (-xmin / xschritt, 0)-(-xmin / xschritt, (ymax - ymin) / yschritt)

FOR i = 0 TO px STEP px / 20 'Gitternetz
    'LINE (i, py)-(i, 0)
NEXT i
FOR j = 0 TO py STEP py / 20
    'LINE (0, j)-(px, j)
NEXT j

x = x0           'Startpunkt
sx = (x - xmin) / xschritt
zy = ymax / yschritt
PSET (sx, zy)
CIRCLE (sx, zy), 3, 10

FOR i = 1 TO n             'Linienzug
    GOSUB funktion
    IF y > ymax THEN EXIT FOR
    zy = (ymax - y) / yschritt
    LINE -(sx, zy), 10
    x = y
    sx = (x - xmin) / xschritt
    LINE -(sx, zy), 10

```

```
NEXT i

PRINT "Grenzpunkt:"
PRINT "x = "; x

END

funktion:          'Festlegung der Iterationsfunktion
  FOR j = 1 TO m
    x = x ^ 2 + a
  NEXT j
  y = x
RETURN
```

```

'-----
'Quellprogramm: GRAFIT2.BAS
'-----
'Grafische Iteration eines Intervalls
'-----
'Eingabewerte:
a = -2          'Parameter a
x0 = .8         'Startwert x0
delx = .05      'Intervallbreite delta x
n = 5           'Anzahl der Iterationen nmax
'-----

SCREEN 12
px = 620: py = 420

xmin = -2: xmax = 2
ymin = -2: ymax = 2

xschritt = (xmax - xmin) / px
yschritt = (ymax - ymin) / py

FOR sx = 0 TO px                                'Iterationsfunktion zeichnen
    x = xmin + sx * xschritt
    GOSUB funktion
    IF y > ymax THEN GOTO 100
    zy = (ymax - y) / yschritt
    PSET (sx, zy), 12
100 : NEXT sx

FOR sx = 0 TO px                                'Winkelhalbierende zeichnen
    x = xmin + sx * xschritt
    y = x
    IF y > ymax THEN EXIT FOR
    zy = (ymax - y) / yschritt
    PSET (sx, zy), 12
NEXT sx

FOR x1 = x0 TO x0 + delx STEP delx / 15          'Iteration
    x = x1
    sx = (x - xmin) / xschritt
    zy = ymax / yschritt
    PSET (sx, zy)
    FOR i = 1 TO n
        GOSUB funktion
        IF y > ymax THEN EXIT FOR
        zy = (ymax - y) / yschritt
        LINE -(sx, zy), 10
        x = y
        sx = (x - xmin) / xschritt
        LINE -(sx, zy), 10
    NEXT i
NEXT x1

END

funktion:
    y = x ^ 2 + a
RETURN

```

```

'-----
'Quellprogramm FEIGE1.BAS
'-----
'Abschaetzung des Feigenbaum-Punktes
'-----
'Eingabewerte:
a1 = -1.3941      'Parameterwerte
a2 = -1.3997
'-----

CLS
del = 4.6692      'Feigenbaum-Konstante

FOR i = 1 TO 8
    a3 = (del + 1) / del * a2 - 1 / del * a1
    PRINT i, a3
    a1 = a2
    a2 = a3
NEXT i

END

```

```

'-----
'Quellprogramm: FEIGE2.BAS
'-----
'Berechnung des Feigenbaum-Diagramms fuer die
'Iteration  $x(n+1) = x(n)^2 + a$ 
'-----

SCREEN 12                                'VGA, 640 x 480 Pixel, 16 Farben
px = 600: py = 400                       'Bildschirmbereich

amin = -2: amax = 1                      'mathematischer Bereich
xmin = -2: xmax = 2

aschritt = (amax - amin) / px             'Schrittweite
xschritt = (xmax - xmin) / py

FOR sx = 0 TO px
a = amin + sx * aschritt                 'Parameter
  x = 0                                  'Startwert
  FOR i = 1 TO 800                        'Iteration
    x = x ^ 2 + a
    IF x > xmax THEN EXIT FOR
    IF i > 400 THEN
      zy = (xmax - x) / xschritt
      PSET (sx, zy), 12
    END IF
  NEXT i
NEXT sx

FOR i = 0 TO px STEP px / 30             'Gitternetz
  LINE (i, py)-(i, 0)
NEXT i
FOR j = 0 TO py STEP py / 20
  LINE (0, j)-(px, j)
NEXT j

END

```

```

DECLARE SUB intervall (xa0!, xe0!, xfix!)
DIM SHARED a, m
'-----
'Programm FIXPKT1.BAS
'-----
'Numerische Berechnung von Fixpunkten der
'm-ten Iterierten von  $x(n+1) = x(n)^2 + a$ 
'mittels Intervallschachtelung
'-----
'Eingabewerte:
a = -1.76                'Parameter a
m = 3                    'm-te Iterierte
xmin = -2                'Gesamtintervall [xmin, xmax]
xmax = 2
k = 20                    'Anzahl der Teilintervalle k
'-----

CLS

PRINT "Fixpunktsuche "; m; "-te Iteration, a = "; a
PRINT "Intervall          Fixpunkt"
PRINT "-----"

xa0 = xmin
xe0 = xmin + (xmax - xmin) / k
CALL intervall(xa0, xe0, xfix)
PRINT USING "[##.### , ##.###]      ###.#####"; xa0; xe0; xfix

FOR i = 1 TO k - 1
    xa0 = xe0
    xe0 = xa0 + (xmax - xmin) / k
    CALL intervall(xa0, xe0, xfix)
    PRINT USING "[##.### , ##.###]      ###.#####"; xa0; xe0; xfix
NEXT i

END

SUB intervall (xa0, xe0, xfix)

xa = xa0                'Intervall
xe = xe0
schritt = (xe - xa) / 100 'Schrittweite

DO                        'Intervallschleife
    y1 = xa              'Anfangswerte
    FOR i = 1 TO m
        y1 = y1 ^ 2 + a
    NEXT i
    y2 = xa
    s2 = SGN(y1 - y2)
    j = 0

    DO                    'Schnittpunktbestimmung
        s1 = s2
        y1 = xa
        FOR i = 1 TO m    'm-te Iterierte
            y1 = y1 ^ 2 + a
        NEXT i
        y2 = xa
        xa = xa + schritt
        s2 = SGN(y1 - y2)
        j = j + 1
    DO WHILE s1 <> s2
        xa = (j * xa + (j + 1) * y1) / (j + 1)
        y1 = xa
    LOOP
    xfix = xa
END SUB

```

IF j > 100000 THEN	'Abbruchbedingung der
xfix = 0	'inneren Schleife
EXIT SUB	
END IF	
LOOP UNTIL s1 <> s2	'Schnittpunktbedingung
xa = xa - 2 * schritt	
xe = xa + schritt	
schritt = (xe - xa) / 100	
LOOP UNTIL schritt < .0000001	'Rechengenauigkeit
IF xa < xa0 OR xe > xe0 THEN	'Prozedurergebnis
xfix = 0	
ELSE	
xfix = (xa + xe) / 2	
END IF	
END SUB	

```

'-----
'Quellprogramm: KWURZEL1.BAS
'-----
'Berechnung der Wurzel aus einer komplexen
'Zahl
'-----
'Eingabewerte:
p = 1                      'Wurzel (p + iq)
q = 1
'-----

CLS

IF q = 0 THEN
    IF p <= 0 THEN
        u1 = 0: u2 = 0
        v1 = SQR(-p): v2 = -v1
    ELSE
        u1 = SQR(p): u2 = -u1
        v1 = 0: v2 = 0
    END IF
ELSE
    u1 = SQR((p + SQR(p ^ 2 + q ^ 2)) / 2): u2 = -u1
    IF u1 = 0 THEN v1 = SQR(-p) ELSE v1 = q / 2 / u1
    v2 = -v1
END IF

PRINT "WURZEL ("; p; " + "; q; "*i)"
PRINT "Z1 = ("; u1; "; "; v1; ")"
PRINT "Z2 = ("; u2; "; "; v2; ")"

END

```



```

'-----
'Quellprogramm: KFOLGE1.BAS
'-----
'Berechnung der Bahn der komplexen Iteration
'z(n+1) = z(n)^2
'-----
'Eingabewerte:
x = .2                      'komplexer Startwert z0 = (x0, y0)
y = .97
'y = SQR(1 - x ^ 2)         'Einheitskreis
nmax = 15                   'Anzahl der Iterationen nmax
'-----

CLS
PRINT
PRINT
zbetrag = SQR(x ^ 2 + y ^ 2)
PRINT USING "###  ##.#####  ##.#####  ##.#####"; 0; x; y; zbetrag

FOR i = 1 TO nmax           'Iterationsschleife
    xx = x ^ 2 - y ^ 2      'Hilfsgroesse
    y = 2 * x * y          'Imaginaerteil
    x = xx                 'Realteil
    zbetrag = SQR(x ^ 2 + y ^ 2)
    PRINT USING "###  ##.#####  ##.#####  ##.#####"; i; x; y; zbetrag
NEXT i

LOCATE 1, 1
PRINT " n    x(n)          y(n)          Betrag von z(n) "
PRINT "-----"
END

```

```

'-----
'Quellprogramm: KFOLGE2.BAS
'-----
'Berechnung der Bahn der komplexen Iteration
'z(n+1) = z(n)^2 + c
'-----
'Eingabewerte:
x = 0                'komplexer Startwert z0 = (x0, y0)
y = 0
a = -1.15            'Parameter c = (a, b)
b = .25
nmax = 500           'Anzahl der Iterationen nmax
'-----

CLS
PRINT
PRINT
zbetrag = SQR(x ^ 2 + y ^ 2)
PRINT USING "###  ##.#####  ##.#####  ##.#####"; 0; x; y; zbetrag

FOR i = 1 TO nmax                'Iterationsschleife
    xx = x ^ 2 - y ^ 2 + a        'Hilfsgroesse
    y = 2 * x * y + b            'Imaginaerteil
    x = xx                       'Realteil
    zbetrag = SQR(x ^ 2 + y ^ 2)
    PRINT USING "###  ##.#####  ##.#####  ##.#####"; i; x; y; zbetrag
NEXT i

LOCATE 1, 1
PRINT " n      x(n)          y(n)          Betrag von z(n) "
PRINT "-----"
END

```

```

'-----
'Quellprogramm: KORBIT1.BAS
'-----
'Berechnung und grafische Darstellung des Orbits,
'der Fixpunkte der ersten Iterierten sowie
'des Divergenzkreises der komplexen Iteration
'z(n+1) = z(n)^2 + c
'-----
'Eingabewerte:
a = -.5: b = .5          'Parameter c = (a, b)
x = .5: y = -.5          'Startwert z0 = (x0, y0)
nmax = 30                'Anzahl der Iterationen nmax
'-----

SCREEN 12
px = 420                  'Bildschirmbereich
py = 420

xmin = -2: xmax = 2      'mathematischer Bereich
ymin = -2: ymax = 2
xschritt = (xmax - xmin) / px  'Schrittweite
yschritt = (ymax - ymin) / py

'Achsenkreuz
LINE (0, ymax / yschritt)-(px, ymax / yschritt)
LINE (-xmin / xschritt, 0)-(-xmin / xschritt, py)

'Fluchtschranke
r = (1 + SQR(1 + 4 * SQR(a ^ 2 + b ^ 2))) / 2
r2 = r ^ 2

'Divergenzskreis
CIRCLE (-xmin / xschritt, ymax / yschritt), r / xschritt, 12

'Einheitskreis
CIRCLE (-xmin / xschritt, ymax / yschritt), 1 / xschritt

'Iteration
sx = (x - xmin) / xschritt  'Startpunkt
zy = (ymax - y) / yschritt
PSET (sx, zy), 12
CIRCLE (sx, zy), 3, 12

FOR i = 1 TO nmax          'Iterationsschleife
    xx = x ^ 2 - y ^ 2 + a
    y = 2 * x * y + b
    x = xx
    IF x ^ 2 + y ^ 2 > r2 THEN EXIT FOR
    sx = (x - xmin) / xschritt
    zy = (ymax - y) / yschritt
    LINE -(sx, zy), 12      'Punktausgabe
    'PSET (sx, zy), 12
NEXT i

'Fixpunkte der 1. Iterierten
p = 1 - 4 * a: q = -4 * b
IF q = 0 THEN
    IF p <= 0 THEN
        u = 0: v = SQR(-p)
    ELSE
        u = SQR(p): v = 0
    END IF

```

```

ELSE
  u = SQR((p + SQR(p ^ 2 + q ^ 2)) / 2)
  IF u = 0 THEN v = SQR(-p) ELSE v = q / 2 / u
END IF
x1 = (1 + u) / 2: y1 = v / 2
x2 = (1 - u) / 2: y2 = -v / 2
sx1 = (x1 - xmin) / xschritt: zy1 = (ymax - y1) / yschritt
sx2 = (x2 - xmin) / xschritt: zy2 = (ymax - y2) / yschritt
CIRCLE (sx1, zy1), 3: CIRCLE (sx2, zy2), 3

PRINT "c-Parameter:"
PRINT "("; a; ";"; b; ")"

END

```

```

'-----
'Quellprogramm: JULIA1.BAS
'-----
'Berechnung der Julia-Menge mit
'Umkehriteration  $z(n+1) = +-SQR(z(n) - c)$ 
'-----
'Eingabewerte:
a = -1: b = 0           'Parameter c = (a,b)
x = -.1: y = 0          'Startwert z0 = (x0, y0)
nmax = 100000           'Anzahl der Iterationen nmax
'-----

SCREEN 12                'VGA, 640 x 480 Pixel, 16 Farben
px = 420: py = 420       'Bildschirmbereich

'z -Ebene                'mathematischer Bereich
xmin = -2: xmax = 2      'Realteil
ymin = -2: ymax = 2      'Imaginaerteil

xschritt = (xmax - xmin) / px  'Schrittweite
yschritt = (ymax - ymin) / py

'Achsenkreuz
LINE (0, ymax / yschritt)-(px, ymax / yschritt)
LINE (-xmin / xschritt, 0)-(-xmin / xschritt, py)

'Einheitskreis
CIRCLE (-xmin / xschritt, ymax / yschritt), 1 / xschritt

'Iterationsschleife
FOR i = 1 TO nmax
  p = x - a: q = y - b
  IF q = 0 THEN
    IF p <= 0 THEN
      u = 0: v = SQR(-p)
    ELSE
      u = SQR(p): v = 0
    END IF
  ELSE
    u = SQR((p + SQR(p ^ 2 + q ^ 2)) / 2)
    IF u = 0 THEN v = SQR(-p) ELSE v = q / 2 / u
  END IF
  x = u: y = v
  'zufaellige Auswahl der beiden Wurzeln
  IF RND > .5 THEN x = -x: y = -y
  'Punktausgabe
  IF i > 50 THEN
    sx = (x - xmin) / xschritt
    zy = (ymax - y) / yschritt
    PSET (sx, zy), 12
  END IF
NEXT i

PRINT "c-Parameter:"
PRINT "("; a; ", "; b; ")"

END

```

```

'-----
'Quellprogramm: JULIA2.BAS
'-----
'Berechnung der ausgefuellten Julia-Menge
'f•r die komplexe Iteration  $z(n+1) = z(n)^2 + c$ 
'nach dem Level-Set-Algorithmus.
'Die Iterationstiefe wird schrittweise erhoeht und
'damit die Julia-Menge mit Niveauflaechen angenaehert.
'-----

'Eingabewerte:
a = -1                      'Parameter c = (a,b)
b = 0
'-----

SCREEN 12                    'VGA, 640 x 480 Pixel, 16 Farben
px = 400: py = 400          'Bildschirmbereich

'z -Ebene                    'mathematischer Bereich
xmin = -4: xmax = 4         'Realteil
ymin = -4: ymax = 4         'Imaginaerteil

xschritt = (xmax - xmin) / px      'Schrittweite
yschritt = (ymax - ymin) / py

r = 2 + SQR(a ^ 2 + b ^ 2)         'Fluchtschranke
r2 = r ^ 2

CLS
FOR nmax = 0 TO 6
  'Berechnung der angenaeherten Julia-Menge durch
  'Abtasten der Bildschirmpunkte mit Test auf Mengenzugehoerigkeit
  FOR sx = 0 TO px              'BildschirmSpalten
    x = xmin + sx * xschritt
    FOR zy = 0 TO py            'BildschirmZeilen
      y = ymax - zy * yschritt
      x1 = x: y1 = y
      FOR n = 1 TO nmax         'Iterationsschleife
        xx = x1 * x1 - y1 * y1 + a      'Hilfsgroesse
        y1 = 2 * x1 * y1 + b           'Imaginaerteil
        x1 = xx                    'Realteil
        IF x1 * x1 + y1 * y1 > r2 THEN EXIT FOR
      NEXT n
      'Punktausgabe
      IF x1 * x1 + y1 * y1 <= r2 THEN PSET (sx, zy), 5 + nmax
    NEXT zy
  NEXT sx
NEXT nmax

nmax = 100
'Berechnung der Julia-Menge
FOR sx = 0 TO px              'BildschirmSpalten
  x = xmin + sx * xschritt
  FOR zy = 0 TO py            'BildschirmZeilen
    y = ymax - zy * yschritt
    x1 = x: y1 = y
    FOR n = 1 TO nmax         'Iterationsschleife
      xx = x1 * x1 - y1 * y1 + a      'Hilfsgroesse
      y1 = 2 * x1 * y1 + b           'Imaginaerteil
      x1 = xx                    'Realteil
      IF x1 * x1 + y1 * y1 > r2 THEN EXIT FOR
    NEXT n
    IF x1 * x1 + y1 * y1 <= r2 THEN PSET (sx, zy), 0
  NEXT zy
NEXT zy

```

```
NEXT sx
```

```
PRINT "c-Parameter:"
```

```
PRINT "("; a; ", "; b; ")"
```

```
END
```

```

'-----
'Quellprogramm: JULIA3.BAS
'-----
'Das Programm berechnet die Julia-Menge fuer die
'Iteration  $z(n+1) = z(n)^2 + c$  nach dem
'Level-Set-Algorithmus, ausserdem alle Fixpunkte
'der 1. und 2. Iterierten mit expliziten Formeln.
'Anziehende Fixpunkte mit hoeherer Periodenzahl
'werden numerisch angenaehert.
'-----
'Eingabewerte:
a = -.1                'Parameter c = (a, b)
b = .75
nmax = 100             'Anzahl der Iterationen nmax
'-----

SCREEN 12              'VGA, 640 x 480 Pixel, 16 Farben
px = 420: py = 420    'Bildschirmbereich

'z-Ebene              'mathematischer Bereich
xmin = -2: xmax = 2   'Realteil
ymin = -2: ymax = 2   'Imaginaerteil

xschritt = (xmax - xmin) / px    'Schrittweite
yschritt = (ymax - ymin) / py

'Fluchtschranke
r = 2 + SQR(a ^ 2 + b ^ 2)
r2 = r ^ 2

CLS

'Berechnung der ausgefuellten Julia-Menge durch
'Test auf Mengenzugehoerigkeit:
x = 0: y = 0
FOR sx = 0 TO px                'Bildschirmspalten
  x = xmin + sx * xschritt
  FOR zy = 0 TO py              'Bildschirmzeilen
    y = ymax - zy * yschritt
    x1 = x: y1 = y
    FOR n = 1 TO nmax            'Iteration
      xx = x1 * x1 - y1 * y1 + a  'Hilfsgroesse
      y1 = 2 * x1 * y1 + b        'Imaginaerteil
      x1 = xx                    'Realteil
      IF x1 * x1 + y1 * y1 > r2 THEN EXIT FOR
    NEXT n
    'Punktausgabe
    IF x1 * x1 + y1 * y1 <= r2 THEN PSET (sx, zy), 9
  NEXT zy
NEXT sx

'Fixpunkte der 1. Iterierten
p = 1 - 4 * a: q = -4 * b
u = SQR((p + SQR(p ^ 2 + q ^ 2)) / 2)
IF u = 0 THEN v = SQR(-p) ELSE v = q / 2 / u
x1 = (1 + u) / 2: y1 = v / 2
x2 = (1 - u) / 2: y2 = -v / 2
sx1 = (x1 - xmin) / xschritt: zy1 = (ymax - y1) / yschritt
sx2 = (x2 - xmin) / xschritt: zy2 = (ymax - y2) / yschritt
CIRCLE (sx1, zy1), 3, 10: CIRCLE (sx2, zy2), 3, 10

'Fixpunkte der 2. Iterierten
p = -3 - 4 * a: q = -4 * b

```



```

u = SQR((p + SQR(p ^ 2 + q ^ 2)) / 2)
IF u = 0 THEN v = SQR(-p) ELSE v = q / 2 / u
x1 = -(1 + u) / 2: y1 = -v / 2
x2 = -(1 - u) / 2: y2 = v / 2
sx1 = (x1 - xmin) / xschritt: zy1 = (ymax - y1) / yschritt
sx2 = (x2 - xmin) / xschritt: zy2 = (ymax - y2) / yschritt
CIRCLE (sx1, zy1), 3, 12: CIRCLE (sx2, zy2), 3, 12

'Iteration zur naeherungsweise Fixpunktberechnung
x = 0: y = 0
FOR i = 1 TO 1000
    xx = x * x - y * y + a
    y = 2 * x * y + b
    x = xx
    IF x * x + y * y > r2 THEN EXIT FOR
    sx = (x - xmin) / xschritt
    zy = (ymax - y) / yschritt
    IF i > 900 THEN CIRCLE (sx, zy), 3
NEXT i

LOCATE 28
PRINT "Berechnungsparameter: a = "; a; "; b = "; b; "; n max = "; nmax

END

```

```

'-----
'Quellprogramm: JULIA4.BAS
'-----
'Das Programm berechnet die Julia-Menge fuer die
'Iteration  $z(n+1) = z(n)^2 + c$  nach dem
'Level-Set-Algorithmus. Mit Umkehriteration wird ein
'Punkt der Julia-Menge angenaehert, der anschliessend
'iteriert wird.
'-----
'Eingabewerte:
a = 0                      'Parameter c = (a, b)
b = .5
n0 = 65                    'Julia-Mengen-Punkt (n0 > 50)
'-----

SCREEN 12                  'VGA, 640 x 480 Pixel, 16 Farben
px = 420: py = 420        'Bildschirmbereich

'z-Ebene                  'mathematischer Bereich
xmin = -2: xmax = 2       'Realteil
ymin = -2: ymax = 2       'Imaginaerteil

xschritt = (xmax - xmin) / px    'Schrittweite
yschritt = (ymax - ymin) / py

'Fluchtschranke
r = 2 + SQR(a ^ 2 + b ^ 2)
r2 = r ^ 2

CLS

'Berechnung der ausgefuellten Julia-Menge nach dem
'Level-Set-Algorithmus:
x = 0: y = 0
FOR sx = 0 TO px          'Bildschirmspalten
  x = xmin + sx * xschritt
  FOR zy = 0 TO py        'Bildschirmzeilen
    y = ymax - zy * yschritt
    x1 = x: y1 = y
    FOR n = 1 TO 100       'Iteration
      xx = x1 * x1 - y1 * y1 + a    'Hilfsgroesse
      y1 = 2 * x1 * y1 + b          'Imaginaerteil
      x1 = xx                      'Realteil
      IF x1 * x1 + y1 * y1 > r2 THEN EXIT FOR
    NEXT n
    'Punktausgabe
    IF x1 * x1 + y1 * y1 <= r2 THEN PSET (sx, zy), 9
  NEXT zy
NEXT sx

'Berechnung eines Punktes in der Julia-Menge
'durch Umkehriteration
x = .01: y = 0
FOR i = 1 TO n0
  p = x - a: q = y - b
  IF q = 0 THEN
    IF p <= 0 THEN
      u = 0: v = SQR(-p)
    ELSE
      u = SQR(p): v = 0
    END IF
  ELSE
    u = SQR((p + SQR(p ^ 2 + q ^ 2)) / 2)

```

```

        IF u = 0 THEN v = SQR(-p) ELSE v = q / 2 / u
    END IF
    x = u: y = v
    'zufaellige Auswahl der beiden Wurzeln
    IF RND > .5 THEN x = -x: y = -y
NEXT i

sx = (x - xmin) / xschritt
zy = (ymax - y) / yschritt
'Punktausgabe
CIRCLE (sx, zy), 3, 12

'Iteration des Julia-Mengen-Punktes
x1 = x: y1 = y
FOR n = 1 TO 20
    xx = x1 * x1 - y1 * y1 + a
    y1 = 2 * x1 * y1 + b
    x1 = xx
    sx = (x1 - xmin) / xschritt
    zy = (ymax - y1) / yschritt
    'Linieausgabe
    LINE -(sx, zy)
NEXT n

LOCATE 28
PRINT "Berechnungsparameter: a = "; a; "  b = "; b; "

END

```

```

'-----
'Quellprogramm : MANDEL1.BAS
'-----
'Berechnung der Mandelbrot-Menge der komplexen Iteration
'z(n+1) = z(n)^2 + c mit dem Level-Set-Algorithmus.
'Wahlweise koennen Niveaulinien dargestellt werden.
'-----

SCREEN 12                                'VGA, 640 x 480 Pixel, 16 Farben
px = 600: py = 400                      'Bildschirmbereich

CLS
PRINT "Berechnung der Mandelbrot-Menge"
PRINT "===== "
PRINT
PRINT "Das Programm stellt die Mandelbrot-Menge grafisch dar."
PRINT "Eingabeparameter sind die Komponenten a und b"
PRINT "fuer den komplexen Bereich der Konstanten c = (a,b)"

marke:
PRINT
'c-Ebene, c = (a,b)                      'mathematischer Bereich
INPUT "amin = ", amin
INPUT "amax = ", amax
INPUT "bmin = ", bmin
bmax = 2 / 3 * (amax - amin) + bmin
PRINT
PRINT "Maximale Anzahl der Iterationen:"
INPUT "nmax = ", nmax
PRINT
INPUT "Darstellung von Niveaulinien (j/n) ? ", nl$
PRINT
INPUT "Gitternetz (j/n) ? ", gi$

aschritt = (amax - amin) / px            'Schrittweite
bschritt = (bmax - bmin) / py

CLS
'Berechnung der Mandelbrot-Menge durch Abtasten der
'Bildschirmpunkte
FOR sx = 0 TO px                        'BildschirmSpalten
  a = amin + sx * aschritt
  FOR zy = 0 TO py                      'BildschirmZeilen
    b = bmax - zy * bschritt
    x = 0: y = 0                        'Startwert z0=0
    FOR n = 1 TO nmax                  'Iteration
      xx = x * x - y * y + a           'Hilfsgroesse
      y = 2 * x * y + b                'Imaginaerteil
      x = xx                           'Realteil
      IF x * x + y * y > 4 THEN n0 = n: EXIT FOR
      'Der Punkt c = (a, b) gehoert nicht zur Mandelbrot-Menge
    NEXT n
    'Punktausgabe
    IF nl$ = "j" THEN
      IF INT(n0 / 2) = n0 / 2 THEN PSET (sx, zy)
      'Darstellung der Niveaulinien
      IF x * x + y * y <= 4 THEN PSET (sx, zy), 12
      'Der Punkt c = (a, b) gehoert zur Mandelbrot-Menge
    ELSE
      IF x * x + y * y <= 4 THEN PSET (sx, zy), 12
      'Der Punkt c = (a, b) gehoert zur Mandelbrot-Menge
    END IF
  NEXT zy
NEXT sx

```

```

NEXT sx

'Gitternetz
IF gi$ = "j" THEN
  FOR i = 0 TO px STEP px / 30
    LINE (i, py)-(i, 0)
  NEXT i
  FOR j = 0 TO py STEP py / 20
    LINE (0, j)-(px, j)
  NEXT j
END IF

LOCATE 28
PRINT "amin = "; amin; ";"; " amax = "; amax; ";";
PRINT " bmin = "; bmin; ";"; " bmax = "; bmax; ";";
PRINT " nmax = "; nmax
INPUT "Weitere Berechnung (j,n) "; antwort$
IF antwort$ = "j" THEN GOTO marke

END

```

```

'-----
'Quellprogramm : JULIA5.BAS
'-----
'Berechnung der Julia-Menge der komplexen Iteration
'z(n+1) = z(n)^2 + c mit dem Level-Set-Algorithmus.
'Wahlweise koennen Niveaulinien dargestellt werden.
'-----

SCREEN 12                                'VGA, 640 x 480 Pixel, 16 Farben
px = 600: py = 400                       'Bildschirmbereich

CLS
PRINT "Berechnung der Julia-Menge"
PRINT "===== "
PRINT
PRINT "Das Programm stellt die Julia-Menge grafisch dar."
PRINT "Eingabewerte sind die Komponenten a und b"
PRINT "des Parameters c = (a,b) sowie die x- und y-"
PRINT "Koordinaten fuer den dynamischen Bereich z = (x,y). "

marke:
PRINT
PRINT "Parameter c = (a,b)"
INPUT "a = ", a
INPUT "b = ", b
PRINT
'z-Ebene, z = (x,y)                        'mathematischer Bereich
PRINT "Dynamischer Bereich"
INPUT "xmin = ", xmin
INPUT "xmax = ", xmax
INPUT "ymin = ", ymin
ymax = 2 / 3 * (xmax - xmin) + ymin
PRINT
PRINT "Maximale Anzahl der Iterationen:"
INPUT "nmax = ", nmax
PRINT
INPUT "Darstellung von Niveaulinien (j/n) ? ", nl$
PRINT
INPUT "Gitternetz (j/n) ? ", gi$

xschritt = (xmax - xmin) / px             'Schrittweite
yschritt = (ymax - ymin) / py

r = 2 + SQR(a ^ 2 + b ^ 2)                'Fluchtschranke
r2 = r ^ 2

CLS
'Berechnung der Julia-Menge durch Abtasten der
'Bildschirmpunkte
FOR sx = 0 TO px                          'BildschirmSpalten
  x = xmin + sx * xschritt
  FOR zy = 0 TO py                        'BildschirmZeilen
    y = ymax - zy * yschritt
    x1 = x: y1 = y                       'Startwert z0
    FOR n = 1 TO nmax                     'Iteration
      xx = x1 * x1 - y1 * y1 + a          'Hilfsgroesse
      y1 = 2 * x1 * y1 + b                'Imaginaerteil
      x1 = xx                             'Realteil
      IF x1 * x1 + y1 * y1 > r2 THEN n0 = n: EXIT FOR
    NEXT n
    'Punktausgabe
    IF nl$ = "j" THEN
      IF INT(n0 / 2) = n0 / 2 THEN PSET (sx, zy)

```

```

'Darstellung der Niveaulinien
  IF x1 * x1 + y1 * y1 <= r2 THEN PSET (sx, zy), 12
ELSE
  IF x1 * x1 + y1 * y1 <= r2 THEN PSET (sx, zy), 12
END IF
NEXT zy
NEXT sx

'Gitternetz
IF gi$ = "j" THEN
  FOR i = 0 TO px STEP px / 30
    LINE (i, py)-(i, 0)
  NEXT i
  FOR j = 0 TO py STEP py / 20
    LINE (0, j)-(px, j)
  NEXT j
END IF

LOCATE 27
PRINT "a = "; a; ";"; " b = "; b
PRINT "xmin = "; xmin; ";"; " xmax = "; xmax; ";";
PRINT " ymin = "; ymin; ";"; " ymax = "; ymax; ";";
PRINT " nmax = "; nmax
INPUT "Weitere Berechnung (j,n) "; antwort$
IF antwort$ = "j" THEN GOTO marke

END

```

```

'-----
'Quellprogramm: QUAT1.BAS
'-----
'Das Programm berechnet die Julia-Menge mit Quaternionen
'fuer die Iteration  $q(n+1) = q(n)^2 + c$ 
'nach dem Level-Set-Algorithmus.
'Die z-Komponente wird mit Hoehenlinien dargestellt.
'-----
'Eingabewerte:
a = -1: b = 0                'Parameter c = (a,b)
nmax = 15                    'Anzahl der Iterationen nmax
'-----

SCREEN 12                    'VGA, 640 x 480 Pixel, 16 Farben
px = 400: py = 400          'Bildschirmbereich

'q-Raum                      'mathematischer Bereich
xmin = -2: xmax = 2
ymin = -2: ymax = 2
zmin = 0: zmax = 1

xschritt = (xmax - xmin) / px    'Schrittweite
yschritt = (ymax - ymin) / py
zschritt = .02

CLS

'Berechnung der Julia-Menge durch Abtasten der
'Bildschirmpunkte mit Test auf Mengenzugehoerigkeit
FOR sx = 0 TO px              'x-Komponente (Bildschirmspalten)
  x = xmin + sx * xschritt
  FOR zy = 0 TO py            'y-Komponente (Bildschirmzeilen)
    y = ymax - zy * yschritt
    FOR z = zmin TO zmax STEP zschritt    'z-Komponente
      x1 = x: y1 = y: z1 = z
      FOR n = 1 TO nmax        'Iterationsschleife
        xx = x1 * x1 - y1 * y1 - z1 * z1 + a
        y1 = 2 * x1 * y1 + b
        z1 = 2 * x1 * z1
        x1 = xx
        IF x1 * x1 + y1 * y1 + z1 * z1 > 16 THEN EXIT FOR
      NEXT n
      IF x1 * x1 + y1 * y1 + z1 * z1 <= 16 THEN
        'Darstellung der z-Komponente durch Hoehenlinien
        FOR i = 0 TO 10
          IF z >= i * .1 AND z < (i + 1) * .1 THEN
            'Punktausgabe
            PSET (sx, zy), 3 + i
            EXIT FOR
          END IF
        NEXT i
      END IF
    NEXT z
  NEXT zy
NEXT sx

PRINT "c-Parameter:"
PRINT "("; a; ", "; b; ")"

END

```



```

'-----
'Quellprogramm FEIGE3.BAS
'-----
'Berechnung der Feigenbaum-Konstanten nach dem
'Newton-Verfahren
'-----

CLS

PRINT "i", "m", "a(i)", "del(i)", "a'(i)"
PRINT "-----";
PRINT "-----"

i = 0
m = 2 ^ i
a(i) = 0
PRINT i, m, a(i)

i = 1
m = 2 ^ i
a(i) = -1      'Parameter a
d(i) = 4      'Feigenbaum-Konstante (1. Naehierung)
'Schaetzwert fuer naechstes a(i+1)
aa(i) = a(i) + (a(i) - a(i - 1)) / d(i)
PRINT i, m, a(i), d(i), aa(i)

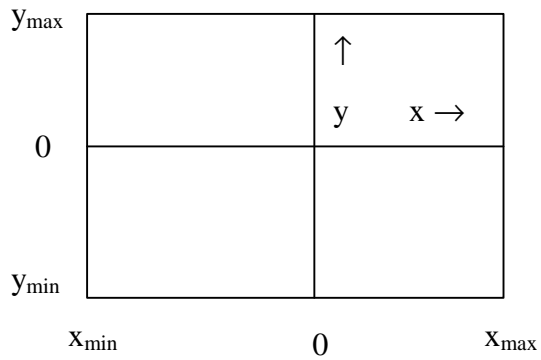
FOR i = 2 TO 6
    m = 2 ^ i
    a = aa(i - 1)
    DO      'Schleife zur Nullstellenbestimmung
        x = 0
        x = x * x + a
        xstr = 1
        FOR k = 2 TO m      'm-te Iterierte
            xstr = 2 * x * xstr + 1      'Ableitung
            x = x * x + a      'Funktion
        NEXT k
        a = a - x / xstr      'Newton-Naehierung
    LOOP UNTIL ABS(x) < .00001
    a(i) = a
    IF i = 2 THEN
        d(i) = 4
    ELSE
        d(i) = (a(i - 2) - a(i - 1)) / (a(i - 1) - a(i))
    END IF
    aa(i) = a(i) + (a(i) - a(i - 1)) / d(i)
    PRINT i, m, a(i), d(i), aa(i)
NEXT i

END

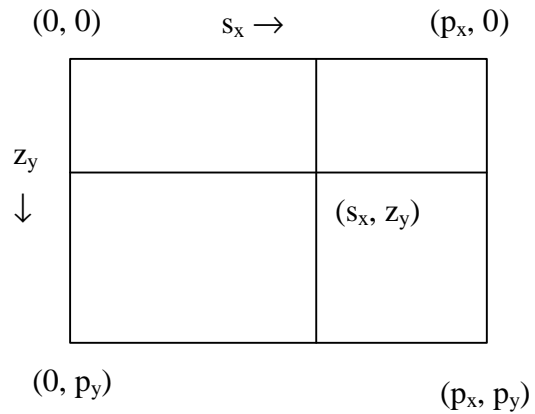
```

## 6.2 Koordinaten – Transformation

### mathematische Koordinaten



### Bildschirmkoordinaten

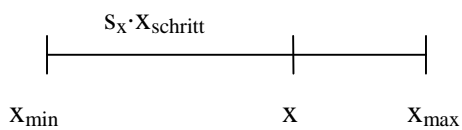


### Schrittweite

$$x_{\text{schrift}} = (x_{\max} - x_{\min}) / p_x$$

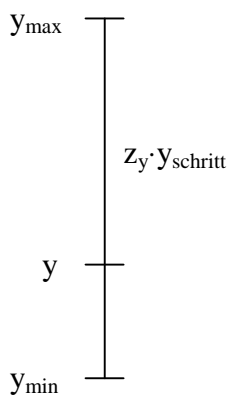
$$y_{\text{schrift}} = (y_{\max} - y_{\min}) / p_y$$

### Transformationsgleichungen



$$x = x_{\min} + s_x \cdot x_{\text{schrift}}$$

$$s_x = (x - x_{\min}) / x_{\text{schrift}}$$



$$y = y_{\max} - z_y \cdot y_{\text{schrift}}$$

$$z_y = (y_{\max} - y) / y_{\text{schrift}}$$

## 6.3 Konvergenzsatz

### Satz

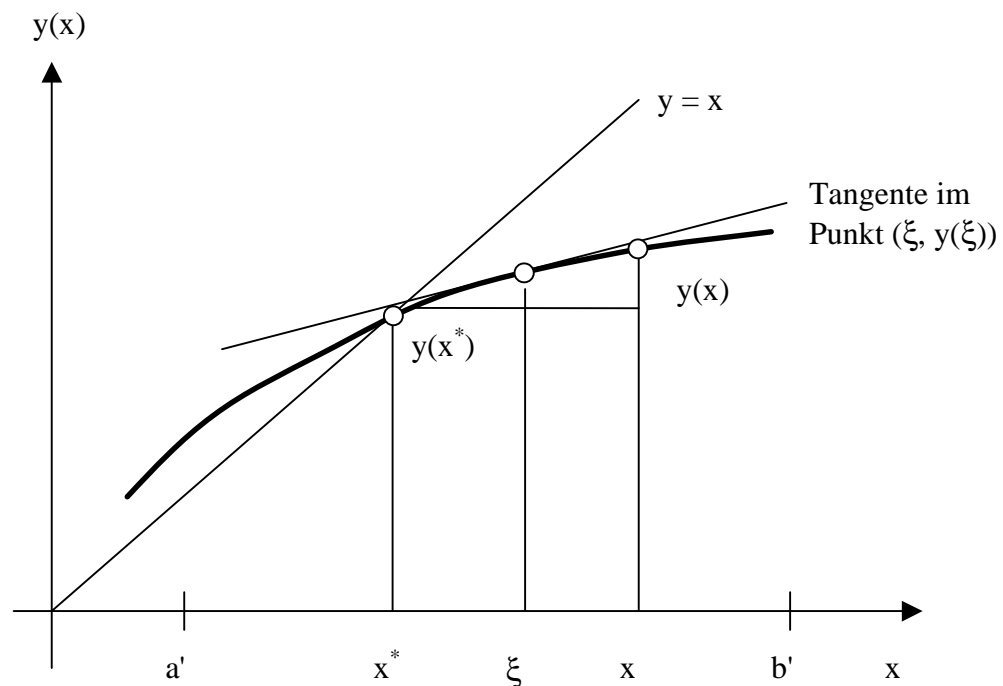
Es sei  $x^*$  Fixpunkt der Iteration  $x_{n+1} = y(x_n)$ . Ferner sei  $y'(x)$  stetig in einem Intervall  $(a, b)$  um den Punkt  $x^*$  herum und es gelte  $|y'(x^*)| < 1$ . Liegt dann der Startwert  $x_0$  der Rekursion nahe genug an  $x^*$ , so strebt  $x_n$  gegen  $x^*$ .

### Beweis

Wegen  $|y'(x^*)| < 1$  lässt sich ein abgeschlossenes Teilintervall  $[a', b']$  von  $(a, b)$  finden, in welchem auch noch  $|y'(x)| < 1$  gilt. Für jeden Punkt  $x$  aus  $[a', b']$  kann man nach dem Mittelwertsatz der Differentialrechnung schreiben:

$$\frac{y(x) - y(x^*)}{x - x^*} = y'(\xi) \quad \text{mit} \quad \xi \in [x^*, x] \subseteq [a', b'],$$

wobei  $\xi$  ein nicht näher angegebbarer Punkt zwischen  $x^*$  und  $x$  ist.



Da  $y(x^*) = x^*$  ist, können wir auch schreiben:

$$y(x) - x^* = y'(\xi) \cdot (x - x^*)$$

Der Startpunkt  $x_0$  der Rekursion möge nun im Intervall  $[a', b']$  liegen. Da  $y(x_0) = x_1$  ist, folgt:

$$x_1 - x^* = y'(\xi_0) \cdot (x_0 - x^*) \quad \text{mit} \quad \xi_0 \in [x^*, x_0]$$

$\xi_0$  liegt in  $[a', b']$  und daher ist  $|y'(\xi_0)| < 1$ , d.h.:

$$|x_1 - x^*| < |x_0 - x^*|$$

Diese Ungleichung besagt, daß  $x_1$  näher am Fixpunkt  $x^*$  liegt als  $x_0$  und auch in  $[a', b']$ .

Indem man  $x_1$  und alle weiteren Rekursionen  $x_n$  ebenso behandelt wie  $x_0$ , stellt man fest, daß alle diese Punkte in  $[a', b']$  liegen und man allgemein schreiben kann:

$$x_{n+1} - x^* = y'(\xi_n) \cdot (x_n - x^*) \quad \text{mit} \quad \xi_n \in [x^*, x_n]$$

Es sei nun  $k$  das Maximum von  $|y'(x)|$ :  $k = \text{Max}(|y'(x)|) < 1$  in  $[a', b']$ .

Da  $\xi_n \in [a', b']$  ist, folgt  $|y'(\xi_n)| \leq k$  und es gilt:

$$|x_{n+1} - x^*| \leq k \cdot |x_n - x^*|$$

Also:

$$|x_1 - x^*| \leq k \cdot |x_0 - x^*|$$

$$|x_2 - x^*| \leq k \cdot |x_1 - x^*| \leq k^2 \cdot |x_0 - x^*|$$

$$|x_3 - x^*| \leq k \cdot |x_2 - x^*| \leq k^3 \cdot |x_0 - x^*|$$

.....

$$|x_n - x^*| \leq k \cdot |x_{n-1} - x^*| \leq k^n \cdot |x_0 - x^*|$$

Wegen  $k < 1$  strebt  $k^n \rightarrow 0$  für  $n \rightarrow \infty$  und es folgt:

$$\lim_{n \rightarrow \infty} |x_n - x^*| = 0$$

oder

$$\lim_{n \rightarrow \infty} x_n = x^* \quad \diamond$$

## 6.4 Berechnung der Feigenbaum–Konstanten

Ausgangspunkt ist die Iterationsgleichung

$$y_a(x) = x^2 + a$$

In Kap. 2.6 haben wir für diese Iteration Parameterwerte  $a_0, a_1, a_2, \dots$  experimentell gefunden, ab denen sich 1er–, 2er–, 4er–, 8er–Zyklen u.s.w. ergeben haben. Mit diesen Parametern konnten wir die Feigenbaum–Konstante  $\delta$  und den Feigenbaum–Punkt  $a_\infty$  abschätzen.

Bei der näherungsweisen, numerischen Berechnung von  $\delta$  geht man von einem anderen Satz von Parameterwerten  $a(i)$  aus und zwar von den sogenannten superattraktiven Parameterwerten, für die die Iteration besonders schnell und direkt konvergiert. Diese sind durch folgenden Zyklus mit dem kritischen Punkt  $\boxed{x_0 = 0}$  als Zykelselement definiert:

$$Z = \{0; y_{a(i)}^1(0) \neq 0; y_{a(i)}^2(0) \neq 0; \dots; y_{a(i)}^{2^{i-1}}(0) \neq 0\}; y_{a(i)}^{2^i}(0) = 0$$

Es sind also Parameterwerte  $a$  gesucht, die die Bedingung

$$y_a^m(0) = g(a) = 0 \quad \text{mit} \quad m = 2^i; \quad i = 0, 1, 2, 3, \dots$$

der minimalen Periode  $m$  erfüllen.  $y_a^m(0)$  ist als Funktion von  $a$  anzusehen.

Die ersten Parameterwerte  $a(i)$  können wir explizit angeben:

$$i = 0: \quad y_a^1(0) = x^2 + a = 0$$

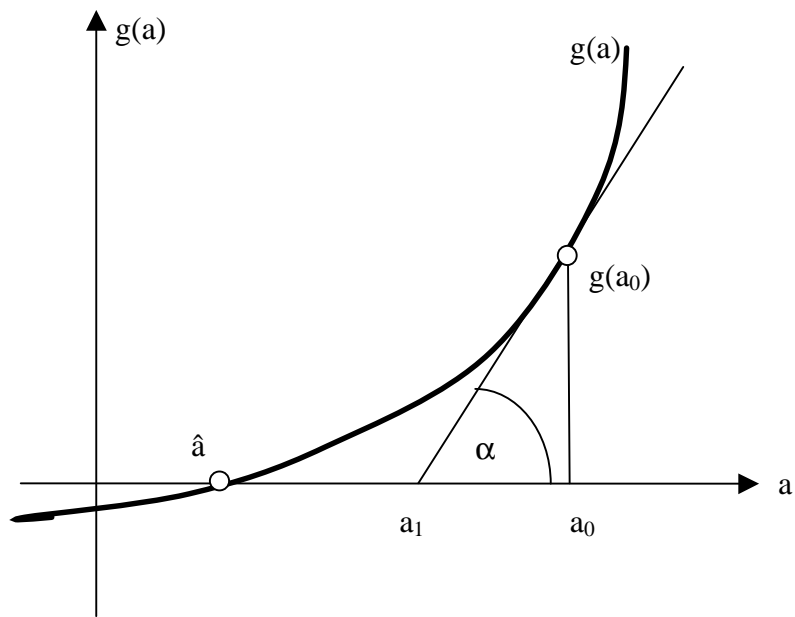
$$\Rightarrow a(0) = 0$$

$$i = 1: \quad y_a^2(0) = (x^2 + a)^2 + a$$

$$= x^4 + 2ax^2 + a(a + 1) = 0$$

$$\Rightarrow a(1) = -1$$

Die weiteren Parameterwerte  $a(2), a(3), \dots$  müssen numerisch berechnet werden. Dabei wenden wir zur Nullstellenberechnung der Gleichung  $g(a) = 0$  das Newton–Verfahren (Isaac Newton, 1643 – 1727) an:



Aus der Abbildung ergibt sich:

$$\tan \alpha = g'(a_0) = \frac{g(a_0)}{a_0 - a_1}$$

Daraus errechnet sich:

$$a_1 = a_0 - \frac{g(a_0)}{g'(a_0)}$$

Der Wert  $a_1$  stellt gegenüber dem Wert  $a_0$  eine bessere Näherung für die Nullstelle  $\hat{a}$  dar. Wird dieses Verfahren wiederholt angewendet nach der Iterationsgleichung

$$a_{k+1} = a_k - \frac{g(a_k)}{g'(a_k)}$$

dann kann die Nullstelle  $\hat{a}$  mit beliebiger Genauigkeit berechnet werden. Eine Abbruchbedingung für die Iteration kann darin bestehen, daß  $|g(a_k)|$  einen vorgegebenen, kleinen Wert  $\epsilon$  unterschreitet.

Das Problem besteht in der Berechnung der Ableitung  $g'(a)$ . Dazu schauen wir uns das folgende Iterationsschema an:

<u>Funktion</u>	<u>Ableitung nach a</u>
$x_0 = 0$	
$x_1 = x_0^2 + a$	$x_1' = 0$
$x_2 = x_1^2 + a$	$x_2' = 2x_1x_1' + 1$
$x_3 = x_2^2 + a$	$x_3' = 2x_2x_2' + 1$
.....	.....
$x_m = y_a^m(x_0 = 0) = g(a)$	$x_m' = y_a^m'(x_0 = 0) = g'(a)$
$= x_{m-1}^2 + a$	$= 2x_{m-1}x_{m-1}' + 1$

Somit kann die m-te Iterierte und deren Ableitung sukzessive berechnet werden.

Die Nullstellenbestimmung liefert dann die weiteren Parameterwerte  $a(2), a(3), \dots, a(n), \dots$

Mit diesen Parameterwerten kann eine Schätzung für die Feigenbaum-Konstante  $\delta$  angegeben werden:

$$\delta(n) = \frac{a(n-2) - a(n-1)}{a(n-1) - a(n)}$$

Außerdem läßt sich damit eine Näherung  $a'(n)$  für den nächsten Parameterwert  $a(n+1)$  berechnen:

$$a'(n) = a(n) + (a(n) - a(n-1)) \cdot \frac{1}{\delta(n)}$$

Dieser Wert wird dann als Startwert für die nächste Berechnung von  $a(n+1)$  benutzt.

Das beschriebene Verfahren wird mit dem Programm FEIGE3 realisiert. Man erhält folgenden Ergebnisausdruck:

i	m	a(i)	del(i)	a'(i)
0	1	0		
1	2	-1	4	-1.25
2	4	-1.310703	4	-1.388378
3	8	-1.381548	4.385675	-1.397701
4	16	-1.396945	4.601014	-1.400292
5	32	-1.400253	4.65493	-1.400964
6	64	-1.400962	4.667452	-1.401114

Die Berechnung der Näherungswerte der Konstanten  $\delta$  und  $a_\infty$  erfolgt bis  $i = 6$  also bis zur 64. Iterierten. Eine weitere Berechnung ist nicht mehr sinnvoll, da die Parameterwerte  $a(n)$  immer dichter liegen und dann die Rechengenauigkeit des Rechners hier nicht mehr ausreicht.

## 6.5 Internet–Adressen

- [A] [www-users.rwth-aachen.de/niko.wilbert/frac/salomo/index.htm](http://www-users.rwth-aachen.de/niko.wilbert/frac/salomo/index.htm)  
(Programm "Salomo" von Niko Wilbert)
- [B] <http://www.physcip.uni-stuttgart.de/phy11733>  
(Programm "Quat" von Dirk Meyer)
- [C] [www.qbasic.de](http://www.qbasic.de)  
(QuickBASIC)



## 6.6 Literaturhinweise

- [1] H.–O. Peitgen; H. Jürgens; D. Saupe:  
Chaos – Bausteine der Ordnung,  
rororo Hamburg 1998
- [2] H.–O. Peitgen; H. Jürgens; D. Saupe:  
Bausteine des Chaos – Fraktale,  
rororo Hamburg 1998
- [3] J. Dufner; A. Roser; F. Unseld:  
Fraktale und Julia–Mengen,  
Deutsch Frankfurt 1998
- [4] R. Behr:  
Ein Weg zur fraktalen Geometrie,  
Klett Stuttgart 1998
- [5] H. Zeitler; W. Neidhardt:  
Fraktale und Chaos  
Wiss. Buchges. Darmstadt 1994
- [6] F. Erwe:  
Differential– und Integralrechnung, Band 1,  
Bibl. Inst. Mannheim 1988
- [7] J. Hückstadt:  
Schnell–Übersicht QuickBasic 4.0/4.5,  
Markt&Technik München 1989