Attraktoren mit Flare-Dynamik in Systemen fern vom Gleichgewicht

D I S S E R T A T I O N

der Fakultät für Chemie und Pharmazie
der Eberhard-Karls-Universität Tübingen

zur Erlangung des Grades eines Doktors
der Naturwissenschaften

2002

vorgelegt von

Georg Hartmann

Tag der mündlichen Prüfung: 14. Dezember 2001
Dekan: Prof. Hans-Georg Probst
1. Berichterstatter: Prof. Otto Rössler
2. Berichterstatter: Prof. Claus Kahlert

Die vorliegende Arbeit wurde vom Januar 1995 bis Ende 2000 unter der Leitung von Prof. Dr. O. E. Rössler am Lehrstuhl für Theoretische Chemie der Universität Tübingen angefertigt.

Zusammenfassung

In der vorliegenden Schrift wird eine neuartige Klasse von Attraktoren in der Disziplin der Dynamischen Systeme definiert und eingehend untersucht. Attraktoren können in chemischen Reaktionssystemen auftreten, wenn sich diese weit entfernt vom thermodynamischen Gleichgewicht befinden. Man spricht dann von „Systemen fern vom Gleichgewicht", englisch „far-from-equilibrium systems". Solche Systeme bilden einen Teil der allgemeineren Klasse dissipativer Dynamischer Systeme.

Nichtlineare dissipative Dynamische Systeme können in der Chemie in offenen chemischen Reaktionen auftreten, wenn — als notwendige Bedingung — mindestens ein nichtlinearer Reaktionsschritt im System auftritt: Autokatalyse (deren Rückreaktion immer nichtlinear ist) oder Selbstinhibition wären Beispiele. Außerdem kann auch die Temperaturabhängigkeit der Reaktionskonstanten ein System nichtlinear machen.

Autokatalyse und Selbstinhibition können bei echtem thermodynamischem Gleichgewicht jedoch nicht auftreten, da dann die jeweilige Rückreaktion in gleichem Maße stattfinden muss. Reaktionssysteme mit derartigen Teilreaktionsschritten können fern vom Gleichgewicht ein komplexes zeitliches (wie hier untersucht) oder räumliches Verhalten aufweisen. Letzteres tritt z.B. in der recht bekannten Belousov-Zhabotinskii-Reaktion in Form von Spiralwellen, konzentrischen Ringen aber auch unregelmäßigen Mustern auf. Man spricht auch von zeitlicher bzw. räumlicher Selbstorganisation.

Ein zuvor offenbar noch nicht klassifiziertes, zeitliches dynamisches Verhalten ist das hier beschriebene „Flare"-Verhalten. Der in diesem Zusammenhang zunächst vielleicht etwas ungewöhnlich anmutende astronomische Begriff „Flare" steht für einen plötzlich unvermittelt auftretenden kurzzeitigen Übergang eines Dynamischen Systems aus einem steady-state-

(„Fließgleichgewichts-")nahen in einen „steady-state-fernen" Zustand und wieder zurück. Vergleicht man ein solches zeitliches Verhalten mittels graphischer Schaubilder mit gemessenen Intensitätsverläufen astronomisch bekannter Flare-Sterne, wie beispielsweise Proxima Centauri, so wird die Verwendung dieses aus der Astrophysik entlehnten Begriffes unmittelbar einleuchtend.

Im Rahmen der vorliegenden Arbeit werden sowohl kontinuierliche als auch diskrete Dynamische Systeme mit Flare-haftem Verhalten vorgestellt. Man erhält unter Anwendung eines definierten „Synthese-Prinzips" dabei stets Phasenraumschaubilder und Zeitreihendarstellungen von einer charakteristischen Morphologie. Die Untersuchung diskreter Dynamische Systeme ist in den Naturwissenschaften vor allem deshalb von Bedeutung, weil Phasenraumdarstellungen kontinuierlicher dynamischer Flüsse (mathematisch durch eine Differentialgleichung darstellbar), wie wir sie bei physikalischen und chemischen Systemen in der Regel vorfinden, im sogenannten „Poincaré-Schnitt" ein diskretes Dynamisches System mit einer um eins niedrigeren Dimenionsion darstellen. Dies erleichert die analytische Betrachtung und Charakterisierung der zugrunde liegenden Dynamik.

In einem abschließenden Kapitel wird darüber hinaus noch ein konservatives (volumenerhaltendes) dynamisches System mit flarehaften Trajektorie vorgestellt; die Systeme der ersten Kapitel sind dagegen vom dissipativen (nicht volumenerhaltenden) Typ.

Publikationen

- O. E. Rössler und G. C. Hartmann. Attractors with flares. *Fractals* **3**, 285–296, 1995.

- G. C. Hartmann und O. E. Rössler. Flaring — A new type of dynamical behaviour. In *Fractal Reviews in the Natural and Applied Sciences*, Chapman & Hall, London, 1995, S. 372–375.

- G. C. Hartmann und O. E. Rössler. A self-similar flare attractor. In *4th Experimental Chaos Conference — Abstracts book* (Boca Raton, FL, U.S.A, 6.–8. August 1997), S. 97–98.

- O. E. Rössler und G. C. Hartmann. A society of flare attractors. In S. C. Müller, Hrsg., *Diffusion and Structure Formation — Abstracts book*, 18th WE-Heraeus Seminar, 10.–12. November 1997, Bad Honnef, S. 22–23.

- G. C. Hartmann und O. E. Rössler. Coupled flare attractors. *Discrete Dynamics in Nature and Society*, **2**, 153–159, 1998.

- G. C. Hartmann, G. Radons, H. H. Diebner und O. E. Rössler. Staircase baker's map generates flaring type time series. *Discrete Dynamics in Nature and Society*, **5**, 107–120, 2000.

- B. Rosser, E. Ahmed und G. C. Hartmann. Volatility via social flaring. *Journal of Economic Behavior and Organization* (eingereicht).

Inhaltsverzeichnis

Zusammenfasssung i

Liste der Publikationen iii

I Grundlagen **1**

1 Dynamische Systeme **3**

1.1 Definition 3

1.2 Nichtlineare Dynamik 4

1.3 Attraktoren 4

1.4 Kontinuierliche versus diskrete Systeme 5

2 Dynamische Systeme in den Naturwissenschaften **9**

2.1 Physikalische Systeme 9

2.2 Chemische und biochemische Systeme 10

2.3 Biologisch-medizinische Systeme 11

3 Dynamische Systeme mit „Flare"-Verhalten **13**

3.1 Einleitung 13

3.2 Das „Flare"-Prinzip 15

3.3 Attraktoren vom Flare-Typ 16

3.4 Vergleich verschiedener Forcings für Flare-Systeme 23

II Modellrechnungen kontinuierlicher Systeme durch numerische Simulation 27

4 Numerisches Lösen von Differentialgleichungssytemen 29

4.1 Analytisches vs. numerisches Lösen von Differentialgleichungssytemen 29

4.2 Analogrechner und SIMULINK 30

5 Numerische Modellsimulationen von kontinuierlichen Flare-Systemen 33

5.1 Geeignete Integrationsmethoden 33

5.2 Chemische Reaktionskinetiksysteme 34

5.3 Reaktionskinetik-Systeme mit Flare-Dynamik 37

III Modellrechnungen diskretisierter Systeme durch numerische Simulation 41

6 Nichtinvertierbare Maps 43

6.1 Einfache Maps 43

6.1.1 Logistische Map 44

6.1.2 Bernoulli-Shift 45

6.1.3 Tent-Map 47

6.2 Ein komplexeres System 47

7 Invertierbare Abbildungen **51**

7.1 Einleitung 51

7.2 Ein erstes invertierbares Modell 51

7.3 Weiterere invertierbare Modellsysteme 54

7.3.1 Baker's Map 54

7.3.2 Mehrfach gefaltete Horseshoe-Map 59

7.3.3 Inverse Baker's-Map 63

8 Gekoppelte Zellen mit Flare-Dynamik **67**

8.1 Einleitung 67

8.2 Eine einfache Summendynamik 68

IV Flare-Dynamik in einem konservativem System 73

9 Variationen zur Baker's Map **75**

9.1 Vorbemerkungen 75

9.2 Die Variation der Baker's Map nach Hopf und Gaspard 75

10 Staircase Baker's Map mit flarehaftem dynamischem Verhalten **77**

10.1 Definition 77

10.2 Numerische Simulationen der Staircase Baker's Map 81

10.3 Poincaré-Rekurrenz bei der Staircase Baker's Map 85

10.4 Die Staircase Baker's Map als Multiple Baker's Map 88

V Diskussion **91**

Anhang **97**

A Source Codes **97**

A.1 Simulationsprogramm für Iterations-Maps 98

A.2 Gekoppelte Flare-Systeme 105

A.3 Staircase Baker's Map mit multiplem Präzisions-Algorithmus . 118

A.4 3D Simulationsplotter 131

Abbildungsverzeichnis **137**

Literaturverzeichnis **140**

Teil I

Grundlagen

Kapitel 1

Dynamische Systeme

1.1 Definition

Ein Dynamisches System ist definiert durch die gesetzmäßige zustandsabhängige Änderung seines Zustands mit der Zeit (t). In der Anwendung unterscheidet man zwischen zwei Typen von Dynamischen Systemen: Zum einen solchen, bei denen der Zeitparameter kontinuierlich ist ($t \in \mathbb{R}$) und zum anderen Systeme mit Zeit-Diskretisierung ($t \in \mathbb{N}$ oder $\mathbb{Z}$).

Kontinuierliche Dynamische Systeme werden im Allgemeinen mittels einer *Differentialgleichung* beschrieben

$$\dot{x} \;=\; \mathbf{X}(\mathbf{x}). \tag{1.1}$$

Geht man zu diskretisierter Zeit über, so stellt man das System mittels *Iteration* von Funktionen dar

$$\mathbf{x}_{t+1} \;=\; f(\mathbf{x}_t), \qquad\qquad t \in \mathbb{N} \text{ oder } \mathbb{Z}. \tag{1.2}$$

In den Formeln (1.1) und (1.2) stellt $\mathbf{x}$ den Zustand des Systems im sogenannten *Phasenraum* (manchmal auch *Zustandsraum* genannt) dar. Oftmals ist dieser Phasenraum ein euklidischer Raum bzw. ein Unterraum desselben.

Es kann allerdings auch eine nicht-euklidische Struktur vorliegen, wie Kreis, Kugel, Torus bzw. irgendeine andere *differenzierbare* Mannigfaltigkeit [1].

Der zeitliche Ablauf der sich ändernden Zustände im Phasenraumschaubild des Dynamischen Systems wird durch die *Trajektorie* beschrieben.

1.2 Nichtlineare Dynamik

Die nichtlineare Dynamik kann man als Teil der irreversiblen Thermodynamik auffassen [2]. Kommen nichtlineare Terme in den Differential- bzw. Iterations-Gleichungen Dynamischer Systeme vor, so können eine ganze Reihe an Lösungstypen der entsprechenden Geschwindigkeitsgleichungen auftreten, wie Grenzzyklus-Oszillationen, quasiperiodische und chaotische Attraktoren. Derartige Dynamiken sind in nicht-dissipativen (auch Hamiltonsch genannten) Dynamischen Systemen nicht existent [3].

1.3 Attraktoren

Eine geometrische Form im Phasenraum, an die sich die Trajektorie eines nichtlinearen dynamischen Systems nach einer gewissen Zeitspanne (der sogenannten *Transienzzeit*) asymptotisch annähert, bezeichnet man als *Attraktor*.

Ein Attraktor ist im einfachsten Fall ein Punkt, kann aber auch ein beliebig hochdimensionales Gebilde sein; innerhalb von drei System-Dimensionen existieren als Attraktoren-Klassen (in Klammer jeweils die Dimensionalität): Punktattraktor (0 D), Grenzzyklus (1 D), Torus (2 D), unendlich oft übereinandergelegtes Band im Falle des Chaos (topologisch 2D). Es existieren (wie im zuletzt genannten Fall) auch Dynamische Systeme mit Attraktoren gebrochener Dimensionalität*. Weist ein Attraktor eine chaotische Dynamik auf, so spricht man auch oft von einem *seltsamen* Attraktor.

*Dimensionalitätsbegriff nach Hausdorff-Besicovitch.

1.4 Kontinuierliche versus diskrete Systeme

Auf eine Idee des Mathematikers Henri Poincaré geht eine der wichtigsten Methoden zur Analyse von Dynamische Systemen zurück, der nach ihm benannte *Poincaré Schnitt* [3]. Er stellt eine Diskretisierungstechnik dar, die ihre Bedeutung darin hat, daß die Dimension des resultierenden Systems um eins verringert ist.

Gegeben sei z.B. ein System von n gewöhnlichen Differentialgleichungen

$$\begin{aligned} \frac{\mathrm{d}x_1}{\mathrm{d}t} &= F_1(x_1, x_2, ..., x_n) \\ \frac{\mathrm{d}x_2}{\mathrm{d}t} &= F_2(x_1, x_2, ..., x_n) \\ &\vdots \\ \frac{\mathrm{d}x_n}{\mathrm{d}t} &= F_n(x_1, x_2, ..., x_n) \end{aligned} \tag{1.3}$$

Führt man Vektoren ein,

$$\begin{aligned} \mathbf{x} &= (x_1, x_2, ..., x_n) \\ \mathbf{F} &= (F_1, F_2, ..., F_n) \end{aligned} \tag{1.4}$$

so kann man das System von Gleichung (1.3) zusammenfassend schreiben als

$$\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t} = \mathbf{F}(\mathbf{x}). \tag{1.5}$$

Man betrachtet nun eine Trajektorie im zugehörigen n-dimensionalen Phasenraum und wählt eine passende Hyperfläche Σ der Dimension $n-1$,

$$g(x_1, x_2, ..., x_n) = 0, \tag{1.6}$$

aus, die von der Phasenkurve überall transversal geschnitten wird. Im Differentialgleichungssystem (1.5) gibt $\mathbf{F}(\mathbf{x})$ lokal die Richtung der Tangente an die Trajektorie c im Punkt $\mathbf{x}$ an.

Man erhält die Flächennormale $\mathbf{n}$ von Σ als

$$\mathbf{n} = |\text{grad } g|^{-1}\text{grad } g,$$

mit

$$\text{grad } g = \left\{ \frac{\partial g}{\partial x_1} \quad \frac{\partial g}{\partial x_2} \quad \cdots\cdots \quad \frac{\partial g}{\partial x_n} \right\}. \tag{1.7}$$

Σ soll die Trajektorie transversal schneiden, und an den Schnittpunkten muss die *Transversalitätsbedingung*

$$\mathbf{F}^t\mathbf{n} \neq 0 \tag{1.8}$$

gelten; nach Möglichkeit wählt man als Schnittfläche eine Hyperebene aus, die durch einen konstanten Wert einer der Variablen (z.B. den Wert Null) gebildet wird. Man bezeichnet die Schnittpunkte von c mit Σ, die sich nach einem vollen Umlauf ergeben, als $P^{(0)}, P^{(1)}, P^{(2)}$ usw.. Durch den momentanen Zustand ist bei invertierbaren Dynamischen Systemen auch der frühere Verlauf der Trajektorie festgelegt, so dass man auch die zurückliegenden Zustände $P^{(-1)}, P^{(-2)}, P^{(-3)}$ ermitteln kann; siehe Abb. 1.4 für das Beispiel eines 3-dimensionalen Phasenraums.

Wird jeder Punkt $P^{(n+1)}$ als Bild des vorhergehenden Punktes $P^{(n)}$ aufgefaßt, so gelangt man zu einer *diskreten* iterativen Abbildung („Map"), die umkehrbar eindeutig ist und als Poincare-Map bezeichnet wird. Sucht man nun umgekehrt — wie in den Abschnitten 7.1 bis 7.3 — nach diskreten Abbildungen, die als Poincaré-Schnitt einer höherdimensionalen kontinuierlichen Dynamik darstellbar sein sollen, so hat man auf *Invertierbarkeit* der Iterationsvorschrift zu achten (speziell darf dabei die Jakobi-Determinante nicht Null werden).

Aus dem ursprünglichen Differentialgleichungssystem wird also ein System von *Differenzengleichungen*, das die Folge der Durchstoßungspunkte beschreibt. Ein System von Differenzengleichungen bezeichnet man in der Literatur auch im Deutschen oft mit dem englischen Fachterminus „Map",

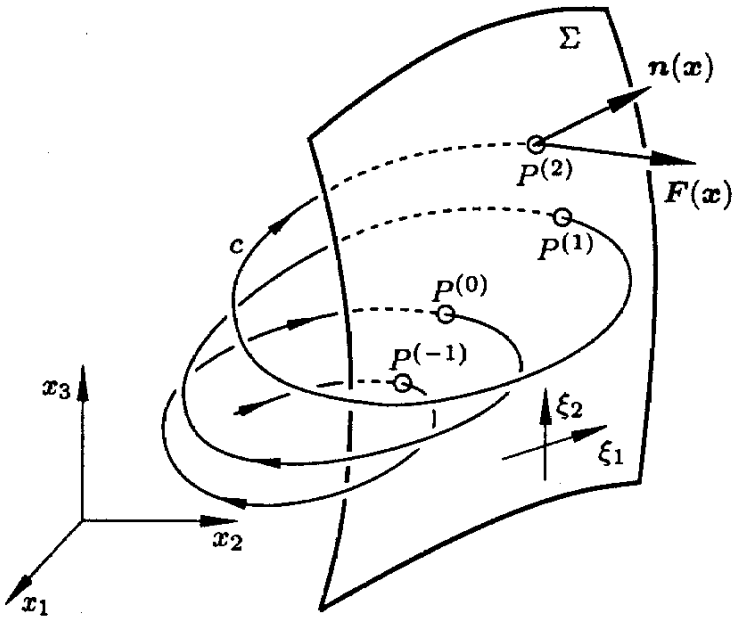


Abbildung 1.1: Technik des Poincaré-Schnitts; aus [4].

es sind aber auch die Ausdrücke „Transformation", „Iteration" oder „Abbildung" gebräuchlich und werden hier — gelegentlich — synonym verwandt.

Da nach dem Suspensions-Theorem („Aufhängungstheorem") von Smale (1967) zu jeder invertierbaren Abbildung ein Dynamisches System mit demselbem Verhalten existiert, ist es „erlaubt" an Stelle von Differentialgleichungen Maps mit gleichartigem Verhalten zu studieren. Mit Hilfe einer Iterationsformel für Poincaré-Abbildungen ist es daher oft möglich, das Langzeitverhalten Dynamischer Systeme besser zu studieren, da sich die Rechenzeit zwischen zwei Schnitten auf eine einzige Operation verkürzt und man nicht mehr mittels rechenaufwendigerer numerischer approximativer Integration die komplette Trajektorie des Systems selbst verfolgen muß. Dies erspart erheblich Rechenzeit und macht die Untersuchung gewisser Dynamischer Systeme überhaupt erst möglich; außerdem erhält man dabei noch eine höhere Rechengenauigkeit, da die unvermeidlichen Fehler durch numerische Integrationverfahren hierbei nicht auftreten können [4][5].

Kapitel 2

Dynamische Systeme in den Naturwissenschaften

2.1 Physikalische Systeme

Ein einfaches physikalisches System, bei dem sich die Grundbegriffe der Theorie Dynamischer Systeme gut verstehen lassen, ist der *harmonische Oszillator*. Liegt ein echter Harmonischer Oszillator vor (ein mathematisches Pendel), so ist die Trajektorie im Orts-Impuls-Phasenraum ein Kreis bzw. eine Ellipse.

Ist — wie bei realen physikalischen Pendeln — Reibung bzw. Dämpfung vorhanden, so läuft das System nach einer gewissen Zeit in der obigen Phasenraum-Darstellung stets auf einen Punkt zu (Punktattraktor). Wird das gedämpfte Pendel jedoch angetrieben, so endet das System bei beliebig gewählten Startbedingungen im Allgemeinen auf einem zyklischen Attraktor (Grenzzyklus oder periodischer Attraktor).*

Zum Verständnis der hier und auch später immer wieder auftauchenden Phasenraumdarstellungen sei noch einmal deutlich gemacht, dass obige Pha-

*Ein periodisch angetriebenes Pendel kann allerdings bis ins Chaos getrieben werden (siehe die folgenden Abschnitte).

senraumbilder nicht etwa die Bahn des Pendels selbst darstellen. Ein Punkt repräsentiert zu jeder Zeit den Zustand des kompletten Dynamischen Systems „Pendel", also sowohl den Ort wie den Impuls (falls dies die beiden Zustandsvariablen sind).

Breite Anwendung findet die Theorie der Dynamischen Systeme auch im Zusammenhang mit der Meteorologie — in einer Arbeit von Lorenz [6] wurde erstmals ein von einem Wetter-Modell abgeleitetes einfaches Dynamisches System mit chaotischer Dynamik† vorgestellt.

2.2 Chemische und biochemische Systeme

In chemischen Systemen kann ein nichtlineares dynamisches Verhalten bereits in homogenen Systemen (wo die Konzentrationen nur zeitabhängig sind) auftreten. Die Dynamik homogener chemischer Systeme kann mittels gewöhnlicher Differentialgleichungen beschrieben werden.

Es existieren aber auch Systeme mit zusätzlicher *ortsabhängiger* Konzentrationsänderung, im einfachsten Fall als sogenannte *Reaktions-Diffusions-Systeme* wie zum Beispiel das *Turing-System* [7]. Solche Systeme lassen sich dann nur noch mit *partiellen* Differentialgleichungen darstellen.

Chemische Modellsysteme mit zeitlich komplexer nichtlinearer Dynamik wurden von Prigogine [8] vorhergesagt und finden sich u.a. in frühen Arbeiten von Rössler [9], Olsen, Degn [10] und Hudson [11]. Erste experiemtell untersuchte chemische Systeme mit (noch nicht komplexen) Phämomenen der nichtlinearen Dynamik sind die Reaktion von Bray (IO_3-katalysierte Zersetzung von Wasserstoffperoxid, 1921) und die hier schon erwähnte Belousov-Zhabotinskii-Reaktion (Malonsäure-Oxidation durch BrO_3 in schwefelsaurer Lösung, katalysiert durch wechselvalente Metallionen wie bspw. Ce^{3+} oder Mn^{2+}) von 1958. Die offenbar erste biochemische Reaktion mit experimentell beliebig langer Oszillationsdauer war die Meerrettichperoxidase-Reaktion

†In der damaligen Arbeit von Lorenz allerdings noch nicht so bezeichnet.

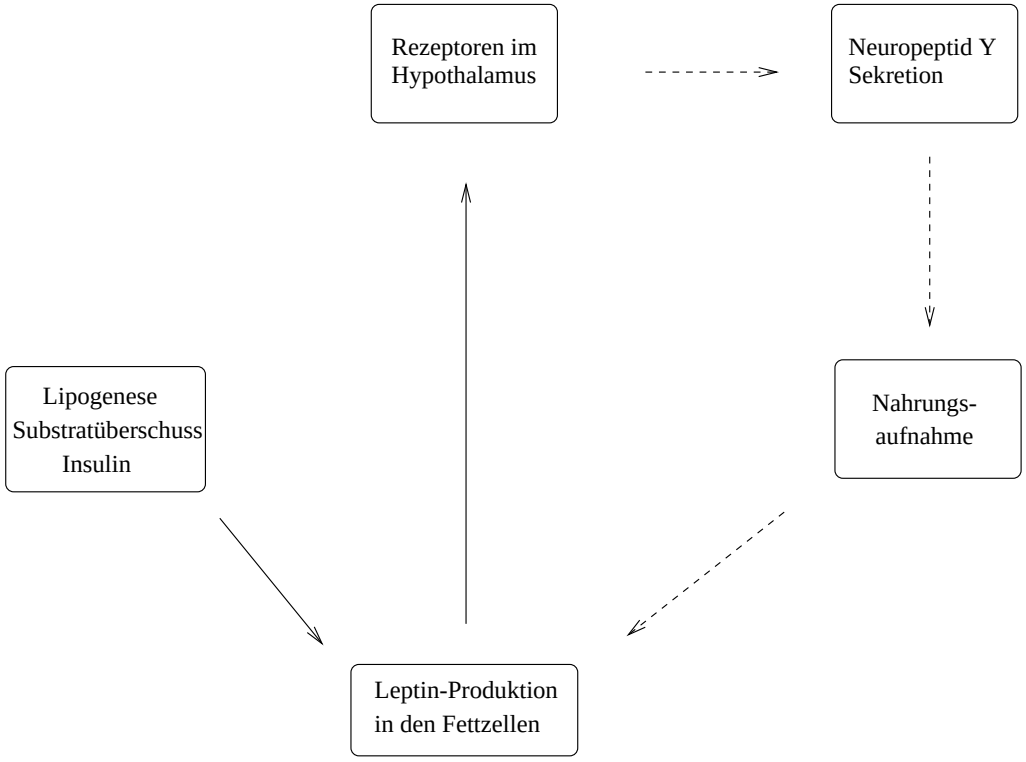


Abbildung 2.1: Physiologischer Regelkreis der Leptin-Sekretion als Beispiel für ein Dynamisches System in der Medizin; Schema nach [14]. $\longrightarrow$ = Aktivierung, $\dashrightarrow$ = Hemmung.

(Yamatzki *et.al.* 1965).

2.3 Biologisch-medizinische Systeme

Auch in der Biologie bzw. Medizin gibt es zahlreiche Beispiele für Dynamische Systeme. Populationsdynamiken sind ein breiter Anwendungsbereich iterierter Dynamischer Systeme (Maps) [12].

In der Physiologie beobachtet man sehr häufig kontinuierliche Dynamische Systeme in Form von sogenannten *Regelkreisen*. Frühe Arbeiten zur Nichtlinearen Dynamik in der Neurophysiologie findet man bei FitzHugh [13].

Als ein exemplarisches Dynamisches System in Form eines physiologischen Regelkreises im humanen Hormonsystem sei hier kurz die 1994 entdeckte dynamische Regulation der Körperfettmasse bei Mäusen aufgeführt. In solch einem einfachen Regelkreis tritt ein System-Attraktor auf: der sogenannte Punkt-Attraktor.

Diese dynamische Regulation wurde nachgewiesen bei genetisch fettsüchtigen ob/ob-Mäusen; beim Menschen wird derselbe Mechanismus diskutiert. Bei Stimulation durch ein Überangebot an Energiesubstrat, Massenzunahme oder aktive Lipogenese sezernieren die Fettzellen das Peptidhormon *Leptin* ins Blut. Durch auf dieses Hormon ansprechenden Leptinrezeptoren im Gehirn werden hypothalamische Zentren erregt, woraufhin dort die Sekretion des sogenannten *Neuropeptid Y* verringert wird. Da ein hoher Spiegel dieses Peptids ein Sättigungsgefühl verhindert und zu verstärkten Energieaufnahme führt, bewirkt die erzielte Senkung des *Neuropeptid Y*-Spiegels also eine Senkung der Energieaufnahme. So können sich Nahrungsaufnahme und Körpermasse stets dynamisch aneinander anpassen [14].

Wird in einem zweidimensionalen Phasenraum-Diagramm für dieses Beispiel Körpermasse und tägliche Nahrungsaufnahme aufgetragen, so resultiert im Zustand der Homöostase, also bei gleichbleibendem innerem Milieu des Körpers und konstanten sonstigen Aussenbedingungen, ein Punkt-Attraktor (in erster Näherung).

Kapitel 3

Dynamische Systeme mit „Flare"-Verhalten

3.1 Einleitung

Aus Arbeiten von J. Milnor [15] ließen sich Systeme ableiten, deren Trajektorien bzw. Zeitreihen „Flare"-Verhalten aufweisen [16]. Wie bereits kurz angesprochen zeigen die Zeitserienbilder solcher Dynamischer Systeme eine erstaunliche Ähnlichkeit in der Morphologie zu astrophysikalisch gemessenen Strahlungsintensitätszeitserien von Flare-Sternen. Exemplarisch seien hier zwei solche Zeitserien einander gegenübergestellt.

An dieser Stelle kann nun auf ein zuvor offenbar nicht bekanntes allgemeines synthetisches Prinzip eingangen werden, das es erlaubt, solches Verhalten sowohl in kontinuierlichen, als auch in diskreten Dynamischen Systemen auf einfache Art zu erzeugen. Die hierbei auftretenden Attraktoren lassen sich als eine eigene Klasse im „Zoo" der Attraktoren der Nichtlinearen Dynamik auffassen [17].

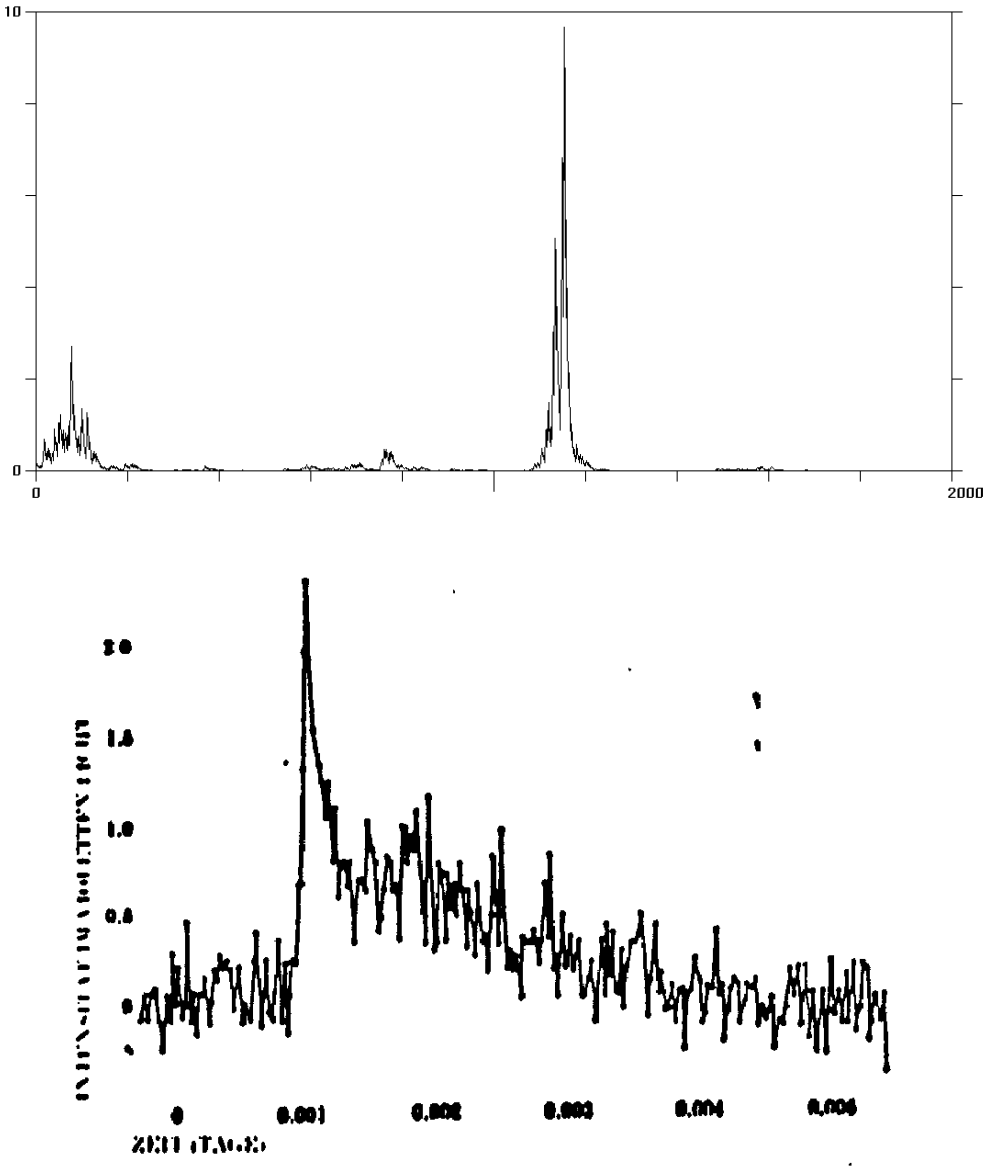


Abbildung 3.1: Zeitserienvergleich: **Oben** Modellsystem vom Flare-Typ — **unten** Intensitätsmessung des Flare-Sterns Wega 424 AB.

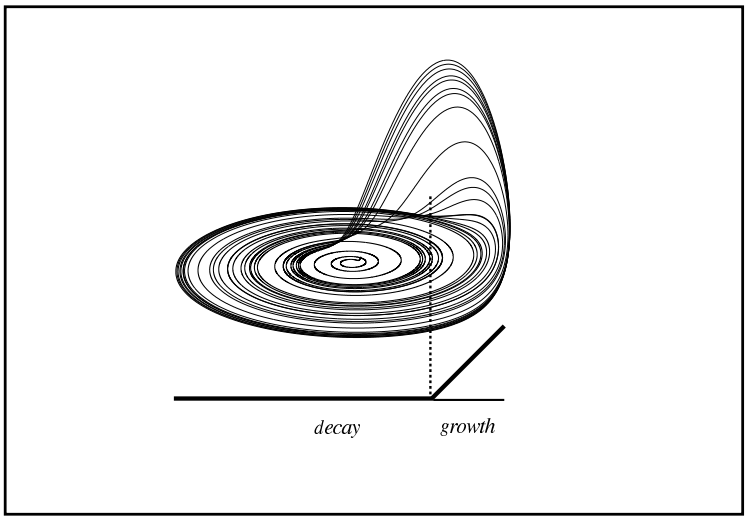


Abbildung 3.2: Schematisches Grundprinzip eines Dynamischen Systems vom Flare-Typ. Ein chaotisches Teilsystem treibt ein nichtlineares System, das wie ein Schalter eine Schwelle besitzt. Je nach dem momentanen Wert des Antriebs — unterschwellig oder überschwellig — kommt es entweder zu Selbstinhibition (Dämpfung) oder zum autokatalytischen Wachstum (Entdämpfung) der angetriebenen Variablen; aus [18].

3.2 Das „Flare"-Prinzip

Es wird ein einfaches Prinzip zur Erzeugung Dynamischer Systeme mit Flare-Verhalten vorgestellt. Dabei stellt sich heraus, dass die Systemgleichungen für Flüsse (kontinuierliche Dynamische Systeme) und diskretisierte Dynamische Systeme analog aufgebaut sind. Abb. 3.2 zeigt anschaulich das Prinzip, wie man zu einem Flare-Attraktor gelangt.

Im Gleichungssystem des Dynamischen Systems muss die Variable, die eine Flare-Dynamik aufweisen soll — in (3.1) jeweils w genannt —, stets einen charakteristischen, schwellenbildenden Term enthalten. Dieser Term ist bei kontinuierlichen und diskreten Systemen potentiell identisch, wie die folgenden beiden Beispiele belegen:

$$\begin{aligned} \textbf{kontinuierlich}: \quad \dot{w} &= w(bx-a)+... \\ \text{oder} \qquad \dot{w} &= w(a-bx)+... \end{aligned} \tag{3.1}$$

$$\begin{aligned} \textbf{diskret}: \quad w_{n+1} &= w_n + w_n(bx_n-a)+... \\ \text{oder} \qquad w_{n+1} &= w_n + w_n(a-bx_n)+... \end{aligned}$$

Zur Erläuterung:

w: Die Zustandsvariable mit Flare-Verhalten.

x: Schaltvariable mit nicht-periodischem (z.B. chaotischem) zeitlichem Verlauf innerhalb eines Werteintervalls.

a, b: Konstanten, deren Werte so eingestellt werden müssen, dass der Term im Zeitmittel dämpfend wirkt (die Klammer im Mittel < 0 ist), aber dennoch zwischenzeitlich entdämpfende Folgen, wo die Klammer > 0 ist, auftreten. Das ist im Allgemeinen ein relativ enger Bereich des Verhältnisses von a und b, den es mit einer Genauigkeit von ca. 1% zu treffen gilt.

Entscheidend ist hier also ein *nichtlinearer Schalter* im jeweiligen „Flare"-Term, über den der im Mittel konstante, nicht-periodisch bzw. chaotisch funktionierende Antrieb eingekoppelt wird. Je nach „Schalterstellung" resultiert so für das Beispiel eines chemischen Reaktionssystems *Autokatalyse* oder *Selbstinhibition*.

3.3 Attraktoren vom Flare-Typ — eine neue Klasse von Attraktoren

Die Idee, das „Flaring" als eine robuste Verhaltensweise von Dynamischen Systemen anzusehen, entstand durch die Betrachtung von numerischen Ergebnissen, die bei der Simulation von Milnor-Attraktoren aufgetaucht waren [19]. Milnor-Attraktoren zeigen im einfachsten Fall unbeschränkte Amplitu-

denwerte für zumindest eine Variable [20][21]. Ein Attraktor im Unendlichen und ein Attraktor im Endlichen (z.B. bei Null) teilen sich dabei die Anfangsbedingungen in einer solchen Weise, dass Punkte im endlichen Bereich mit einer mehr oder weniger gleich großen Wahrscheinlichkeit entweder zu dem einen oder zu dem anderen Attraktor tendieren, also entweder zum Bassin des einen oder des anderen Attraktors gehören. Man spricht daher auch von „durchlöcherten Bassins" (englisch „riddled basins"), weil das eine Einzugsgebiet das andere in beliebig kleinen Ausschnitten durchdringt [22].

Eine Illustration zu einer solchen fraktalen gegenseitgen fingerartigen Durchlöcherung zwischen schwarz und weiß zeigt Abb. 3.3. Die schwarzen Finger gehören zum Attraktor bei Unendlich, die weißen zum Attraktor bei Null. Man kann sich vorstellen, dass das Sichdurchdringen der weißen und schwarzen Finger tatsächlich bis in unendlich kleine Ästchen zwischen beiden weiter geht. Die zugehörige Gleichung zu dieser Abbildung (die durch Abtasten der Anfangsbedingungen entstanden ist) lautet konkret:

$$x_{n+1} = \begin{cases} x_n < \frac{1}{3} & : \quad x_{n+1} = 3x_n \\ x_n \geq \frac{1}{3} & : \quad x_{n+1} = \frac{3}{2}(1 - x_n) \end{cases}$$

$$y_{n+1} = \begin{cases} x_n < \frac{1}{3} & : \quad y_{n+1} = \frac{6}{5}y_n + \frac{1}{10} \\ x_n \geq \frac{1}{3} & : \quad y_{n+1} = \frac{3}{5}y_n + \frac{1}{10} \end{cases} \tag{3.2}$$

Flare-Attraktoren stellen nun sozusagen eine „gezähmte" Art von Milnor-Attraktoren dar, da sie nur endliche Amplituden aufweisen und vor allem „generisch" sind. Mit anderen Worten, ihr Verhalten kann nicht mehr durch unendlich kleine Änderungen oder Einfügungen eines unendlich kleinen weiteren Parameters qualitativ verändert werden. Ein Beispiel kann dies klar

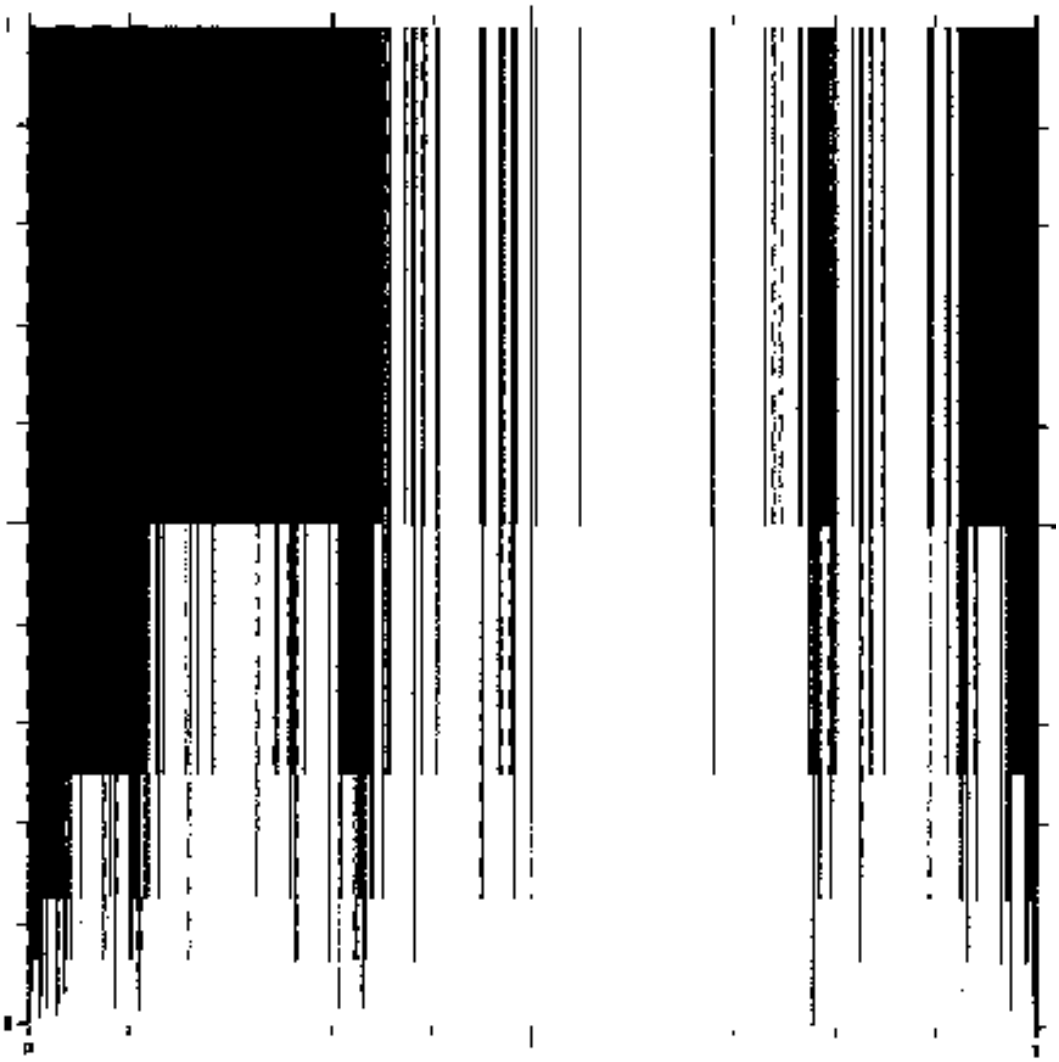


Abbildung 3.3: „Riddled Basins" Attraktor gemäß Gl. 3.2; Abszisse x [0,1], Ordinate y [0,1]

machen. In der folgenden Gleichung ist die erste Variable die sogenannte logistischen Abbildung, die mit dem Paramter 4 bekanntlich einen idealisierten Fall von Chaos erzeugt (vgl. [23]):

$$\begin{aligned} x_{n+1} &= 4x_n \cdot (1 - x_n) \\ w_{n+1} &= w_n + w_n(x_n - \mathit{Schwellenwert}) - \epsilon {w_n}^2 \end{aligned} \tag{3.3}$$

Die Variable der zweiten Gleichung wächst autokatalytisch, solange ihr Input, x_n, größer als der „Schwellenwert" ist. Der Schwellenwert könnte z.B. den Zahlenwert 0,63 haben. In dieser Gleichung erhält man einen Milnor-Attraktor, wenn ϵ exakt Null ist, sodass w beliebig hoch anwachsen kann, d.h. nicht wachstumbeschränkt ist (bei positiven Anfangsbedingungen!). Offensichtlich ist dies eine strukturell instabile Situation: Jeder noch so kleine Wert von ϵ begrenzt das maximale Wachstum von w auf endliche Werte und der resultierende „Flare-Attraktor" ist ein gewöhnlicher Attraktor mit einem beschränkten und nicht durchlöcherten Einzugsgebiet. Dennoch erinnert er in seinem Zeitverhalten, wenn man genügend kurze Zeiträume betrachtet, so dass das Ausbleiben sehr grosser Peaks nicht auffällt, sehr stark an das eines Milnor-Attraktors.

Flare-Attraktoren gehören im invertierbaren Fall* ihrerseits in die größere Klasse der Kaplan-Yorke-Attraktoren [24]. Das bedeutet, sie besitzen ein ungewöhnliches Spektrum ihrer Lyapunov-charakteristischen Exponenten (LCE)†. Der eine positive Lyapunov-charakteristische Exponent, der durch das Chaos der ersten Variablen bedingt ist, ist in seinem Betrag grösser als der kleinste negative Lyapunov-Exponent, der im vorliegenden Fall der mittleren Dämpfung der zweiten Variablen entspricht.

Als Folge springt die fraktale Dimension des Attraktors, die sogenannte Lynapunov-Dimension, um 1 nach oben. Das bedeutet, dass sich die

*Zur Invertierbarkeit von Flare-Atttrakoren, vgl. Kap. 7.1 bis 7.3

†Die LCE stellen den gemittelten Eigenwert des auf die Trajektorie als Pseudo-Steady-State bezogenen linearisierten System dar (Mitflug-Dynamik nach Lyapunov). Kontinuierliche Dynamische Systeme haben immer einen LCE, der exakt Null ist. Die Zahl der LCE ist hierbei immer gleich der Variablenzahl.

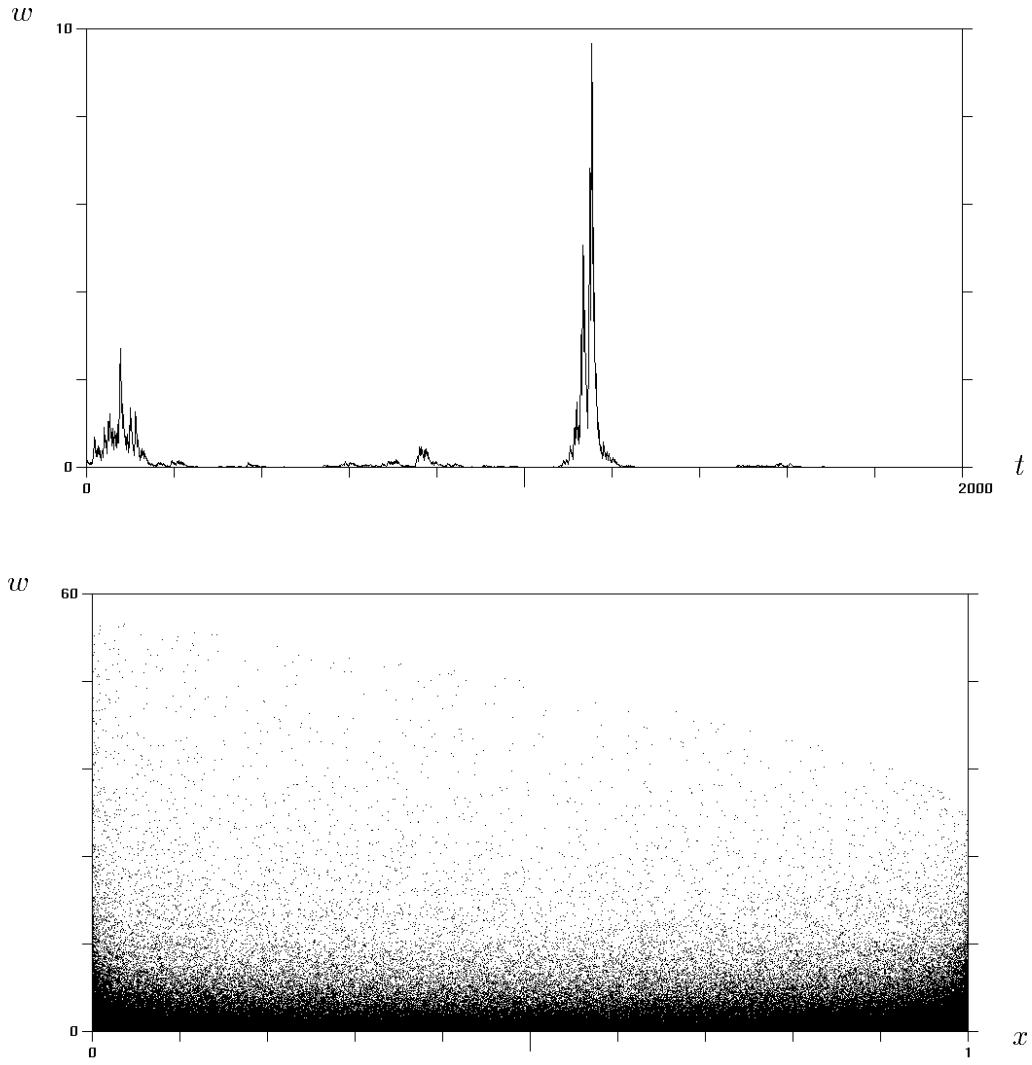


Abbildung 3.4: Ein einfacher Flare-Attraktor basierend auf der logistischen Map (siehe Gl. 3.3). **Oben**: Zeitserie der Flaring-Variablen w, wobei aufeinanderfolgende Iterationspunkte graphisch verbunden sind; 2000 Iterationen. **Unten**: Phasenraum-Darstellung; 1 000 000 Iterationen. Parameterwerte: $\mathit{Schwellenwert} = 0.7, \epsilon = 0.01$. Startbedingungen: $x_0 = 0, w_0 = 1$.

effektive Attraktor-Dimension nicht von der eines sogenannten hyperchaotischen Attraktors (mit einem zweiten positiven Lyapunov-charakteristischen Exponenten) unterscheidet [24]. Kaplan-Yorke-Attraktoren besitzen im Allgemeinen einen Schnitt durch den Attraktor, der die Gestalt einer stetigen, aber nirgendwo differenzierbaren Funktion aufweist; allerdings sind beide Eigenschaften (die Stetigkeit und die Nicht-Differenzierbarkeit) auf eine zugrundeliegende Cantor-Menge eingeschränkt. Das heißt, Kaplan-Yorke-Attraktoren gehören zur Klasse der sogenannten singulär-stetigen, nirgendwo-differenzierbaren Attraktoren [25][26][27]. Diese Eigenschaften erstrecken sich im invertierbaren Fall auch auf die Attraktoren vom Flare-Typ. Um zu zeigen, wie solch ein Querschnitt im invertierbaren Fall aussieht, sei die folgende Abbildung 3.5 aufgeführt. Sie zeigt einen Querschnitt durch einen Flare-Attraktor in einem differenzierbaren dynamischen System (einer invertierbaren Map aus drei Variablen).

Man erhält Attraktoren vom Flare-Typ auch, wenn anstelle eines chaotischen Inputs, wie oben in Gleichung 3.3, ein hyperchaotischer Input verwendet wird, also zwei oder mehr verschiedene chaotische Antriebe vorliegen. Auch jede andere stochastische Zeitvariable mit zeitlich konstantem Mittelwert ist im Prinzip geeignet, wenn das obige Schwellenprinzip mit einem geeignet angepassten Schwellenwert verwendet wird.

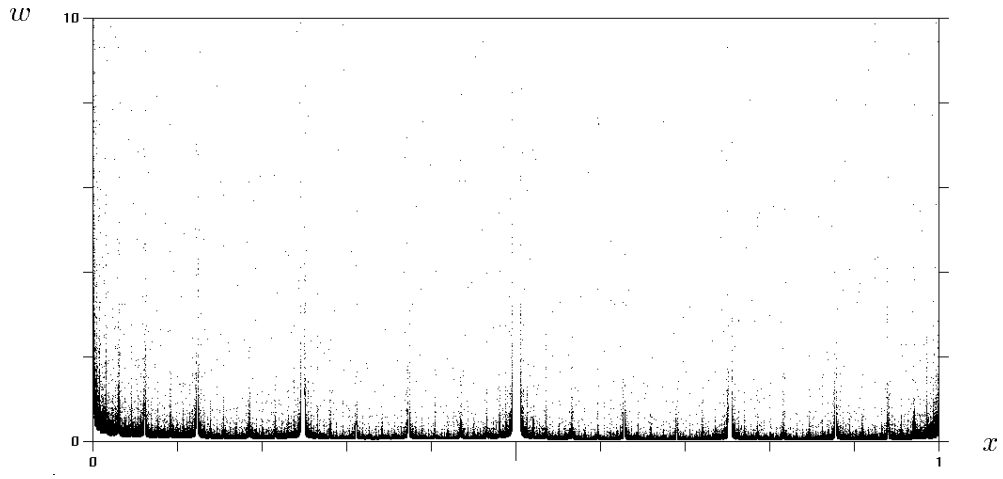


Abbildung 3.5: „Löwentatzen-Fraktal": Querschnitt eines differenzierbaren Flare-Attraktors. Nach Gl. 7.7 in Kapitel 7.3.3, Abszisse: forcende Variable x, Ordinate: Flaring-Variable w; aus [18].

3.4 Vergleich verschiedener Forcings für Flare-Systeme

Im Folgenden werden als Forcings für ein potentielles System mit Flare-Verhalten die bisher bekannten qualitativ verschiedenen Typen von zeitlichem Verhalten[‡] eingesetzt:

a) periodisch

b) quasiperiodisch

c) chaotisch-determinitisch

d) hyperchaotisch-determinitisch

e) stochastisch (*nicht* deterministisch).

Die typischen flare-haften Zeitserien finden wir nur bei den letzten drei Typen von Forcings, wobei zwischen diesen im Resultat nicht unterschieden werden kann. Bei periodischem bzw. quasi-periodischem Forcing treten dagegen nie Flare-Zeitserien auf.

Nachfolgend einige typische Beispiele für Zeitserien bzw. Phasenraumdiagramme der jeweiligen Typen.

Zunächst der wichtigste Fall: Chaotisch-deterministische Forcings (Fall c) stellen die Standardmethode dieser Arbeit zur Erzeugung der Flare-Dynamiken dar. Eine beispielhafte Zeitreihe nebst zugehöriger Phasenraumdarstellung zeigt Abb. 3.4.

Echtes stochastisches Verhalten (Fall e) wäre zum Beispiel eine abgespeicherte Zerfallstätigkeit eines radioaktiven Präparats. Zeitreihen mit gleichem probabilistischem Verhalten lassen sich im Rechner nur simulieren (sogenannte Zufallsgeneratoren, z.B. die Funktion `rnd(x)` in der Programmiersprache

[‡]Mit der kleinen Ausnahme des Zeitverhaltens eines sog. „strange-nonchaotic attractor", welches noch zu prüfen ist.

C). Derartige Funktionen stellen in der Realität chaotische bzw. hyperchaotische Iterationsabbildungen dar; vergl. auch [7].

Hyperchaotische Forcings (Fall d) ergeben ebenfalls die charakteristischen Flare-Zeitreihen. Ein solches Beispiel (Flare-Dynamik mittels eines hyperchaotischen Forcings) ist das System von Abb. 8.1d in Kapitel 8; diese Phasenraumdarstellung zeigt die typische Morphologie der Flare-Attraktorenklasse, d.h. den uns schon bekannten „Fingerprint" einer nach außen immer dünner werdenden Punktwolke.

In den Fällen a) und b) schließlich lassen sich keine Flare-Attraktoren der beschriebenen Art erzeugen. Typische Zeitserien, wenn man solche Systeme als Forcings für ein potentielles Flare-System einsetzt, zeigt die Abbildung 3.6 für den Fall eines quasi-periodischen Forcings, wobei das folgende System verwendet wurde:

$$\begin{aligned} x_{n+1} &= (x_n + \frac{(\sqrt{5}-1)}{2}) \bmod 1 \\ w_{n+1} &= w_n + w_n(\frac{2}{5} - x) + 10^{-2} - 10^{-4}w^2 \end{aligned} \tag{3.4}$$

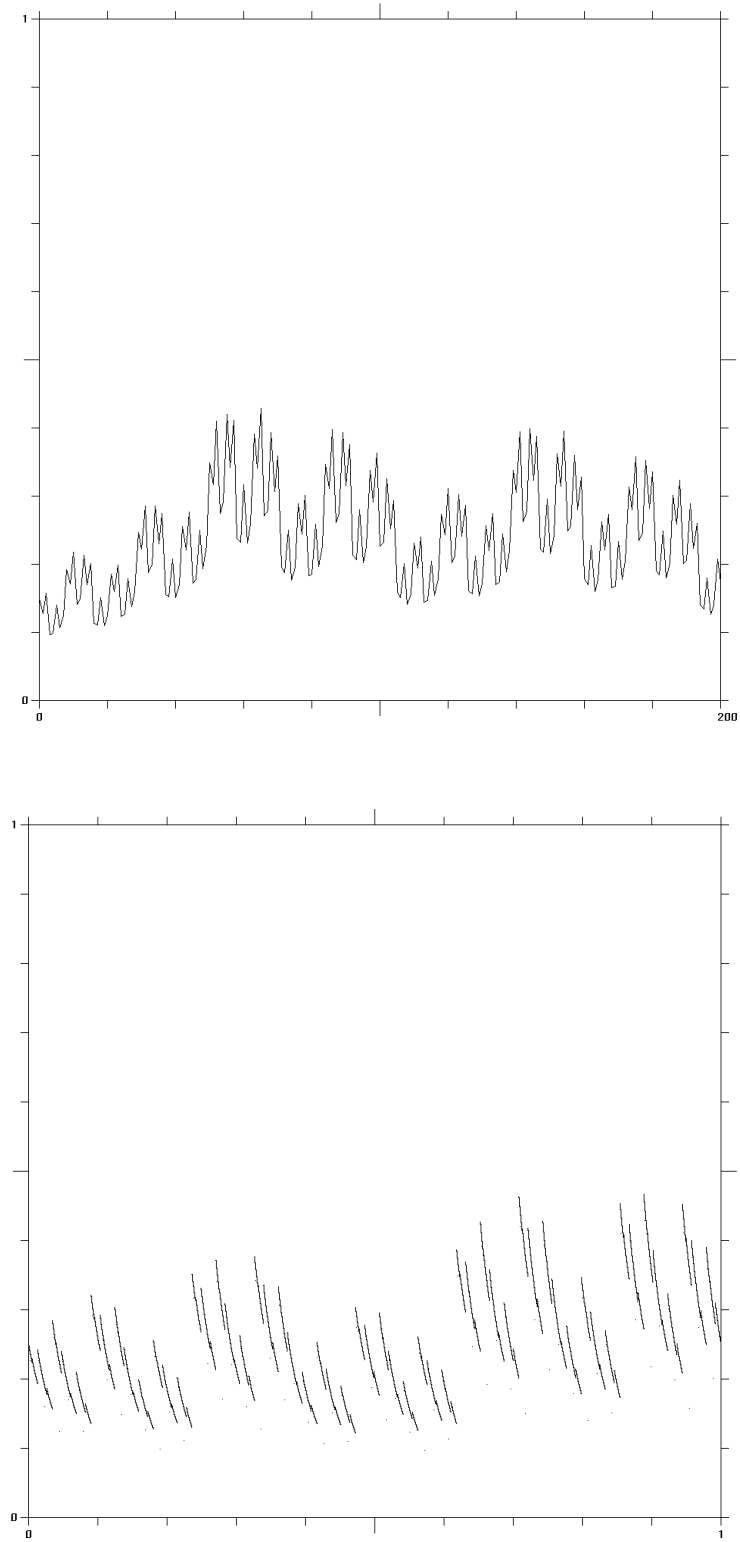


Abbildung 3.6: Quasiperiodisches Forcing eines potentiellen Flare-Systems gemäß Gl. 3.4: **Oben** Zeitserie (Abszisse t, Ordinate w), 200 Iterationen. **Unten** die dazugehörige Phasenraumdarstellung (x,w) 1 000 000 Iterationen.

Teil II

Modellrechnungen kontinuierlicher Systeme durch numerische Simulation

Kapitel 4

Numerisches Lösen von Differentialgleichungssytemen

4.1 Analytisches vs. numerisches Lösen von Differentialgleichungssytemen

Die im Rahmen der Arbeit simulierten Flüsse bzw. Reaktionskinetik-Systeme stellen mathematisch nichtlineare vier- oder höherdimensionale Differentialgleichungssysteme dar. Die für einfache lineare Differentialgleichungssysteme bestehende Möglichkeit einer exakten, sogenannten *analytischen* Lösung mit den bekannten Lösungsverfahren wie Variablenseparation o. ä. besteht in der Regel nicht. Es müssen deshalb schrittweise rechnende numerische Näherungsverfahren eingesetzt werden — im englischen Terminus technicus auch *ODE-Solver** genannt. Hierfür existieren bereits fertige Implementationen bzw. Bibliotheken für die verschiedenen Programmiersprachen. Besonders bequem ist die Anwendung solcher Routinen, wenn diese in den modernen, symbolischen Mathematikpaketen wie MATHEMATICA, MAPLE, MATLAB u.ä. implementiert sind, da diese keine Erstellung von jeweils einzelnen, ausführbaren Binärdateien erfordern und somit eine schnellere und

*ODE steht für engl. **O**rdinary **D**ifferential **E**quation.

kompaktere Arbeitsweise gestatten. Ihr Nachteil besteht leider darin, dass je nach Programm nur wenige Parameter des DGL-Solvers justiert werden könnnen, was bei schwierigem Lösungsverhalten (z.B. „steife" Probleme, s.u.) evtl. nicht ausreicht. Außerdem erfordern solche Programme natürlich für eine bestimmte Aufgabe in der Regel sehr viel mehr Rechnerresourcen als ein selbst erstelltes und kompiliertes Programm in einer modernen Computersprache.

Vor der Verfügbarkeit leistungsfähiger Digitalrechner stellten die **Analogrechner** das Mittel der Wahl zur Simulation von chemischen und biologischen Dynamischen Systemen dar [28]. Der hohe apparative Aufwand rechtfertigt heute ihren Einsatz für normale Untersuchungen Dynamischer Systeme nicht mehr. Dennoch stellen sie prinzipiell eine realistischere Abbildung eines beispielsweise chemischen dynamischen Systems dar, da die Binärdarstellung im digitalen Computer letztlich immer nur eine — zudem oft unanschaulichere — Approximation an ein reales, analoges† System darstellen kann. Was die Analogrechner an Genauigkeit verlieren, gewinnen sie an qualitativer Ähnlichkeit des Verhaltens, da bei ihnen ähnlich den dynamischen Prozessen in der Natur (bzw. Chemie) zusätzlich „Nebenreaktionen" (mit Streukapazitäten und Kriechströmen beispielsweise) auftreten können.

4.2 Analogrechner und SIMULINK

Nachdem die Anwendung von Analogrechnern zur Simulation von Dynamischen Systemen im Laufe der Zeit auf Grund der zu geringen Rechenleistung und umständlichen Handhabung nahezu vollständig durch Digitalrechner abgelöst worden ist, erlebt die bessere Anschauung, die diese Systeme boten, für Dynamische Systeme via analogrechner-ähnlicher Schaltbilder eine Renaissance seit der Entwicklung der MATLAB-Toolbox **SIMULINK** [29]. Diese erlaubt es, Dynamische Systeme wie beim Analogrechner mittels Schaltel-

†Von den Grenzen der Kontinuumsphysik und -chemie im Partikel- und Quantenbereich sei hier abgesehen.

elementen zusammenzusetzen, was bei bestimmten Problemen (insbesondere in den Ingenieurwissenschaften) [30] von großem Vorteil sein kann. Die Zu- bzw. Abflüsse durch Drahtverbindungen des Analogrechners haben hier ihre unmittelbaren visuellen Pendants. Desweiteren kann jedes System wieder als ein einzelnes Subsegment eines größeren Systems dienen, was vor allem den Überblick beim Zusammenstellen größerer, komplexerer Systeme erleichtert.

Die letztendliche Simulation des Differentialgleichungssystem erfolgt dann nach den oben vorgestellten Methoden auf dem Digitalrechner, auf dem das MATLAB/SIMULINK-System installiert ist. Es kann die jeweils günstigste Methode zur Integration ausgewählt werden und es können sowohl Trajektorien als auch Phasenraumdiagramme ausgeplottet werden.

Als ein einfaches Beispiel soll hier das bekannte *Van der Pol* System in ein SIMULINK-Schaltbild umgesetzt werden:

Das Van der Pol System wird durch die folgenden zwei Differentialgleichungen erster Ordnung beschrieben:

$$\begin{aligned} \dot{x_1} &= x_1(1 - x_2^2) - x_2 \\ \dot{x_2} &= x_1 \end{aligned} \tag{4.1}$$

Abbildung 4.1 zeigt das zugehörige SIMULINK-Modell:

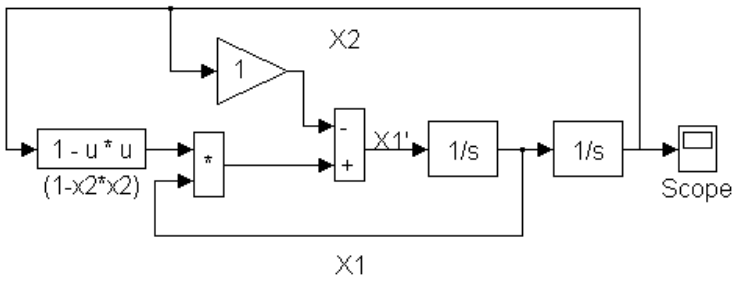


Abbildung 4.1: Beispiel eines Simulink-Schaltbilds — das van-der-Pol System (vgl. Text).

Kapitel 5

Numerische Modellsimulationen von kontinuierlichen Flare-Systemen

5.1 Geeignete Integrationsmethoden

Differentialgleichungssysteme mit Flare-Verhalten gehören zu den Systemen mit sogenanntem *steifem* Verhalten: Steifheit kann bei Differentialgleichungssystemen auftreten, wenn die abhängigen Variablen größenordnungmäßig sehr stark verschiedene Änderungen pro Zeitschritt erfahren.

Ein simples Beispiel für ein solches System wäre das folgende Zweivariablensystem:

$$\begin{aligned} \dot{u} &= 999u + 1998v \\ \dot{v} &= -999u - 1999v \end{aligned} \tag{5.1}$$

Steife Systeme sind mit einfacheren numerischen Verfahren wie Euler oder

Standard-Runge-Kutta-Verfahren nicht bzw. nur schlecht, d.h. mit minimalen Schrittweiten (und daher evtl. sehr großen Rechenzeiten), zu lösen. Die Crux steifer Differentialsgleichungssysteme liegt darin, dass man für stabiles Lösungsverhalten die Integrationsschrittweite minimal halten muss, auch wenn die gewünschte Ergebnisgenauigkeit eine um Größenordnungen geringere Rechenpräzision zuließe [31]. So versagen deshalb auch die einfacheren ODE-Solver von symbolischen Mathematik-Programmen wie MATHEMATICA oder MAPLE meist bei steifen Systemen; allerdings liegt der Fokus dieser Programme auch nicht bei der numerischen Lösung komplexerer Diffentialgleichungssysteme.

Die numerische Lösung gelingt etwa mit einem Runge-Kutta-Merson Algorithmus vierter Ordnung mit adaptiver Schrittweite; allerdings erst, wenn man hierbei die maximal mögliche Schrittweite genügend klein wählt. Am besten errechnet man die numerische Lösung solcher Systeme — auch was die Rechengeschwindigkeit anbetrifft — mit Prädiktor/Korrektor-Methoden wie bspw. den Adams und Gear-Methoden, welche speziell für steife Differentialgleichungsprobleme entwickelt wurden. Die SIMULINK-Toolbox von MATLAB bot (zum Zeitpunkt der Arbeit) als einzige der erhältlichen symbolischen Mathematik-Pakete eine solche Methode an; dies war der Hauptgrund für den Einsatz des MATLAB-Pakets bei der Untersuchung dieser Systeme.

Abbildung 5.1 zeigt das vollständige SIMULINK-Schaltbild eines Dynamischen Systems mit Flare-Verhalten.

5.2 Chemische Reaktionskinetiksysteme

Nichtlineare Dynamiken mit nichttrivialen (d.h. nicht nur einen einzigen, global stabilen Steady State besitzenden) dynamischem Verhalten — wie wir sie bei den Flare-Systemen dieser Arbeit stets vorliegen haben — können in der Chemie nur bei *offenen* Systemen auftreten; Selbstorganisation, d.h. in unserem Fall komplexe Attraktoren, können nur auftreten, wenn sich ein solches System *fern* von Gleichgewicht befindet. Chemische Reaktionssysteme

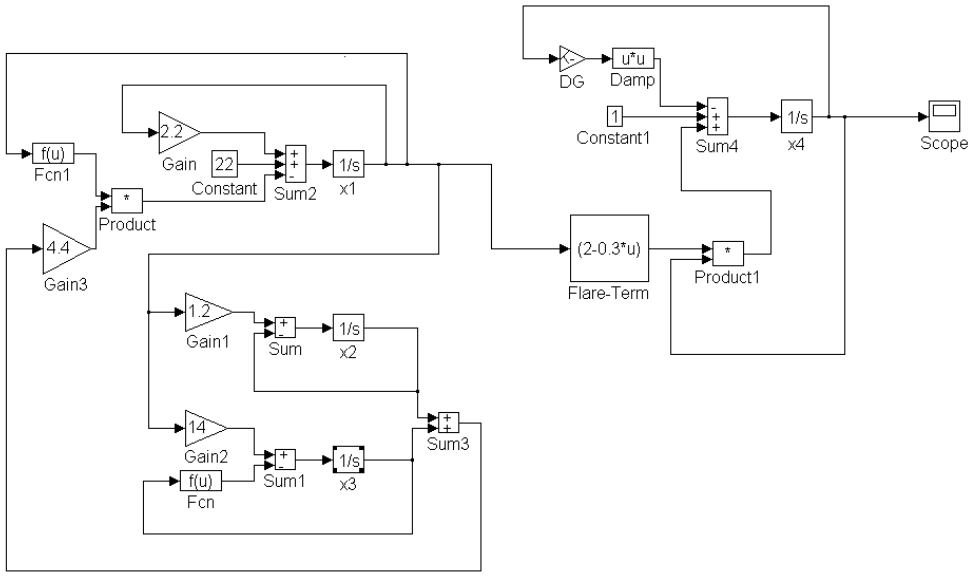


Abbildung 5.1: Simulink-Schaltbild zur Erzeugung einer Flare-Dynamik gemäß Gleichung 5.3

solcher Art sind also von einem prinzipiell anderen Typ als die Reaktionssysteme der gewohnten Synthese-Chemie, welche bei ihren stöchiometrischen und reaktionskinetischen Betrachtungen in der Regel von einem thermodynamischen Gleichgewichtszustand ausgeht. Dies ist bei der Untersuchung nichtlinearer dynamischer chemischer Systeme gerade nicht der Fall. Chemische Systeme der nichtlinearen Dynamik, wie sie hier betrachtet werden — also offen und fern vom Gleichgewicht —, ähneln dagegen viel eher den chemischen Prozessen in biologischen Zellen und Organismen, welche in thermodynamischer Sprache ebenfalls offene Systeme fern vom Gleichgewicht darstellen.

Zur Untersuchung/Realisation von nichtlineaeren, chemischen Systemen im Labor bedient man sich üblicherweise des *Continuous (Flow) Stirred Tank Reaktors* (CSTR), bei welchem unter ständigem Rühren Edukte und Produkte kontinuierlich zu- bzw. abfließen. Bei Annahme einer konstanten Edukt-Konzentration kann man auch unter extremen Konzentrationsüberschuss der Edukte arbeiten (sog. *chemical-pool* Näherung, z.B. im Brüsselator-System von Prigogine, vgl. dazu die Monographie von Scott[32]). Zur Betrachtung

der nichtlinearen Reaktionskinetik offener Systeme kann man die üblichen Rechenansätze der Reaktionskinetik durchaus weiterverwenden, man muss diese nur um nichtlineare Terme erweitern.*

Die sogenannten Raten- oder Geschwindigkeitsgleichungen (Differentialgleichungssysteme) besitzen eine Reihe von alternativen Lösungstypen, welche durchaus im CSTR auch beobachtet werden können: bistabile oder multistablie stationäre Zustände, periodisch gedämpfte, ungedämpfte und quasiperiodische Oszillationen, sowie deterministisches Chaos [33].

Chemische Reaktionssysteme aus zwei Variablen können Oszillationen erzeugen, falls eine Variable die andere zum wachsen anregt (d.h. katalysiert oder in sie übergeht) und die andere die erste Variable inhibiert (bzw. deren Abbaureaktion fördert). Für chaotisches zeitliches Verhalten genügen drei Variablen; das Chaos entsteht hierbei im einfachsten Fall als „Verschränkung" von zwei schwingenden Subsystemen (2 mal zwei verschiedene 2-Variablen Subsysteme). Das bekannteste abstrakte Reaktionsmodell mit periodischen Zuständen als Lösungen der Geschwindigkeitsgleichungen geht auf Prigonine† zurück und wird deshalb als *Brüsselator* bezeichnet.

Das einfachste chaos-erzeugende Reaktionssystem hat die folgenden Form:

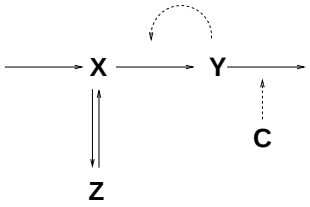


Abbildung 5.2: Das einfachste Reaktionssystem mit deterministischem Chaos; $\dashrightarrow$ = Katalyse; wobei **C** keine Variable, sondern in der hier angenommenen Michaelis-Menten-Approximation eine Konstante ist[34].

*Ein anderer Ansatz wäre es, die nichtlineare Dynamik als Teil der irreversiblen Thermodynamik aufzufassen [33][2].

†NP für Chemie im Jahr 1977

5.3 Reaktionskinetik-Systeme mit Flare-Dynamik

Hat man nun ein Subsystem mit sich chaotisch verhaltenden Konzentrationen zur Verfügung, so kann man zwecks Erzeugung einer Dynamik mit zeitlich *flarehaftem* Verhalten eine weitere (vierte) Reaktionsvariable mit autokatalytischer Potenz multiplikativ ankoppeln. Ein solches 4-Variablen-System ist in Abb. 5.3 dargestellt (aus [16]). Das antreibende Chaos[‡] ist unkritisch, solange der Mittelwert konstant ist (im Fall der Abb. 5.3 stellt das System x, y, z ein sogenanntes *Spiralchaos* zur Verfügung [35]); entscheidend ist die Kinetik der autokatalytischen vierten Variablen (w). Das Spiralchaos wurde als Forcing ausgewählt, weil es ein recht stark „durchmoduliertes" Chaos aufweist (hohes Verhältnis zwischen größter und kleinster Amplitude); die Parameterwerte können hierbei modifiziert werden, solange man noch eine chaotische Zeitserie für x erhält. Die Parameterwerte der Gleichung der Flaring-Variablen w werden dann so gewählt, dass man maximale Amplitude bei den Flaring-Peaks erhält (was, wie man der Konstante 0.33 mit ihren nur wenigen Stellen in der vierten Zeile in Gl. 5.3 ansieht, recht unkritisch ist).

Das typisch flarehafte Konzentrationsverhalten der Variablen w zeigt Abb. 5.4.

Das zugehörige Differentialgleichungssystem lautet:

$$\begin{aligned} \dot{x} &= 22 + 2.2x - 4.4\frac{(y+z)x}{x+0.01} \\ \dot{y} &= 1.2x - y \\ \dot{z} &= 14x - \frac{140z}{z+0.05} \\ \dot{w} &= 1 + w(2 - 0.33x) - 10^{-5}w^2. \end{aligned} \tag{5.2}$$

Eine vollständige Phasenraumdarstellung des Systems ist — da vier Dimensionen vorhanden sind — nicht möglich; in der dreidimensionalen Ab-

[‡]Wie schon mehrfach erwähnt, würde auch eine kontinuierliche, nicht-chaotische Stochastizität genügen.

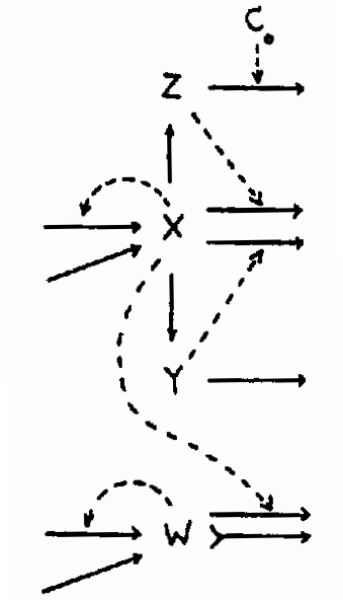


Abbildung 5.3: Reaktionskinetisches Modell nach Gl. 5.3 mit Flare-Verhalten bei einer Reaktionsvariablen (w); $\longrightarrow$ = Aktivierung, $\dashrightarrow$ = Hemmung.

bildung 5.5 des Phasenraums wurde deshalb eine Variable nicht berücksichtigt — die Dynamik eines flarehaften Flusses ist trotzdem gut erkennbar: Meist hält sich das System nahe bei einem Quasi-Steadystate in der Nähe von Null auf. Nur ganz selten verläßt die Systemtrajektorie diesen Bereich, um kurzzeitig zu quasisteadystatefernen Systemzuständen zu gelangen. Ein ganz ähnliches dynamisches Prinzip also, wie es auch bei den dreidimensionalen Iterationsabbildungen mit Flare-Verhalten im Kapitel 6 sichtbar werden wird.

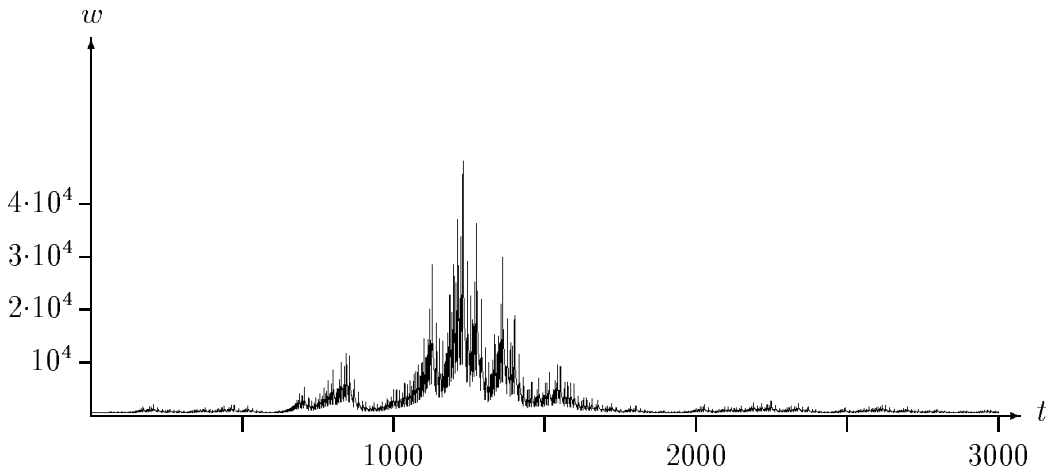


Abbildung 5.4: Numerische Integration mittels Adams-Gear-Methode eines Flare-Systems gemäß dem Reaktionsschema der Abb. 5.3: Zeitliches Verhalten ($t : 0 \rightarrow 3000$) der Konzentration von w.

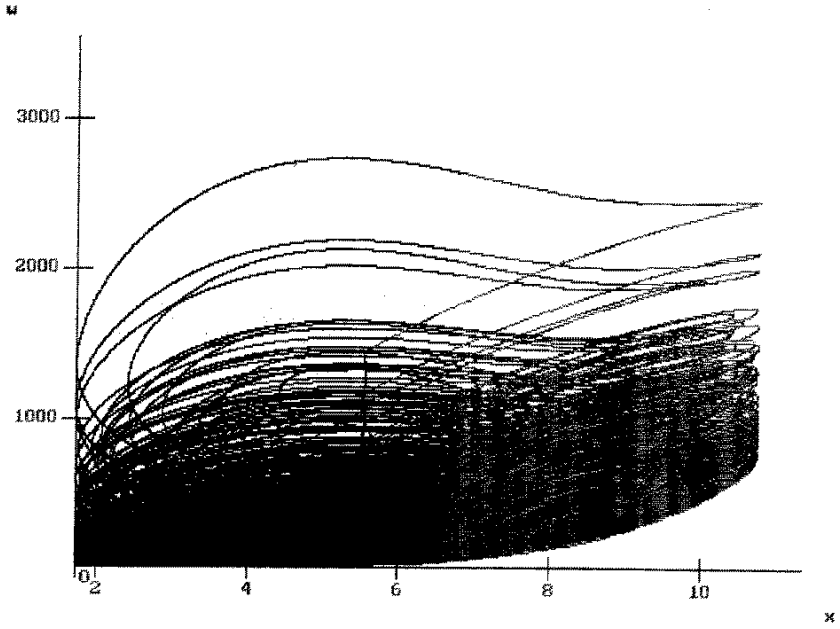


Abbildung 5.5: Dreidimensionale Phasenraumdarstellung des reaktionskinetischen Systems gemäß Gleichung 5.3 (die oberste Trajektorie läuft nach rechts).

Teil III

Modellrechnungen diskretisierter Systeme durch numerische Simulation

Kapitel 6

Nichtinvertierbare Maps

6.1 Einfache Maps

Iterationsabbildungen lassen sich ganz besonders schnell und effektiv mittels eines Computers untersuchen.* Deshalb liegt hier der Schwerpunkt der im Rahmen der Arbeit untersuchten Dynamischen Systeme.

Im Gegensatz zu den numerischen Simulationen von kontinuierlichen Systemen im vorigen Kapitel reicht bei diskreten Iterationssystemen zur Erzeugung einer unregelmäßigen (chaotischen) Zeitserie bereits ein **1**-Variablen-System aus, was das Simulieren noch zusätzlich vereinfacht und schneller macht. Es ergeben sich demgemäß für die einfachsten Flare-Attraktor-Systeme Iterationssysteme mit lediglich 2 Variablen.

Es wurden nun zahlreiche chaoserzeugende Abbildungen daraufhin untersucht, ob sie als Forcing für ein Flare-System dienen können. Es zeigte sich, daß dies für einen gewissen Parametersatz in der Gleichung der Flare-Variablen (in den folgenden Systemen stets mit b_n bezeichnet) immer der Fall ist; daraus läßt sich mit genügender Sicherheit schließen, dass wohl jede Chaos-generierende Iterationsabbildung ausreicht um bei geeignetem Para-

*Zur Relevanz von diskreten Abbildungen für chemische/physikalische Flüße siehe die Ausführungen im Abschnitt 1.2.

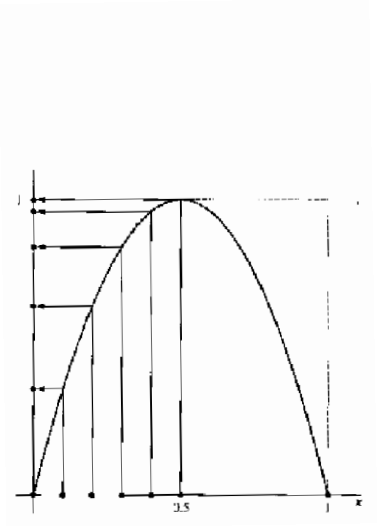


Abbildung 6.1: Abbildungsprinzip der logistischen Map

metersatz das Flare-Verhalten auslösen zu können.

In den folgenden drei Abschnitten werden jeweils Simulationen für die drei bekanntesten und zugleich einfachsten diskreten Iterationsabbildungen vorgestellt: die logistische Abbildung, die Zelt-Abbildung („Tent-Map") und die Sägezahn-Abbildung („Bernoulli-Shift").

6.1.1 Logistische Map

Ein Flare-System mit Forcing durch die logistische Map wurde bereits in Abschnitt 3.3 als ein erstes Beispiel für einen Flare-Attraktor präsentiert (Gl. 3.3, Abb. 3.4). Das Abbildungsprinzip dieser chaotische Zeitserien erzeugenden Iterations-Map zeigt Abbildung 6.1. Man sieht, dass die x-Achse zwischen Null und Eins bei jedem Iterationsschritt im Durchschnitt auf die doppelte Länge (plus Faltung) abgebildet wird. Vgl. [12].

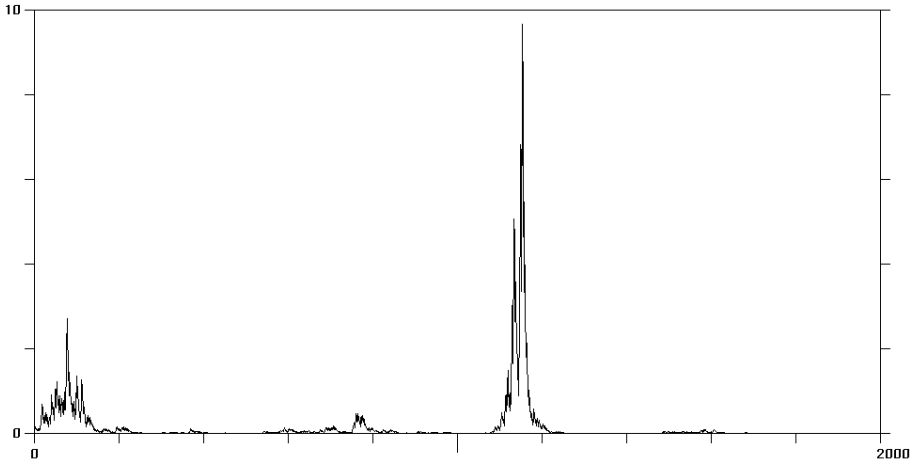


Abbildung 6.2: Typische Flare-Zeitserie eines durch die logistische Map geforcten Systems (Ausschnitt aus Abb. 3.4).

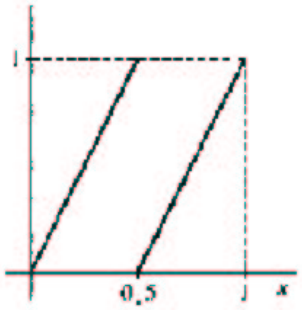


Abbildung 6.3: Abbildungsprinzip des Bernoulli-Shift

6.1.2 Bernoulli-Shift

Das in Abb. 6.3 gezeigte Abbildungsschema dieser Map entspricht einer Sägezahnkurve. Auch diese Abbildung bildet jedes Halbintervall auf die doppelte Länge ab und erzeugt so eine chaotische Zeitserie.

Ein typisches Flare-System auf dieser Basis lautet dann wie folgt:

$$\begin{aligned} x_{n+1} &= 2x_n \bmod 1 \\ w_{n+1} &= w_n + w_n(\frac{1}{3} - x_n) + 0.01 \end{aligned} \tag{6.1}$$

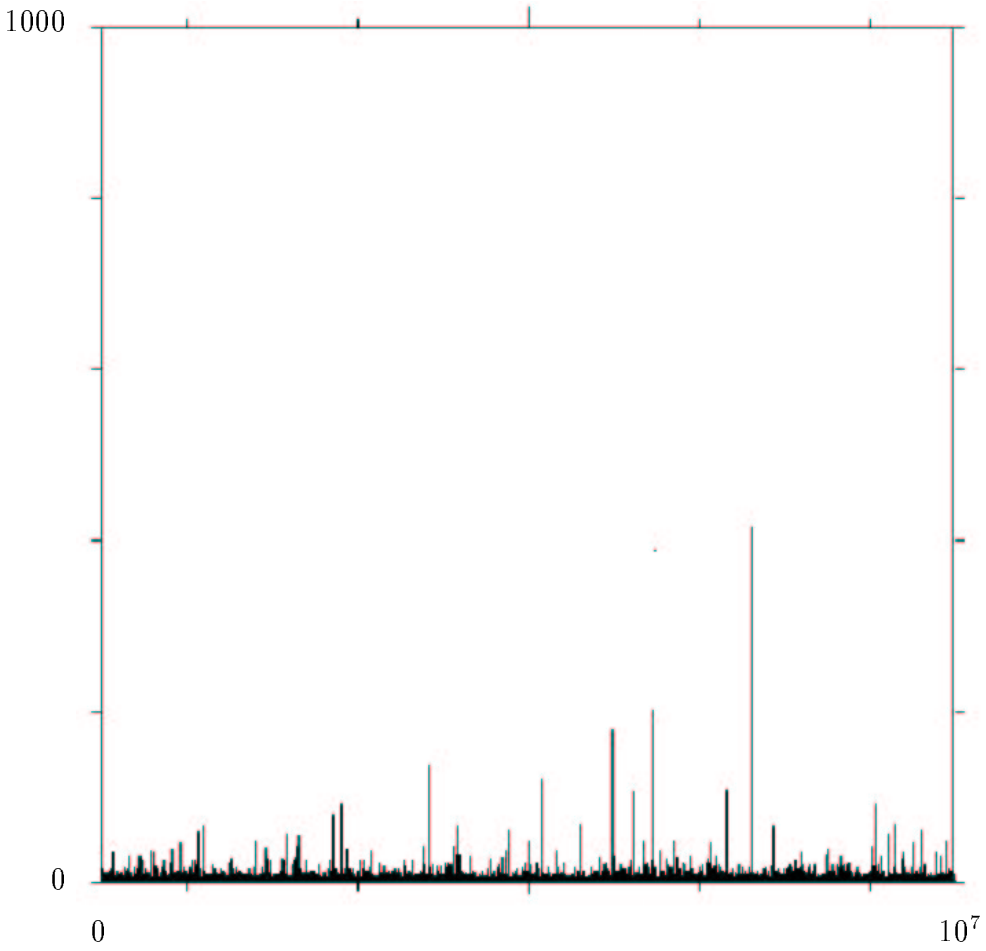


Abbildung 6.4: Zeitserie durch Forcing des Systems mittels des Bernoulli-Shift gemäß Gl. 6.1: Verhalten der 2. Variablen als Funktion der Zeit.

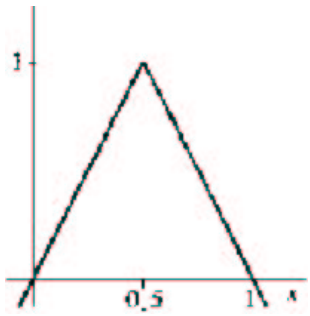


Abbildung 6.5: Abbildungsprinzip der Tent-Map

6.1.3 Tent-Map

Diese Art von Map kann man unmittelbar aus der Sägezahnabbildung ableiten; die erste Hälfte der Iterationsvorschrift (für Werte von $x < 0.5$) ist identisch; für Werte $x > 0.5$ besitzt dagegen die Steigung der Iterationsgeraden bei gleichem Betrag ein negatives Vorzeichen. Abbildung 6.5 zeigt das Prinzip. Die System-Gleichungen des vollständigen Systems lauten dann wie folgt:

$$\begin{aligned} x_{n+1} &= 2\left|x - \frac{1}{2}\right| + 1 \\ w_{n+1} &= w_n + w_n(0.4 - x_n) + 0.01 - 0.001 w_n^2. \end{aligned} \tag{6.2}$$

Das Verhalten dieses Systems ist in Abb. 6.2 dargestellt.

6.2 Ein komplexeres System

Es wurde weiterhin das folgende Iterations-System untersucht:

$$\begin{aligned} x_{n+1} &= x_n e^{1.63(1-x_n^2)} + w_n \\ w_{n+1} &= w_n(1 + 2.2(0.4 - x_n)). \end{aligned} \tag{6.3}$$

Die erste Zeile der Gl. 6.3 ohne den w_n-Term ist eine chaoserzeugende 1-dimensionale Abbildung, die zur logistischen Gleichung nahe verwandt

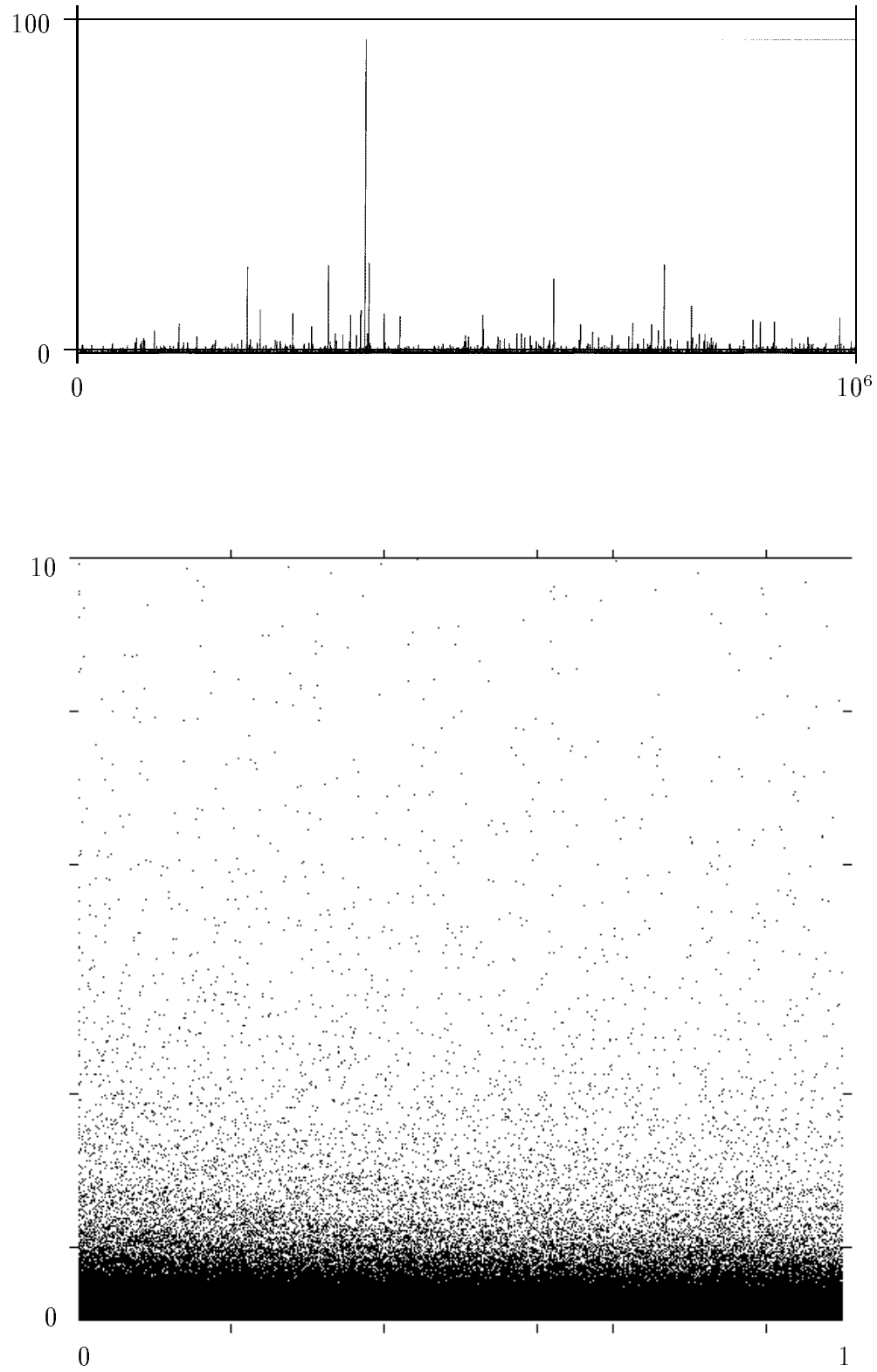


Abbildung 6.6: Oben: Zeitserie der Flaring-Variablen w durch Forcing des Systems mittels der Tent-Map gemäß Gl. 6.2, 1 000 000 Iterationsschritte. Unten: Dazugehörige Phasenraumdarstellung (x, w)

ist, aber weniger anfällig gegen das Explodieren der Werte ist, wenn Rückkopplungsterme (wie hier der w_n-Term in der ersten Zeile) eingeführt werden. Die zweite Variable wird wie üblich durch die erste chaotisch angetrieben. Es wurde ein maximal grosser Rückkopplungsterm ($1 \cdot w_n$) eingeführt. Merkwürdigerweise verhält sich die vorliegende Gleichung trotz des Rückkopplungsterms praktisch genauso, wie wenn dieser nicht vorhanden wäre (eine eng verwandte Gleichung ohne den Rückkopplungsterm wurde in [16] beschrieben).

Das Beispiel der Gleichung 6.3 könnte noch weiter optimiert werden, da der äußerste Bereich des Flare-Attraktors (unten) nicht genügend stark verdünnt ist. Diese Gleichung wird uns übrigens noch einmal weiter unten als Baustein für einen ersten invertierbaren Flare-Attraktor in einer Map wiederbegegnen (siehe Kapitel 7, Abb. 7.1, Gl. 7.2).

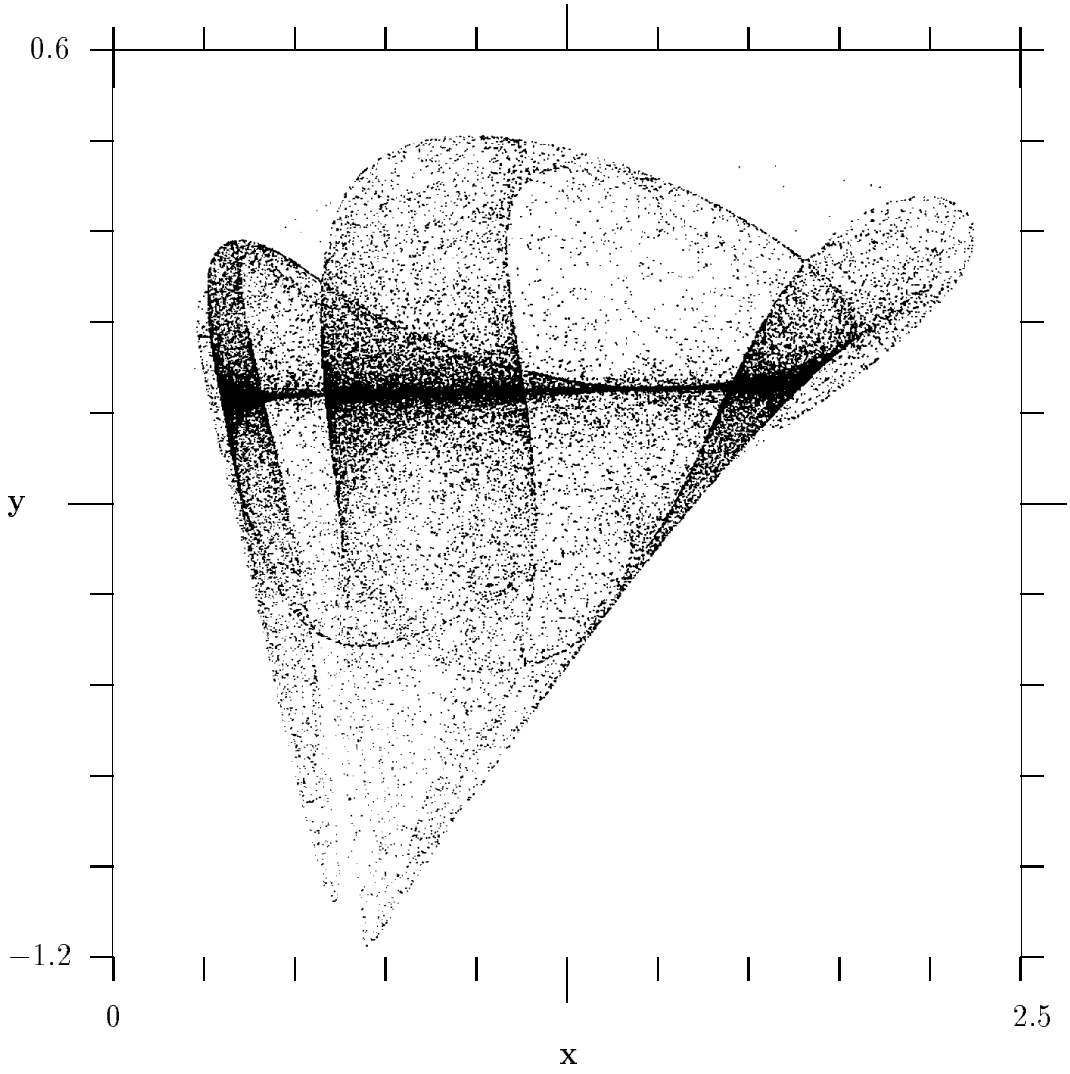


Abbildung 6.7: Phasenraumdarstellung des Flare-Attraktors nach Gl. 6.3, 1 000 000 Iterationen.

Kapitel 7

Invertierbare Abbildungen

7.1 Einleitung

Wie oben in Abschnitt 1.2 bereits ausgeführt, müssen Differenzengleichungssysteme, die als Poincaré-Schnitt einer Kontinuums-Dynamik interpretierbar sein sollen, die Bedingung der Invertierbarkeit erfüllen. Deshalb bildet die Simulation von **invertierbaren** Dynamischen Systemen einen eigenen Schwerpunkt bei der Untersuchung des Verhaltens von Iterations-Abbildungen mit Flare-Dynamik.

7.2 Ein erstes invertierbares Modell

Die folgende Drei-Variablen-Abbildung ist das erste im Rahmen der invertierbaren Systeme aufgefundene Modell:

$$\begin{aligned} x_{n+1} &= x_n e^{1.57(1-x_n^2)} + y_n \\ y_{n+1} &= y_n(1 + 2.2(0.4 - x_n)) + 0.04 w_n \\ w_{n+1} &= x_n \end{aligned} \tag{7.1}$$

Das System ist eng mit der oben zuletzt besprochenen 2-Variablen-Gleichung, Gl. 6.3, verwandt. Die hinzugekommene dritte Variable (hier w genannt) wurde mit einem eigenen kleinen Kopplungsterm (der sich dem Wert Null nähern kann) an die zweite Variable (hier y genannt) angekoppelt; siehe dazu den letzten Term in der zweiten Zeile.

Die hinzugekommene Variable macht die gesamte Gleichung invertierbar, weil die folgenden Bedingungen erfüllt sind:

1. Die Jakobi-Determinante der Gleichung 7.2 ist überall von Null verschieden.

2. Keiner der möglichen Ausnahmefälle, die die Invertierbarkeit trotzdem verhindern könnten, liegt vor (es liegt weder ein Sinus-Term noch ein Modulo-Term vor), vergl. [36].

Die Jakobi-Determinante der Jakobi-Dreiermatrix (7.2)

$$\mathbf{J} = \begin{pmatrix} \frac{\partial f(x)}{\partial x} & \frac{\partial f(x)}{\partial y} & \frac{\partial f(x)}{\partial w} \\ \frac{\partial f(y)}{\partial x} & \frac{\partial f(y)}{\partial y} & \frac{\partial f(y)}{\partial w} \\ \frac{\partial f(w)}{\partial x} & \frac{\partial f(w)}{\partial y} & \frac{\partial f(w)}{\partial w} \end{pmatrix} \tag{7.2}$$

läßt sich nach der *Regel von Sarrus** berechnen:

$$\begin{aligned} \det(\mathbf{J}) \;=\; & \frac{\partial f(x)}{\partial x} \cdot \frac{\partial f(y)}{\partial y} \cdot \frac{\partial f(w)}{\partial w} + \frac{\partial f(x)}{\partial y} \cdot \frac{\partial f(y)}{\partial w} \cdot \frac{\partial f(w)}{\partial x} + \frac{\partial f(x)}{\partial w} \cdot \frac{\partial f(y)}{\partial x} \cdot \frac{\partial f(w)}{\partial y} \\ & - \frac{\partial f(w)}{\partial x} \cdot \frac{\partial f(y)}{\partial y} \cdot \frac{\partial f(x)}{\partial w} - \frac{\partial f(w)}{\partial y} \cdot \frac{\partial f(y)}{\partial w} \cdot \frac{\partial f(x)}{\partial x} - \frac{\partial f(w)}{\partial w} \cdot \frac{\partial f(y)}{\partial x} \cdot \frac{\partial f(x)}{\partial y} \\ & = \frac{\partial f(x)}{\partial x} \cdot \frac{\partial f(y)}{\partial y} \cdot 0 + 1 \cdot 0.04 \cdot 1 + 0 \cdot \frac{\partial f(y)}{\partial x} \cdot 0 \\ & -1 \cdot \frac{\partial f(y)}{\partial y} \cdot 0 - 0 \cdot 0 \cdot \frac{\partial f(x)}{\partial x} - 0 \cdot \frac{\partial f(y)}{\partial x} \cdot \frac{\partial f(x)}{\partial w} t \\ & = 0 + 0.04 + 0 - 0 - 0 - 0 = \mathbf{0.04} \end{aligned} \tag{7.3}$$

*gültig nur für Determinanten dritter Ordnung

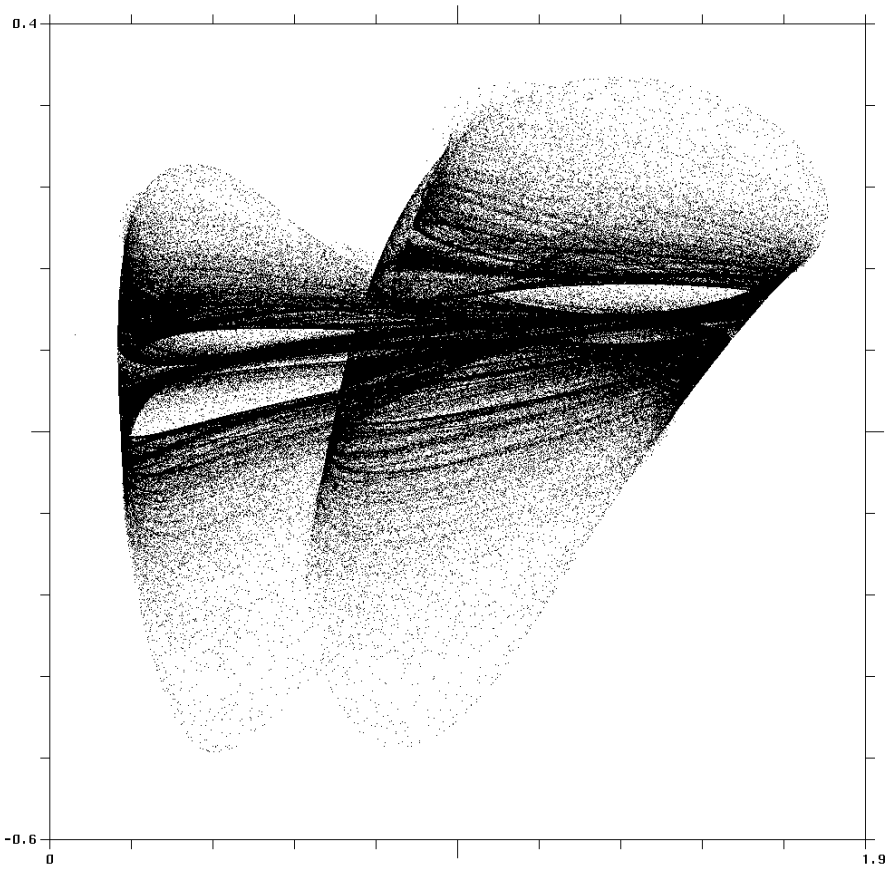


Abbildung 7.1: Flare-Attraktor, erzeugt durch die invertierbare Iterations-Abbildung von Gl. 7.2. Projektion auf die x-y-Ebene. Die Abszisse ist die x-Achse, die Ordinate die y-Achse. 2 000 000 Iterationspunkte sind dargestellt. Anfangsbedingungen: 0.01 für alle drei Variablen.

Der Wert der Jakobi-Determinante ergibt sich hier also zu 0.04. Das bedeutet, wenn man Gleichung 7.2 als die mathematische Beschreibung eines Knetvorgangs interpretiert, dass man das Volumen des gekneteten Teigs bei jedem Schritt um dem Faktor 25 (= 1/0.04) schrumpfen läßt — z.B. durch das Abtropfen von ausgepresster Flüssigkeit.

Abbildung 7.1 zeigt, daß die hinzugefügte dritte Variable das Flare-Phänomen intakt lässt. Wir können daraus folgern, daß flarehaftes Verhalten auch in kontinuierlichen Systemen (z.B. chemischen Modell-Systemen) prinzipiell existent sein kann, denn invertierbare Abbildungen können, wie aus dem Suspensionstheorem von Smale [37] (bzw. dessen Umkehr) folgt, im Prinzip immer auch von kontinuierlichen Systemen als Poincaré-Querschnitt realisiert werden. Siehe dazu auch die Ausführungen im Abschnitt 1.4 (und das Beispielsystem der Abbildungen 3.3 bis 3.5).

Abbildung 7.2 zeigt noch eine Vertikal-Projektion (von oben) auf den Attraktor der Abbildung 7.1. Man sieht einen „banana-peel" Attraktor. Einen Querschnitt durch den „Hals" des Attraktors (an der Stelle der unteren Biegung bei $w = 0,48$) wird Abb. 7.3 gezeigt. Eine Teil-Vergrößerung findet sich schließlich in Abb. 7.4; man sieht eine „Cantor-Streifen"-Struktur. Die zunehmende „Ausdünnung" ist ein charakteristisches Zeichen für das Flare-Verhalten. Außerdem sieht man, dass die verschiedenen Schichten stark unterschiedlich gefüllt sind und in ihrem Oberteil verschieden lang sind.

7.3 Weiterere invertierbare Modellsysteme

7.3.1 Baker's Map

Die Baker's Map stellt eine Abwandlung des Bernouilli-Shifts (der Sägezahn-Abb. 6.3) dar. Die hinzugefügte zweite Variable sorgt für die Invertierbarkeit des vollständigen, nunmehr 2-dimensionalen Systems.

Die *Baker's Map* wurde von Eberhard Hopf im Jahre 1937 in die Theo-

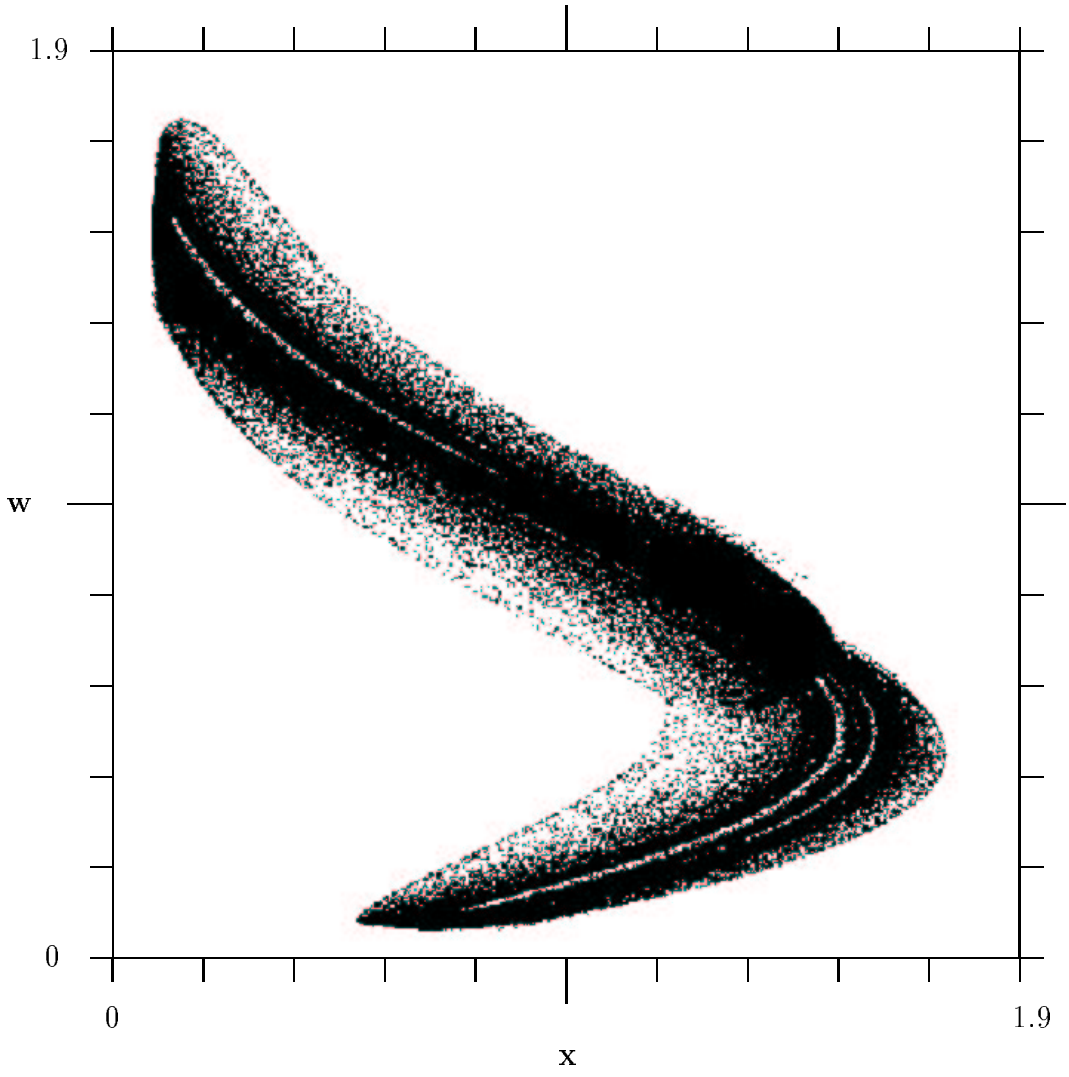


Abbildung 7.2: Eine andere Projektion des Attraktors von Abbildung 7.1 (Projektion auf die x-w-Ebene). Die Abszisse ist wiederum die x-Achse; die Ordinate ist hier aber statt der y-Achse jetzt die w-Achse.

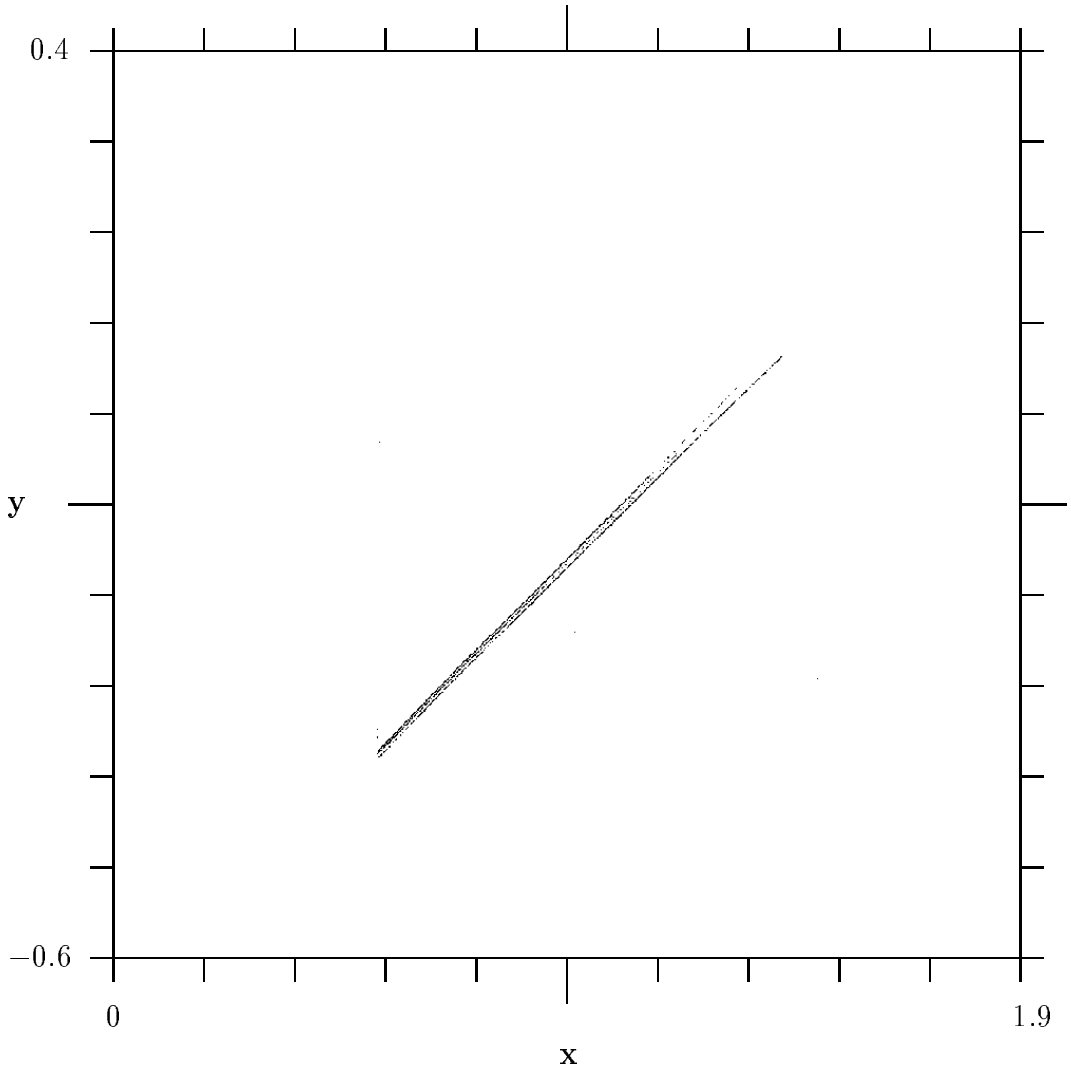


Abbildung 7.3: Querschnitt durch den Attraktor der Abbildung 7.1 mit exakt identischer Projektion (Abszisse x, Ordinate w). Dargestellt ist der Schnitt zwischen $w = 0.4800$ und $w = 0.48014$. 125 000 000 Iterationen.

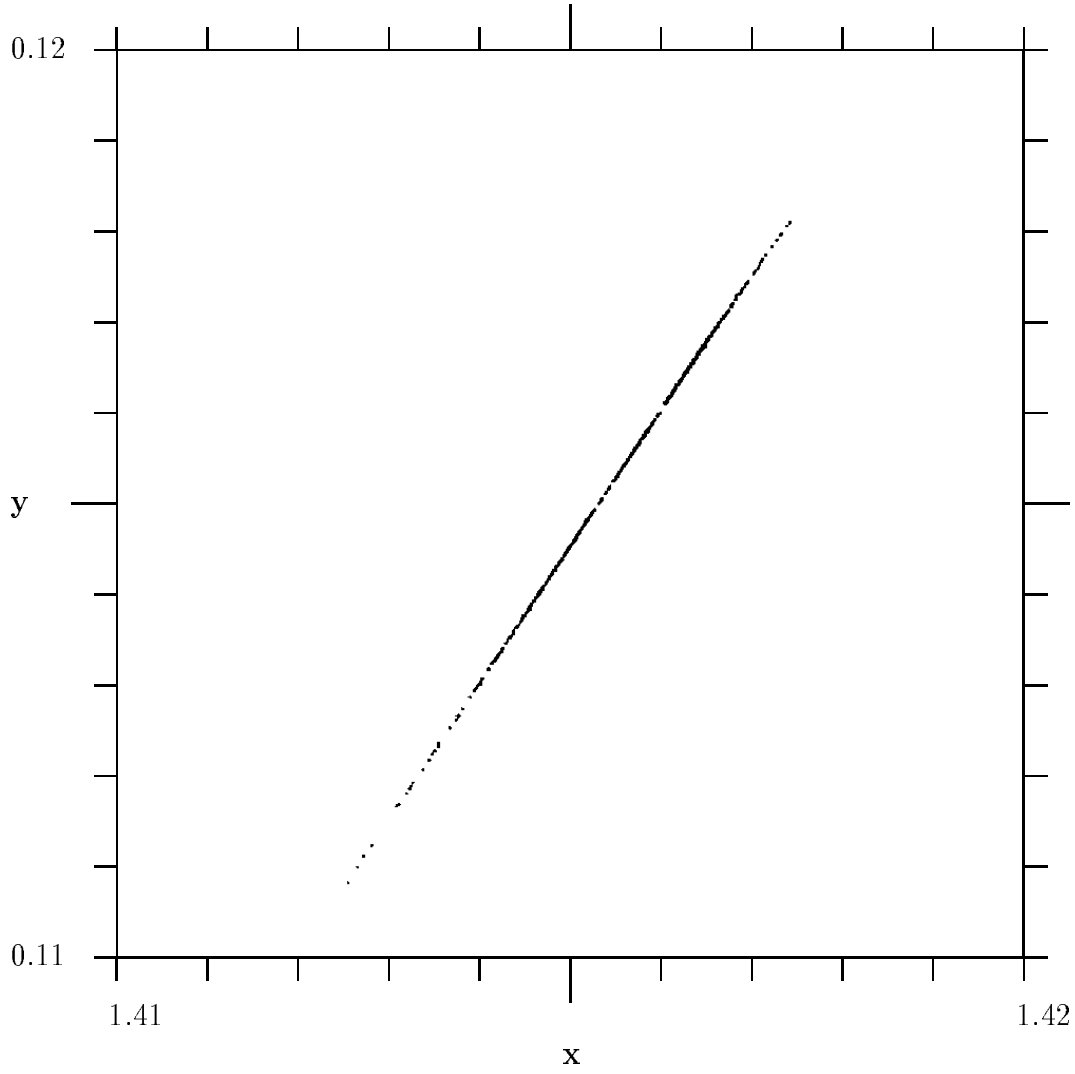


Abbildung 7.4: Detail-Vergrößerung der Abbildung 7.3 aus einem Segment (der obersten rechten Ecke); 1 000 000 000 Iterationen.

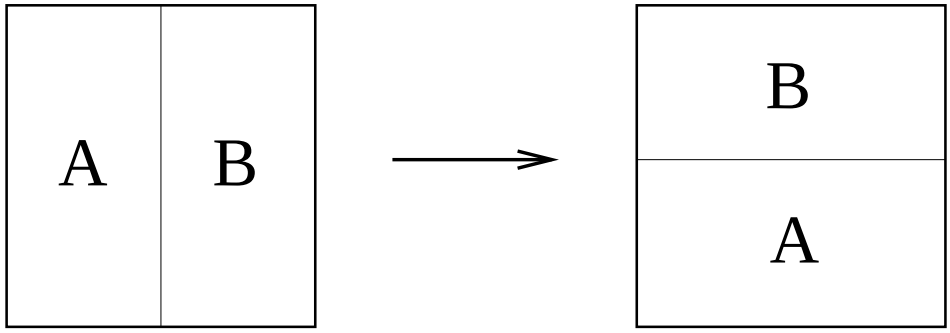


Abbildung 7.5: Baker's map

rie der Dynamischen Systeme eingeführt [38]. Sie hat grosse Bedeutung, weil sie einen intuitiven Zugang zu einem deterministisches Chaos aufweisenden System gewährt. Diese Transformation kann als protypisches Modell eines physikalischen Systems angesehen werden, das wegen seiner sensiblen Abhängigkeit von den Anfangsbedingungen in einem gewissen Sinn nur sehr unvollständig bekannt ist, da man keine Vorraussagen über den exakten Ort der späteren Bildpunkte machen kann [4].

Die Original-Map ist eine nicht-dissipative (also volumen- [bzw. hier flächen-]erhaltende) Abbildung. Dies trifft zu, obwohl der zugrundeliegende eindimensionale *Bernouilli-Shift* (die Sägezahnabbildung der ersten Variablen) für sich genommen ein dissipatives (nicht Volumen erhaltendes) System darstellt.

Die vollständigen System-Gleichungen lauten wie folgt:

$$\begin{aligned} x_{n+1} &= 2x_n \mod 1 \\ y_{n+1} &= \tfrac{1}{2}y_n + \begin{cases} x_n \leq \frac{1}{2} & : \quad 0 \\ x_n > \frac{1}{2} & : \quad \frac{1}{2} \end{cases} \end{aligned} \tag{7.4}$$

Fügt man einen Term in die zweite Gleichung ein, der für eine Phasenraumvolumenkontraktion sorgt, so gelangt man zur sogenannten *dissipativen* Baker's Map, vgl. Agyris [4].

Setzt man nun die Baker's Map als „Forcing" für ein Flaring-System ein,

so lautet eine exemplarische Gleichung für die flarende dritte Variable des Gleichungssystem wie folgt:

$$w_{n+1} = w_n + w_n(\frac{1}{3} - w_n) + 0.01 - 10^{-5} w_n^2 \tag{7.5}$$

Abb. 7.6 zeigt eine dreidimensionale Darstellung des durch die Gleichungen 7.4 und 7.5 beschriebenen invertierbaren Flare-Systems. Die Variable mit dem Flare-Verhalten (w) ist nach oben aufgetragen. Man erkennt so gut den fast ausschließlichen Aufenthalt des Trajektorienpunkts nahe der Ebene mit $w = 0$. Aus dieser Ebene treten einzelne quasisteadystate-ferne Flare-Werte heraus.

Zur genaueren strukturellen Untersuchung dieser dreidimensionalen Abbildungen wurde eine interaktive 3D-Darstellung (einen Bildschirm-Dump auf einer SGI-Workstation zeigt die Abb. 7.6) entwickelt. Dadurch wurde das Betrachten des dreidimensionalen Attraktors aus beliebigem Winkel und Richtung ermöglicht. Die graphischen Bedienelemente des 3D-Plotters wurden mittels eines zusätzlichen, die normalen Plotroutinen umfassenden TCL/TK-Programms erstellt, welches im Sourcen-Anhang dieser Arbeit abgedruckt ist.

7.3.2 Mehrfach gefaltete Horseshoe-Map

In diesem Abschnitt wird nun eine Abwandlung der sogenannten *Horseshoe* Map von Smale (oder genauer der expliziten gefalteten Map von Hénon) als Forcing einsetzt. Auch diese neuentwickelte Map ist wie die Baker's Map invertierbar.

Abbildung 7.7 zeigt den zugrundeliegenden chaotischen Attraktor, der ähnlich einer schwach kontrahierenden Baker's-Map (allerdings ohne Zerschneiden) gebaut ist: Die Map ist „nahezu flächenfüllend" — mit ihren überabzählbar vielen fraktalen Separierungen zwischen den horizontalen Segmenten in allen Größen.

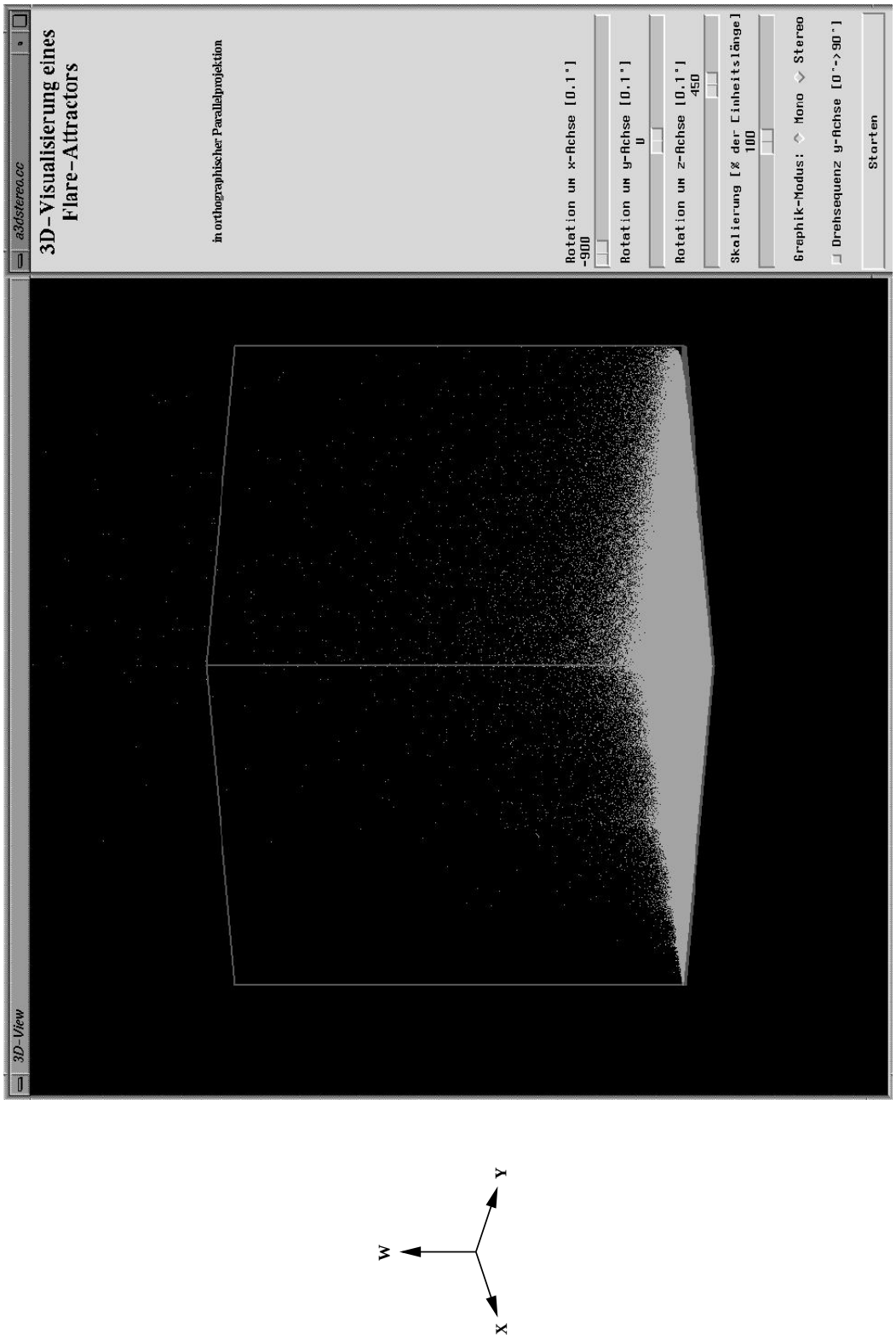


Abbildung 7.6: 3D Flare-Attraktor: Forcing durch die Baker's Map gemäß Gl. 7.5; x- und y-Achse [0,1], w-Achse [0,10].

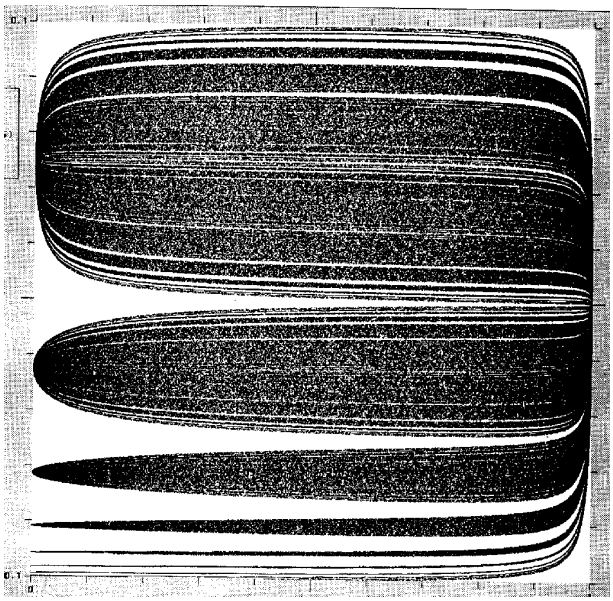


Abbildung 7.7: Horseshoe-Attraktor gemäß dem x-y Untersystem von Gl. 7.6; Abszisse x [0,1], Ordinate y [0,1].

Die Gleichung des vollständigen Dreivariablen-Systems lautet wie folgt:

$$\begin{aligned} x_{n+1} &= 4x_n(1-x_n) \\ y_{n+1} &= \frac{0.505(y_n-0.1)(1-2x_n)}{\sqrt{(1-2x_n)^2+0.01}} \\ w_{n+1} &= w_n + w_n(0.37-w_n) + 0.01 \end{aligned} \tag{7.6}$$

Abbildung 7.8 zeigt alle drei System-Dimensionen. Die Flares stellen auch hier Ausbrüche aus der quasisteadystate-fernen Ebene dar. Die senkrecht zueinanderstehenden zweidimensionalen Schnitte (y,z und x,z) durch den dreidimensionalen Attraktor offenbaren nun einen wichtigen strukturellen Unterschied zwischen dissipativen und nicht-dissipativen Flare-Attraktoren (wie im Fall der Abb. 7.3).

Ein Mittelschnitt durch den Würfel (oberhalb der Mitte) der Abbildung 7.8 liefert das Querschnittsbild der Abbildung 7.9: Die y,w-Darstellung —

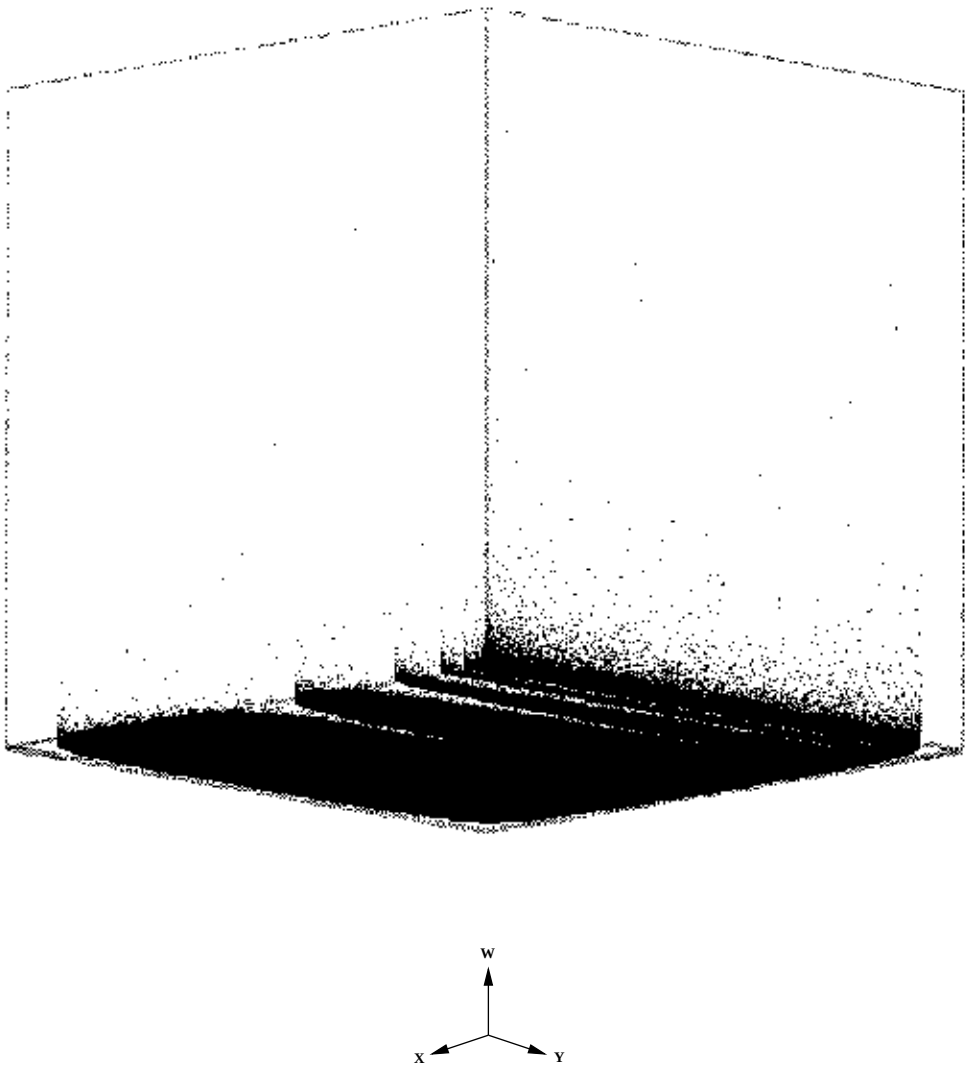


Abbildung 7.8: 3D Flare-Attraktor, geforct durch die vielfach gefaltete Horseshoe Map gemäß Abb. 7.6 und Gl. 7.6; x- und y-Achse [0,1], w-Achse [0,10].

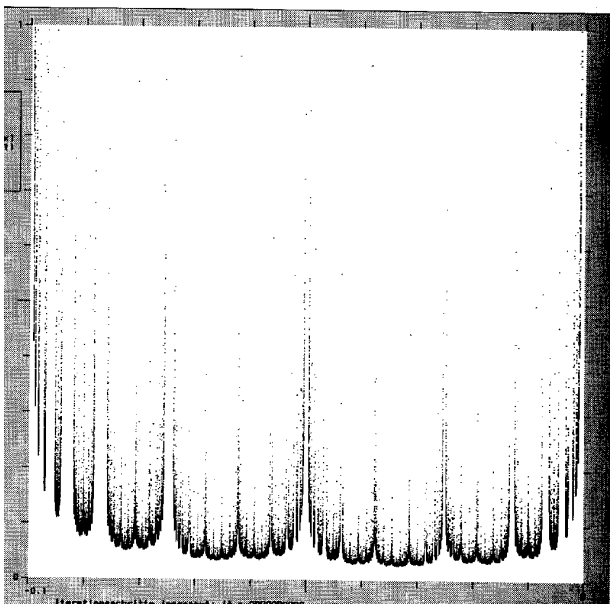


Abbildung 7.9: x, w-Schnitt durch den 3D Horseshoe-Flare-Attraktor; Abszisse x [0,1], Ordinate w [0,1].

eine charakteristische, fraktale Struktur („löwentatzenartig"). Sie ist typisch für Flare-Attraktoren mit sehr geringer Dissipation. Ein strukturell analoges Bild wird uns weiter unten bei der Tent-Baker's-Map begegnen (Abb. 7.11c).

Wenn man den Mechanismus der geforcten Variable („Flare-Variable") alteriert, indem man das Vorzeichen des nichtlinearen Schalt-Terms umkehrt†, erhält man einen strukturell völlig unterschiedlichen („feuerzungenartigen") Attraktor: Abbildung 7.10. Auch dieser Typ tritt ebenso bei der Tent-Baker's-Map des folgenden Abschnitts auf (dort nicht dargestellt).

7.3.3 Inverse Baker's-Map

Aus dem durch die Tent-Map angetriebende System kann man durch Hinzufügen einer dritten Variablen (analog zur Ableitung der Baker-Map aus

†Diese Möglichkeit wurde bereits in Abschnitt 3.2 als prinzipielle Variation des üblichen Prinzips geschildert.

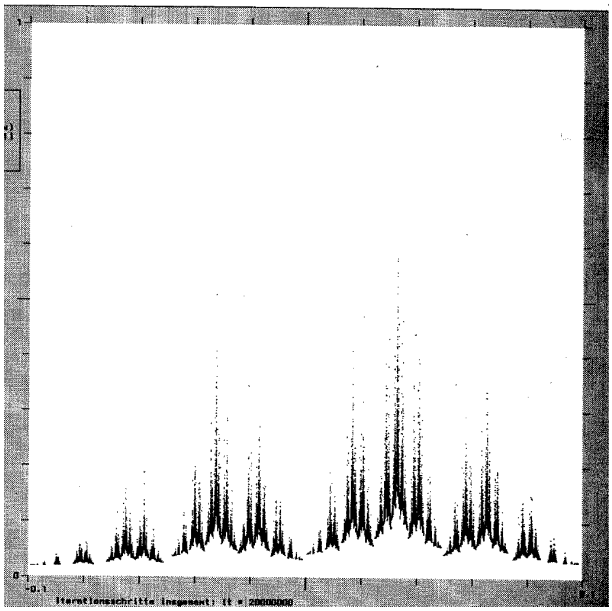


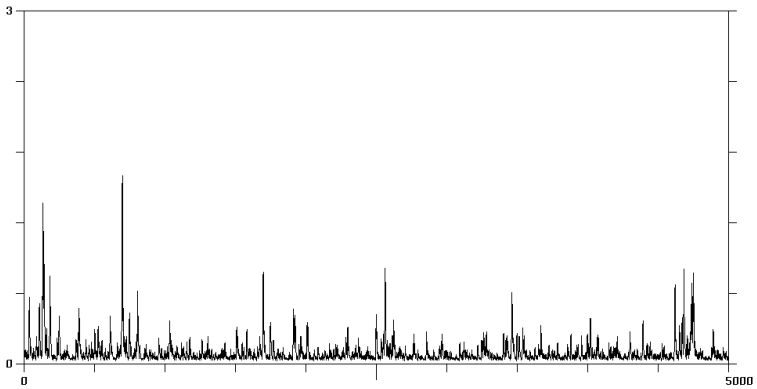
Abbildung 7.10: y, w-Schnitt durch den 3D Horseshoe-Flare-Attraktor („feuerzungenartig"); Abszisse y [0,1], Ordinate w [0,1].

dem Bernoulli-Shift) das folgende invertierbare System, ebenfalls mit Dissipation, erhalten[‡]

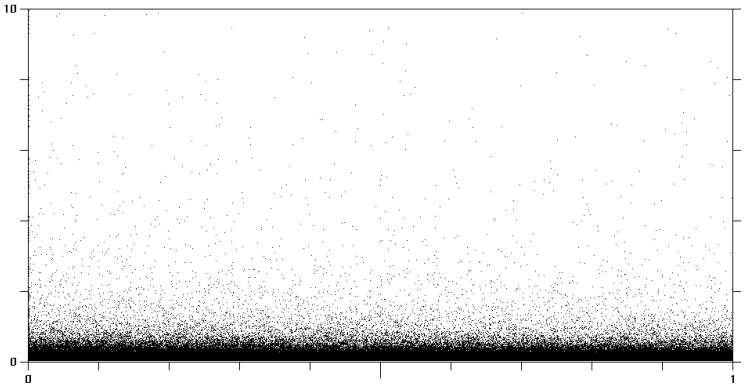
$$\begin{aligned} x_{n+1} &= 2\left|\frac{1}{2} - x\right| + 1 \\ y_{n+1} &= \begin{cases} x_n \leq \frac{1}{2}: & 1 - (\frac{1}{2} - 0.05)y_n \\ x_n > \frac{1}{2}: & (\frac{1}{2} - 0.05)y_n \end{cases} \\ w_{n+1} &= w_n + w_n(0.37 - x_n) + 10^{-2} - 10^{-3}w_n^2. \end{aligned} \tag{7.7}$$

Auch mit der Tent-Baker's-Map bekommen wir wieder dieselben Strukturen. In Abb. 7.11 sehen wir oben das Zeitverhalten von w, darunter eine Seitenansicht des Würfels (**b**), schließlich die andere Seitenansicht mit der „Löwentatzen"-Struktur (**c**).

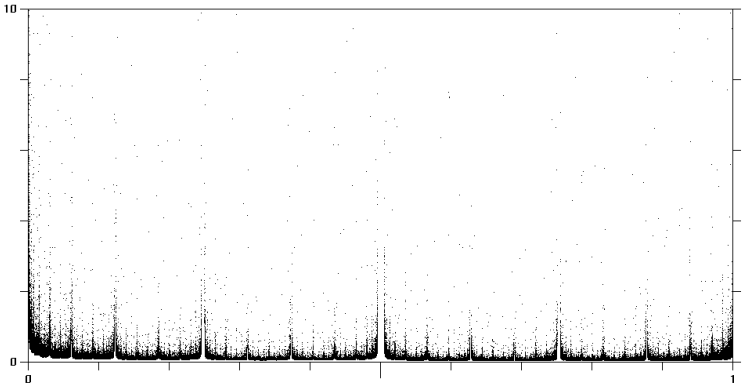
[‡]Solche Abbildungen werden deshalb manchmal auch als *Tent-Baker's-Map* bezeichnet.



a



b



c

Abbildung 7.11: Flare-Attraktor, erzeugt durch Gl. 7.7. **a**: Zeitserie von w, 5 000 Iterationen. **b**: Seiten-Ansicht (x,w Plot) **c**: Queransicht: y,w Plot. Ein Querschnitt zwischen $x = 0$ und $x = 1$ ist dargestellt. 1 000 000 Iterationen abgebildet (in **b** und **c**). Anfangsbedingungen: $x_0 = \frac{1}{10}\sqrt{2}$, $y_0 = 0.1$, $w_0 = 0.1$.

Kapitel 8

Gekoppelte Zellen mit Flare-Dynamik

8.1 Einleitung

In den folgenden Abschnitten werden größere Dynamische Systeme untersucht, welche sich aus einzelnen kleineren, diffusiv gekoppelten Untersystemen (dynamischen Zellen) zusammensetzen. Hierbei zeigt sich, dass das bisher beschriebene dynamische Verhalten „Flaring" äußerst stabil gegen kleinere Störeinflüsse von außen ist — eine Eigenschaft die in der Theorie der Dynamischen Systeme als *Robustheit** bezeichnet wird [18][40].

*Das Erfüllen der Bedingung „Robustheit" ist Vorraussetzung für den Einsatz von Dynamischen Systemen zu Simulationen von beispielsweise ökonomischen Modellen [39]. Der Ökonomie-Lehrstuhlinhaber Barkley Rosser (Harrisonburg, U.S.A.) benutzt Flare-Systeme für einfache Modellsysteme in Stockmarket-Analysen: Dem Umschalten zwischen Autokatalyse und Selbstinhibition mittels des nichtlinearen Schalters entspricht im Stockmarket-Modell ein Umschalten zwischen der Chartisten- (investiert v.a. in steigende Werte) und der Antizyklisten- (investiert v.a. in fallende bzw. gefallene Werte) Anlagestrategie.

8.2 Eine einfache Summendynamik

Im Folgenden werden nun mehrere Flare-Dynamiken gekoppelt. Abb. 8.1 zeigt in **b** und **c** zwei verschiedene Summendynamiken von verschieden großen gekoppelten Systemen.

Das erste System besteht nur aus drei Flare-Zellen, das zweite System enthält sechs, das dritte achtzehn Zellen. Das nun folgende Gleichungssystem stellt das im Modell realisierte 18-Zellen-System dar, wobei die diffusive Kopplung über die s-Variable des Gleichungssystems erfolgt. Jedes einzelne Untersystem besitzt eine eigene, von allen anderen Untersystemen verschiedene Flare-Dynamik, die durch eine chaotische Dynamik — erzeugt durch eine logistische Abbildung — angetrieben wird; vergl. Gl. 3.3 und Abb. 3.4. Die Gleichungssysteme für drei bzw. sechs Zellen kann man leicht aus diesem größten Gleichungssystem ableiten in dem man nur die Gleichungen bis $x_{n+1}^{(3)}, b_{n+1}^{(3)}$ bzw. $x_{n+1}^{(6)}, b_{n+1}^{(6)}$ verwendet. Die Summenvariable s_{n+1} setzt sich dann entsprechend nur aus drei bzw. sechs b_n-Elementen zusammen; s ist die Summenvariable die im System für die diffusive Kopplung der Untersysteme sorgt.

Das vollständige Gleichungssystem für alle achtzehn Zellen lautet nun wie folgt:

$$
\begin{aligned}
x_{n+1}^{(1)} &= 3.99 x_n^{(1)}(1-x_n^{(1)}) \\
b_{n+1}^{(1)} &= b_n^{(1)} + b_n^{(1)}(0.565 - x_n^{(1)}) - 10^{-3} {z_n^{(1)}}^2 + 10^{-3} s \\
x_{n+1}^{(2)} &= 3.99 x_n^{(2)}(1-x_n^{(2)}) \\
b_{n+1}^{(2)} &= b_n^{(2)} + b_n^{(2)}(0.566 - x_n^{(2)}) - 10^{-3} {z_n^{(2)}}^2 + 10^{-3} s \\
x_{n+1}^{(3)} &= 3.99 x_n^{(3)}(1-x_n^{(3)}) \\
b_{n+1}^{(3)} &= b_n^{(3)} + b_n^{(3)}(0.567 - x_n^{(3)}) - 10^{-3} {z_n^{(3)}}^2 + 10^{-3} s \\
x_{n+1}^{(4)} &= 3.99 x_n^{(4)}(1-x_n^{(4)}) \\
b_{n+1}^{(4)} &= b_n^{(4)} + b_n^{(4)}(0.568 - x_n^{(4)}) - 10^{-3} {z_n^{(4)}}^2 + 10^{-3} s \\
x_{n+1}^{(5)} &= 3.99 x_n^{(5)}(1-x_n^{(5)}) \\
b_{n+1}^{(5)} &= b_n^{(5)} + b_n^{(5)}(0.569 - x_n^{(5)}) - 10^{-3} {z_n^{(5)}}^2 + 10^{-3} s
\end{aligned}
$$

$$
\begin{aligned}
x_{n+1}^{(6)} &= 3.99 x_n^{(6)}(1-x_n^{(6)}) \\
b_{n+1}^{(6)} &= b_n^{(6)} + b_n^{(6)}(0.570 - x_n^{(6)}) - 10^{-3} {z_n^{(6)}}^2 + 10^{-3} s \\
x_{n+1}^{(7)} &= 3.99 x_n^{(7)}(1-x_n^{(7)}) \\
b_{n+1}^{(7)} &= b_n^{(7)} + b_n^{(7)}(0.571 - x_n^{(7)}) - 10^{-3} {z_n^{(7)}}^2 + 10^{-3} s \\
x_{n+1}^{(8)} &= 3.99 x_n^{(8)}(1-x_n^{(8)}) \\
b_{n+1}^{(8)} &= b_n^{(8)} + b_n^{(8)}(0.572 - x_n^{(8)}) - 10^{-3} {z_n^{(8)}}^2 + 10^{-3} s \\
x_{n+1}^{(9)} &= 3.99 x_n^{(9)}(1-x_n^{(9)}) \\
b_{n+1}^{(9)} &= b_n^{(9)} + b_n^{(9)}(0.573 - x_n^{(9)}) - 10^{-3} {z_n^{(9)}}^2 + 10^{-3} s \\
x_{n+1}^{(10)} &= 3.99 x_n^{(10)}(1-x_n^{(10)}) \\
b_{n+1}^{(10)} &= b_n^{(10)} + b_n^{(10)}(0.574 - x_n^{(10)}) - 10^{-3} {z_n^{(10)}}^2 + 10^{-3} s \\
x_{n+1}^{(11)} &= 3.99 x_n^{(11)}(1-x_n^{(11)}) \\
b_{n+1}^{(11)} &= b_n^{(11)} + b_n^{(11)}(0.575 - x_n^{(11)}) - 10^{-3} {z_n^{(11)}}^2 + 10^{-3} s \\
x_{n+1}^{(12)} &= 3.99 x_n^{(12)}(1-x_n^{(12)}) \\
b_{n+1}^{(12)} &= b_n^{(12)} + b_n^{(12)}(0.576 - x_n^{(12)}) - 10^{-3} {z_n^{(12)}}^2 + 10^{-3} s \\
x_{n+1}^{(13)} &= 3.99 x_n^{(13)}(1-x_n^{(13)}) \\
b_{n+1}^{(13)} &= b_n^{(13)} + b_n^{(13)}(0.577 - x_n^{(13)}) - 10^{-3} {z_n^{(13)}}^2 + 10^{-3} s \\
x_{n+1}^{(14)} &= 3.99 x_n^{(14)}(1-x_n^{(14)}) \\
b_{n+1}^{(14)} &= b_n^{(14)} + b_n^{(14)}(0.578 - x_n^{(14)}) - 10^{-3} {z_n^{(14)}}^2 + 10^{-3} s \\
x_{n+1}^{(15)} &= 3.99 x_n^{(15)}(1-x_n^{(15)}) \\
b_{n+1}^{(15)} &= b_n^{(15)} + b_n^{(15)}(0.579 - x_n^{(15)}) - 10^{-3} {z_n^{(15)}}^2 + 10^{-3} s \\
x_{n+1}^{(16)} &= 3.99 x_n^{(16)}(1-x_n^{(16)}) \\
b_{n+1}^{(16)} &= b_n^{(16)} + b_n^{(16)}(0.580 - x_n^{(16)}) - 10^{-3} {z_n^{(16)}}^2 + 10^{-3} s \\
x_{n+1}^{(17)} &= 3.99 x_n^{(17)}(1-x_n^{(17)}) \\
b_{n+1}^{(17)} &= b_n^{(17)} + b_n^{(17)}(0.581 - x_n^{(17)}) - 10^{-3} {z_n^{(17)}}^2 + 10^{-3} s \\
x_{n+1}^{(18)} &= 3.99 x_n^{(18)}(1-x_n^{(18)}) \\
b_{n+1}^{(18)} &= b_n^{(18)} + b_n^{(18)}(0.582 - x_n^{(18)}) - 10^{-3} {z_n^{(18)}}^2 + 10^{-3} s \\
s_{n+1} &= b_n^{(1)} + b_n^{(2)} + \ldots
\end{aligned}
\tag{8.1}
$$

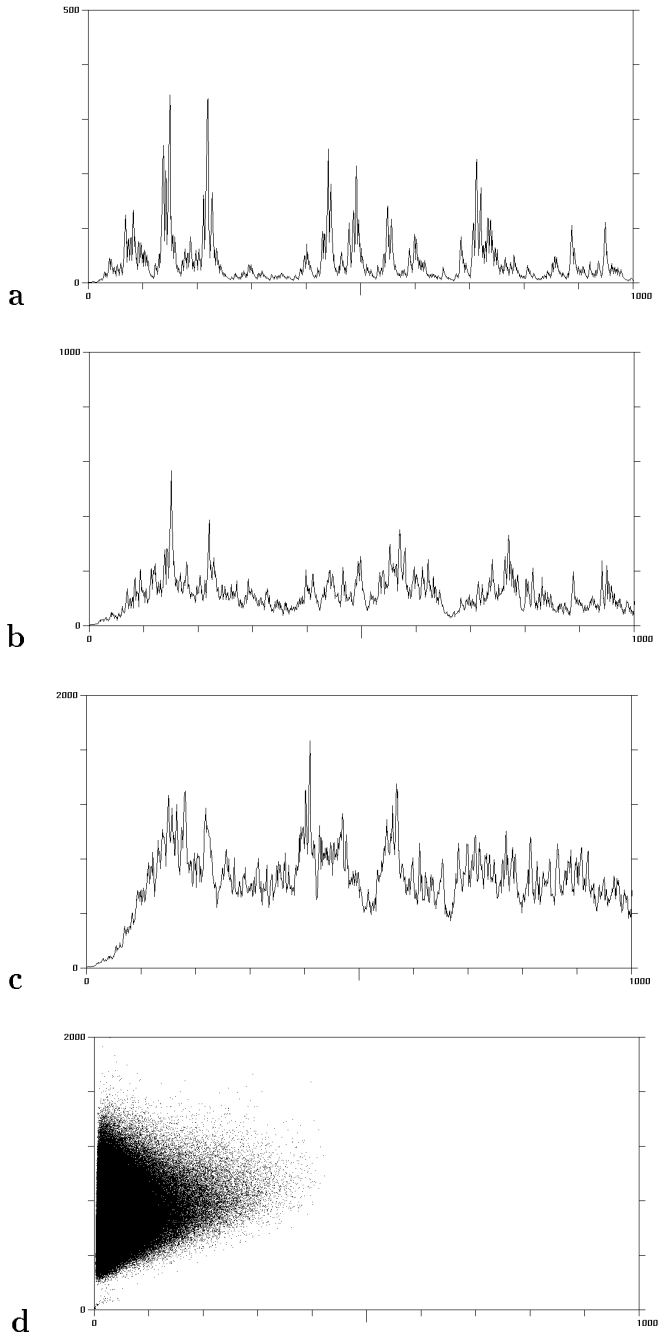


Abbildung 8.1: Gekoppelte Flaresysteme mit 3, 6 und 18 Zellen: Numerische Simulation von 8.1. **a**: Zeitserie von $b^{(1)}$, ungekoppelt. Ein identisches Verhalten erhält man, wenn man den letzten Term, $10^{-3}s$, der zweiten Zeile von Gl. 8.1 durch die Konstante 0.6 ersetzt. **b**: Zeitserie von $b^{(1)}$ bis $b^{(6)}$ als Summendarstellung (s). **c**: Zeitserie von $b^{(1)}$ bis $b^{(18)}$ als Summendarstellung (s). (**a**–**c**): Es sind jeweils 1 000 Iterationen dargestellt. **d**: $b^{(18)}, s$-Darstellung, 1 000 000 Iterationen. Startbedingungen für $x^{(j)}$: 0.010, 0.011, 0.012, etc.; für $b^{(j)}$: jeweils 0.2.

Zur Untersuchung des gegenseitigen Störverhaltens besitzt jede einzelne Zelle eine eigene, von allen anderen Zellen verschiedene Dynamik. Dies wird erreicht, in dem zum einen für jede Zelle unterschiedliche Startbedingungen für $x^{(i)}$ verwandt werden, was schon ausreicht um ein jeweils völlig eigenes dynamisches Zeitverhalten zu erzeugen; zum anderen kommen jeweils unterschiedliche *Schwellenwerte* im Gleichungsterm der zweiten Variable ($b^{(i)}$) zur Anwendung, wodurch ein auch bei gleichem Input unterschiedliches Flare-Verhalten garantiert ist.

Man erkennt aus Abb. 8.1 b) und c), dass die größeren Flare-Peaks als Maxima auch bei stärkerer Kopplung weiterhin an gleicher Stelle auftauchen!

Abb. 8.1 d) zeigt das Verhalten einer *einzelnen* Flareattraktor-Variablen, gegen die Summenvariable s aller Teilsysteme aufgetragen. Die charakteristische sich „ausdünnende" Morphologie aller Flare-Attraktoren (in der Phasenraumdarstellung) ist deutlich zu erkennen.

Wir kommen nun zu den drei Zeitserien in Abb. 8.2. In diesen Simulationsrechnungen fungiert die Summenvariable s als ein *zeitlicher Integrator*. Anstatt nur in jedem Iterationschritt $(x_n^{(j)}, b_n^{(j)} \rightarrow x_{n+1}^{(j)}, b_{n+1}^{(j)})$ die momentane Summe aus den momentanen $b_n^{(j)}$-Werten darzustellen, lautet die neue Summenvariable in diesem Fall:

$$s_{n+1} = s_n + b_n^{(1)} + b_n^{(2)} + ... - as_n. \tag{8.2}$$

Im Fall von $a = 1$ erhält man wieder das ursprüngliche, rein instantane Aufsummieren der b-Werte, wie in Abb. 8.1c, zurück. Geht man zu Werten von a kleiner als 1 über, so zeigt sich bei den Zeitserien wegen der nun auftretenden integrativen Wirkung ein Glättungseffekt, der bei kleineren a-Werten immer stärker wird.

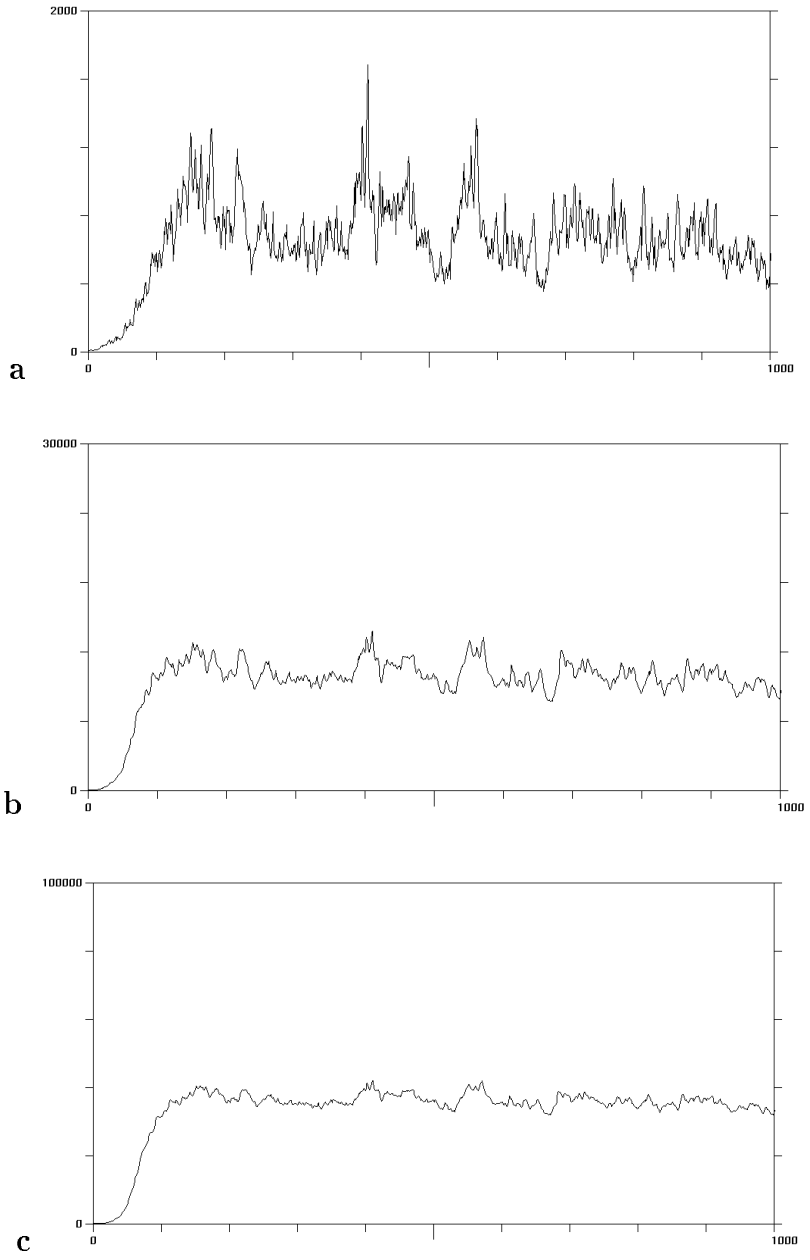


Abbildung 8.2: Numerische Simulation von Gleichung 8.1, wobei die letzte Zeile durch die Gleichung 8.2 ersetzt ist. **a**: $a = 0.88$, nahezu kein glättender Effekt im Vergleich zur Abb. 8.1d. **b**: $a = 0.2$, mittelstarke Glättung. **c**: $a = 0.05$, starke Glättung. Vgl. Text.

Teil IV

Flare-Dynamik in einem konservativem System

Kapitel 9

Variationen zur Baker's Map

9.1 Vorbemerkungen

Mit einer Abwandlung der ursprünglichen Baker's Map gelingt es, Zeitreihen mit flare-hafter Dynamik auch in einem *konservativen* Dynamischen System zu erzeugen. Dieses System kann auch als ein stark vereinfachtes Modell zur Analyse sogenannter dynamischer *Poincaré-Zyklen* aufgefasst werden.

9.2 Die Variation der Baker's Map nach Hopf und Gaspard

Die Baker's Map wurde bereits in Abschnitt 7.3.1 ausführlich vorgestellt. Ein von der ursprünglichen Baker's Map (Abb. 7.5) abgeleitetes iteriertes Dynamisches System ist die Map nach *Hopf-Gaspard*.

Die Ableitung der Hopf-Gaspard Abbildung aus der Baker's Map wird sofort einsichtig, wenn man in der Abbildung 9.1 die Teile α und **d** des mittleren Drittels durch die Teile A und D ersetzt. Man erhält dann einfach eine Vier-Teile Version der ursprünglichen Baker's Map [38][41].

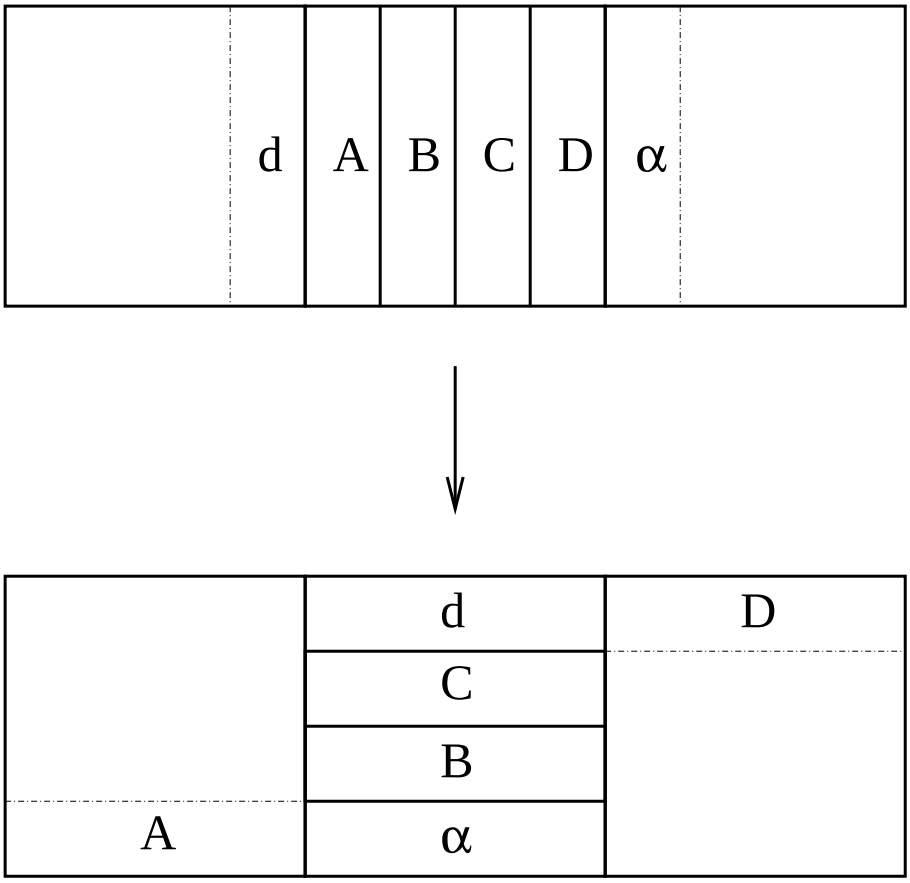


Abbildung 9.1: Iterationsschema zur Hopf-Gaspard Map, nach [41]

Im Prinzip ist es möglich noch einen Schritt weiter zu gehen: Aus der Hopf-Gaspard Abbildung kann eine treppenstufenartige Abwandlung entwickelt werden. Diese bringt ebenfalls flare-haften dynamische Zeitserien hervor, wie im folgenden Kapitel gezeigt wird.

Kapitel 10

Staircase Baker's Map mit flarehaftem dynamischem Verhalten

10.1 Definition

Das Prinzip der Abbildung lässt sich leicht anhand eines graphischen Schemas veranschaulichen, vergl. [42]. Abbildung 10.1 zeigt das Prinzip.

Die fertige treppenhafte Bäcker-Abbildung („Staircase Baker's Map") ist, wie man sieht, nichts anderes als eine gewöhnliche 4-teilige Baker's Map vom Hopf-Gaspard-Typ (Abb. 9.1), die mehrfach nach rechts fortgesetzt ist. Die neue Variante ergibt sich durch die abnehmende Stufenhöhe. Mit jedem Schritt wird trotzdem nach wie vor zwischen je zwei benachbarten Elementen dasselbe Stück Fläche („Teig-Volumen") ausgetauscht. Bei jeder Stufenpaar reduziert sich das Volumen um einen frei zu wählenden Faktor*.

In Abb. 10.1 ist der Fall von nur 4 Treppenstufen dargestellt. Man erkennt, dass jeweils die Breite des ganz rechts gelegenen Segmentes einer Stufe

*Aus Gründen leichterer Anschauung und des Rechen-Komforts wählt man am besten den Faktor 1/2; prinzipiell kann aber jeder Faktor kleiner Eins gewählt werden!

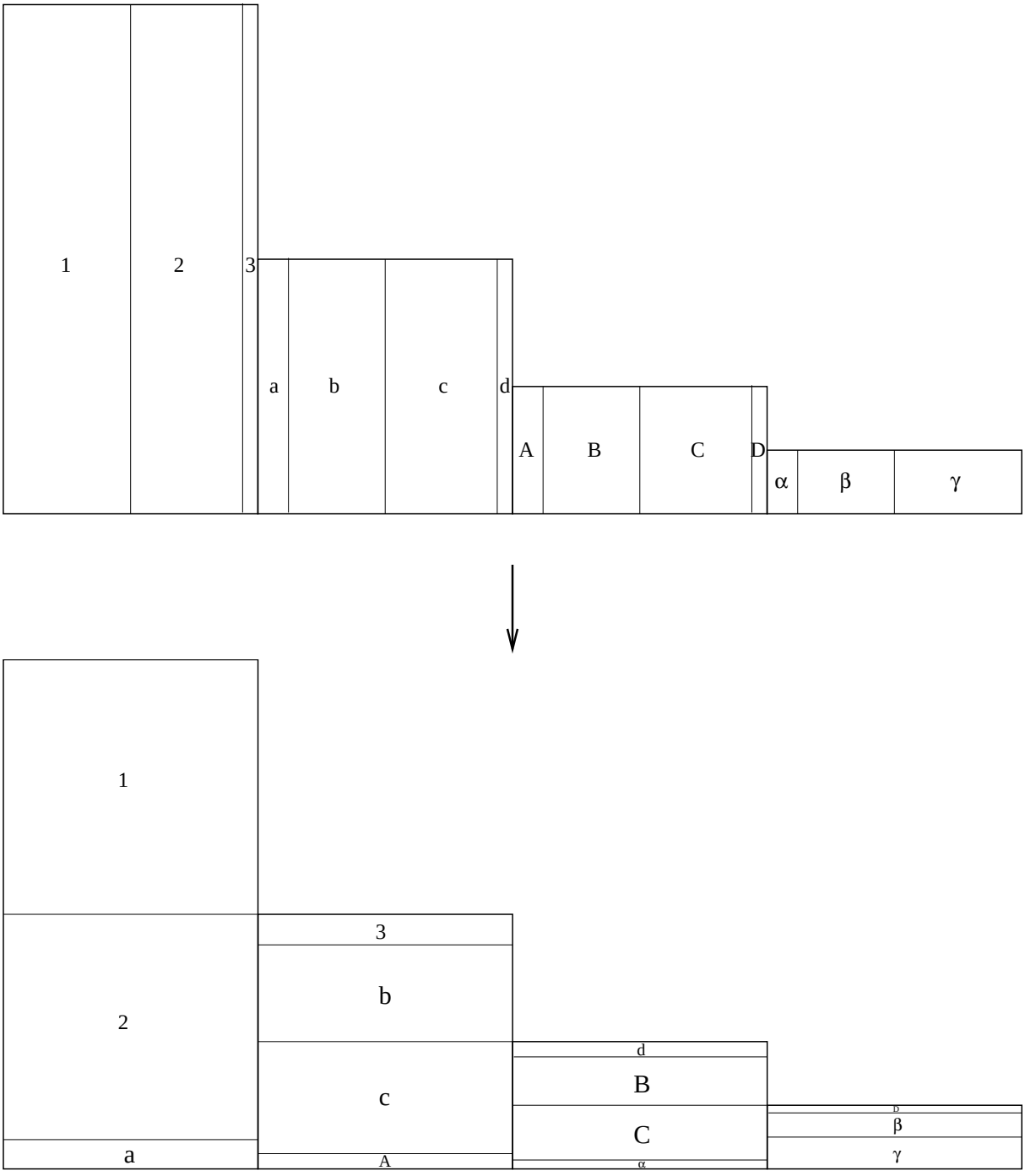


Abbildung 10.1: Staircase Baker's Map im Fall von 4 Stufen

(bspw. D) und die Breite des ganz links gelegenen Segmentes der nächsten Stufe (z.B. α) sich um einen konstanten Faktor unterscheiden (hier also um den gewählten Faktor 2), um einen Austausch von gleichen Flächenstücken zu gewährleisten. Es wird daher in der Tat eine gleich große Menge von „Teig" zwischen zwei benachbarten Stufen pro Iterationsschritt ausgetauscht, so wie dies auch bei der Map nach Hopf-Gaspard mit konstanter Höhe (Abb. 9.1) der Fall war.

Anstatt nur für den Fall $n = 4$ (wie in Abb. 10.1 zu sehen) ein explizites Gleichungssystem anzugeben, wird im Folgenden ein allgemeinerer Algorithmus für eine beliebige Stufenanzahl n angegeben. Als Verkleinerungsfaktor von Stufe zu Stufe wurde aus den schon beschriebenen Gründen wieder 1/2 gewählt. Die Aufteilung der 4 Elemente der inneren (nicht Rand-) Treppenstufen wurde im Verhältnis $12 : 38 : 44 : 6$ gewählt.† Lediglich in der ersten und der letzten Stufe wurden jeweils die beiden äußeren Elemente zusammengefasst, da diese Stufen nicht mehr nach außen austauschen (also $38 + 12 = 50$ bzw. $44 + 6 = 50$).

Der vollständig ausformulierte Algorithmus ist auf der folgenden Seite dargestellt:

†Auch dieses Aufteilungsverhältnis kann natürlich variiert werden!

80 Staircase Baker's Map mit flarehaftem dynamischem Verhalten

Staircase Baker's Map nach Abb. 10.1, mit n Stufen, n beliebig:

$$
x_{n+1} = \begin{cases}
[\text{ für } \quad 0 < x_n \le 1\text{: }] & & & & \\
\frac{100}{50} x_n & \text{für} & 0 & < x_n \le & \frac{50}{100} \\
\frac{100}{44}(x_n - \frac{50}{100}) & \text{für} & \frac{50}{100} & < x_n \le & \frac{94}{100} \\
\frac{100}{6}(x_n - \frac{94}{100}) + 1 & \text{für} & \frac{94}{100} & < x_n \le & 1 \\
[\text{ für } \quad 1 < x_n \le m\text{: }] & & & & \\
\frac{100}{12}(x_n - k_n) + k_n - 1 & \text{für} & k_n & < x_n \le & (k_n + \frac{12}{100}) \\
\frac{100}{38}(x_n - (k_n + \frac{12}{100})) + k_n & \text{für} & (k_n + \frac{12}{100}) & < x_n \le & (k_n + \frac{50}{100}) \\
\frac{100}{44}(x_n - (k_n + \frac{50}{100})) + k_n & \text{für} & (k_n + \frac{50}{100}) & < x_n \le & (k_n + \frac{94}{100}) \\
\frac{100}{6}(x_n - (k_n + \frac{94}{100})) + k_n + 1 & \text{für} & (k_n + \frac{94}{100}) & < x_n \le & (k_n + 1) \\
[\text{ für } \quad m < x_n \le (m+1)\text{: }] & & & & \\
\frac{100}{12}(x_n - m) + m - 1 & \text{für} & m & < x_n \le & (m + \frac{12}{100}) \\
\frac{100}{38}(x_n - (m + \frac{12}{100})) + m & \text{für} & (m + \frac{12}{100}) & < x_n \le & (m + \frac{50}{100}) \\
\frac{100}{50}(x_n - (m + \frac{50}{100})) + m & \text{für} & (m + \frac{50}{100}) & < x_n \le & (m + 1)
\end{cases}
$$

$$
y_{n+1} = \begin{cases}
\frac{50}{100} y_n + \frac{50}{100} & \text{für} & 0 & < x_n \le & \frac{50}{100} \\
\frac{44}{100} y_n + \frac{6}{100} & \text{für} & \frac{50}{100} & < x_n \le & \frac{94}{100} \\
\frac{6}{100} y_n + \frac{44}{100} & \text{für} & \frac{94}{100} & < x_n \le & 1 \\
\frac{12}{100} y_n + \frac{88}{100} \cdot \frac{1}{2^{k_n}} - \frac{88}{100} \cdot \frac{1}{2^{k_n}} & \text{für} & k_n & < x_n \le & (k_n + \frac{12}{100}) \\
\frac{38}{100} y_n + \frac{50}{100} \cdot \frac{1}{2^{k_n}} & \text{für} & (k_n + \frac{12}{100}) & < x_n \le & (k_n + \frac{50}{100}) \\
\frac{44}{100} y_n + \frac{6}{100} \cdot \frac{1}{2^{k_n}} & \text{für} & (k_n + \frac{50}{100}) & < x_n \le & (k_n + \frac{94}{100}) \\
\frac{6}{100} y_n + \frac{88}{100} \cdot \frac{1}{2^{k_n+1}} & \text{für} & (k_n + \frac{94}{100}) & < x_n \le & (k_n + 1) \\
\frac{12}{100} y_n + \frac{88}{100} \cdot \frac{1}{2^m} - \frac{88}{100} \cdot \frac{1}{2^m} & \text{für} & m & < x_n \le & (m + \frac{12}{100}) \\
\frac{38}{100} y_n + \frac{50}{100} \cdot \frac{1}{2^m} & \text{für} & (m + \frac{12}{100}) & < x_n \le & (m + \frac{50}{100}) \\
\frac{50}{100} y_n & \text{für} & (m + \frac{50}{100}) & < x_n \le & (m + 1)
\end{cases}
$$

(10.1)

In der vorstehenden Gleichung wurde eine Hilfsgröße k_n eingesetzt mit der folgenden Definition:

$$k_n = \text{trunc}(x_n) = x_n - (x_n \bmod 1). \tag{10.2}$$

Durch den ermittelten Wert von k_n wird also jeweils im Algorithmus die Stufe ausgewählt, in der sich der Trajektorienpunkt zu diesem Iterationszeitpunkt gerade befindet.

10.2 Numerische Simulationen der Staircase Baker's Map

Abbildung 10.2 zeigt eine numerische Simulation für 40 Treppenstufen ($n = 40$), wobei wegen des exponentiellen Abfalls der Stufenhöhe und der begrenzten graphischen Auflösung nur die Stufen 1 bis 10 dargestellt sind. Detailvergrößerungen der noch weiter rechts gelegenen Stufen wurden ebenfalls gerechnet; wegen der weitaus geringeren Aufenthaltswahrscheinlichkeit des Zustandspunktes in den immer kleineren Stufen steigt dabei die notwendige Rechenzeit drastisch an und es ergeben sich gewisse Artefakte durch die eingeschränkte Rechengenauigkeit der Computerfließkommaarithmetik[‡].

Da die Höhe jedes Stufenelements mit steigender Stufenzahl exponentiell absinkt, empfiehlt sich für eine genauere Analyse die Anwendung mathematischer Funktionsbibliotheken, die eine höhere Genauigkeit als die des im Rahmen dieser Arbeit normalerweise angewandten Standards ANSI-C erlauben. Aus Kostengründen und wegen der etwas begrenzten Hauptspeicherkapazität der Workstation, auf der die angewendeten Simulationsprogramme implementiert sind, empfahl sich die Anwendung der frei verfügbaren GNU Multiple Precision Library (kurz GMP genannt), welche die Verwendung solcher Routinen auch bei vergleichsweise geringen Rechenkapazitäten erlaubt[§].

[‡]Zu diesem Punkt siehe auch den folgenden Abschnitt 10.3.

[§]32 MB Hauptspeicher der SGI XS4000 Workstation

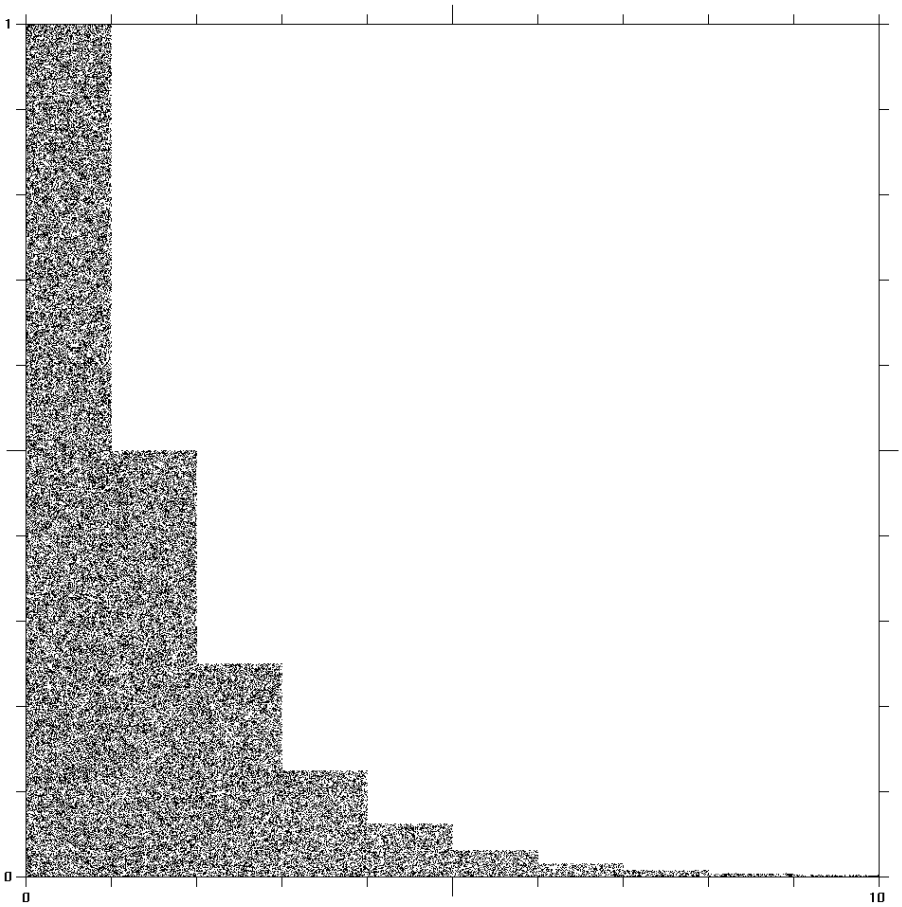


Abbildung 10.2: Phasenraumdarstellung der Staircase Baker Map mit 10-facher Überhöhung der y-Variablen. Numerische Berechnung der Gl. 10.1 mit Double-Precision-Fließkomma-Arithmetik. 100 000 Iterationen, Stufenzahl insgesamt $m = 40$, nur die 10 größten Stufen sind dargestellt.

Die GMP Bibliothek gestattet prinzipiell jeden beliebigen Fließkommagenauigkeitsgrad!

Ein gewisser Nachteil dieser Bibliothek ist jedoch, dass das Programm in einer neuen Syntax zu erstellen ist. Zur Veranschaulichung dieser Problematik sei hier kurz eine 1:1 Übertragung eines kurzen Stücks C-Quellcode der Implementierung von Gl. 10.1 in den entsprechenden C-Code mit GMP-Extension angeführt:

- Ursprünglicher ANSI-C Code:

```
if (x<=0.50) {
          x = 2.0*x;
          y = 0.5*y + 0.5;
}
```

- C-Code mit GMP-Extension:

```
if (mpf_cmp(xx,o5)<=0)  {
  mpf_mul(xx,two0,xx);
  mpf_mul(du,o5,yy);
  mpf_add(yy,du,o5);
}
```

Um die Übertragung überschaubar zu halten, empfiehlt sich die Verwendung von deskriptiven neuen Variablennamen[¶].

Während in der vorherigen Abbildung das Verhalten beider Variablen im gesamten Phasenraum dargestellt wurde, betrachten wir nun das Verhalten von nur einer Variablen in der Zeit — die horizontal elongierte x-Variable (also die momentane Treppenlänge).

[¶]Der vollständige Source-Code der GMP-Implementierung findet sich im Abschnitt A.3 des Anhangs.

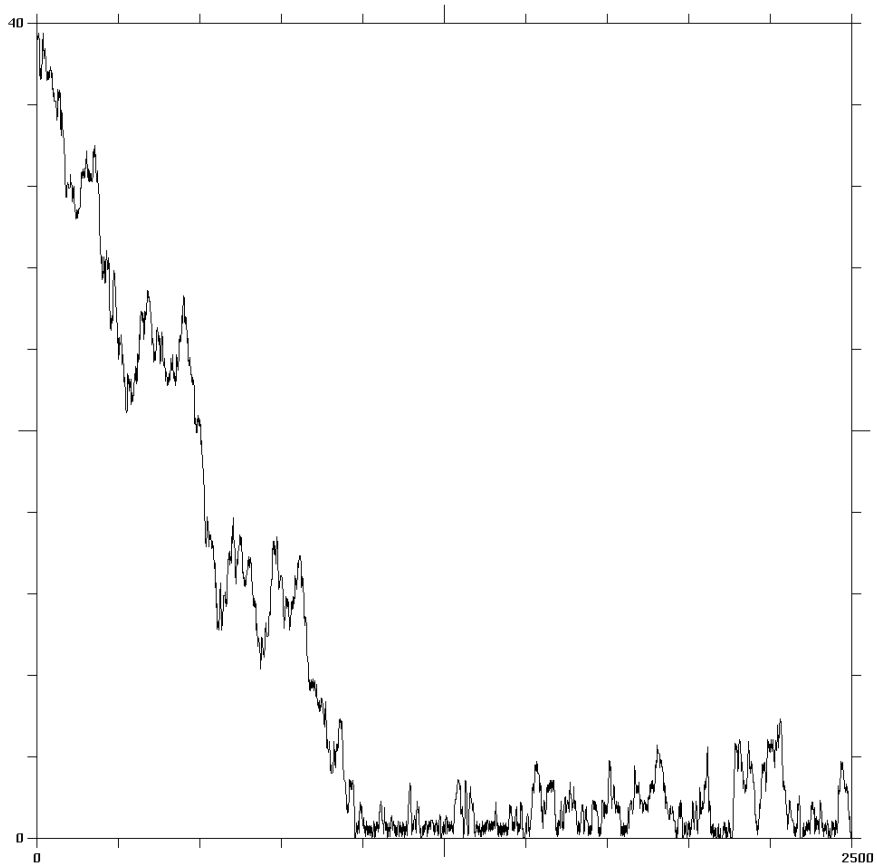


Abbildung 10.3: Zeitverhalten der horizontalen Variablen x der Staircase Baker's Map mit 40 Stufen für einen Anfangspunkt ganz rechts im Fuß der Treppe, also bei maximaler x-Amplitude, $x \approx 40$. Anfangsbedingungen: $x_0 = 39,9$, $y_0 = 0,0$. 2500 Iterationen bei 16 Dezimalstellen Fließkommagenauigkeit (C double percision).

Abb. 10.3 zeigt einen kurzen Abschnitt des zeitlichen Verhaltens der (nach oben aufgetragenen) horizontalen Amplitude der Staircase Map von Gl. 10.1. Es wurde ein Startpunkt in der äußersten 40. Treppenstufe gewählt.

Man beobachtet eine rasche Abnahme der horizontalen x-Amplitude hin zu einer fluktierenden „Gleichgewichtsamplitude" bei niedrigen x-Werten, also nahe der nach links abschließenden größten Stufe mit $x = 0$ an deren linkem Rand. Diese Stufe besitzt den größten Flächeninhalt und damit auch die größte Aufenthaltswahrscheinlichkeit für den Trajektorienpunkt.

Qualitativ dasselbe Resultat, aber mit glatterem Verlauf der Zeitserie, erhält man zweifellos, wenn man zu geringeren Abstufungen übergeht (Höhenabnahme beispielsweise nur 10% statt 50% pro Stufe). Die zugehörige kompliziertere Version von Gl. 10.1 kann ebenfalls erarbeitet werden, erfordert jedoch einen ungleich höheren Aufwand in allen Stufen ihrer Erstellung (wovon deshalb wie erwähnt abgesehen wurde).

Schon bei einer 10fach längeren Simulation erkennt man die morphologische Analogie zu den in den Teilen I bis III dieser Arbeit definierten und simulierten Flare-Zeitserien. Dies zeigt Abb. 10.4.

10.3 Poincaré-Rekurrenz bei der Staircase Baker's Map

Die Abb. 10.5 zeigt abschließend eine extrem lange Zeitseriensimulation der x-Variablen nach Gl. 10.1. Der erste Simulationsteil (das erste Zehntel der Darstellung) wurde dabei mit vergrößerter Fliesskommagenauigkeit mittels der GMP Library berechnet ($\approx$ 2600 Dezimalstellen entsprechend 8192 Bits)[‖]. Der Rest der Abbildung ist dann wieder mit Standard-Rechengenauigkeit (double) dargestellt, wobei das erste Zehntel in dieser Abbildung nicht dargestellt, jedoch gerechnet wurde.

[‖]Eine kürzere Zeitserie mit 2^{18} Bits, entsprechend ca. 260 000 Dezimalstellen, ist hier nicht dokumentiert, da sie keine höheren Peaks enthielt

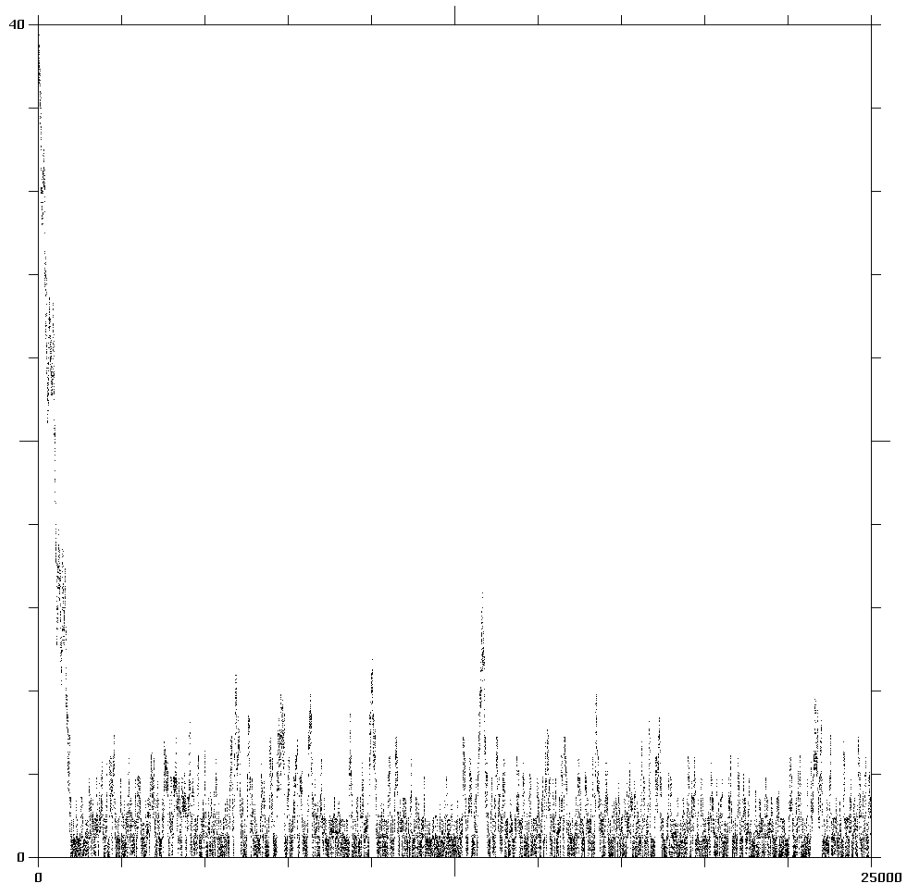


Abbildung 10.4: Simulation wie Abb. 10.3 mit zehnfacher Iterationsdauer (25 000 Iterationsschritte).

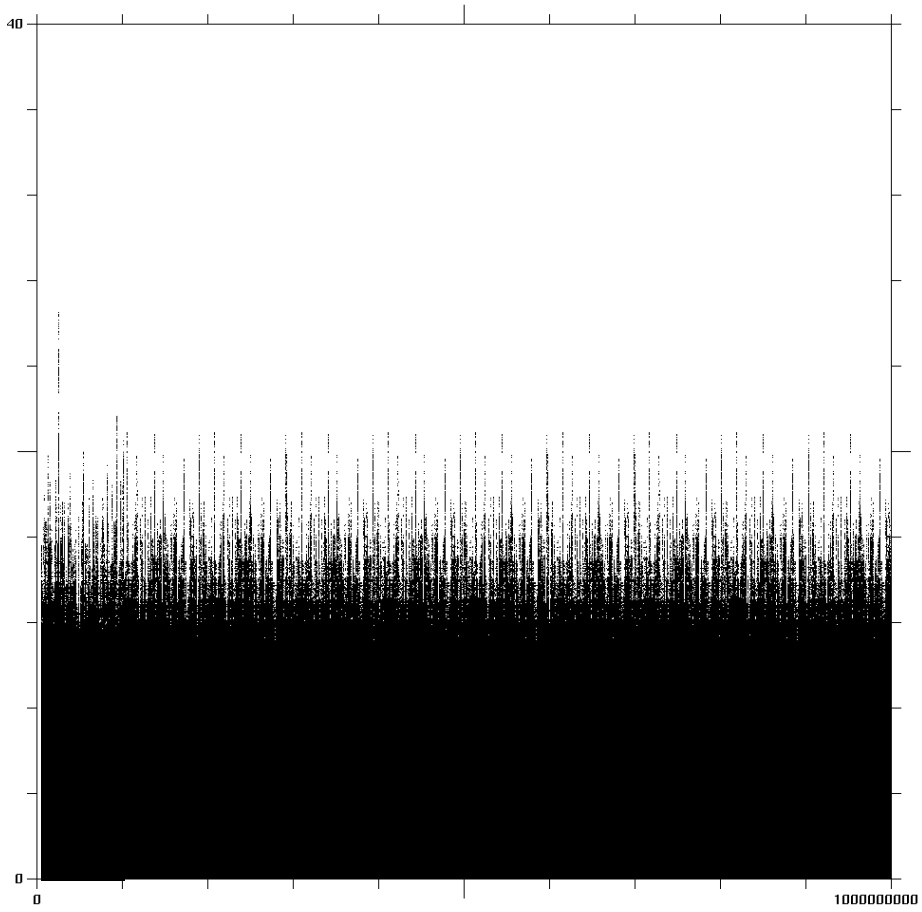


Abbildung 10.5: Hundert mal längere Simulation als in Abb. 10.4, dargestellt in 2 Stücken. Die ersten 10 Prozent (100 Millionen Zeitschritte) wurden mit 8192 Bit Dezimaldarstellung gerechnet, die 900 Millionen restlichen Zeitschritte wurden mit normaler double Genauigkeit gerechnet. Beim größeren zweiten Zeitintervall erkennt man deutlich eine Periodizität mit einer Periode von knapp 100 Millionen Iterationsschritten. Beim linken Zehntel (mit multipler Rechengenauigkeit) erkennt man einen sehr deutlich höheren Peak, der nicht gut Teil einer sich in 100 Millionen Iterationsschritten wiederholenden Sequenz sein kann (vgl. Text).

Der mit der erweiterten Genauigkeit gerechnete Teil der Abbildung offenbart ein neues Phänomen: Die bei Standard-Genauigkeit im Verlaufe der Zeit auftretende Periodizität tritt hier nicht auf! Man sieht, dass Zustände mit weit höherer x-Amplitude erreicht werden, die, wenn sie ebenfalls Teil einer Periodizität wären, eine wesentlich höhere Periode erforderlich machen würden. Die Frage, ob dieses Signal bei 100fach längerer Iterationszeit nun ebenfalls periodisch werden würde oder nicht, muss bis jetzt offen bleiben.

Eine solche periodische Eigenschaft dieses Flare-Zeitverhalten stellt eine gewisse Analogie zum sogenannten Poincaré-Rekurrenz-Phänomen dar, wie dies bei konservativen Vielteilchensystemen zuerst von Poincaré postuliert wurde, aber in Simulationen bisher nicht demonstriert werden konnte. Ein zum allerersten Teil der Abb. 10.3 passendes qualitatives Verhalten wurde für das Phasenraumvolumen von reversibel gerechneten, modellhaften Vielteilchensystemen (einer sogenannten molekulardynamischen Simulation) von Diebner et.al. untersucht [43].

10.4 Die Staircase Baker's Map als Multiple Baker's Map

Die Staircase Baker's Map stellt einen neuen Vertreter in der Reihe der sogenannten Multiple Baker's Abbildungen dar, wie sie von Hopf eingeführt worden sind. Dieser führte nicht nur eine semi-infinite Baker's Map ein, sondern bewies auch bereits die mathematische Eigenschaft des Mischens für diese Klasse von dynamischen Systemen. Weitere sogenannte „ergodische" Eigenschaften für bi-infinite Ketten von Baker's Maps wurden von Goldstein und Lebowitz im Jahr 1974 bewiesen [44].

Später rückten die Eigenschaften von multiplen Baker's Maps wieder in den Mittelpunkt der Untersuchung dynamischer Systeme durch die Beobachtung von sogenannten „fraktalen Eigenzuständen" durch Hasegawa und Tasaki [45][46].

Vor kurzem nun wurden inhomogene Varianten von Bäcker-Abbildungen (die Zellenhöhe variiert hier unregelmäßig als Funktion der horizontalen Position) in Verbindung mit sogenannten „disordered chaotic maps" untersucht [47][48].

Bei dem in dieser Arbeit vorgestellten Vertreter der multiplen Baker's Map schwankt die Zellengröße nicht unregelmäßig, sondern sie nimmt exponentiell in eine Richtung ab. Die Staircase Baker's Map stellt daher eine eigenständige Variante der von Radons definierten Kategorie der multiplen Bäcker-Abbildungen dar [49].

90 Staircase Baker's Map mit flarehaftem dynamischem Verhalten

Teil V

Diskussion

Diskussion

Die außerordentliche Komplexität des chemisch-physikalischen Verhaltens von Flammen ist lange bekannt. Heute weiß man, dass man chaotische Dynamiken in Flammen schon lange beobachten konnte, bevor das Chaos in der Belousov-Zhabotinskii-Reaktion entdeckt war. Nun sind die in dieser Arbeit geschilderten flare-haften Zeitserien allerdings keine chaotischen Zeitserien. Ihre Unberechbarkeit ist im Grunde noch weit größer, da sie analog dem namengebenden Naturschauspiel nahezu unvorhersagbar kommen und gehen, und dies mit extremen Amplitudenausschlägen. Das heißt, zu einer Flare-Dynamik gehört ein fast „unendlicher" Amplitudenbereich, der vollständig ausgefüllt wird, aber nicht (wie dies beim Chaos der Fall ist) mehr oder weniger gleichverteilt, sondern mit einer exponentiell abnehmenden Häufigkeit der großen Ausbrüche.

Es kann daher vielleicht nicht überrraschen, dass auf der Grundlage des Chaos und seiner Unregelmäßigkeit die besonderen Eigenschaften der Flare-Systeme erhalten werden können, wenn eine einzige weitere Variable ins Spiel gebracht wird. Wenn Chaos also ein typisch dreidimensionales dynamisches Verhalten ist (mit einem dreidimensionalem Zustandsraum, z.B. aus drei miteinander reagierenden Substanzen unter wohlgerührten Bedingungen), kommt bei den Flare-Systemen eine weitere Dimension hinzu, so dass das Flare-Verhalten als „typisch vierdimensionales dynamisches Verhalten" klassifiziert werden kann.

Während dreidimensionale dynamische Systeme in ihrem qualitativen dynamischen Verhalten dank der Jahrzehnte intensiver Untersuchung dyna-

mischer Systeme mit chaotischem Verhalten heute relativ wohlverstanden sind, sind dagegen vierdimensionale dynamische Systeme immer noch beinahe ein „Buch mit sieben Siegeln". Nur einige Inseln im unbekannten, wie Hyperchaos (maximales Chaos in vier Dimensionen — so wie bei zwei unabhängigen dreidimensionalen chaotischen Systemen, nur eben bereits in vier Dimensionen) und Kaplan-Yorke-Verhalten sind relativ gut untersucht. Bei dem letzteren Verhalten ist die vierte Variable, die im einfachsten Fall passiv angekoppelt sein darf, nur schwach gedämpft. Dadurch kumuliert sich der chaotische Einfluss in einem gewissen Sinn, so dass bei oberflächlicher Betrachtung ein Hyperchaos-ähnliches Verhalten resultiert. Die fraktale Dimension des Attraktors ist dadurch tatsächlich vom gleichen Typ wie beim (echten) Hyperchaos.

Das Flare-Verhalten ist, so man will, ein komplexerer Fall des Kaplan-Yorke-Verhaltens. Auch hier muss die vom Chaos angetriebene („geforcte") vierte Variable relativ schwach gedämpft sein. Das allein reicht, wie gesagt, bereits aus zur Erzeugung eines außerordentlich erratischen Verhaltens. Doch es kommt etwas Zweites hier hinzu: Je nach den besonderen momentanen Eigenschaften des antreibenden Chaos — also seinen zufälligen momentanen Eigenschaften — kann es passieren, dass die vierte Variable nun gar nicht gedämpft ist, sondern im Gegenteil sogar autokatalytisch anwächst.

Das chemische Reaktionssystem der Abbildung 5.3 im Kapitel 5 macht diese Tatsache besonders deutlich. Das Chaos läßt es bei bestimmten Zeitfolgen zu, dass das angetriebene chemische Subsystem autokatalytisch anwächst, und dies über längere Zeiträume hinweg. Bei anderen Zeitfolgen ist es gedämpft und schrumpft (negative Autokatalyse $\equiv$ Selbstinhibition). Je nach der Verteilung der einen oder der anderen Art von Antrieb (im chaotischen Ausgangssignal) kommt es in absolut unvorraussagbarer Weise zu extremem Wachstum oder zum völligen Zusammenbruch der angetriebenen vierten Variablen.

Numerische Untersuchungen nicht-linearer Gleichungssysteme besitzen heutzutage eine Art von Universalität in zahlreichen Forschungsgebieten, wel-

che zunächst nicht unmittelbar verwandt zu sein scheinen; so ergab es sich in diesem Zusammenhang auch, dass die im Rahmen dieser Arbeit beschriebene dynamischen Modelle mit Flare-Verhalten von Barkley Rosser an der James Madison University Harrisburg, U.S.A., für die Simulation von kleinen ökonomischen Stockmarket-Modellsystemen eingesetzt werden konnte [50].

Wie in Kapitel 3.1 geschildert, wurde das Flare-Verhalten empirisch entdeckt, nachdem ein sehr instabiler Attraktor vom Milnor-Typ von Sommerer, Ott und Yorke als ein interessantes neues dynamisches Phänomen in die Debatte geworfen worden war (und Chico Doria so freundlich war, uns auf diese Arbeit aufmerksam zu machen). Durch „Simplifizierung" und Abwandlung dieses Attraktortyps entstand überraschend der erste Flare-Attraktor. Dazu genügte es, das bis ins Unendliche reichende maximale Wachstum dieses Attraktors zu begrenzen, also — chemisch gesprochen — die bei jeder Autokatalyse unvermeidliche quadratische Rückreaktion als (wenigstens infinitesimalen) „begrenzenden Abflussterm" einzubauen. Denn „irreversible" homogene Reaktionen gibt es bekanntlich nicht. Dieser „Tropfen Wermut im Wasser" der Milnor-Attraktoren gebar sozusagen zwangsweise die robuste Klasse der Flare-Attraktoren. Sie repräsentieren damit einen neuen Typus von Intermittenz, der in realistischen Dynamischen Systemen auftreten kann — z.B. in gut gerührten Reaktionssytemen von Typ der Zhabotinskii-Reaktion. Eine weitere Bedeutung dieser Flare-Systeme besteht in ihrem möglichen Einsatz als ein einfacher Detektor für Unregelmäßigkeit (Stochastizität) in ansonsten überwiegend periodischen (bzw. quasiperiodischen) Zeitserien. Hierin läge wohl zunächst die wahrscheinlichste Implikation der Flare-Systeme für eine technisch-experimentelle Anwendung.

Anhang

Anhang A

Source Codes

In den folgenden Abschnitten finden sich exemplarisch für alle Simulationsmodelle die verschiedenen Grund-Computerprogrammtypen, welche für die spezifischen Simulationen in der Regel nur in den Zeilen für die numerischen Berechnungen jeweils auf das zu erstellende Modell angepaßt wurden.

Alle Listings umfassen sowohl den numerischen Algorithmus als auch die Realisierung der zur graphischen Ausgabe nötigen Programmierung durch Graphik-Routinen der hier verwendeten GL/OpenGL-Graphikbibliothek. Alle Programme sind in ANSI-C geschrieben und wurden mittels SGI-C und GNU-C Compilern auf einer SGI Workstation erstellt.

A.1 Simulationsprogramm für Iterations-Maps

Das nachfolgende abgedruckte ANSI-C Programm wurde zur Simulation aller nicht-gekoppelten Iterations-Maps angewandt; es erlaubt die Darstellung der errechneten Iterationspunkte während des Programmlaufs. Dies ist je nach Aufruf des Programm in der Phasenraumdarstellung zweier Variablen gegeneinander oder in einer „live" sich immer weiter verlängerden Zeitreihendarstellung einer Variablen möglich (im abgedruckten Beispiel wird ein quasiperiodisches Forcing eingesetzt):

```
/****************************** quasi2.c ***********************************
*****************************************************************************/

#include <stdio.h>
#include <gl/gl.h>
#include <stdlib.h>
#include <device.h>
#include <time.h>
#include <math.h>
#include <ctype.h>
#include <unistd.h>

#define XM modf((x + 0.5*(sqrt(5.0)-1.0)),&int_q)
#define XS1 "x =(x + 0.5*"
#define XS2 "*(sqrt(5.0)-1.0)) mod 1"
#define YM
#define YS1 ""
#define YS2 ""
#define BM b + b*(0.5-x) + 0.01 - 0.0001*b*b
#define BS1 "b = b + b*(0.5-x) + 0.01"
#define BS2 "    - 0.0001*b*b"

#define START_X 0.1
#define START_B 0.1

#define SCALE 90
#define ANZK 900
#define DIGIT 100000

#define UX1 302
#define UY1 57
#define UX2 1202
#define UY2 957

/* wenn definiert, werden Datenpunkte in Zeitreihen miteinander
```

```
verbunden */
#define ZUG

main  (int argc, char *argv[])
{
        if (argc != 9) {
                printf("Falsche Anzahl an Argumenten!!!!\n");
                printf("Korrekter Aufruf: logi0 <Minit> <Maxit> <Right>
<Right_Min> <Right_Max>
                  <Up> <Up_Min> <Up_Max>\n");
                return -1;
        }

        initialize(argv[2]);
        drawrast(argv[1],argv[2],argv[3],argv[4],argv[5],argv[6],argv[7],argv[8]);
        eval_pxl(argv[1],argv[2],argv[3],argv[4],argv[5],argv[6],argv[7],argv[8]);
        while(TRUE)
        {
                checkinput();
        }
}

initialize(char *siter)
{
        char wint[70];
        sprintf(wint,"Ott-Yorke-Attraktor bei %s Iterationsschritten --- Zeitentwicklung",siter);
        prefposition(28, 1272, 8, 992);
        winopen("Attraktor");
        wintitle(wint);
}

checkinput()
{
        short val;
        switch(qread(&val))
        {
        case REDRAW:
                reshapeviewport();
                break;
        }
}

drawrast(char *sitmin, char *sitmax, char *right, char *shorimi, char *shorima,
char *up, char *svertmi, char *svertma)
{
        Icoord m,n;
        int i;
        char wistr[35];
        time_t t;
```

```
struct tm *local;


color(WHITE);
clear();
color(BLACK);
rect(UX1,UY1,UX2,UY2);

m = UX1 + SCALE;
n = UY1;
while(n <= UY2 + 10)
{
        for(i=0; i<9; i++)
        {
                move2i(m,n-10);
                draw2i(m,n);
                m = m + SCALE;
        }
        m = UX1 + SCALE;
        n = n + (UY2 - UY1 + 10);
}
move2((UX2-UX1)/2+UX1,UY1-20);
draw2((UX2-UX1)/2+UX1,UY1-10);
move2((UX2-UX1)/2+UX1,UY2+10);
draw2((UX2-UX1)/2+UX1,UY2+20);

move2(UX1,UY1-10);
draw2(UX1,UY1);
move2(UX2,UY2);
draw2(UX2,UY2+10);

move2(UX1,UY2+10);
draw2(UX1,UY2);
move2(UX2,UY1);
draw2(UX2,UY1-10);

m = UX1;
n = UY1 + SCALE;
while(m <= UX2 + 10)
{
        for(i=0; i<9; i++)
        {
                move2(m-10,n);
                draw2(m,n);
                n = n + SCALE;
        }
        n = UY1 + SCALE;
        m = m + (UX2 - UX1 + 10);
}
```

```
        move2(UX1-20,5*SCALE+UY1);
        draw2(UX1-10,5*SCALE+UY1);
        move2(UX2+10,5*SCALE+UY1);
        draw2(UX2+20,5*SCALE+UY1);

        move2(UX1-10,UY1);
        draw2(UX1,UY1);
        move2(UX2,UY2);
        draw2(UX2+10,UY2);

        move2(UX2+10,UY1);
        draw2(UX2,UY1);
        move2(UX1,UY2);
        draw2(UX1-10,UY2);

        cmov2(UX1-3,UY1-26);
        charstr(shorimi);
        cmov2(UX2-3,UY1-26);
        charstr(shorima);

        cmov2(UX1-strlen(svertmi)*9-13,UY1-4);
        charstr(svertmi);
        cmov2(UX1-strlen(svertma)*9-13,UY2-4);
        charstr(svertma);

        cmov2(15, 37);
        charstr(__FILE__);
/*
        cmov2((UX1 + 45), 12);
        sprintf(wistr,"Iterationsschritte insgesamt: it = %s",sitmax);
        charstr(wistr);
*/
        time(&t);
        local=localtime(&t);
        strftime(wistr,25,"%d.%m.%y  %H:%M:%S",local);
        cmov2(15,12);
        charstr(wistr);
#ifdef CUTSTART
        cmov2(58, UY1 + 200);
        sprintf(wistr,"X_CUT(MIN) = %.4f",CUTSTART);
        charstr(wistr);
        cmov2(58, UY1 + 170);
        sprintf(wistr,"X_CUT(MAX) = %.4f",CUTSTOP);
        charstr(wistr);
#endif
        cmov2(83, UY1 + 120);
        sprintf(wistr,"x_0 = %.2f",START_X);
        charstr(wistr);
```

```
        cmov2(83, UY1 + 90);
        sprintf(wistr,"b_0 = %.2f",START_B);
        charstr(wistr);
/*
        cmov2(15, UY1 + 765);
        charstr(XS1);
        rect(5, UY1 + 628, strlen(BS1)*10-5, UY1 + 790);
        cmov2(15, UY1 + 750);
        charstr(XS2);

        cmov2(15, UY1 + 710);
        sprintf(wistr,BS1);
        charstr(wistr);

        cmov2(15, UY1 + 695);
        sprintf(wistr,YS2);
        charstr(wistr);

        cmov2(15, UY1 + 655);
        charstr(ZS1);
        cmov2(15, UY1 + 640);
        charstr(ZS2);
*/
        move2(63,UY1+315);
        draw2(63,UY1+405);
        move2(58,UY1+400);
        draw2(63,UY1+405);
        move2(68,UY1+400);
        draw2(63,UY1+405);
        cmov2(43,UY1+355);
        charstr(up);

        move2(78,UY1+300);
        draw2(168,UY1+300);
        move2(163,UY1+305);
        draw2(168,UY1+300);
        move2(163,UY1+295);
        draw2(168,UY1+300);
        cmov2(118,UY1+280);
        charstr(right);
}


eval_pxl(char *sitmin, char *sitmax, char *right, char *shorimi, char *shorima,
char *up, char *svertmi, char *svertma)
{
        int minit, maxit;
        double x, b, ka, xs, bs, hori_min, hori_max, vert_min, vert_max, nach_hori, nach_vert;
        double *hori, *vert;
```

```
double int_q, skal_hori, skal_vert;
register i,k;
register short phori, pvert;
char wistr[30] = "0";



minit = atoi(sitmin);
maxit = atoi(sitmax);
hori_min = atof(shorimi);
hori_max = atof(shorima);
vert_min = atof(svertmi);
vert_max = atof(svertma);
x = START_X;
b = START_B;
color(WHITE);
rectfi(UX1+1,UY1+1,UX2-1,UY2-1);
color(BLACK);

switch(right[0])
{
        case 'X':
                hori = &x;
                break;
        case 'B':
                hori = &b;
                break;
        case 'k':
                hori = &ka;
                break;
        default:
                printf("Ung\"ultige Angabe der x-Achsenvariable!!\n");
}

switch(up[0])
{
        case 'X':
                vert = &x;
                break;
        case 'B':
                vert = &b;
                break;
        case 'k':
                hori = &ka;
                break;
        default:
                printf("Ung\"ultige Angabe der y-Achsenvariable!!\n");
}
```

```
        for(k=0;k<=maxit;k++)
          {
            xs = XM;
            bs = BM;
            ka = k;

            x = xs;
            b = bs;
            /*   for(i=0;i<50000;i++) */
              ;

            if (k < minit)
                  continue;
            if (!(k%DIGIT))
              {
                cmov2(UX1+495,12);
                color(WHITE);
                charstr(wistr);
                color(RED);
                cmov2(UX1+495,12);
                sprintf(wistr,"Iterationsschritte momentan: it = %d",k);
                charstr(wistr);
                color(BLACK);
              }
#ifdef CUTSTART
            if (x < CUTSTART)
                  continue;
            if (x > CUTSTOP)
                  continue;
#endif
#ifdef EXPLO
               if (k >= EXPLO)
                  printf("%.1f %.1f %.1f\n",x,b,z);
#endif
#ifdef LIMIT
               if (abs(y)>LIMIT)
                  {
                    cmov2(30,550);
                    sprintf(wistr,"(%.1e/%.1e/%.1e)",x,b);
                    charstr(wistr);
                    cmov2(100,500);
                    sprintf(wistr,"n = %d",k);
                    charstr(wistr);
                    break;
                  }
#endif
if (*hori >= hori_min && *hori <= hori_max && *vert >= vert_min && *vert <= vert_max)
                {
```

```
                        nach_vert = modf(900 * ((*vert - vert_min)/(vert_max - vert_min)),&skal_vert);
                        nach_hori = modf(900 * ((*hori - hori_min)/(hori_max - hori_min)),&skal_hori);
                        pvert = skal_vert;
                        phori = skal_hori;
                        if(nach_vert >= 0.5)
                                pvert++;
                        if(nach_hori >= 0.5)
                                phori++;
#ifdef ZUG
                        if (*hori == ka) {
                          if(k == minit)
                            move2(UX1,UY1);
                          draw2(UX1+phori,UY1+pvert);
                        }
                        else
#endif
                          pnt2(UX1+phori,UY1+pvert);
                }
#ifdef DATOUT
                printf("%.16f  %.16f  %.16f  %d\n",x,b,k);
#endif


        }
        color(WHITE);
        cmov2(UX1+495,12);
        charstr(wistr);
}
```

A.2 Gekoppelte Flare-Systeme

Mit diesem ANSI-C Programm wurden die gekoppelten Reaktions-Diffusions-Modelle des Kapitels 8 simuliert; es ist wie immer eine simultane Betrachtung der errechneten Punkte möglich:

```
/************************** ensembl8.c ***********************************
*************************************************************************/

#include <stdio.h>
#include <gl/gl.h>
#include <stdlib.h>
#include <device.h>
#include <time.h>
#include <ctype.h>
```

```
#include <unistd.h>

#define SCALE 90
#define ANZK 900
#define DIGIT 100000

#define UX1 302
#define UY1 57
#define UX2 1202
#define UY2 957

/* wenn definiert, werden Datenpunkte in Zeitreihen miteinander
verbunden */
#define ZUG

main  (int argc, char *argv[])
{
        if (argc != 9) {
                printf("Falsche Anzahl an Argumenten!!!!\n");
                printf("Korrekter Aufruf: ensembl4 <Minit> <Maxit> <Right>
<Right_Min> <Right_Max>
                  <Up> <Up_Min> <Up_Max>\n");
                return -1;
        }

        initialize(argv[2]);
        drawrast(argv[1],argv[2],argv[3],argv[4],argv[5],argv[6],argv[7],argv[8]);
        eval_pxl(argv[1],argv[2],argv[3],argv[4],argv[5],argv[6],argv[7],argv[8]);
        while(TRUE)
        {
                checkinput();
        }
}

initialize(char *siter)
{
        char wint[70];
        sprintf(wint,"Flare-Attraktor bei %s Iterationsschritten --- Zeitentwicklung",siter);
        prefposition(28, 1272, 8, 992);
        winopen("Attraktor");
        wintitle(wint);
}

checkinput()
{
        short val;
        switch(qread(&val))
        {
        case REDRAW:
```

```
                reshapeviewport();
                break;
        }
}

drawrast(char *sitmin, char *sitmax, char *right, char *shorimi, char *shorima,
char *up, char *svertmi, char *svertma)
{
        Icoord m,n;
        int i;
        char wistr[35];
        time_t t;
        struct tm *local;


        color(WHITE);
        clear();
        color(BLACK);

        m = UX1 + SCALE;
        n = UY1;
        while(n <= UY2 + 10)
        {
                for(i=0; i<9; i++)
                {
                        move2i(m,n-10);
                        draw2i(m,n);
                        m = m + SCALE;
                }
                m = UX1 + SCALE;
                n = n + (UY2 - UY1 + 10);
        }
        move2((UX2-UX1)/2+UX1,UY1-20);
        draw2((UX2-UX1)/2+UX1,UY1-10);
        move2((UX2-UX1)/2+UX1,UY2+10);
        draw2((UX2-UX1)/2+UX1,UY2+20);

        move2(UX1,UY1-10);
        draw2(UX1,UY1);
        move2(UX2,UY2);
        draw2(UX2,UY2+10);

        move2(UX1,UY2+10);
        draw2(UX1,UY2);
        move2(UX2,UY1);
        draw2(UX2,UY1-10);

        m = UX1;
        n = UY1 + SCALE;
```

```
        while(m <= UX2 + 10)
        {
                for(i=0; i<9; i++)
                {
                        move2(m-10,n);
                        draw2(m,n);
                        n = n + SCALE;
                }
                n = UY1 + SCALE;
                m = m + (UX2 - UX1 + 10);
        }
/*      move2(UX1-20,5*SCALE+UY1);
        draw2(UX1-10,5*SCALE+UY1);
        move2(UX2+10,5*SCALE+UY1);
        draw2(UX2+20,5*SCALE+UY1);
*/
        move2(UX1-10,UY1);
        draw2(UX1,UY1);
        move2(UX2,UY2);
        draw2(UX2+10,UY2);

        move2(UX2+10,UY1);
        draw2(UX2,UY1);
        move2(UX1,UY2);
        draw2(UX1-10,UY2);

        cmov2(UX1-3,UY1-26);
        charstr(shorimi);
        cmov2(UX2-3,UY1-26);
        charstr(shorima);

        cmov2(UX1-strlen(svertmi)*9-13,UY1-4);
        charstr(svertmi);
        cmov2(UX1-strlen(svertma)*9-13,UY2-4);
        charstr(svertma);

        cmov2(15, 37);
        charstr(__FILE__);
/*
        cmov2((UX1 + 45), 12);
        sprintf(wistr,"Iterationsschritte insgesamt: it = %s",sitmax);
        charstr(wistr);
*/
        time(&t);
        local=localtime(&t);
        strftime(wistr,25,"%d.%m.%y  %H:%M:%S",local);
        cmov2(15,12);
        charstr(wistr);
```

```
        move2(63,UY1+315);
        draw2(63,UY1+405);
        move2(58,UY1+400);
        draw2(63,UY1+405);
        move2(68,UY1+400);
        draw2(63,UY1+405);
        cmov2(43,UY1+355);
        charstr(up);

        move2(78,UY1+300);
        draw2(168,UY1+300);
        move2(163,UY1+305);
        draw2(168,UY1+300);
        move2(163,UY1+295);
        draw2(168,UY1+300);
        cmov2(118,UY1+280);
        charstr(right);
}


eval_pxl(char *sitmin, char *sitmax, char *right, char *shorimi, char *shorima,
char *up, char *svertmi, char *svertma)
{
        long unsigned int minit, maxit;
        double x, y, z, w, a, b, ka;
        double a1, b1, a2, b2,a3, b3,a4, b4,a5, b5,a6, b6,a7, b7,a8, b8,a9,
b9,a10, b10,a11,
          b11,a12, b12,a13, b13,a14, b14,a15, b15;
        double xs, ys, zs, ws, as, bs;
        double a1s, b1s, a2s, b2s,a3s, b3s,a4s, b4s,a5s, b5s,a6s, b6s,a7s,
b7s,a8s, b8s,a9s,
          b9s,a10s, b10s,a11s, b11s,a12s, b12s,a13s, b13s,a14s, b14s,a15s, b15s;
        double hori_min, hori_max, vert_min, vert_max, nach_hori, nach_vert;
        double *hori, *vert;
        double int_q, skal_hori, skal_vert;
        double m;
        register i,k;
        register short phori, pvert;
        char wistr[30] = "0";



        minit = atoi(sitmin);
        maxit = atoi(sitmax);
        hori_min = atof(shorimi);
        hori_max = atof(shorima);
        vert_min = atof(svertmi);
        vert_max = atof(svertma);
```

```
x = 0.01;
z = 0.2;
y = 0.011;
w = 0.2;
a = 0.012;
b = 0.2;
a1 = 0.013  ;
b1 = 0.2;
a2 = 0.014  ;
b2 = 0.2;
a3 = 0.015  ;
b3 = 0.2;
a4 = 0.016  ;
b4 = 0.2;
a5 = 0.017  ;
b5 = 0.2;
a6 = 0.018  ;
b6 = 0.2;
a7 = 0.019  ;
b7 = 0.2;
a8 = 0.020  ;
b8 = 0.2;
a9 = 0.021  ;
b9 = 0.2;
a10 = 0.022  ;
b10 = 0.2;
a11 = 0.023  ;
b11 = 0.2;
a12 = 0.024  ;
b12 = 0.2;
a13 = 0.025  ;
b13 = 0.2;
a14 = 0.026  ;
b14 = 0.2;
a15 = 0.027  ;
b15 = 0.2;

m = 0.0;


color(WHITE);
rectfi(UX1+1,UY1+1,UX2-1,UY2-1);
color(BLACK);

switch(right[0])
{
        case 'X':
                hori = &x;
```

```
        break;
case 'Y':
        hori = &y;
        break;
case 'Z':
        hori = &z;
        break;
case 'W':
        hori = &w;
        break;
case 'A':
        hori = &a;
        break;
case 'B':
        hori = &b;
        break;
        /*
case 'A1':
        hori = &a1;
        break;
case 'B1':
        hori = &b1;
        break;
case 'A2':
        hori = &a2;
        break;
case 'B2':
        hori = &b2;
        break;
case 'A3':
        hori = &a3;
        break;
case 'B3':
        hori = &b3;
        break;
case 'A4':
        hori = &a4;
        break;
case 'B4':
        hori = &b4;
        break;
case 'A5':
        hori = &a5;
        break;
case 'B5':
        hori = &b5;
        break;
case 'A6':
        hori = &a6;
```

```
        break;
case 'B6':
        hori = &b6;
        break;
case 'A7':
        hori = &a7;
        break;
case 'B7':
        hori = &b7;
        break;
case 'A8':
        hori = &a8;
        break;
case 'B8':
        hori = &b8;
        break;
case 'A9':
        hori = &a9;
        break;
case 'B9':
        hori = &b9;
        break;
case 'A10':
        hori = &a10;
        break;
case 'B10':
        hori = &b10;
        break;
case 'A11':
        hori = &a11;
        break;
case 'B11':
        hori = &b11;
        break;
case 'A12':
        hori = &a12;
        break;
case 'B12':
        hori = &b12;
        break;
case 'A13':
        hori = &a13;
        break;
case 'B13':
        hori = &b13;
        break;
case 'A14':
        hori = &a14;
        break;
```

```
        case 'B14':
                hori = &b14;
                break;
        case 'A15':
                hori = &a15;
                break;
        case 'B15':
                hori = &b15;
                break; */
        case 'M':
                hori = &m;
                break;
        case 'k':
                hori = &ka;
                break;
        default:
                printf("Ung\"ultige Angabe der x-Achsenvariable!!\n");
}

switch(up[0])
{
        case 'X':
                vert = &x;
                break;
        case 'Y':
                vert = &y;
                break;
        case 'Z':
                vert = &z;
                break;
        case 'W':
                vert = &w;
                break;
        case 'A':
                vert = &a;
                break;
        case 'B':
                vert = &b;
                break;
        case 'M':
                vert = &m;
                break;
                /*
        case 'B1':
                vert = &b1;
                break;
        case 'A2':
                vert = &a2;
                break;
```

```
case 'B2':
        vert = &b2;
        break;
case 'A3':
        vert = &a3;
        break;
case 'B3':
        vert = &b3;
        break;
case 'A4':
        vert = &a4;
        break;
case 'B4':
        vert = &b4;
        break;
case 'A5':
        vert = &a5;
        break;
case 'B5':
        vert = &b5;
        break;
case 'A6':
        vert = &a6;
        break;
case 'B6':
        vert = &b6;
        break;
case 'A7':
        vert = &a7;
        break;
case 'B7':
        vert = &b7;
        break;
case 'A8':
        vert = &a8;
        break;
case 'B8':
        vert = &b8;
        break;
case 'A9':
        vert = &a9;
        break;
case 'B9':
        vert = &b9;
        break;
case 'A10':
        vert = &a10;
        break;
case 'B10':
```

```
                vert = &b10;
                break;
        case 'A11':
                vert = &a11;
                break;
        case 'B11':
                vert = &b11;
                break;
        case 'A12':
                vert = &a12;
                break;
        case 'B12':
                vert = &b12;
                break;
        case 'A13':
                vert = &a13;
                break;
        case 'B13':
                vert = &b13;
                break;
        case 'A14':
                vert = &a14;
                break;
        case 'B14':
                vert = &b14;
                break;
        case 'A15':
                vert = &a15;
                break;
        case 'B15':
                vert = &b15;
                break;     */
        default:
                printf("Ung\"ultige Angabe der y-Achsenvariable!!\n");
}


for(k=0;k<=maxit;k++)
  {

    xs = 3.9999*x*(1.0-x);
    zs = z + z*(0.565-x) - 0.001*z*z + 0.001*m ;
    ys = 3.9999*y*(1.0-y);
    ws = w + w*(0.566-y) - 0.001*w*w + 0.001*m ;
    as = 3.9999*a*(1.0-a);
    bs = b + b*(0.567-a) - 0.001*b*b + 0.001*m ;
    a1s = 3.9999*a1*(1.0-a1);
    b1s = b1 + b1*(0.568-a1) - 0.001*b1*b1 + 0.001*m ;
    a2s = 3.9999*a2*(1.0-a2);
```

```
            b2s = b2 + b2*(0.569-a2) - 0.001*b2*b2 + 0.001*m ;
            a3s = 3.9999*a3*(1.0-a3);
            b3s = b3 + b3*(0.570-a3) - 0.001*b3*b3 + 0.001*m ;
            a4s = 3.9999*a4*(1.0-a4);
            b4s = b4 + b4*(0.571-a4) - 0.001*b4*b4 + 0.001*m ;
            a5s = 3.9999*a5*(1.0-a5);
            b5s = b5 + b5*(0.572-a5) - 0.001*b5*b5 + 0.001*m ;
            a6s = 3.9999*a6*(1.0-a6);
            b6s = b6 + b6*(0.573-a6) - 0.001*b6*b6 + 0.001*m ;
            a7s = 3.9999*a7*(1.0-a7);
            b7s = b7 + b7*(0.574-a7) - 0.001*b7*b7 + 0.001*m ;
            a8s = 3.9999*a8*(1.0-a8);
            b8s = b8 + b8*(0.575-a8) - 0.001*b8*b8 + 0.001*m ;
            a9s = 3.9999*a9*(1.0-a9);
            b9s = b9 + b9*(0.576-a9) - 0.001*b9*b9 + 0.001*m ;
            a10s = 3.9999*a10*(1.0-a10);
            b10s = b10 + b10*(0.577-a10) - 0.001*b10*b10 + 0.001*m ;
            a11s = 3.9999*a11*(1.0-a11);
            b11s = b11 + b11*(0.578-a11) - 0.001*b11*b11 + 0.001*m ;
            a12s = 3.9999*a12*(1.0-a12);
            b12s = b12 + b12*(0.579-a12) - 0.001*b12*b12 + 0.001*m ;
            a13s = 3.9999*a13*(1.0-a13);
            b13s = b13 + b13*(0.580-a13) - 0.001*b13*b13 + 0.001*m ;
            a14s = 3.9999*a14*(1.0-a14);
            b14s = b14 + b14*(0.581-a14) - 0.001*b14*b14 + 0.001*m ;
            a15s = 3.9999*a15*(1.0-a15);
            b15s = b15 + b15*(0.582-a15) - 0.001*b15*b15 + 0.001*m ;

            m =  m + z + w + b + b1 + b2 + b3 + b4 + b5 + b6 + b7 + b8 + b9 +
b10
                 + b11 + b12 + b13 + b14 + b15 - 0.2*m;

/*            m = z + w + b + b1 + b2 + b3; */
            ka = k;

            x = xs;
            y = ys;
            z = zs;
            w = ws;
            a = as;
            b = bs;
            a1 = a1s;
            b1 = b1s;
            a2 = a2s;
            b2 = b2s;
            a3 = a3s;
            b3 = b3s;
            a4 = a4s;
            b4 = b4s;
```

```
a5 = a5s;
b5 = b5s;
a6 = a6s;
b6 = b6s;
a7 = a7s;
b7 = b7s;
a8 = a8s;
b8 = b8s;
a9 = a9s;
b9 = b9s;
a10 = a10s;
b10 = b10s;
a11 = a11s;
b11 = b11s;
a12 = a12s;
b12 = b12s;
a13 = a13s;
b13 = b13s;
a14 = a14s;
b14 = b14s;
a15 = a15s;
b15 = b15s;


/*   for(i=0;i<50000;i++) Zeitlupenschleife */
  ;

if (k < minit)
      continue;
if (!(k%DIGIT))
  {
    cmov2(UX1+495,12);
    color(WHITE);
    charstr(wistr);
    color(RED);
    cmov2(UX1+495,12);
    sprintf(wistr,"Iterationsschritte momentan: it = %d",k);
    charstr(wistr);
    color(BLACK);
  }
if (*hori >= hori_min && *hori <= hori_max && *vert >= vert_min && *vert <= vert_max)
    {
            nach_vert = modf(450 * ((*vert - vert_min)/(vert_max - vert_min)),&skal_vert);
            nach_hori = modf(900 * ((*hori - hori_min)/(hori_max - hori_min)),&skal_hori);
            pvert = skal_vert;
            phori = skal_hori;
            if(nach_vert >= 0.5)
                    pvert++;
            if(nach_hori >= 0.5)
```

```
                                phori++;
#ifdef ZUG
                        if (*hori == ka) {
                          if(k == minit)
                            move2(UX1,UY1);
                          draw2(UX1+phori,UY1+pvert);
                        }
                        else
#endif
                          pnt2(UX1+phori,UY1+pvert);
                }

        }
        color(WHITE);
        cmov2(UX1+495,12);
        charstr(wistr);
        color(BLACK);
        move2(UX1,UY1+5*SCALE);
        draw2(UX1-10,UY1+5*SCALE);
        move2(UX2,UY1+5*SCALE);
        draw2(UX2+10,UY1+5*SCALE);
        cmov2(UX1-strlen(svertma)*9-13,UY1+5*SCALE-4);
        charstr(svertma);
        rect(UX1,UY1,UX2,UY1+5*SCALE);
}
```

A.3 Staircase Baker's Map mit multiplem Präzisions-Algorithmus

Hier das vollständige ANSI-C Programm mit der Erweiterung durch die GMP-Routinen:

```
/********************** mpshoe2.c ***********************************
*************************************************************************
******* Kompilieren mit gcc mpshoe2.c -lgl_s -lX11_s -lm -lgmp ***************/
```

```
#include <stdio.h>
#include <gl/gl.h>
#include <stdlib.h>
#include <device.h>
#include <time.h>
#include <ctype.h>
#include <math.h>
#include <gmp.h>

#define BOXZAHL 45
#define SW

#define START_X 39.9
#define START_Y 0.0
#define START_Z 0.1

/* #define CUTSTART 0.100
#define CUTSTOP 0.110

#define XM1 2.0*x - 1.000000000000001
#define XS1 "x = 2.0*x - 1"
#define XM2 2.0*x
#define XS2 "x = 2.0*x"
#define YM1 0.5*y+0.5
#define YS1 "y = 0.5*y + 0.5"
#define YM2 0.5*y
#define YS2 "y = 0.5*y"
#define ZM 0.1*y + 0.99*z
#define ZS1 "z = 0.1*y + 0.99*z"
#define ZS2 ""
*/


/* #define LIMIT 1000 */

#define SCALE 90
#define ANZK 900
#define DIGIT 100000
#define PRAEZ 30

#define UX1 302
#define UY1 57
#define UX2 1202
#define UY2 957

/* wenn definiert, werden Datenpunkte in Zeitreihen miteinander
verbunden */
/* #define ZUG */
```

```
main  (int argc, char *argv[])
{
        if (argc != 9) {
                printf("Falsche Anzahl an Argumenten!!!!\n");
                printf("Korrekter Aufruf: shoea0 <Minit> <Maxit> <Right>
<Right_Min> <Right_Max>
          <Up> <Up_Min> <Up_Max>\n");
                return -1;
        }

        initialize(argv[2]);
        drawrast(argv[1],argv[2],argv[3],argv[4],argv[5],argv[6],argv[7],argv[8]);
        eval_pxl(argv[1],argv[2],argv[3],argv[4],argv[5],argv[6],argv[7],argv[8]);
        while(TRUE)
        {
                checkinput();
        }
}

initialize(char *siter)
{
        char wint[70];
        sprintf(wint,"Ott-Yorke-Attraktor bei %s Iterationsschritten --- Zeitentwicklung",siter);
        prefposition(28, 1272, 8, 992);
        winopen("Attraktor");
        wintitle(wint);
}

checkinput()
{
        short val;
        switch(qread(&val))
        {
        case REDRAW:
                reshapeviewport();
                break;
        }
}

drawrast(char *sitmin, char *sitmax, char *right, char *shorimi, char *shorima,
char *up, char *svertmi, char *svertma)
{
        Icoord m,n;
        int i;
        char wistr[35];
        time_t t;
        struct tm *local;

#ifdef SW
```

```
        color(WHITE);
#else
        color(CYAN);
#endif
        clear();
        color(BLACK);

        m = UX1 + SCALE;
        n = UY1;
        while(n <= UY2 + 10)
        {
                for(i=0; i<9; i++)
                {
                        move2i(m,n-10);
                        draw2i(m,n);
                        m = m + SCALE;
                }
                m = UX1 + SCALE;
                n = n + (UY2 - UY1 + 10);
        }
        move2((UX2-UX1)/2+UX1,UY1-20);
        draw2((UX2-UX1)/2+UX1,UY1-10);
        move2((UX2-UX1)/2+UX1,UY2+10);
        draw2((UX2-UX1)/2+UX1,UY2+20);

        move2(UX1,UY1-10);
        draw2(UX1,UY1);
        move2(UX2,UY2);
        draw2(UX2,UY2+10);

        move2(UX1,UY2+10);
        draw2(UX1,UY2);
        move2(UX2,UY1);
        draw2(UX2,UY1-10);

        m = UX1;
        n = UY1 + SCALE;
        while(m <= UX2 + 10)
        {
                for(i=0; i<9; i++)
                {
                        move2(m-10,n);
                        draw2(m,n);
                        n = n + SCALE;
                }
                n = UY1 + SCALE;
                m = m + (UX2 - UX1 + 10);
        }
        move2(UX1-20,5*SCALE+UY1);
```

```
draw2(UX1-10,5*SCALE+UY1);
move2(UX2+10,5*SCALE+UY1);
draw2(UX2+20,5*SCALE+UY1);

move2(UX1-10,UY1);
draw2(UX1,UY1);
move2(UX2,UY2);
draw2(UX2+10,UY2);

move2(UX2+10,UY1);
draw2(UX2,UY1);
move2(UX1,UY2);
draw2(UX1-10,UY2);

cmov2(UX1-3,UY1-26);
charstr(shorimi);
cmov2(UX2-50,UY1-26);

/*      cmov2(UX2-10,UY1-26);   */
charstr("100000000");
/*        charstr(shorima);
 */
cmov2(UX1-strlen(svertmi)*9-13,UY1-4);
charstr(svertmi);
cmov2(UX1-strlen(svertma)*9-13,UY2-4);
charstr(svertma);

cmov2(15, 37);
charstr(__FILE__);

cmov2((UX1 + 45), 12);
sprintf(wistr,"Iterationsschritte insgesamt: it = %s",sitmax);
charstr(wistr);

time(&t);
local=localtime(&t);
strftime(wistr,25,"%d.%m.%y  %H:%M:%S",local);
cmov2(15,12);
charstr(wistr);

cmov2(83, UY1 + 120);
sprintf(wistr,"X_0 = %.7f",START_X);
charstr(wistr);

cmov2(83, UY1 + 90);
sprintf(wistr,"Y_0 = %.7f",START_Y);
charstr(wistr);

cmov2(23, UY1 + 45);
```

```
        sprintf(wistr,"Genauigkeit = %d Bit",PRAEZ);
        charstr(wistr);

        /*
        cmov2(135, UY1 + 765);
        charstr(XS1);
        cmov2(15, UY1 + 755);
        charstr("if (x>0.5):");
        cmov2(135, UY1 + 750);
        charstr(YS1);


        cmov2(135, UY1 + 710);
        charstr(XS2);
        cmov2(15, UY1 + 700);
        charstr("if (x<=0.5):");
        cmov2(135, UY1 + 695);
        charstr(YS2);

        cmov2(15, UY1 + 655);
        charstr(ZS1);
        rect(5, UY1 + 628, strlen(XS1)*10+125, UY1 + 790);
        cmov2(15, UY1 + 640);
        charstr(ZS2);

        rect(UX1,UY1,UX2,UY2);
        */
#ifdef SW
        rect(UX1,UY1,UX2,UY2);
#endif
        move2(63,UY1+315);
        draw2(63,UY1+405);
        move2(58,UY1+400);
        draw2(63,UY1+405);
        move2(68,UY1+400);
        draw2(63,UY1+405);
        cmov2(43,UY1+355);
        charstr(up);

        move2(78,UY1+300);
        draw2(168,UY1+300);
        move2(163,UY1+305);
        draw2(168,UY1+300);
        move2(163,UY1+295);
        draw2(168,UY1+300);
        cmov2(118,UY1+280);
        charstr(right);
}
```

```
eval_pxl(char *sitmin, char *sitmax, char *right, char *shorimi, char *shorima,
char *up, char *svertmi, char *svertma)
{
        mpf_t xx, yy, mm, ll, du, du2, two0, o5, o94, one0, four4, o44, o06,
o12, o38, o50,
          hundertover6, hundertover12, hundertover38, hundertover44, pwl_1,
pwl1, pwl, pwm_1,
          pwm1, fiftyover44, cdu0, cdu1, cdu2, cdu3, cdu4, cdu5, cdu6;
        int minit, maxit;
        double x, y, z, ka, hori_min, hori_max, vert_min, vert_max, nach_hori, nach_vert;
        double *hori, *vert;
        double skal_hori, skal_vert;
        register k,flag;
        short m;
        long int pre;
        register short l, phori, pvert;
        char wistr[30] = "0";
        pre = PRAEZ;
        mpf_set_default_prec(12);
        mpf_init(xx);
        mpf_init(yy);
        mpf_init(mm);
        mpf_init(ll);
        mpf_init(du);
        mpf_init(du2);
        mpf_init(cdu0);
        mpf_init(cdu1);
        mpf_init(cdu2);
        mpf_init(cdu3);
        mpf_init(cdu4);
        mpf_init(cdu5);
        mpf_init(cdu6);

        mpf_init(two0);
        mpf_init(o5);
        mpf_init(o94);
        mpf_init(one0);
        mpf_init(fiftyover44);
        /*      mpf_init(four4); */
        mpf_init(o44);
        mpf_init(o06);
        mpf_init(o12);
        mpf_init(o38);
        mpf_init(o50);
        mpf_init(hundertover6);
        mpf_init(hundertover12);
        mpf_init(hundertover38);
        mpf_init(hundertover44);
```

```
mpf_init(pwl_1);
mpf_init(pwl);
mpf_init(pwl1);
mpf_init(pwm_1);
mpf_init(pwm1);


minit = atoi(sitmin);
maxit = atoi(sitmax);
hori_min = atof(shorimi);
hori_max = atof(shorima);
/*****/  hori_max = 1000000000;

vert_min = atof(svertmi);
vert_max = atof(svertma);
x = START_X;
y = START_Y;
z = START_Z;
color(WHITE);
rectfi(UX1+1,UY1+1,UX2-1,UY2-1);
color(BLACK);

switch(right[0])
{
case 'X':
        hori = &x;
        break;
case 'Y':
        hori = &y;
        break;
case 'Z':
        hori = &z;
        break;
case 'k':
        hori = &ka;
        break;
default:
        printf("Ung\"ultige Angabe der x-Achsenvariable!!\n");
}

switch(up[0])
{
case 'X':
        vert = &x;
        break;
case 'Y':
        vert = &y;
        break;
case 'Z':
```

```
        vert = &z;
        break;
default:
        printf("Ung\"ultige Angabe der y-Achsenvariable!!\n");
}

m = BOXZAHL-1;
mpf_set_d(xx,x);
mpf_set_d(yy,y);
mpf_set_d(mm,m*1.0);
mpf_set_d(cdu4,m+0.12);
mpf_set_d(cdu5,m+0.50);
mpf_set_d(cdu6,m+1.0);

mpf_set_d(two0,2.0);
mpf_set_d(o5,0.50);
mpf_set_d(o94,0.94);
mpf_set_d(one0,1.0);
mpf_set_d(fiftyover44,50.0/44.0);
/*      mpf_set_d(four4,4.4); */
mpf_set_d(o06,0.06);
mpf_set_d(hundertover6,100.0/6.0);
mpf_set_d(o44,0.44);
mpf_set_d(o50,0.50);
mpf_set_d(o12,0.12);
mpf_set_d(hundertover12,100.0/12.0);
mpf_set_d(hundertover38,100.0/38.0);
mpf_set_d(hundertover44,100.0/44.0);
mpf_set_d(o38,0.38);



for(k=0;k<=maxit;k++)
{
        /*   1. Box */
        if (mpf_cmp(xx,o5)<=0) {
          mpf_mul(xx,two0,xx);
          mpf_mul(du,o5,yy);
          mpf_add(yy,du,o5);
        }
        else if (mpf_cmp(xx,o5)>0 && mpf_cmp(xx,o94)<=0) {
          mpf_mul(du,two0,xx);
          mpf_sub(du,du,one0);
          mpf_mul(xx,du,fiftyover44);

          mpf_mul(du,o44,yy);
          mpf_add(yy,du,o06);
        }
        else if (mpf_cmp(xx,o94)>0 && mpf_cmp(xx,one0)<=0) {
```

```
  mpf_sub(du,xx,o94);
  mpf_mul(du,du,hundertover6);
  mpf_add(xx,du,one0);

  mpf_mul(du,o06,yy);
  mpf_add(yy,du,o44);
}
else if (mpf_cmp(xx,one0)>0 && mpf_cmp(xx,mm)<0) {
        /* Box-Algorithmus */
        flag=0;
        for (l=1;l<m;l++) {
                mpf_set_d(ll,l*1.0);
                mpf_set_d(cdu0,l+0.12);
                mpf_set_d(cdu1,l+0.50);
                mpf_set_d(cdu2,l+0.94);
                mpf_set_d(cdu3,l+1.0);

                if (mpf_cmp(xx,ll)>0 && mpf_cmp(xx,cdu0)<=0) {
                  mpf_sub(du,xx,ll);
                  mpf_mul(du,du,hundertover12);
                  mpf_add(du,du,ll);
                  mpf_sub(xx,du,one0);

                  mpf_set_d(pwl_1,pow(2.0,l-1.0));
                  mpf_mul(du,o12,yy);
                  mpf_mul(du2,o44,pwl_1);
                  mpf_add(du,du,du2);
                  mpf_sub(yy,du,du2);
                        flag = 1;
                }
                else if (mpf_cmp(xx,cdu0)>0 && mpf_cmp(xx,cdu1)<=0) {
                  mpf_add(du,ll,o12);
                  mpf_sub(du,xx,du);
                  mpf_mul(du,hundertover38,du);
                  mpf_add(xx,du,ll);

                  mpf_set_d(pwl1,pow(2.0,l+1.0));
                  mpf_div(du,one0,pwl1);
                  mpf_mul(du2,o38,yy);
                  mpf_add(yy,du2,du);
                        flag = 1;
                }
                else if (mpf_cmp(xx,cdu1)>0 && mpf_cmp(xx,cdu2)<=0) {
                  mpf_add(du,ll,o50);
                  mpf_sub(du,xx,du);
                  mpf_mul(du,hundertover44,du);
                  mpf_add(xx,ll,du);

                  mpf_set_d(pwl,pow(2.0,l*1.0));
```

```
                    mpf_div(du,o06,pwl);
                    mpf_mul(du2,o44,yy);
                    mpf_add(yy,du,du2);
                          flag = 1;
                  }
                  else if (mpf_cmp(xx,cdu2)>0 && mpf_cmp(xx,cdu3)<=0) {
                    mpf_add(du,ll,o94);
                    mpf_sub(du,xx,du);
                    mpf_mul(du,hundertover6,du);
                    mpf_add(du,ll,du);
                    mpf_add(xx,du,one0);

                    mpf_set_d(pwl,pow(2.0,l*1.0));
                    mpf_div(du,o44,pwl);
                    mpf_mul(du2,o06,yy);
                    mpf_add(yy,du,du2);
                          flag = 1;
                  }
                  if (flag==1)
                    break;
          }
}
else if (mpf_cmp(xx,mm)>0 && mpf_cmp(xx,cdu4)<=0) { /* Abschluss-Box */
  mpf_sub(du,xx,mm);
  mpf_mul(du,hundertover12,du);
  mpf_add(du,mm,du);
  mpf_sub(xx,du,one0);

  mpf_set_d(pwm_1,pow(2.0,m-1.0));
  mpf_sub(du,pwm_1,pwm_1);
  mpf_div(du,o44,du);
  mpf_mul(du2,o12,yy);
  mpf_add(yy,du2,du);
        }
else if (mpf_cmp(xx,cdu4)>0 && mpf_cmp(xx,cdu5)<=0) {
  mpf_add(du,mm,o12);
  mpf_sub(du,xx,du);
  mpf_mul(du,hundertover38,du);
  mpf_add(xx,du,mm);

  mpf_set_d(pwm1,pow(2.0,m+1.0));
  mpf_div(du,one0,pwm1);
  mpf_mul(du2,o38,yy);
  mpf_add(yy,du2,du);
        }

else if (mpf_cmp(xx,cdu5)>0 && mpf_cmp(xx,cdu6)<=0) {
  mpf_add(du,mm,o50);
  mpf_sub(du,xx,du);
```

```
                mpf_mul(du,two0,du);
                mpf_add(xx,mm,du);
                mpf_mul(yy,yy,o50);
                      }

              ka = k;

              if (k < minit)
                      continue;
#ifdef EXPLO
              if (k >= EXPLO)
                      printf("%.1f %.1f %.1f\n",x,y,z);
#endif
#ifdef LIMIT
              if (abs(y)>LIMIT)
              {
                      cmov2(30,550);
                      sprintf(wistr,"(%.1e/%.1e/%.1e)",x,y,z);
                      charstr(wistr);
                      cmov2(100,500);
                      sprintf(wistr,"n = %d",k);
                      charstr(wistr);
                      break;
              }
#endif
              x = mpf_get_d(xx);
              y = mpf_get_d(yy);

#ifdef CUTSTART
              if (*hori >= hori_min && *hori <= hori_max && *vert >= vert_min && *vert <= vert_max && x >= CUTSTART &&
                  x <= CUTSTOP)/*&& z > 0.5)*/
#else
                      if (*hori >= hori_min && *hori <= hori_max && *vert >= vert_min && *vert <= vert_max)
#endif
                      {
                              nach_vert = modf(900 * ((*vert - vert_min)/(vert_max - vert_min)),&skal_vert);
                              nach_hori = modf(900 * ((*hori - hori_min)/(hori_max - hori_min)),&skal_hori);
                              pvert = skal_vert;
                              phori = skal_hori;
                              if(nach_vert >= 0.5)
                                      pvert++;
                              if(nach_hori >= 0.5)
                                      phori++;
#ifdef ZUG
                              if (*hori == ka) {
                                      if(k == hori_min) /* minit */
                                              move2(UX1,UY1);
                                      draw2(UX1+phori,UY1+pvert);
                              }
```

```
                                else
#endif
                                        pnt2(UX1+phori,UY1+pvert);
                        }
                /*                  else
                  printf("\a\n");
*/
#ifdef DATOUT
                printf("%.16f  %.16f  %.16f  %d\n",x,y,z,k);
#endif


                if (!(k%DIGIT))
                {
                        cmov2(UX1+495,12);
#ifndef SW
                        color(CYAN);
#else
                        color(WHITE);
#endif
                        charstr(wistr);
                        color(RED);
                        cmov2(UX1+495,12);
                        sprintf(wistr,"Iterationsschritte momentan: it = %d",k);
                        charstr(wistr);
                        color(BLACK);
                }
        }

        mpf_clear(xx);
        mpf_clear(yy);
        mpf_clear(mm);
        mpf_clear(ll);
        mpf_clear(du);
        mpf_clear(du2);
        mpf_clear(cdu0);
        mpf_clear(cdu1);
        mpf_clear(cdu2);
        mpf_clear(cdu3);
        mpf_clear(cdu4);
        mpf_clear(cdu5);
        mpf_clear(cdu6);
        mpf_clear(two0);
        mpf_clear(o5);
        mpf_clear(o94);
        mpf_clear(one0);
        mpf_clear(fiftyover44);
        mpf_clear(o44);
        mpf_clear(o06);
```

```
        mpf_clear(o12);
        mpf_clear(o38);
        mpf_clear(o50);
        mpf_clear(hundertover6);
        mpf_clear(hundertover12);
        mpf_clear(hundertover38);
        mpf_clear(hundertover44);
        mpf_clear(pwl_1);
        mpf_clear(pwl);
        mpf_clear(pwl1);
        mpf_clear(pwm_1);
        mpf_clear(pwm1);

#ifndef SW
        color(CYAN);
#else
        color(WHITE);
#endif
        cmov2(UX1+495,12);
        charstr(wistr);
}
```

A.4 3D Simulationsplotter

Hierbei wurde zur Berechnung der Iterationswerte ein C^{++}-Programm erstellt. Um die Eingabe und Wahl der Projektionsparameter zu erleichtern, wurde ein graphisches Interface mit Schiebereglern und Schaltflächen im X-Window-System mittels der TCL/TK-Toolbox eingebaut. Beide Programmteile sind nachfolgend aufgeführt.

a) Simulationsprogramm in C^{++}:

```
// a3dstereo2.cc:
// Stereo 3D-View mit ortographic Worldspace
// Hervorhebung der x-y-Ebene durch linewidth(2) und
// zeichnen als letztes.
// Bildverschiebungswinkel: YRot_Diff
// Fuer Movies Version ohne Visualisierungswuerfel!!!!
#include <gl/gl.h>
#include <device.h>
#include <math.h>
#include <stdlib.h>
#include <stdio.h>
```

```
#include <iostream.h>

const double Alfa = 1.0;
const double Beta = 0.04;
const double Gamma = 1.0;
const double Logf = 1.57;
const double Mal = 0.4;
const double Fak_1 = 2.2;
const double Summ1 = 1.0;
const double Eps = 0.001;

const int YRot_Diff = 70;
const int Minit = 0;
const int Maxit = 500000;
const double X_min = 0.0;
const double X_max = 2.0;
const double Y_min = -0.6;
const double Y_max = 0.4;
const double Z_min = 0.0;
const double Z_max = 2.0;

const double Start_x = 0.01;
const double Start_y = 0.01;
const double Start_z = 0.01;

void initialize(int, int, int, int, int);
void processinput(int, int, int, int ,int);
void drawimage(int, int, int, int, int);
void drawcube(void);

int main(int argc, char * argv[])
{
    if (argc != 6){
        cout << "Falsche Anzahl an Kommanozeilen-Argumenten!!!\n"
             << "Korrekter Aufruf: a.out <X_rot> <Y_rot>"
             << "<Z_rot> <mag> <stereo_boolean\n";
        return -1;
    }
    initialize(atoi(argv[1]), atoi(argv[2]), atoi(argv[3]),
               atoi(argv[4]), atoi(argv[5]));
    drawimage(atoi(argv[1]), atoi(argv[2]), atoi(argv[3]),
              atoi(argv[4]), atoi(argv[5]));
    while (TRUE)
        processinput(atoi(argv[1]), atoi(argv[2]), atoi(argv[3]),
                     atoi(argv[4]),atoi(argv[5]));
}

void initialize(int x_rot, int y_rot, int z_rot, int mag, int bo_st)
```

```
{
    if (!(bo_st)) {
//      prefsize(980,980);
        prefposition(0, 980, -3, 980);
//      noborder();
        winopen("3D-View");
        wintitle("3D-View");
//      keepaspect(1,1);
        winconstraints();
        ortho(-mag, mag, -mag, mag, -mag, mag);
        rotate(x_rot, 'x');
        rotate(y_rot, 'y');
        rotate(z_rot, 'z');
    }
    else {
        char wint[60];
        sprintf(wint,"Flare-Attraktor: 3D-StereoView bei %d Iterationsschritten",Maxit);
        prefposition(8, 1272, 8, 700);
        winopen("3D-Stereoview");
        wintitle(wint);
        color(BLACK);
        clear();
        prefposition(8, 640, 8, 700);
        noborder();
        winopen("Stereo-Links");
        ortho(-mag, mag, -mag, mag, -mag, mag);
        rotate(x_rot, 'x');
        rotate(y_rot, 'y');
        rotate(z_rot, 'z');
    }



}

void processinput(int x_rot, int y_rot, int z_rot, int mag, int bo_st)
{
    short val;

    switch(qread(&val)) {
    case REDRAW:
        reshapeviewport();
        drawimage(x_rot, y_rot, z_rot, mag, bo_st);
        break;
    }
}


void drawimage(int x_rot, int y_rot, int z_rot, int mag, int bo_st)
```

```
{

    drawcube();
    if (bo_st) {
        prefposition(640, 1272, 8, 700);
        noborder();
        winopen("Stereo-Rechts");
        ortho(-mag, mag, -mag, mag, -mag, mag);
        rotate(x_rot, 'x');
        rotate((y_rot + YRot_Diff), 'y');
        rotate(z_rot, 'z');
        drawcube();
    }

}

void drawcube(void)
{
    int minit, maxit;
    double x_min, x_max, y_min, y_max, z_min, z_max;
    double x, y, z, xs, ys;
    register k;
    minit = Minit;
    maxit = Maxit;
    x_min = X_min;
    x_max = X_max;
    y_min = Y_min;
    y_max = Y_max;
    z_min = Z_min;
    z_max = Z_max;
    x = Start_x;
    y = Start_y;
    z = Start_z;

    color(BLACK);
    clear();
    color(CYAN);


    for(k=0;k<=maxit;k++)
        {
            xs = x*exp(Logf*(1-x*x)) + Alfa*y;
            ys = y*(Summ1 + Fak_1* (Mal-x)) + Beta * z;
            z = Gamma * x;
            x = xs; y = ys;
            pnt((((x-x_min)/(x_max-x_min))*116.0-58.0),
                (((y-y_min)/(y_max-y_min))*116.0-58.0),
                (((z-z_min)/(z_max-z_min))*116.0-58.0));
        }
```

```
}
```

b) TCL/TK-Skript als Aufsatz auf das oben beschriebene `a3dstereo2.cc`:

```
#! /usr/local/bin/wish
wm title . "a3dstereo.cc"
wm geometry . +975+0
set mode 0
set pzs -5
set tk_strictMotif 1
message .msg -width 7c -justify center -text "3D-Visualisierung eines
Flare-Attractors"
  -font -adobe-times-bold-r-*-*-*-180-*-*-*-*-*-*
label .msg3 -text "in orthographischer Parallelprojektion" -justify center
  -font -adobe-times-bold-r-*-*-*-100-*-*-*-*-*-*
label .msg2 -text "des dynamischen Systems" -justify center
  -font -adobe-times-bold-r-*-*-*-100-*-*-*-*-*-*
frame .space -height 0.5c
frame .space2 -height 10.2c
canvas .c -width 6c -height 2.5c
.c create bitmap 4c 1.2c -bitmap @~/pro/3d/equ3.xbm
scale .x_rot -label "Rotation um x-Achse \[0.1\]" -from -900 -to 900 -length
7c -orient horizontal
  -font -*-international-*-*-*-*-*-*-*-*-*-*-*-*
scale .y_rot -label "Rotation um y-Achse \[0.1\]" -from -900 -to 900 -length
7c -orient horizontal
  -font -*-international-*-*-*-*-*-*-*-*-*-*-*-*
scale .z_rot -label "Rotation um z-Achse \[0.1\]" -from -900 -to 900 -length
7c -orient horizontal
  -font -*-international-*-*-*-*-*-*-*-*-*-*-*-*
scale .mag -label "Skalierung \[% der Einheitslnge\]" -from 1 -to 200 -length
                                % 7c -orient horizontal
  -font -*-international-*-*-*-*-*-*-*-*-*-*-*-*
.mag set 100
.x_rot set 0
.y_rot set 0
.z_rot set 0
frame .dummy0
frame .dummy1
pack .msg .space .msg2 .c .msg3 -fill x
pack .space2 .x_rot .y_rot .z_rot .mag .dummy0 .dummy1 -fill x
label .dummy0.modus -text " Graphik-Modus:" -font -*-international-*-*-*-*-*-*-*-*-*-*-*-*
radiobutton .dummy0.mono -text "Mono" -variable mode -value 0
-activebackground gray89
  -font -*-international-*-*-*-*-*-*-*-*-*-*-*-* -selectcolor yellow
radiobutton .dummy0.stereo -text Stereo -variable mode -value 1
-activebackground gray89
```

```
  -font -*-international-*-*-*-*-*-*-*-*-*-*-* -selectcolor yellow
checkbutton .dummy1.sequenz -text "Drehsequenz y-Achse \[0->90\]" -variable
seque -activebackground gray89
  -font -*-international-*-*-*-*-*-*-*-*-*-*-*  -selectcolor yellow
button .dummy1.start -text Starten -command {
    if {!$seque} {
        exec kill [expr ($pzs +1)] &
        set pzs [bo_seq $mode $seque]} else {
            bo_seq $mode $seque
        }
    } -font -*-international-*-*-*-*-*-*-*-*-*-*-*
pack .dummy0.modus .dummy0.mono .dummy0.stereo -side left -pady 0.3c
pack .dummy1.sequenz .dummy1.start -side top -fill x -pady 0.2c
# pack .dummy.mono .dummy.stereo -side left
proc bo_seq {a b} {
    if {$b == 0} {
        exec ~/pro/3d/a3dstereo [.x_rot get] [.y_rot get] [.z_rot get] [.mag get] $a &
    } else {
    for {set i 0} {$i <= 900} {incr i +90} {
        exec ~/pro/3d/a3dstereo [.x_rot get] [expr [.y_rot get] + $i] [.z_rot get] [.mag get] $a
    }
  }
}
```

Abbildungsverzeichnis

1.1 Technik des Poincaré-Schnitts 7

2.1 Physiologischer Regelkreis der Leptin-Sektretion als Beispiel für ein Dynamisches System in der Medizin 11

3.1 Zeitserienvergleich: Modellystem vom Flare-Typ — Intensitätsmessung des Flare-Sterns Wega XXX 14

3.2 Schematisches Grundprinzip eines Dynamischen Systems vom Flare-Typ. Ein chaotisches Teilsystem treibt ein nichtlineares System, das wie ein Schalter eine Schwelle besitzt. Je nach dem momentanen Wert des Antriebs — unterschwellig oder überschwellig — kommt es entweder zu Selbstinhibition (Dämpfung) oder zum autokatalytischen Wachstum (Entdämpfung) der angetriebenen Variablen 15

3.3 „Riddled Basins" Attraktor gemäß Gl. 3.2; Abszisse x [0,1], Ordinate y [0,1] 18

3.4 Ein einfacher Flare-Attraktor basierend auf der logistischen Map 20

3.5 „Löwentatzen-Fraktal": Querschnitt eines differenzierbaren Flare-Attraktors. Nach Gl. 7.7 in Kapitel 7.3.3, Abszisse: forcende Variable x, Ordinate: Flaring-Variable w; aus [18]. . . . 22

3.6 Quasiperiodisches Forcing eines potentiellen Flare-Systems gemäß Gl. 3.4: **Oben** Zeitserie (Abszisse t, Ordinate w), 200 Iterationen. **Unten** die dazugehörige Phasenraumdarstellung (x,w) 1 000 000 Iterationen. 25

4.1 Beispiel eines Simulink-Schaltbilds — das van-der-Pol System (vgl. Text). 31

5.1 Simulink-Schaltbild zur Erzeugung einer Flare-Dynamik gemäß Gleichung 5.3 35

5.2 Das einfachste Reaktionssystem mit deterministischem Chaos 36

5.3 Reaktionskinetisches Modell mit Flare-Verhalten einer Reaktionsvariablen 38

5.4 Numerische Integration mittels Adams-Gear-Methode eines Flare-Systems gemäß dem Reaktionsschema der Abb. 5.3: Zeitliches Verhalten (t : 0 $\rightarrow$ 3000) der Konzentration von w. 39

5.5 Dreidimensionale Phasenraumdarstellung des reaktionskinetischen Systems gemäß Gleichung 5.3 (die oberste Trajektorie läuft nach rechts). 40

6.1 Abbildungsprinzip der logistischen Map 44

6.2 Typische Flare-Zeitserie eines durch die logistische Map geforcten Systems 45

6.3 Abbildungsprinzip des Bernoulli-Shift 45

6.4 Zeitserie durch Forcing des Systems mittels des Bernoulli-Shift gemäß Gl. 6.1: Verhalten der 2. Variablen als Funktion der Zeit. 46

6.5 Abbildungsprinzip der Tent-Map 47

6.6 Oben: Zeitserie der Flaring-Variablen w durch Forcing des Systems mittels der Tent-Map gemäß Gl. 6.2, 1 000 000 Iterationsschritte. Unten: Dazugehörige Phasenraumdarstellung (x, w) 48

6.7 Phasenraumdarstellung des Flare-Attraktors nach Gl. 6.3, 1 000 000 Iterationen. 50

7.1 Flare-Attraktor, erzeugt durch die invertierbare Iterations-Abbildung von Gl. 7.2. Projektion auf die x-y-Ebene. Die Abszisse ist die x-Achse, die Ordinate die y-Achse. 2 000 000 Iterationspunkte sind dargestellt. Anfangsbedingungen: 0.01 für alle drei Variablen. 53

7.2 Top-down Darstellung des Attraktors von Abbildung 7.1 . . . 55

7.3 Querschnitt durch den Attraktor der Abbildung 7.1 mit exakt identischer Projektion (Abszisse x, Ordinate w). Dargestellt ist der Schnitt zwischen $w = 0.4800$ und $w = 0.48014$. 125 000 000 Iterationen. 56

7.4 Detail-Vergrößerung der Abbildung 7.3 aus einem Segment (der obersten rechten Ecke); 1 000 000 000 Iterationen. 57

7.5 Baker's map 58

7.6 3D Flare-Attraktor: Forcing durch die Baker's Map gemäß Gl. 7.5; x- und y-Achse [0,1], w-Achse [0,10]. 60

7.7 Horseshoe-Attraktor gemäß dem x-y Untersystem von Gl. 7.6; Abszisse x [0,1], Ordinate y [0,1]. 61

7.8 3D Flare-Attraktor, geforct durch die vielfach gefaltete Horseshoe Map gemäß Abb. 7.6 und Gl. 7.6; x- und y-Achse [0,1], w-Achse [0,10]. 62

7.9 x, w-Schnitt durch den 3D Horseshoe-Flare-Attraktor; Abszisse x [0,1], Ordinate w [0,1]. 63

7.10 y, w-Schnitt durch den 3D Horseshoe-Flare-Attraktor („feuerzungenartig"); Abszisse y [0,1], Ordinate w [0,1]. 64

7.11 Flare-Attraktor, erzeugt durch Gl. 7.7. **a**: Zeitserie von w, 5 000 Iterationen. **b**: Seiten-Ansicht (x,w Plot) **c**: Queransicht: y,w Plot. Ein Querschnitt zwischen $x = 0$ und $x = 1$ ist dargestellt. 1 000 000 Iterationen abgebildet (in **b** und **c**). Anfangsbedingungen: $x_0 = \frac{1}{10}\sqrt{2}$, $y_0 = 0.1$, $w_0 = 0.1$. 65

8.1 Gekoppelte Flaresysteme mit 3, 6 und 18 Zellen 70

8.2 Numerische Simulation von Gleichung 8.1, wobei die letzte Zeile durch die Gleichung 8.2 ersetzt ist. 72

9.1 Iterationsschema zur Hopf-Gaspard Map, nach [41] 76

10.1 Staircase Baker's Map im Fall von 4 Stufen 78

10.2 Phasenraumdarstellung der Staircase Baker Map mit 10-facherÜberhöhung der y-Variablen 82

10.3 Zeitverhalten der horizontalen Variablen x der Staircase Baker's Map mit 40 Stufen 84

10.4 Simulation wie Abb. 10.3 mit zehnfacher Iterationsdauer (25 000 Iterationsschritte). 86

10.5 Hundert mal längere Simulation als in Abb. 10.4, dargestellt in 2 Stücken 87

Literaturverzeichnis

[1] D. K. Arrowsmith, C. M. Place: *Dynamische Systeme*. Spektrum Akademischer Verlag, Heidelberg 1994.

[2] P. Gray, S. K. Scott: *Chemical Oscillations and Instabilities*, International Series of Monographs in Chemistry, Band 21. Clarendon Press, Oxford 1994.

[3] H. Poincaré: *Les méthodes nouvelles de la méchanique céleste.* Gauthier-Villar, Paris 1899.

[4] J. Agyris, G. Faust, M. Haase: *Die Erforschung des Chaos — Studienbuch für Ingenieure und Naturwissenschaftler.* Vieweg Verlag, Braunschweig 1995.

[5] P. Kloeden: *Numerical Solutions of Stochastic Differential Equations.* Springer Verlag, Berlin, 2. Aufl. 1999.

[6] E. N. Lorenz: Deterministic nonperiodic flow. *J. Atmos. Sci.*, **20** (1963) 130–141.

[7] F. F. Seelig: Systemtheorie I. Skriptum zur Vorlesung, Universität Tübingen 1993.

[8] I. Prigogine, R. Lefever: Symmetry-breaking instabilities in dissipative systems II. *J. Chem. Phys.*, **48** (1968) 1665.

[9] O. E. Rössler: Chemical turbulence: Chaos in a simple reaction-diffusion system. *Z. Naturforsch.*, **31a** (1976) 1168.

[10] L. F. Olsen, H. Degn: Chaos in an enzyme reaction. *Nature*, **267** (1977) 177–178.

[11] R. A. Schmitz, K. R. Graciani, J. L. Hudson: Experimental evidence of chaotic states in the Belousov-Zhabotinskii reaction. *J. Chem .Phys.*, **67** (1977) 3040–3044.

[12] R. M. May: Simple mathematical models with very complicated dynamics. *Nature*, **261** (1976) 459–467.

[13] R. FitzHugh: Impulses and physiological states in theoretical models of nerve membrane. *Biophys. J.*, **1** (1961) 445–466.

[14] G. Löffler, P. E. Petrides: *Biochemie und Pathobiochemie.* Springer-Verlag, Berlin 1997.

[15] J. Milnor: On the concept of attractor. *Commun. Math. Phys.*, **99** (1985) 177–195.

[16] O. E. Rössler, G. C. Hartmann: Attractors with flares. *Fractals*, **3** (1995) 372–376.

[17] G. C. Hartmann, O. E. Rössler: Flaring — a new type of dynamical behaviour. In M. M. Novak, editor, *Fractal Reviews in the Natural and Applied Sciences* Chapman & Hall, London 1995.

[18] G. C. Hartmann, O. E. Rössler: Coupled flare attractores. *Discrete Dynamics in Nature and Society*, **2** (1998) 153–159.

[19] G. C. Hartmann: Flare-Attraktoren in Chemie und nichtlinearer Dynamik.. Diplomarbeit, Universität Tübingen 1994.

[20] J. C. Alexander, J. A. Yorke, Z. You, I. Kan: Riddled basins. *Intern. J. Bifurcation and Chaos*, **2**(4) (1992) 795–813.

[21] J. C. Sommerer, E. Ott: A physical system with qualitatively uncertain dynamics. *Nature* **365** (1993) 138.

[22] P. BECKMANN: Skriptum zur Vorlesung „Dynamische Flüsse in Zustandsräumen". Universität Mainz 1997.

[23] S. GROSSMANN, S. THOMAE: Invariant distributions and stationary correlation functions of one-dimensional discrete processes. *Z. Naturforsch.*, **32** (1997) 1353–1363.

[24] J. L. KAPLAN, J. A. YORKE: Chaotic behavior of multidimensional difference equations. *Lecture Notes in Mathematics*, **730** (1979) 204–227.

[25] O. E. RÖSSLER, R. WAIS, R. RÖSSLER: Singular-continuous Weierstrass-function attractors. In *Proceedings of the 2nd International Conference on Fuzzy Logic & Neural Networks*, S. 909–912, Iizuka, Japan, 17.–22. Juli 1992.

[26] O. E. RÖSSLER, C. KNUDSEN, J. L. HUDSON, I. TSUDA: Nowhere-differentiable attractors. *Int. J. Intell. Systems*, **10** (1995) 15–23.

[27] O. E. RÖSSLER: Singular-continuous nowhere-differentiability in attractors. In A. E. Andersson, S. I. Andersson, U. Ottoson (Hrsg.), *Dynamical Systems — Theory and Applications*, S. 205–232. World Scientific, Singapore 1993.

[28] H. RÖPKE, J. RIEMANN: *Analogcomputer in Chemie und Biologie.* Springer Verlag 1971.

[29] MATHWORKS INC: *SIMULINK — Dynamic System Simulation Software for Technical Education.* Prentice Hall, New Jersey 1996.

[30] J. HOFFMANN: *MATLAB und SIMULINK: Beispielorientierte Einführung in die Simulation dynamischer Systeme.* Addison-Wesley-Longman, Bonn 1998.

[31] W. H. PRESS (et.al.): *Numerical Recipes in C: The Art of Scientific Programming.* Cambridge University Press 1992.

[32] S. K. SCOTT: *Chemical Chaos.* Clarendon Press, Oxford, 1993.

[33] F. W. SCHNEIDER, A. F. MÜNSTER: *Nichtlineare Dynamik in der Chemie.* Spektrum Akademischer Verlag, Heidelberg 1996.

[34] O. E. RÖSSLER, J. L. HUDSON: Chaos in simple three and four variable chemical systems. *Lecture Note in Biomathematics*, **55** (1984) 135–145.

[35] O. E. RÖSSLER: Chaos in abstract kinetics: Two prototypes. *Bull. Math. Biol.*, **39** (1977) 275.

[36] I. GUMOWSKI, C. MIRA: *Dynamique Chaotique.* Cépadues Editions, Toulouse 1980.

[37] S. SMALE: Differentiable dynamical systems. *Bull. Am. Math. Soc.*, **73** (1967) 747–817.

[38] E. HOPF: *Ergodentheorie.* Springer Verlag, Berlin 1937.

[39] G. SILVERBERG, B. VERSPAGEN: Evolutionary theorizing on economic growth. In K. Dopfer, editor, *The Evolutionary Principle of Economics.* Kluwer Academic Publishers, Norwell, MA, 1995.

[40] O. E. RÖSSLER, G. C. HARTMANN: A society of flare attractors. In S. C. Müller (Hrsg.) *Diffusion and Structure Formation, 18. WE-Heraeus Seminar*, Bad Honnef, 1.–12. November 1997, Abstracts S. 22–23.

[41] P. GASPARD: What is the role of chaotic scattering in irreversible processes? *Chaos*, **3**(4) (1993) 427–442.

[42] G. C. HARTMANN, G. RADONS, H. H. DIEBNER, O. E. RÖSSLER: Staircase baker's map generates flaring type time series. *Discrete Dynamics in Nature and Society*, **5** (2000) 107–120.

[43] H. H. DIEBNER, O. E. RÖSSLER: A deterministic entropy based on the instantaneous phase space volume. *Z. Naturforsch.*, **53**a (1998) 51–60.

[44] S. GOLDSTEIN, J. L. LEIBOWITZ: Ergodic properties of an infinite system of particles moving independently in a periodic field. *Comm. Math. Phys.*, **37** (1974) 1–18.

[45] H. H. HASEGAWA, D. J. DRIEBE: Intrinsic irreversibility and the validity of the kinetic description of chaotic systems. *Phys. Rev. E*, **50**(3) (1994) 1781–1809.

[46] S. TASAKI, P. GASPARD: Fick's law and fractality of nonequilibrium states in a reversible multibaker map. *J. Stat. Phys.*, **81** (1995) 935–987.

[47] G. RADONS: Vorträge beim „Workshop on Stochasticity and Structure Formation", U. Behn, A. Kühnel, K. Schiele (Org.), Leipzig 1996, und beim „Workshop on Hyperbolic Systems with Singularities", D. Szazs, P. Choquard (Org.), Erwin-Schrödinger-Institut,Wien 1996.

[48] G. RADONS: Disordered phenomena in chaotic systems. *Festkörperprobleme, Advances in Solid State Physics*, **38** (1999) 439–451.

[49] O. E. RÖSSLER, G. C. HARTMANN: Vortrag beim „Workshop on Statistical Mechanics", G. Radons (Org.). Dresden 1997.

[50] B. ROSSER: Volatility via social flaring. *Journal of Economic Behavior and Organization* (eingereicht).

Verzeichnis der akademischen Lehrer

Meine akademischen Lehrer waren die Damen und Herren Dozenten und Professoren:

Baier, Bayer, Christen, Eckstein, Gauglitz, Göpel, Häfelinger, Hagenmaier, Hanack, D. Hoffmann, V. Hoffmann, Jäger, Jung, Kahlert, Kemmler-Sack, Kneifel, Koch, Krug, Kunze, Lindner, Lorenz, Nakel, Oberhammer, Oelkrug, Pauschmann, Pausewang, Pommer, Rössler, Schurig, Seelig, Stegmann, Strähle, Zeller.