Protokoll zum Versuch Nichtlineare Dynamik und Chaos

Nicolas Heimann, Jesse Hinrichsen

Universität Hamburg

2015

Zusammenfassung

1 Einleitung

Alle Plottss und Simulationen aus dem Versuch haben wir im Rahmen des Versuches selber implementiert. Dafür wählten wir als Programmiersprache Python2.7 und nutzten OpenGL4.1 und OpenCL für Visualisierungen und Berechnungen. Der Quellcode ist online über github einsehbar: `https://github.com/keksnicoh/gl_plotting_experimental`.

Im folgenden bezeichnet $f^2(x) = f(f(x))$

2 Logistische Abbildung

Die logistische Abbildung ist gegeben durch $f(x_n) = x_{n+1} = rx_n(1 - x_n)$. Es zeigt sich das diese einfache Funktionsvorschrift bereits chaotisches Verhalten an den Tag legt welches wir im folgenden Abschnitt genauer untersucht haben. Zunächst haben wir ein Bifurkations Diagramm der logistischen Abbildung erzeugt indem wir den Parameter r gegen Iterationspunkte aufgetragen haben. Dabei fixierten wir jeweils ein r und erzeugten eine Folge $x_0...x_{1000}$ von welcher wir $x_{500}...x_{1000}$ auf die y-Achse aufgetragen haben (Abbildung N). Das Bifurkationsdiagramm lässt sich in mehrere Bereiche unterteilen. Bis $r = 3$ laufen die $x_{500}...x_{1000}$ auf den gleichen Fixpunkt zu. An $r = 3$ gabelt sich das Diagramm in zwei Äste auf (Periodenverdoppelung). An $r = 3.449$ gibt es eine weitere Perdionenverdopplung und es ist eine Selbstähnlichkeit mit dem Bereich um $r = 3$ zu erkennen (Fraktale Strukturen). Ab $r = 3.569$ entsteht ein chaotischer Bereich in welchem sich aber noch Strutkuren feststellen lassen (Bögen, Punkte auf geraden, freie Bereiche).

2.1 Fixpunkte / Stabilitätsbedigung

Bildet die Funktion einen Punkt idempotent ab, so handelt es sich um einen Fixpunkt, es gilt: $f^n(x^*) = x^* \forall n \iff x^* ist Fixpunkt$. Stabilitätsbedingung

$$|f'(x^*)| < 1 \iff Fixpunkt - stabil$$

$$|f'(x^*)| > 1 \iff Fixpunkt - instabil$$

Mit $f(x^*) = rx^*(1 - x^*) = x^* \iff x^* = 1 - 1/r \vee x' = 0$ ließ sich so analytisch bestimmen, dass die Fixpunkte für $r \in [0, 1) \cup (3, 4]$ instabil für $r \in (1, 3)$ stabil sind. Obwohl für den Parameter $0 < r < 1$ der Fixpunkt instabil ist konvergieren $f^n(x)$ gegen 0 $\forall x \neq x^*$, denn $f^n(x)$ ist ein Polynom vom grad $2n$.

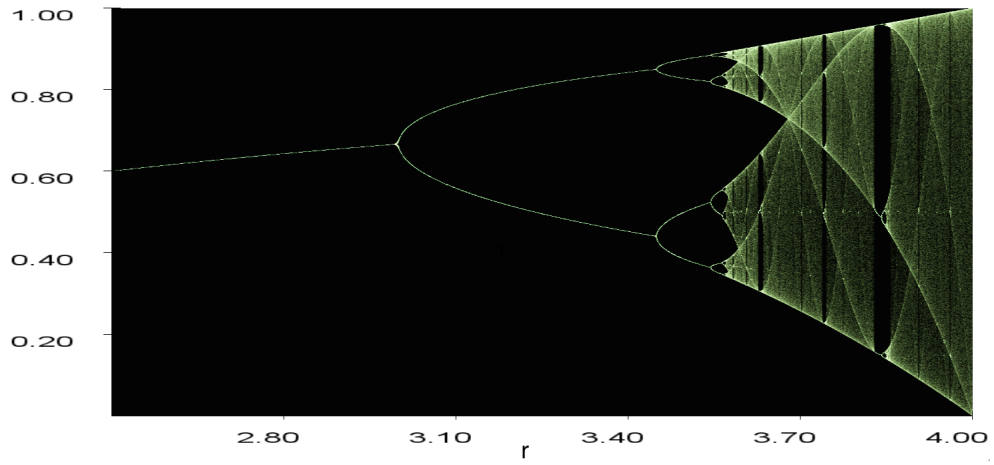

Abbildung 1: Bifurkationsdiagramm der logistischen Abbildung im Bereich $r \in [2.6, 4]$. sourcecode: prak/birukation-logistisch-no-opt.py

Abbildung N zeigt das Verhalten der logisitschen Funktion für zwei solche Werte für r. So lässt sich auf die Aufspaltung im Bifurkationsdiagramm (Abblidung N) für $r > 3$ verstehen. Abbildung N zeigt wie sich die Iteration jeweils für einen stabilen und einen Instabilen Wert von r verhält. Für den zweierzyklus $f^2(x)$ ließen sich folgende Fixpunkte bestimmen welche in Abbildung N zusammen mit dem Einerzyklus Fixpunkten dargestellt sind ermitteln.

$$x_{n+1} = f_r(x_n) = rx_n(1 - x_n)$$
$$\Rightarrow x_{n+2} = r^2 x_n(1 - x_n)(1 - rx_n(1 - x_n))$$

Fixpunktgleichung (Zweierzyklus):

$$x = r^2 x(1 - x)(1 - rx(1 - x))$$
$$\Rightarrow x_{3,4} = \pm\frac{\sqrt{r^2 - 2r - 3} + r + 1}{2r}$$

Damit $x_{3,4}$ reel bleibt muss $r^2 - 2r - 3 \geq 0$

$$\Rightarrow r \leq -1 \wedge r \geq 3$$

Für diesen Bereich gibt es folglich 2 weitere Fixpunkte $x_{3,4} \Leftrightarrow$ Perdiodenverdopplung. Im Fall der logistischen Abbildung gilt

$$\frac{d}{dx} f(x) = r - 2rx = r(1 - 2x)$$
$$\frac{d}{dx} f^2(x) = -r^2(2x - 1)(2r(x - 1)x + 1)$$

AB HIER TODO: Grafisch lässt sich ablesen, dass der Fixpunkt $x_3 = \frac{\sqrt{r^2-2r-3}+r+1}{2r}$ (grüner Graph) für folgende Bereiche stabil ist:

$$-1.45 < r < -0.82 \Rightarrow -1.45 < r \leq -1$$
$$2.82 < r < 3.45 \Rightarrow 3 \geq r > 3.45$$

Der Fixpunkt $x_4 = \frac{-\sqrt{r^2-2r-3}+r+1}{2r}$ (grauer Graph) ist im gesamten Bereich $-1.45 < r < 3.45$ stabil aber da der Fixpunkt ebenfalls nur für $r \leq -1 \wedge r \geq 3$ existiert gilt der selbe Bereich wie für x_3. Die Fixpunkt sind dort stabil, wo sich der graue und der grüne Graph in der Abbildung überlagern.

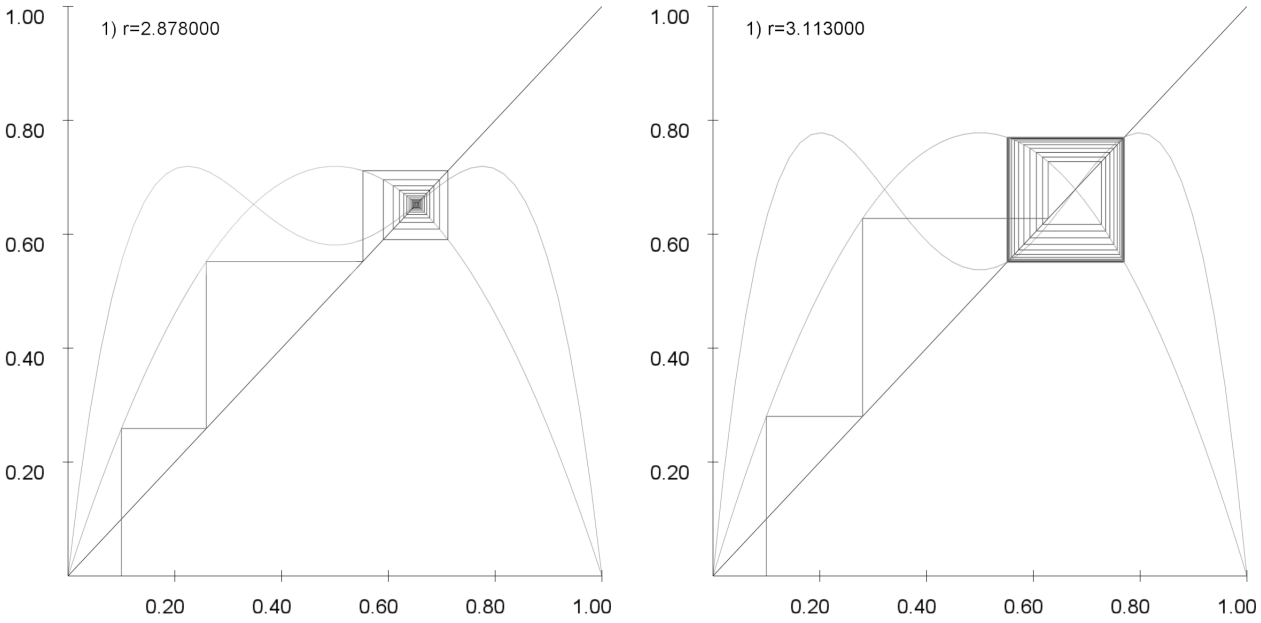

Abbildung 2: Verlauf der Iterationen bei festen Parameter r. Linkes Bild zeigt einen stabilen Fixpunkt bei r=2.878, rechtes Bild zeigt instabilen Fixpunkt bei r=3.113. Der Verlauf der logistischen Funktion $f(x)$, $f^2(x)$ sowie die Einheitsgerade $y = x$ sind aufgetragen . Als Linie Verbunden geplottet ist die Folge $(x_n, 0.0), (x_n, x_{n+1}), (x_{n+1}, x_{n+1}), (x_{n+1}, x_{n+2}), (x_{n+2}, x_{n+2}), (x_{n+2}, x_{n+3}), ...$ Sourcecode: prak/logisitsch-no-opt-behavior.py

.

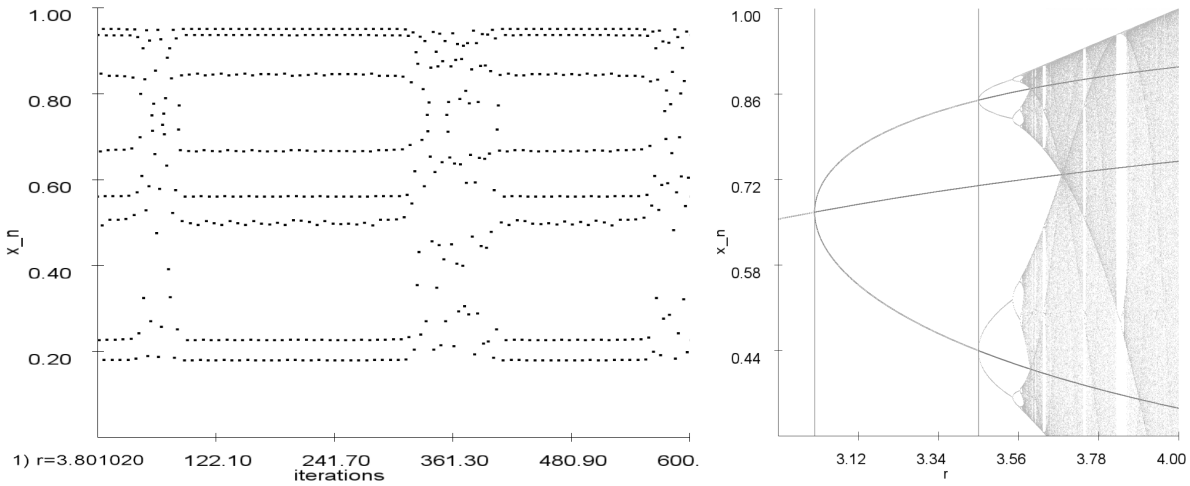

Abbildung 3: Logistische Funktion. Links: Intermittenz bei $r = 3.80102$. Rechts: Die stabile Lösung des Einerzyklus x^* und die beiden Lösungen des Zweierzyklus $x_{3,4}$. Im Hintergrund das Bifurkationsdiagramm. Die vertikalen Linien sind an $r = 3.0$ und $r = 444$ und markieren die Stellen wo $|f'(x)| = 1$, $|\frac{d}{dx} f^2(x)| = 1$ sind. Sourcecode: prak/intermittenz.py, prak/logis-zyklen.py

.

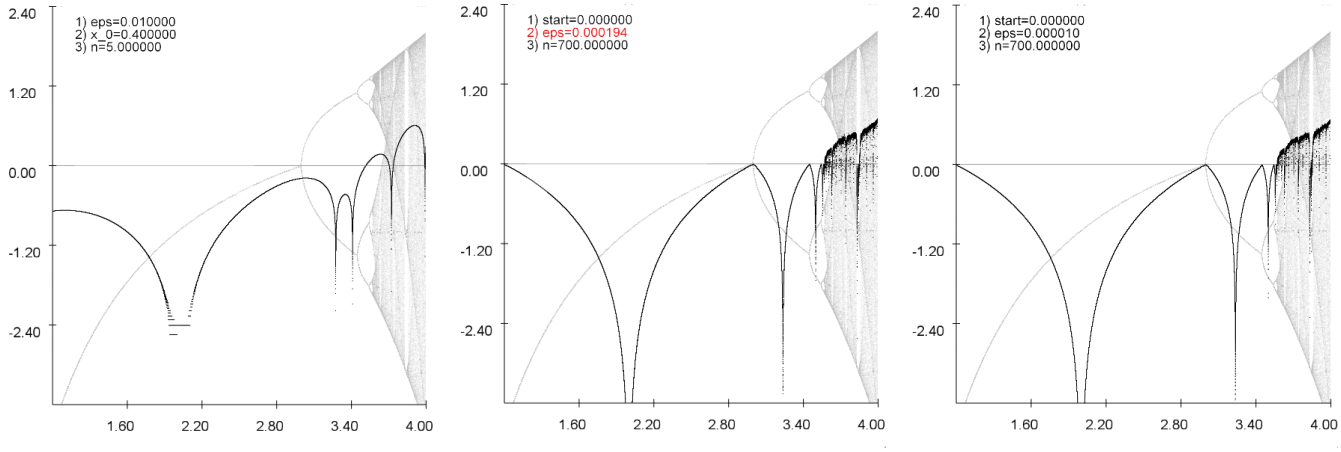

Abbildung 4: Drei verschiedene Implementation des Lyapunov Exponent. Links: Definition, Mitte: Analytisch, Rechts: Renormiert. Die linke Implementation weißt deutliche Abweichungen im Vergleich zu den anderen beiden Implementation auf und ist somit unbrauchbar. Sourcecode: prak/lyapunov.py

.

2.2 Lyapunov Exponent

Der Lyapunov Exponent beschreibt mit welcher Geschwindigkeit sich zwei naheliegende Punkte voneinander entfernen. Es gibt drei Wege den Lyapunov Exponenten zu implementieren:

(1) Definition $\lambda(x_0) = \lim_{N\to\infty} \lim_{\epsilon\to 0} \frac{1}{N} \log | \frac{f^N(x_0+\epsilon)-f^N(x_0)}{\epsilon} |$

(2) Analytisch $\lambda(x_0) = \lim_{N\to\infty} \frac{1}{N} \sum_{i=0}^{N-1} \log f'(x)$

(3) Renormiert: Nach jedem Iterationsschritt wird der Abstand ϵ neu gesetzt.

In Abbildung N haben wir die drei Möglichkeiten auf die logisitsche Funktion angewendet. Es zeigte sich, dass die erste Möglichkeit sehr schlechte Ergebnisse im Vergleich zu den beiden letzten Methoden ergibt. Dies liegt daran, dass die selbt bei kleinen EPSILON die Funktionswerte sehr schnell divergieren und somit die Definition des Differenzenquotienten keinen Sinn ergibt. Die Renomierung hält diesen Abstand in jedem Iterationsschritt klein, weshalb sich der Lyapunov Exponent trotzdem ausrechnen lässt. Der Lyapunov Exponent hat seine Nullstellen dort wo die Abbildung ihre superattraktiven Stellen hat. Umgekehrt divergiert $\lambda(x_0)$ an den superattraktiven Stellen. Man erhält also Information über das Verhalten der Abbildung für bestimmte x_0. Tratsächlich kann man den Lyapunov-Exponenten über den mittleren Informationsverlust ausdrücken $\lambda(x_0) = -log(2) * \delta I$ (QUELLE). Im folgenden ist der OpenCL Quellcode der renomierten Formel des Lyapunov Exponenten:

```
float g(float r, float x) {
   return r * x * (1-x);
}
vec4 f(vec4 x) {
   float x0 = 0.4;
   float eps = 0.0001;
   float n = 10000;
   float summe = x0;
   for (int i=1; i < n; i++) {
      x0 = g(x.x, x0);
      summe += log(abs(g(x.x, x0+eps)-g(x.x, x0))/eps);
   }
   return vec4(x.x, summe/n, 0, 0.5);
}
```

2.2.1 Vergleich der analytischen und renomierten Implementation

Wir haben nun die analytische und die renormierte Implementation des Lyaponovexponenten weiter untersucht. Dabei stellten wir fest, dass bei Stellen mit Periodenverdoppelung die renormierte Formel etwas schneller gegen die 0 konvergiert als die analytische Formel. An einer weiteren Stelle ($r = 3.05$) ist zu erkennen wie beide Implementationen einmal von oben (renormiert) und ein mal von unten (analytisch) gegen einen scheinbar gemeinsamen Wert streben (Abbildung N). Die Vermutung liegt nahe, dass man mit dem Mittelwert aus beiden Implementationen an solchen stellen wesentlich schneller zum Ergebniss kommt. Es zeigt sich aber, dass dieses Verhalten nicht regelmäßig auftritt weshalb wir es nicht mehr weiter untersucht haben. Nach weiteren Stichproben scheint die analytische Implementation bis zum 500ten Iterationsschritt etwas schneller an seinen Grenzwert zu konvergieren als es die renormierte tut. Beide Versionen zeigten aber in Stichproben, dass sie stets gegen den gleichen Grenzwert strebten.

2.3 Feigenbaumkonstante

Die Feigenbaumkonstante ist eine universelle Größe. Sie tritt in chaotischen nicht linearen System auf und lässt sich wie folgt bestimmen:

$$\delta_i = \frac{b_i - b_{i+1}}{b_{i+1} - b_{i+2}}$$

oder

$$\delta_i = \frac{s_i - s_{i+1}}{s_{i+1} - s_{i+2}}$$

$$\delta = \lim_{i \to \infty} \delta_i = 4.669201609102991 (2.3.1)$$

(https://oeis.org/A006890) wobei s_i, b_i die Folgen der superattraktiven und periodenverdoppelden Stellen sind. Also lässt sich die Feigenbaumkonstante mit dem Lyapunov Exponenten bestimmen. Als erstes haben wir versucht die Nullstellen des Lyapunov Exponenten zu bestimmen. Wir wendeten dabei das folgende Kriterium für eine Nullstelle an:

$$\lambda(x - \epsilon) < \lambda(x) \wedge \lambda(x + \epsilon) < \lambda(x) \wedge |\lambda(x)| < \epsilon$$

Es zeigte sich aber, dass bei genauen Betrachten des numerisch bestimmten Lyapunov Exponenten sehr große Schwankungen vorhanden waren, weshalb dieses Kriterium nicht mit unseren Verfügbaren Rechenleistungen praktikabel war (Abbildung N). Aufgrund der Probleme mit den Periodenverdoppelden Stellen haben wir uns dazu entschieden die superattraktiven Stellen numeritsch zu bestimmen. Unser Algorithmus startet im Suchmodus bei gegeben Startwert x_{start} und geht in kleinen Schritten Δx die x-Achse ab. In jedem Schritt wird $l_1 = \lambda(x_0)$ $l_2 = \lambda(x_0 + \Delta x)$ berechnet. Ist

$$l_2 - l_1 \leq 0 (1)$$

wandert der Iterationsschritt zum superattraktiven Fall. Dies wird so lange fortgesetzt bis die Bedingung (1) nicht mehr hält. Es wird nun um Δx zurückgegangen und anschließend die Schrittweite $\Delta x \mapsto \frac{\Delta x}{10}$ verkleinert. Nun wird erneut so lange iteriert, bis (1) nicht mehr hält. Der Vorgang wiederholt sich 8 mal. Anschließend wird $(2 * x_0 + \Delta x)/2$ als Ergebniss gespeichert. Als nächstes befindet sich der Algorithmus im Anfangspunkt-Modus. Es wird so lange die x-Achse abgetestet bis die Bedingung

$$\lambda(x_0) < \lambda(x_0 + \Delta x) < \lambda(x_0 + 2 * \Delta x)$$

nicht mehr erfüllt ist und somit ein neues x_{start} gefunden wurde. Der Suchmodus wird aktiviert. (Quellcode: prak/feigenbaum.py) Im folgenden ist die Terminal Ausgabe des Algorithmusses beigelegt:

```
searching from 1.9
looking for next start_r from 2.00000000002
searching from 2.99950000003
looking for next start_r from 3.23606797751
```

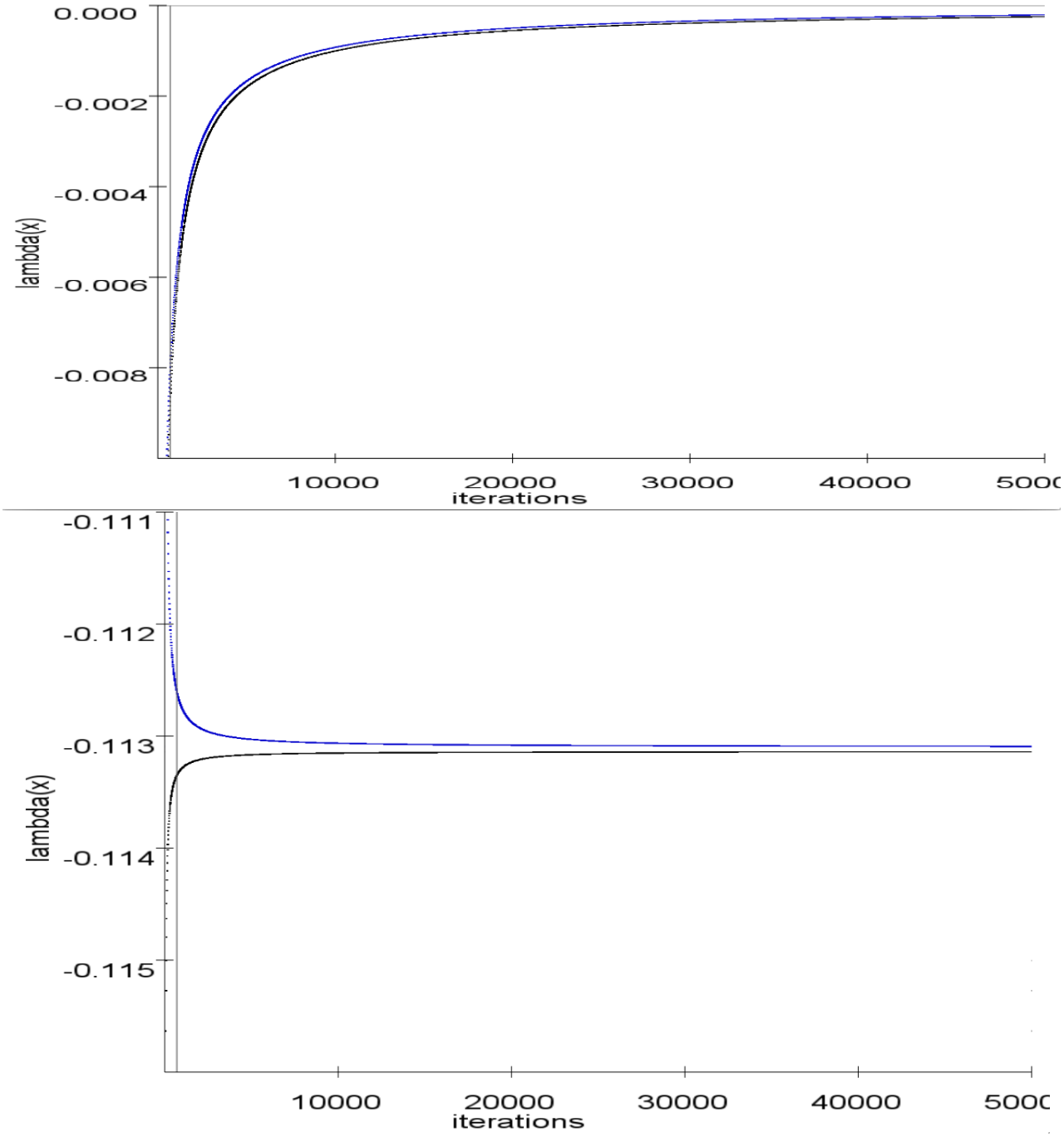

Abbildung 5: Vergleich des analytischen(Schwarz) und renormierten(Blau) Iterationsverhalten des Lyapunov Exponenten. Vertikale Linie bei N=700. Oberes Bild zeigt Iterationsverhalten bei $r = 3.0$. Unteres Bild zeigt Iterationsverhalten bei $r = 3.05$

.

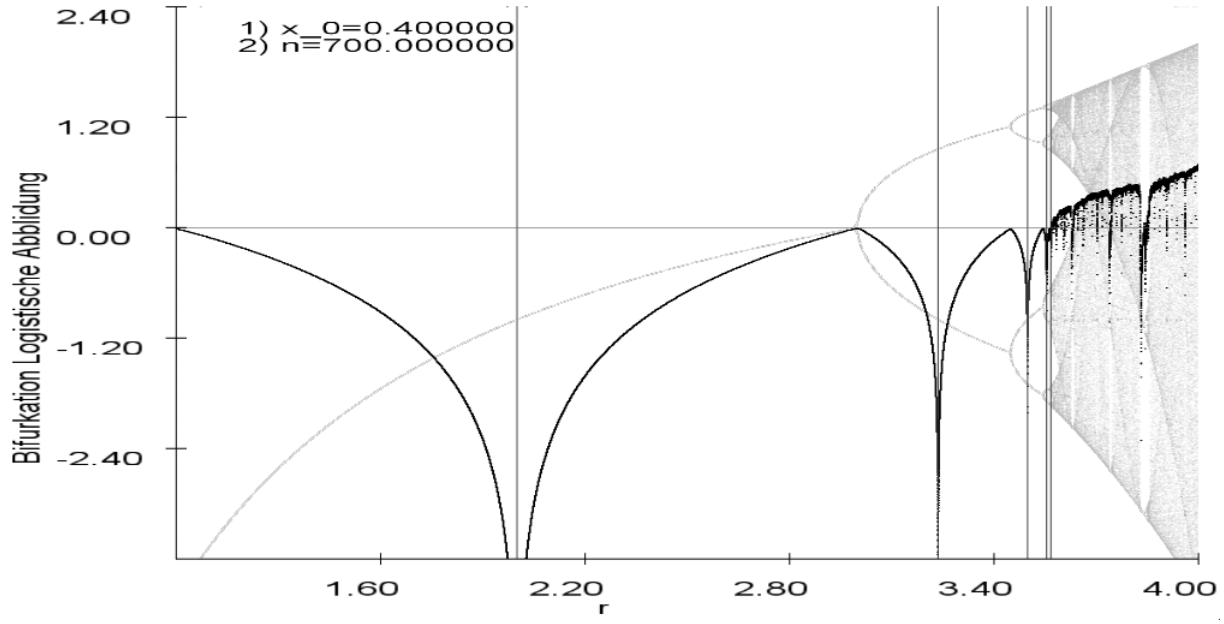

Abbildung 6: Analyse des Bifurkationsdiagrammes der logistischen Funktion. Eingezeichnet sind die y=0 Achse, sowie die ersten 5 Superattraktiven Stellen für r. Das Bifurkationsdiagramm wurde so translatiert und skaliert, dass es hinter dem Lyapunov Exponenten erscheint.

.

```
searching from 3.44927797752
looking for next start_r from 3.49856169934
searching from 3.54400769935
looking for next start_r from 3.55464086278
searching from 3.56439786279
looking for next start_r from 3.56666737986
found values [2.0000000000249916, 3.236067977509959, 3.498561699344952,
   3.554640862779951, 3.5666673798649517]
delta_0=4.70894301336
delta_1=4.68077099865
delta_2=4.66295961155
```

Somit konnten wir numerisch für $i = 2$ eine Feigenbaumkonstante von $\delta_2 = 4.66295961155$ berechnen. Dieser Wert weicht um 0.133671789% vom tatsächlichen Wert (2.3.1) ab. Die Grenzen des Algorithmus sind bereits nach 5 gefunden superattraktiven Stellen erreicht. Ab $r > 3.57$ fängt die numerische Implementation des Lyapunov Expontenten an zu "rauschen"(Abbildung N). Der Algorithmus kann daher nicht mehr präzise seinen Suchmodus ausführen. Ebenfalls liegen die nächsten Superattraktiven Fällen noch dichter bei Sammen als es für s_4, s_5 der Fall war, was ebenfalls vom Algorithmus nicht mehr detektiert wird.

3 Sinus Abbildung

Die Sinusabbildung ist gegeben durch $x_{n+1} = r * sin(x)$. Wir haben die bereits implementierten Programme nun auf die Sinus Funktion angewendet und kamen zu folgendem Ergebniss: TABELLE MIT ALLEN SACHEN:

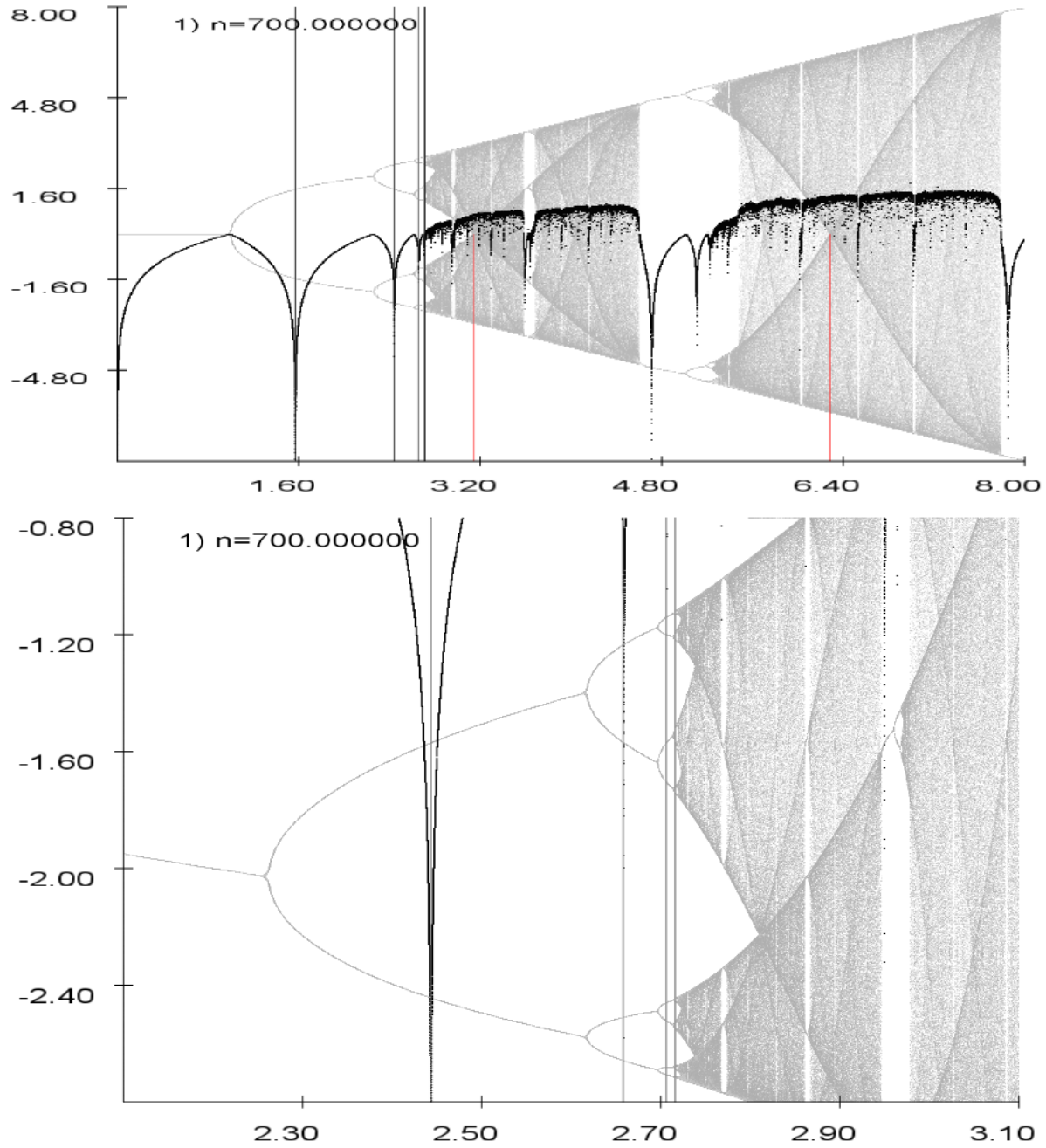

Abbildung 7: Oben: Analyse des Bifurkationsdiagrammes der sinus Abbildung. Eingezeichnet sind die y=0 Achse, sowie die ersten 6 Superattraktiven Stellen für r (Erste Superattraktive Stelle bei $r = 0$. Ebenfalls wurden π und 2π eingezeichnet. Unten: Vergrößerung des Ausschnittes $r \in [2.1, 3.1], y \in [-2.8, -0.8]$

.

4 Duffing-Gleichung

Wir betrachten nun einen angetrieben und gedämpften Oszillator. Als Unterschied zum gewöhnlichen Harmonischen Oszillator tritt hier ein kubischer Dämpfungsterm auf.

$$\ddot{x} + \lambda\dot{x} + \beta x^3 = \epsilon \cos \Omega t$$

Diese DGL lässt sich nun nicht mehr analytisch berechnen. Im Folgenden lösen wir die Gleichung mit der Euler-Methode als auch mit dem Runge-Kutta Verfahren.

$$\frac{dy}{dt} = \epsilon \cos\theta - \lambda y - \beta x^3$$

$$\frac{dx}{dt} = y$$

$$\frac{d\theta}{dt} = \Omega$$

4.1 Attraktoren

Parameter:

$$\epsilon = 0,2, \lambda = 0,08, \beta = 1, \Omega = 1$$

- Unterschiede Runge Kutta Euler (unterschiedliche attraktoren)
- stabile / instabile Trajektorien –> parameter $h = \frac{Zeit}{Iterationen}$
- Optimierung durch Schrittweiten adaptierung
- lyapunov??

4.2 Poincareschnitt

Die gezeigten Phasenraumportaits sind Projektionen des dreidimensionalen Phasenraums (x, y, θ) auf die (x, y) Ebene. Der Poincaree Schnitt ist eine Abbildung aller (x, y) welche eine bestimmte Ebene im Phasenraum schneiden. In Abbildung XYZ ist ein Bereich Poincaree Schnitt des Duffing-Oszillators mit $\epsilon = 7.72$ und $\theta = 0$ gezeigt. Zur Implementation des Poincaree Schnittes wählten wir $\theta = 0$ um die Praktikumsanleitung als Test unserer Software nutzen zu können. Dabei nutzten wir $sin(\theta_1) * sin(\theta_2) \leq 0.0 \iff$ *Ebenenschnitt*:

```
pos = sin(theta);
if (last_pos*pos <= 0.0f) {
   result[k*2] = last_x;
   result[k*2+1] = last_y;
   k++;
}
last_pos = pos;
```

Dieses Verfahren zeigt bei genauerer Betrachtung aber leichter Ungenauigkeiten. So wird nicht exakt das (x, y) duplet abgebildet bei welchen die Ebene geschnitten wurden, stattdessen wird das (x, y) Duplet bei θ_2 angezeit. Eine Möglichkeit dies zu Optimieren wäre den Mittelwert $(\frac{x_1+x_2}{2}, \frac{y_1+y_2}{2})$ als Schnittpunkt zu identifizieren.

5 LDR-Oszillator

Im folgenden untersuchen wir einen realen nichtlinearen Schwingkreis.
TODO: Schaltskizze

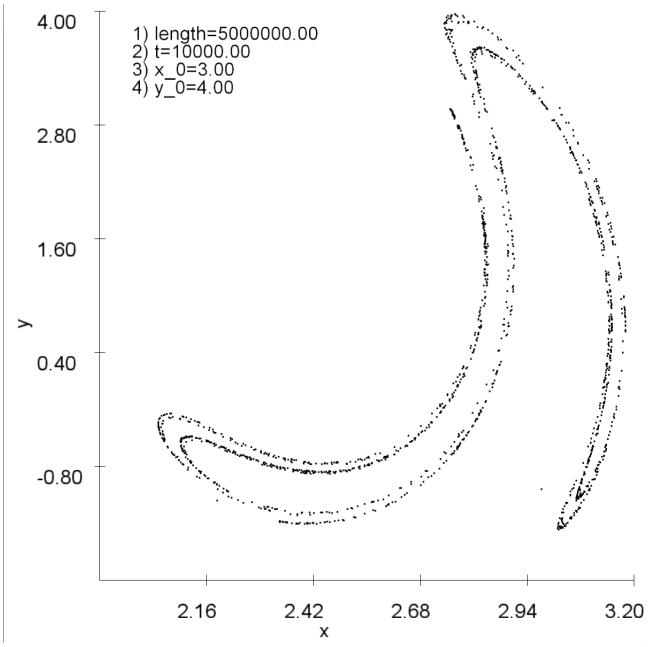

Abbildung 8: Poincare Schnitt des Duffing Oszillators für $\epsilon = 7.72x = 3.0, y = 4.0, \lambda = 0.2, \beta = 1, \theta = 1.$ als Referenz aus dem VORBEREITUNGSHEFT-LITERATUR-S38

5.1 Theorie

Zunächst lässt sich ein linearer Schwingkreis der aus einem Widerstand, einem Kondensator und einer Spule besteht über die Spannungen an den einzelnen Bauteilen beschreiben

$$V_g = V_c + V_l + V_r \text{ (Kirschoff'sches Gesetz)}$$

Daraus folgt mit $V_g = V_s \cos \omega t$ die Differentialgleichung

$$\ddot{Q}L + \dot{Q}R + \frac{Q}{C} = V_s \cos \omega t$$

welche mit dem angetriebener, gedämpfter Oszillator vergleichbar ist. Dabei ist Q die Ladung, L die Induktivität der Spule, R der Widerstand, C die Kapazität des Kondensators V_s, die angelegte Spannung und ω die Frequenz der angelegten Wechselspannung.
Wird nun der Kondensator durch eine Diode ausgetauscht erhalten wir, wie bei Duffing-Oszillator, einen nichtlinearen Term. Während beim linearen Schwingkreis $I = \frac{dQ}{dt}$ gilt lässt sich die Diode als parallel geschalteter Kondensator und Widerstand mit

$$I = I_f(1 - \exp(-\frac{V_d}{V_t})) + \frac{dQ}{dt}$$

beschreiben, wobei I_f und V_t Konstanten sind. Dies führt zu

$$\frac{dV_d}{dt} = \frac{I - I_f(1 - \exp(-\frac{V_d}{V_t}))}{C_f \exp(-\frac{V_d}{V_t})}$$

in Durchlassrichtung und

$$\frac{dV_d}{dt} = \frac{I - I_f(1 - \exp(-\frac{V_d}{V_t}))}{C_r(1 + \frac{V_d}{\phi})^\gamma}$$

in Sperrrichtung. Weiterhin gilt

$$\frac{dI}{dt} = \frac{V_s \cos\theta - V_d - RI}{L}$$

$$\frac{d\theta}{dt} = \omega$$

Damit lässt sich nun das Euler-Verfahren anwenden (siehe 5.2)

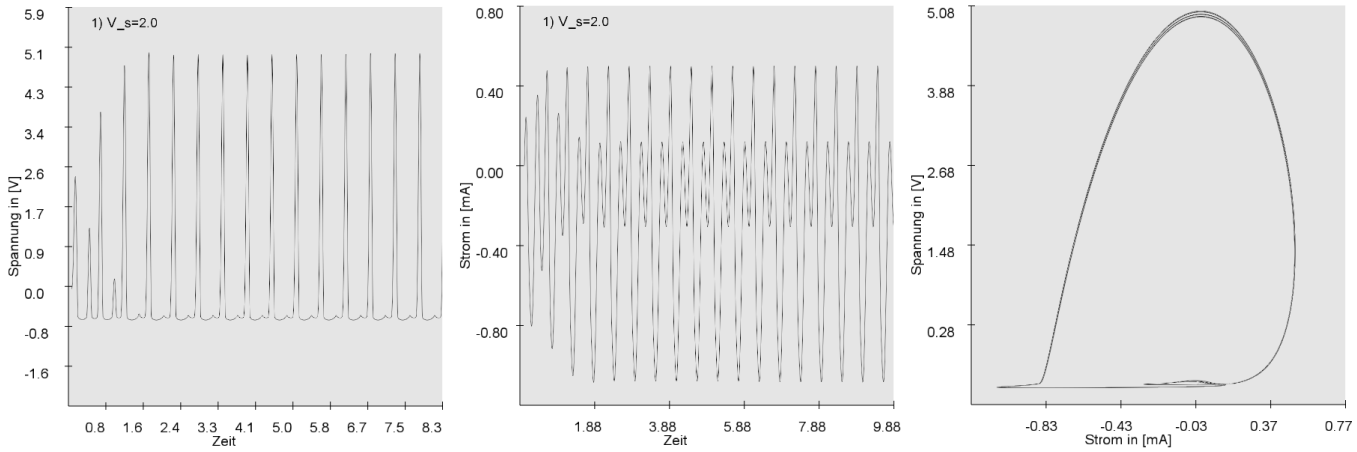

Abbildung 9: LDR-Schwingkreis bei einer Anregungsspannung von $V_s = 2V$. Numerisch mit Euler-Cauchy-Verfahren gelöst, bei einer Schrittweite $h = 10^{-9}$ Links: Zeitlicher Verlauf der Spannung V_d an der Diode (10^5Iterationen). Mitte: Zeitlicher Verlauf des Stroms I ($10^5 Iterationen$). Rechts: Phasenraumdiagram nach 10^5 Iterationen für weitere $4 \cdot 10^5$ Iterationen

.

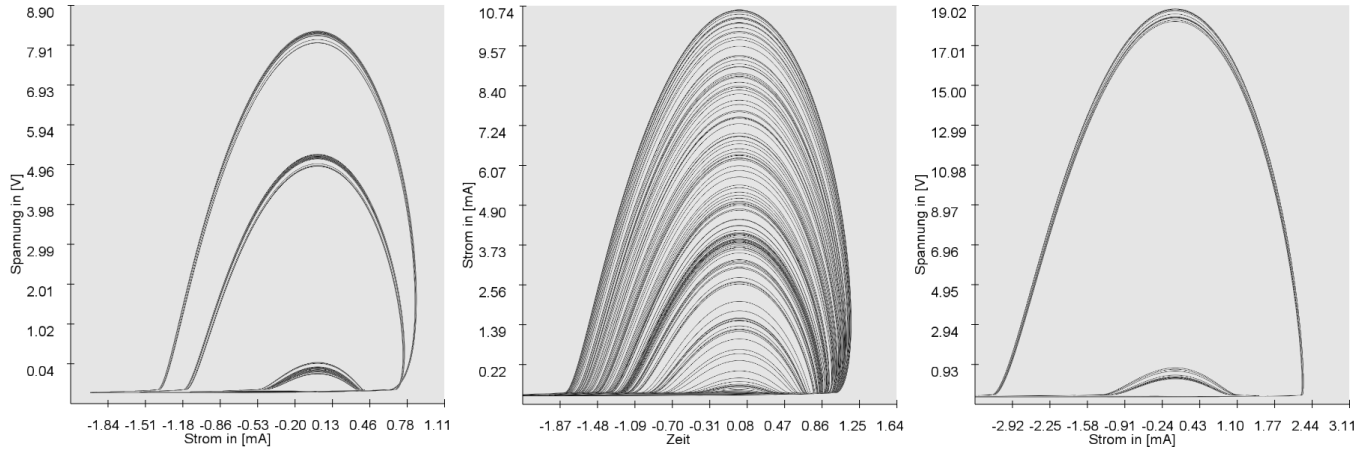

Abbildung 10: Phasenraumdiagramme bei unterschiedlichen Anregungsspannungen V_s nach 10^5 Iterationen für weitere $4 \cdot 10^5$ Iterationen. Links: $V_s = 4V$. Mitte: $V_s = 6V$. Rechts: $V_s = 15.67V$

5.2 Numerische Berechnungen

Unter Verwendung folgender Paramter

R	100Ω	beschreibung
L	$2367 \cdot 10^{-6}H$	beschreibung
C_r	$82pF$	beschreibung
C_f	$56 \cdot 10^{-6}pF$	beschreibung
I_f	$2,8pA$	beschreibung
γ	$0,44$	beschreibung
ϕ	$0,6V$	beschreibung
V_t	$34mV$	beschreibung

haben wir für unterschiedliche Anregungsspannungen V_s numerisch die Gleichungen aus 5.1 gelöst. Dabei sind wir davon ausgegangen, dass für $V_d > -0.6$ die Diode sperrt und ansonsten leitet.
Für unterschiedlichen Anregungsspannungen erhalten wir teilweise deutliche Akttraktoren als auch chaos (siehe Abbildung 11).

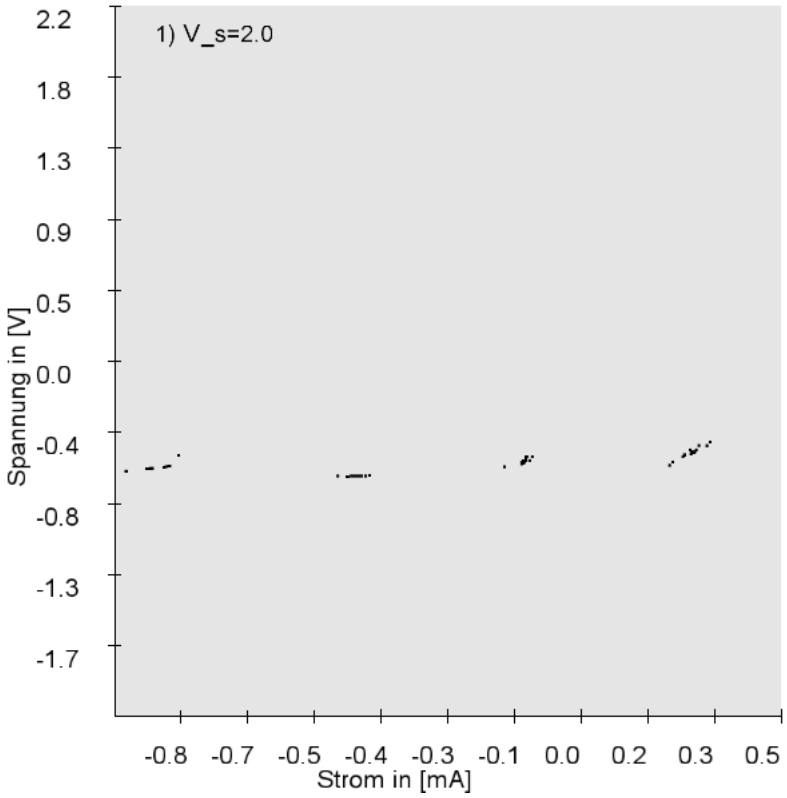

Abbildung 11: Poincaré-Schnitt für $V_s = 2V$ durch die Ebene bei $\sin(\theta) = 0$ angefangen nach 10^4 Iterationen. Insgesamt 1500 Punkte

5.3 Versuchsaufbau

Der in diesem Versuch verwendete Schwingreis ist aufgebaut aus einer Spule, einem Widerstand und einer Diode.....

5.4 Versuchsdurchführung

5.5 Zusammenfassung

6 Literatur

- Nichtlineare Dynamik und Chaos - Physikalisches Praktikum für Fortgeschrittene Universität Hamburg