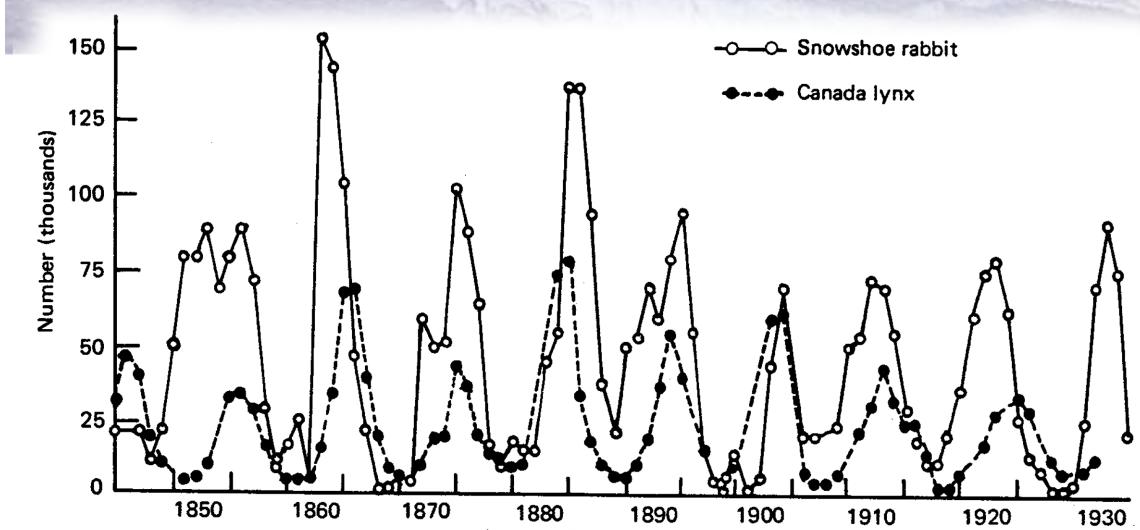


|||||| HEAD ===== 3e11b17a22f17134c5776c45facead6eab419466

Projektarbeit: Das Räuber-Beute-Modell

pray-predator-model



Tobias Möhle
211204256

7. August 2013

Inhaltsverzeichnis

1 Einleitung	1
2 Deterministische Betrachtung	1
3 Betrachtung mit der Master-Gleichung	3
4 Berechnungen mit dem Gillespie-Algorithmus	4
5 Auswertung	7
6 Abbildungsverzeichnis	8

1 Einleitung

Das Räuber-Beute-Modell ist ein Modell aus der Ökologie, welches ein Minimal-System aus der Ökologie beschreibt: Eine Raubtier- und eine Beutepopulation. Diese beiden Populationen beeinflussen sich gegenseitig, da die Beutepopulation von den Raubtieren klein gehalten wird, gleichzeitig jedoch deren Nahrungsgrundlage ist, sodass die Raubtierpopulation ebenfalls nicht zu groß werden kann. Diese gegenseitige Beeinflussung kann durch unterschiedliche Modelle beschrieben werden. Eine Betrachtungsweise ist ein deterministisches Modell, welches durch die Lotka-Volterra-Gleichungen¹

$$\dot{x} = x(k_1A - k_{12}y) = 0$$

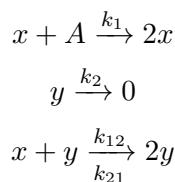
$$\dot{y} = y(k_{21}x - k_2) = 0$$

beschrieben wird. Hierbei stellt x die Beute- und y die Räuberpopulation dar; die Parameter haben die anschauliche Bedeutung

- k_1A ist eine Größe, welche die Fertilität der Beutetiere beschreibt. Diese wird einerseits durch populationsspezifische Größen (Häufigkeit der Geburten, Anzahl der Kinder je Geburt etc.), welche in k_1 zusammengefasst sind, und andererseits durch die vorhandene Nahrungsmenge, symbolisiert mit A , beschrieben.
- k_2 beschreibt die Mortalität der Raubtiere
- k_{21} und k_{12} sind Größen, welche die Wechselwirkung der Populationen beschreibt, also die Anzahl der gerissenen Beutetiere auf der einen Seite und die Fertilität der Räuber auf der anderen Seite (welche von deren Nahrungsgrundlage, den Beutetieren, abhängt).

Die Lotka-Volterra-Gleichungen sind allgemein nicht analytisch lösbar, jedoch kann man an ihnen Betrachtungen zum Langzeit- und Stabilitätsverhalten durchführen. Diese Analysen werden in Kapitel 2 vorgenommen.

Eine zu den Lotka-Vorterra-Gleichungen analoge Beschreibung erhält man durch die drei Reaktionsgleichungen



Dieser Beschreibung liegt, wie allen chemischen Prozessen, ein stochastisches Modell zu Grunde, welches mit Hilfe entsprechender Methoden analysiert werden kann. Eine Betrachtung durch die Master-Gleichung und eine Analyse mit Hilfe des Gillespie-Algorithmus¹ werden in den Kapiteln 3 und 4 vorgestellt.

¹<http://de.wikipedia.org/wiki/Lotka-Volterra-Gleichungen> (am 25.01.2013)

2 Deterministische Betrachtung

Die deterministische Betrachtung des Räuber-Beute-Systems folgt, wie bereits oben erwähnt, den beiden gekoppelten Differenzialgleichungen 1. Ordnung

$$\dot{x} = x(k_1 A - k_{12}y)$$

$$\dot{y} = y(k_{21}x - k_2)$$

Da diese analytisch nicht lösbar sind, werden hier lediglich die stationären Lösungen, das heißt Lösungen, bei welchen sich das System nicht mehr verändert, betrachtet.

Ein stationärer Fall ist eingetreten, wenn $\dot{x} = \dot{y} = 0$. Bei den Lotka-Volterra-Gleichungen ergeben sich zwei stabile Punkte: $x_1 = 0, y_1 = 0, x_2 = \frac{k_2}{k_{21}}, y_2 = \frac{k_1 A}{k_{12}}$.

Nun gilt es, zu untersuchen, wie stabil das System gegenüber kleinen Schwankungen δ an diesen Punkten ist. Dies lässt sich mit der Gleichung $\begin{pmatrix} \dot{x} \\ \dot{y} \end{pmatrix} = J\delta \begin{pmatrix} x \\ y \end{pmatrix}$ wobei J die Jakobimatrix

$$J = \begin{pmatrix} k_1 A - k_{12}y & -k_{12}x \\ k_{21}y & k_{21}x - k_2 \end{pmatrix}.$$

ist, untersuchen.

Für x_1, y_1 gilt nun:

$$\begin{vmatrix} k_1 A - \lambda & 0 \\ 0 & -k_2 - \lambda \end{vmatrix} = (\lambda - k_1 A)(\lambda + k_2).$$

An dieser Stelle ist das System offensichtlich instabil, da λ eine positive Lösung hat.

Für x_2, y_2 gilt analog:

$$\begin{vmatrix} -\lambda & -k_{12}\frac{k_2}{k_{21}} \\ k_{21}\frac{k_1 A}{k_{12}} & -\lambda \end{vmatrix} = \lambda^2 + k_1 k_2 A.$$

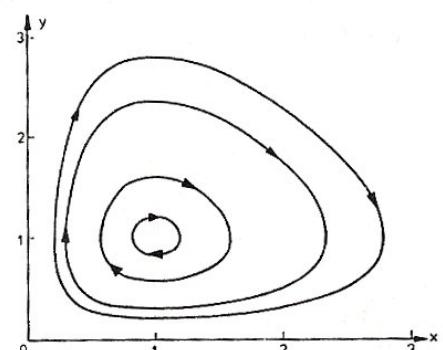
Dieses System hat lediglich die imaginären Lösungen $\lambda = \pm i\sqrt{k_2 k_1 A}$. Rein imaginäre Lösungen beschreiben weder stabile noch instabile Systeme sondern ozillierende Lösungen (analog zu Schwingungsgleichungen in der Mechanik).

Diese Oszillation verschwindet an dem stabilen Punkt; andere Lösungen oszillieren entsprechend um diesen.

Es lässt sich weiter zeigen, dass, analog dem mathematischen Pendel oder anderen oszillierenden Systemen, hier eine Erhaltungsgröße analog der Energie bzw. Hamilton-Funktion eingeführt werden kann^a. Diese lautet:

$$H = k_{21}x - k_2 \ln(x) + k_{12} - k_1 A \ln(y) = \text{const}$$

Im Phasenraum ergeben sich damit nebenstehend dargestellte Trajektorien. Dabei entsprechen die unterschiedlichen Ringe unterschiedlichen Werten für H .



^aR. Mahnke „Nichtlineare Physik in Aufgaben“ Verlag: B.G. Teubner, Stuttgart 1994; S. 125

Abbildung 1: Dynamik des Räuber-Beute-Systems im Phasenraum

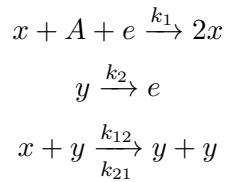
3 Betrachtung mit der Master-Gleichung

Es gibt zwei unterschiedliche Master-Gleichungen, welche prinzipiell das gegebene System beschreiben: zum einen ist das die diskrete Master-Gleichung

$$\frac{\partial}{\partial t} P(n, m, t) = k_1 A(n-1)P(n-1, m, t) + k_{21}(n+1)(m-1)P(n+1, m-1, t) + k_2(m+1)P(n, m+1, t) -$$

$$[k_1 An + k_{21}nm + k_2m] P(n, m, t)$$

bei welcher n die Größe der Beute-Population und m die der Räuber-Population ist. Diese ist jedoch nicht analytisch lösbar, da es hier zunächst unendlich viele miteinander gekoppelte Gleichungen gibt. Dieses Problem ist zwar durch die Einführung einer maximalen Populationsanzahl durchaus zu beheben, jedoch wird das System nicht handlicher, da bereits bei einer Maximalpopulation von 3 Tieren 10 gekoppelte Differentialgleichungen zu lösen sind. Hier gelten nun die Reaktionsgleichungen



dabei ist e die Anzahl „freier“ Plätze in dem System. Damit muss sich auch die erste Übergangsgröße verändern. Aus $k_1 An$ wird nun $k_1 Ane$. Für den Kontinuitätsübergang werden nun noch die Größen $x = \frac{n}{V}$, $y = \frac{m}{V}$ eingeführt. Aber auch hier ist keine Lösung möglich, weshalb ein anderer Lösungsansatz benötigt wird. Im Folgenden soll dies mit Hilfe des Gillespie-Algorithmus' geschehen, welcher einen empirischen Ansatz verfolgt und das Problem numerisch lösen kann.

4 Berechnungen mit dem Gillespie-Algorithmus

Der Gillespie-Algorithmus untersucht stochastische Prozesse mit einem statistischen Ansatz: Es werden eine Anzahl von Durchläufen des Prozesses mit Hilfe von Zufallsfunktionen berechnet und anschließend über diese gemittelt. Wurden genügend Durchläufe gemacht und sind die verwendeten Zufallsalgorithmen gut, so entsprechen die mit dem Gillespie-Algorithmus gewonnenen Ergebnisse denen der analytische Lösung. Das genaue Vorgehen beim Gillespie-Algorithmus lautet wie folgt²:

1. Initialisiere zunächst alle Startwerte (Anfangsbedingungen, Konstanten etc). In meiner Implementierung wird dieser Schritt separat bereits im constructor des Programms vorgenommen.
2. Generiere zwei Zufallszahlen τ , ρ , wobei τ eine exponentiell verteilte Zahl sein soll: Kleine Zahlen sollen also sehr häufig vorkommen, größere immer seltener (wird zum Beispiel, wie in meiner Implementierung, durch eine gleich-verteilte Zufallsvariable erzeugt, welche einer e-Funktion übergeben wird.). ρ soll eine Zahl zwischen 0 und 1 sein. Die Werte von ρ sind gleichverteilt (entsprechende Algorithmen bietet jede höhere Programmiersprache). In meiner Implementierung kommen keine Zeitsprünge größer als 1 Zeiteinheit vor:

```
double changetime=exp(-double(rand())/double(rand()))
double reaction=double(rand())/RAND_MAX;
```

3. Nun wird ein Reaktionsschritt durchgeführt: Die Zeit wird um die Variable τ weiter gesetzt (Zeitpunkt, zu dem die Reaktion stattfindet) und ρ wird auf die möglichen Reaktionsschritte abgebildet.

Da bei dem Räuber-Beute-Modell die Wahrscheinlichkeit von den Populationsgrößen abhängt, wird hier jeweils die Reaktionskonstante (k_1, k_2, k_{21}) mit der entsprechenden Populationsgröße addiert und durch die Gesamtgröße geteilt. In meiner Implementierung:

```
double foo=k1*prepre[i].prey[j];
double bar=foo+k2*prepre[i].pred[j];
double foobar=bar+(k21+k12)*prepre[i].pred[j]*prepre[i].prey[j]/2;
```

Nun kann durch Vergleich der Größen einer der Prozesse durchgeführt werden:

```
if (reaction<=foo/foobar){ //a new prey is born
else if(reaction<=bar/foobar){ // a pred died
else { //a pred ate a prey
```

4. Ist die zu Beginn festgelegte Simulationsdauer noch nicht erreicht, beginne wieder bei Schritt 2. Dabei muss darauf geachtet werden, dass die Wahrscheinlichkeiten beim nächsten Reaktionsschritt anders verteilt sind.

Die gesamte Implementierung des Algorithmus' kann auf der Seite
<https://github.com/BalticPinguin/Prey-Predator-System.git>
eingesehen und heruntergeladen werden.

Mit Hilfe dieses Algorithmus' lässt sich das Räuber-Beute-Modell nun gut beschreiben. Allerdings stellt sich die Suche nach geeigneten Parametern als deutlich schwerer dar, als man zunächst erwarten würde, da die Zusammenhänge, wie auch an den unten dargestellten Graphiken zu erkennen ist, nicht linear sind. Stirbt beispielsweise die Räuber-Population bei den Simulationen stets aus, hilft es nicht, ihr Sterberate einfach zu senken; vielmehr hilft ihnen eine höhere Geburtenzahl bei den Beutetieren, da sie sich so besser erholen und damit die Räuber-Population stärken.

²<http://www.co-nan.eu/pdf/df2.pdf> (28.01.2013)

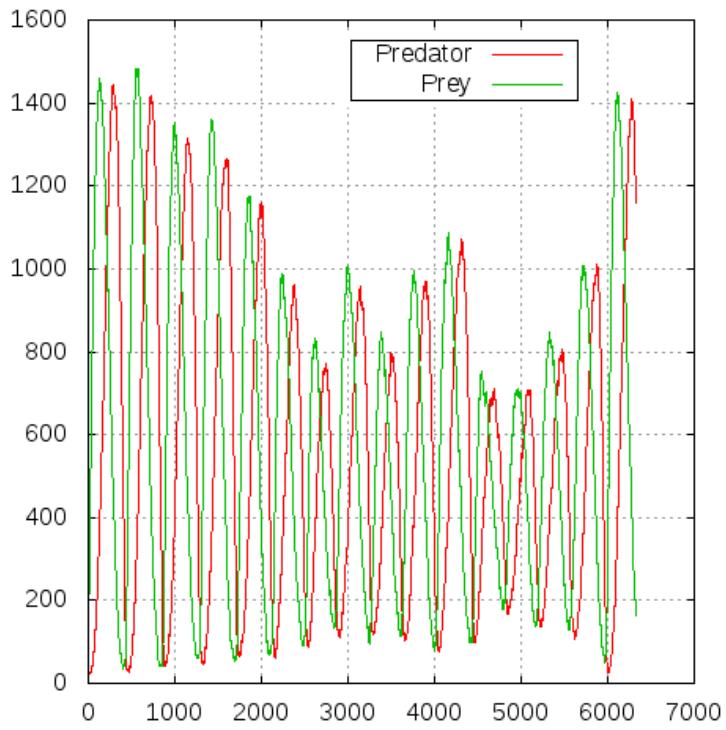


Abbildung 2: zeitliche Entwicklung der Populationen (Einzeltrajektorie)

Die nebenstehenden Graphiken zeigen eine mit dem Algorithmus berechnete Einzeltrajektorie; einmal als zeitlicher Verlauf (oben) und einmal im „Phasenraum“, welcher von den Populationgrößen aufgespannt wird. In diesem sieht man also die Kopplung der beiden Größen untereinander. Auch zeigt sich hier, abgesehen von dem spiralförmigen Verlauf, dass der berechnete Prozess offensichtlich dem oben betrachteten deterministischen Modell ähnlich ist, da die Prozesse in der gleichen leicht dreieckigen Form ablaufen. Allerdings ist die hier abgebildete Trajektorie für die in diesem Prozess genutzten Parameter eher untypisch. Mittelt man hier über viele Prozesse, so zeigt sich, dass meist die Räuber-Population mehr oder weniger bald ausstirbt. Dadurch, dass es sich hier um eine Einzeltrajektorie handelt, sieht man hier aber sehr gut die stochastischen Schwankungen, denen der Zufallsprozess unterliegt.

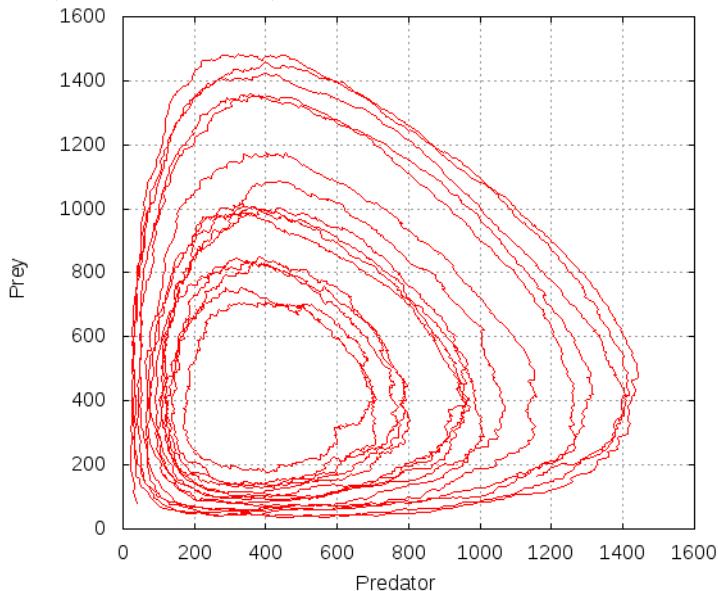


Abbildung 3: Entwicklung der beiden Populationen im Phasenraum (Einzeltrajektorie)

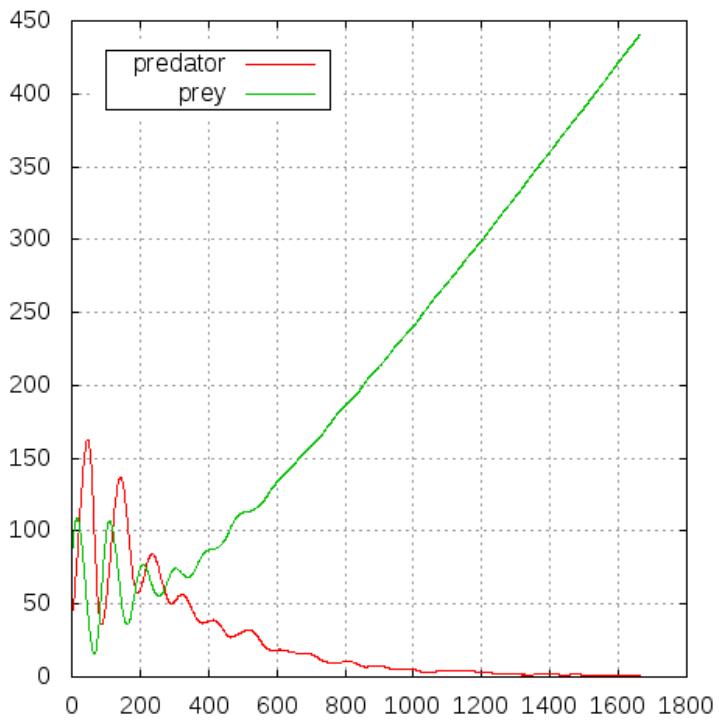


Abbildung 4: zeitliche Populationsentwicklung, wenn Räuber-Population ausstirbt

Die hier abgebildeten Graphiken (oben zeitlicher Verlauf, unten Trajektorie im Phasenraum) entstanden aus einer Mittlung über 1000 Simulationsdurchläufe. Dabei wurden die Parameter hier so gewählt, dass sie die in der deterministischen Betrachtung oben gefundenen Bedingungen $x = \frac{k_2}{k_{21}}$, $y = \frac{k_1 A}{k_{12}}$ genügen. Offensichtlich unterscheidet sich dieses Modell von dem deterministischen quantitativ recht stark. Die Parameter haben offensichtlich in diesem Modell doch eine etwas andere Bedeutung als in dem deterministischen Modell. Allerdings scheinen sie in ihrer Proportionalität denen oben zu entsprechen. So sind auch bei stabileren Durchläufen die Konstanten k_{21} und k_{12} um einige Größenordnungen kleiner als die anderen beiden.

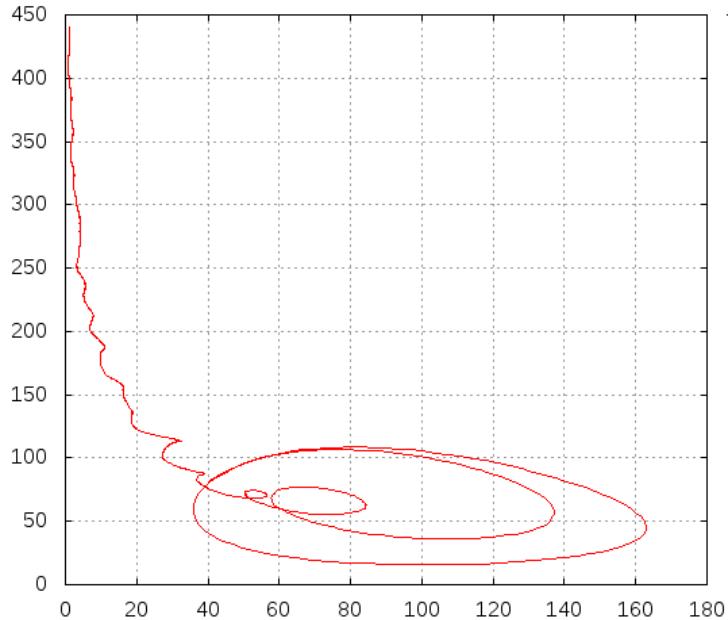


Abbildung 5: obige Entwicklung im Phasenraum

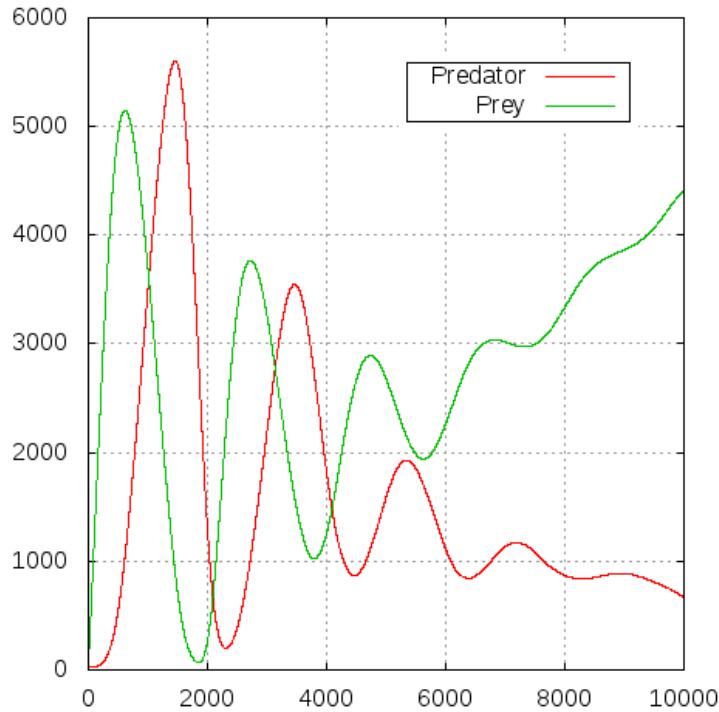


Abbildung 6: gemittelte Trajektorie in einem fast stabilen Zustand

Wie die nebenstehenden Bilder zeigen, hat nicht nur die gewählte Auswertungsmethode, welche zu sehr großen Fehlern führt (welcher jedoch hier außerdem durch einen mittlerweile behobenen Fehler im Programm kommt), sondern auch zu keinen schönen Zyklen führt. Dies liegt vor allem an den Anfangsbedingungen^a, welche in vielen Fällen keine stabilen Zyklen erzeugen. Im Wesentlichen sind die Parameter offensichtlich so gewählt, dass die Räuber-Population gegen Ende häufig ausstirbt, da die Beute-Population entsprechend ansteigt. Die Dämpfung wird jedoch, wie auch die Fehlerkurve nahelegt, vermutlich auch durch die Mittelung deutlich verstärkt.

^anature.berkeley.edu/biocon/BC Class Notes/ 73-77 Predator Models.pdf (am 07.08.2013)

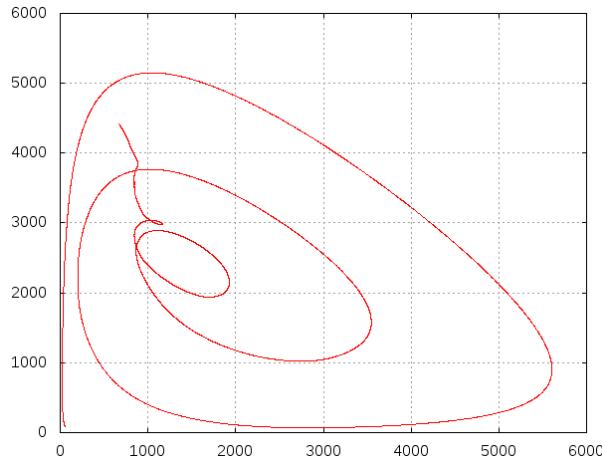


Abbildung 7: Trajektorie im Phasenraum

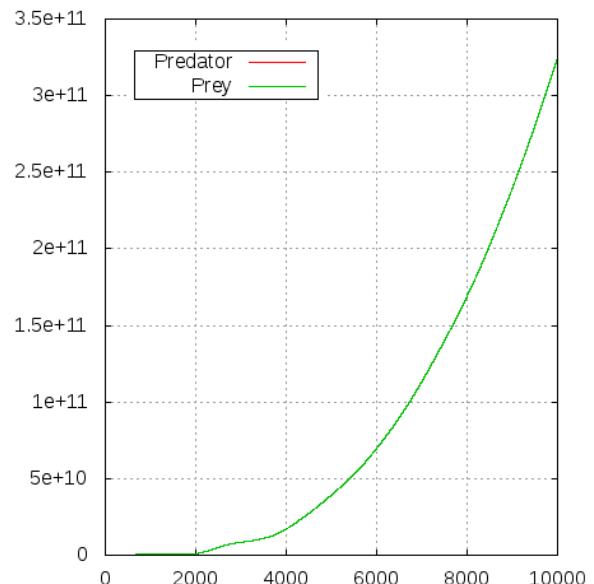


Abbildung 8: zeitliche Entwicklung der Standardabweichung im fast stabilen Zustand

Weitere Beispiele für Simulationsdurchläufe sowie die jeweils genutzten Parameter können unter <https://github.com/BalticPinguin/Prey-Predator-System> im Odner „Graphiken“ heruntergeladen werden.

5 Auswertung

Das Räuber-Beute-Modell ist offensichtlich ein sehr ergibiges Problem, welches sich, wie die obigen Betrachtungen zeigen, auf verschiedene Weise beschreiben und untersuchen lässt. Dabei hat jede Untersuchungsweise ihre Vor- und Nachteile; in der vorliegenden Arbeit konnte auch gezeigt werden, dass die beiden hier untersuchten Modelle einander doch recht ähnlich sind, da sie offensichtlich

äquivalente Ergebnisse liefern.

Im Weiteren wären Untersuchungen über die genauen Einflüsse der Parameter sowie Stabilisierungsversuche sehr interessant, da diese weiteren Aufschluss über das System geben würden. Im Rahmen dieser Arbeit musste darauf verzichtet werden, da dies einen Aufwand mit sich bringen würde und bei der vorliegenden Implementierung wohl allgemein zu aufwändig wäre. Die Suche nach geeigneten Parametern für eine langfristig stabile Simulation hat mir jedoch bereits gezeigt, dass diese Zusammenhänge recht komplex zu sein scheinen, was ihre Untersuchung um so interessanter gestalten würde.

6 Abbildungsverzeichnis

- Quellen Titelbild: http://iqa.evergreenps.org/science/biology/predator-prey_files/pred-prey.jpg (13.02.2013),
<http://www.globalchange.umich.edu/globalchange1/current/lectures/predation/tmp26.gif> (04.01.2013)
- Abbildung 1: R. Mahnke „Nichtlineare Physik in Aufgaben“ Verlag: B.G. Teubner, Stuttgart 1994; S. 125
- Abbildung 2-3: Parameter: 80, 40, 6500, 4.8, 2.0, 0.01, 0.01, 1
- Abbildung 4-5: Parameter: 80, 40, 1700, 5.6, 2.8, 0.07, 0.07, 1000
- Abbildung 6-8: Parameter: 80, 60, 50000, 11.2, 8.1, 0.011, 0.011, 1000

Hinweis zu der Parameterangabe: Reihenfolge der angegebenen Zahlen:

Beutepopulation, Räuber-Population (jew. Startwerte) , zu durchlaufende Zeiteinheiten, $k_1 A$, k_2 , k_{21} , k_{12} , Anzahl der berechneten Simulationen