Biomathematisches Praktikum am PC 2017

SIMULATION DES WATOR RÄUBER-BEUTE MODELLS UND ERWEITERUNGEN

8. Juni 2017

Sebastian von der Thannen
Thomas Heitzinger

Technische Universität Wien

Somewhere, in a direction that can only be called recreational at a distance limited only by one's programming prowess, the planet WATORswims among the stars. It is shaped like a torus, or doughnut, and is entirely covered with water. The two dominant denizens of Wa-Tor are sharks and fish, so called because these are the terrestrial creatures they most closely resemble. The sharks of Wa-Tor eat the fish and the fish of Wa-Tor seem always to be in plentiful supply.

Alexander Keewatin Dewdney 1984

1 Das klassische WATOR Programm

Das WATOR Programm (abgeleitet von Water-Torus) beschreibt ein einfaches Räuber-Beute Modell, die Beute - in unserem Fall Fische haben immer ausreichend Futter und keine natürliche Sterberate. Einzig die Räuber - in unserem Fall Haie können ihnen gefährlich werden, diese können ohne Fische als Nahrungsquelle nicht überleben und würden nach kurzer Zeit aussterben. Wäre das auch schon das ganze Modell, so könnte man diese Zusammenhänge durch die klassischen Lotka-Volterra Gleichungen beschreiben

$$\begin{aligned} x_B'(t) &= x_B(t)\,(\alpha - \beta x_R(t)) \\ x_R'(t) &= x_R(t)\,(\gamma x_B(t) - \delta) \end{aligned} \tag{1.1}$$

Das Wachstum der Beute $b(t)$ ist abhängig von der aktuellen Beutepopulation, sowie auf positive Weise von der Reproduktionsrate $\alpha > 0$, und negativ durch die von den Räuber verursachte Sterberate $\beta > 0$. Ganz ähnlich ist das zeitliche Verhalten der Räuberpopulation proportional zur Fressrate pro Beutelebewesen $\gamma > 0$ und zur natürlichen Sterberate $\delta > 0$ wenn keine Beute vorhanden ist.

Wir wollen die Sache jedoch ein wenig genauer wissen. Anstatt idealisierte stetige Populationsgrößen x_B, x_R anzunehmen, wollen wir tatsächlich eine diskrete Anzahl von einzelne Lebewesen simulieren, und deren Erfolg (oder Scheitern) von ihrem Aufenthaltsort auf WATOR, sowie anderen Lebewesen in unmittelbarer Nähe abhängig machen. Das WATOR Programm enthält einige einfache Regeln, die das Verhalten der Fische und Haie bestimmen. Der Ozean, in dem sie sich tummeln, besteht aus einem rechteckigen Gitter, dessen gegenüberliegende Seiten verbunden werden. Das heißt einfach, dass ein Fisch oder Hai der sich etwa in einer Zelle am rechten Rand befindet und nach rechts schwimmt, in der entsprechenden Zelle am linken Rand wieder auftaucht. Die Zeit vergeht in diskreten Zeitabschnitten die Chronen genannt werden, und jeder Fisch oder Hai darf sich pro Chrone um eine Zelle entweder nach Norden, Osten, Süden

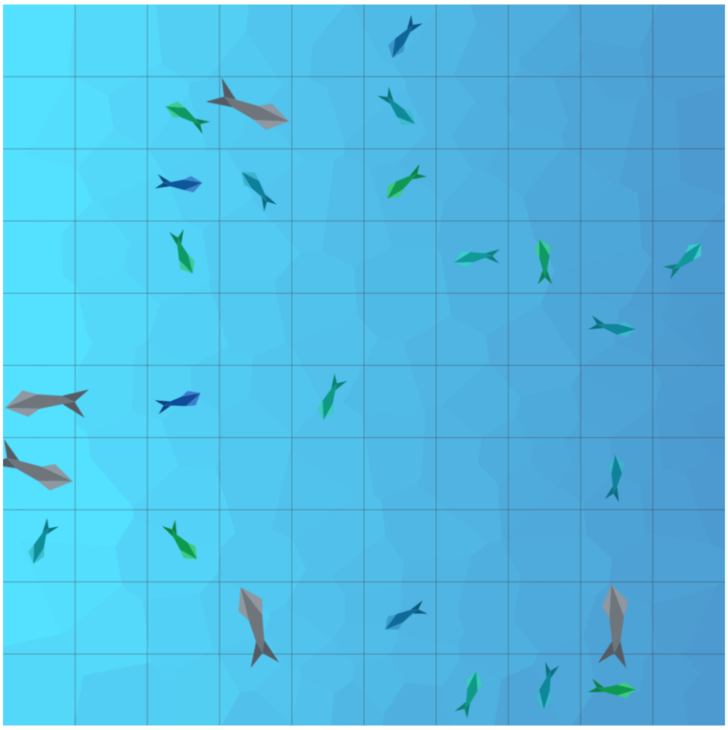

Abbildung 1: Ein typischer Tag auf WATOR

oder Westen bewegen. Falls jedoch bereits alle benachbarten Zellen durch die eigene Spezies besetzt sind, so entfallen diese Regeln. Die Wahl der Bewegungsrichtung ist für Fische ganz einfach: Wähle zufällig. Haie haben aufgrund ihrer Natur als Jäger jedoch ein wenig mehr zu beachten. Es gilt: Befindet sich in den benachbarten Zellen ein oder mehrere Fische, so wähle eine dieser Zellen und bewege dich dort hin – der Fisch wird dabei gefressen. Falls diese Regel nicht anwendbar ist, so verhalte dich wie ein Fisch und wähle zufällig.

Zusätzlich zu diesen Bewegungsregeln dürfen sich unsere WATOR Bewohner unter bestimmten Voraussetzungen vermehren. Für unsere Fische führen wir dafür einen zusätzlichen Parameter `fbreed` ein, welcher die Zeit – also die Anzahl der Chronen – angibt die die Fische am Leben sein müssen bevor sie sich vermehren dürfen. Ganz analog verwenden wir für die Haie den Wert `hbreed` und noch einen zusätzlich Parameter `starve` der angibt wie viele Chronen ein Hai überleben kann ohne gefressen zu haben. Für eine erfolgreiche Simulation ist es schlussendlich noch notwendig initiale Parameter für den Anbeginn der Zeit, also $t = 0$ zu wählen. Das sind genau die Anzahl der Fische und Haie und deren Position auf WATOR. Die Positionen werden wir in unserer Simulation ganz einfach zufällig wählen, und für die initiale Anzahl von Fischen und Haien verwenden wir die Parameter `nfish` und `nshark`.

Von den idealisierten Lotka-Volterra Gleichungen würden wir uns periodische, fast Sinusähnliche, Populationsverläufe erwarten. In der Realität sieht das ganze jedoch naturgemäß komplizierter aus, insbesondere wenn wir nur einen kleinen Planeten simulieren spielt der Zufall eine große Rolle. In Abbildung 3 sehen wir einen typischen Verlauf der Populationen über 500 Chronen. Es ist deutlich ersichtlich, wie die beiden Populationen wesentlich durch die jeweils andere beeinflusst werden. Während die Haie besonders durch vermehrtes Fischvorkommen gedeihen, gilt gerade das Gegenteil für die Fische bei starkem Haizuwachs. Zudem bemerken wir, dass die Maxima der Räuberpopulation relativ konstant etwa 10% – 15% der Maxima

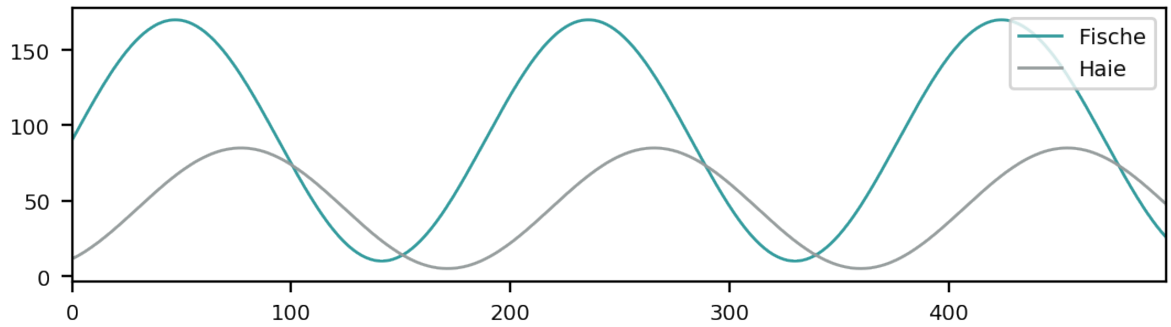

Abbildung 2: Die Lotka-Volerra Gleichungen prognostizieren zwei zeitverschobene, periodische Populationsverläufe.

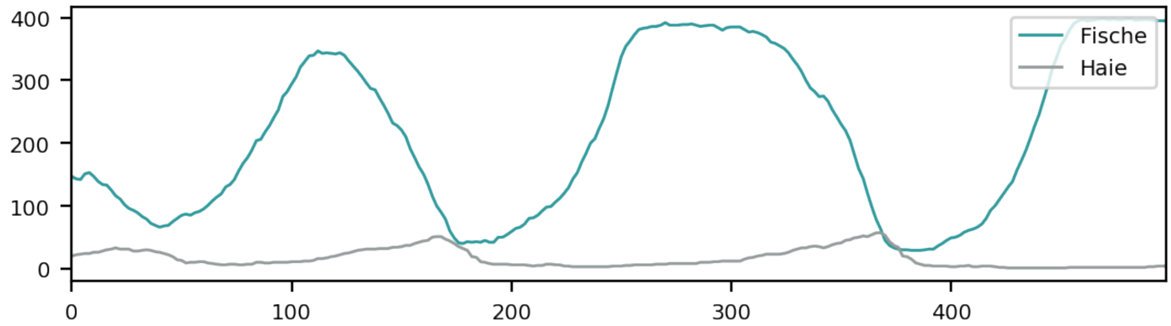

Abbildung 3: Beispiel eines Polulationsverlaufs auf einem kleinen Planeten mit 20×20 Zellen.

der Beutepopulation entspricht, es wird also trotz starkem Einfluss durch zufällige Ereignisse in den vielen Fällen zumindest für einen gewissen Zeitraum ein Äquilibrium erreicht. Dieser annähernd stabile Zustand wird jedoch meist durch einen von zwei Fällen beendet. Im ersten Fall versäumen es die Jäger sich ausreichend über den gesamten Planeten zu verteilen, wodurch nach kurzer Zeit das lokale Beuteaufkommen zur Neige geht und die Jäger verhungern. Daraufhin können sich die Fische ungehindert vermehren, bis der gesamte Planet WATOR mit Fischen bedeckt ist. Im zweiten Fall gelingt es den Haien sich bei der Jagd so gewitzt anzustellen, dass schlussendlich jeder einzelne Fisch gefressen wird, was wiederum wie im ersten Fall den Untergang der Jägerpopulation zur Folge hat. Sofern sich kein perfektes Äquilibrium bildet, ist das Schicksal der Jäger hier also immer früher oder später auszusterben.

Dieses Schicksal der Jäger kann durch geschickte Wahl der fünf Parameter `fbreed`, `hbreed`, `starve`, `nfish` und `nshark`, sowie insbesondere durch die Simulation eines größeren Planeten hinausgezögert werden. Tatsächlich ist es möglich Parameter so zu wählen, dass die beiden Populationen Stunden, oder sogar Tagelang stabil bleiben. In Abbildung 4 sehen wir etwa die Simulation eines Planeten der Größe 45×45. Während auch hier die Anzahl der Haie relativ zu den Fischen nur etwa 10% ausmacht, sieht es hier zumindest absolut gesehen deutlich besser aus. Im Allgemeinen sind hier zu jedem Zeitpunkt mindestens noch 50 Haie am Leben, wodurch ein Aussterben statistisch zumindest auf kurze Zeit gesehen unwahrscheinlich wird.

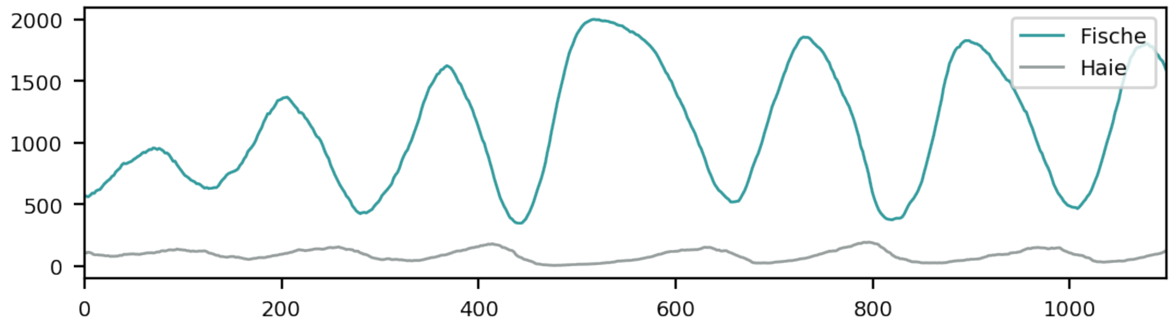

Abbildung 4: Beispiel eines Polulationsverlaufs auf einem größeren Planeten mit 45×45 Zellen.

2 Das stetige WATOR Programm

Im ersten Schritt ließen wir die Bewohner auf einem vordefinierten festen Gitter schwimmen. Das führt allerdings dazu, dass es eine vom Gitter induzierte natürliche Beschränkung für die Anzahl an Bewohnern gibt. Die Populationsschwankung ist somit deutlich eingeschränkt und stößt fast immer an ihre Grenzen. Ziel ist es also eine stabile Räuber-Beute-Beziehung zu modellieren, bei der im Idealfall niemand ausstirbt und die Populationsschwankung nur in Ausnahmefällen künstlich nach oben beschränkt werden muss. Dazu verringern wir die Gitterbreite auf ein Minimum. Somit schwimmen die Bewohner nun auf den einzelnen Pixel. Zusätzlich lassen wir zu, dass sich die Bewohner auch zur gleichen Zeit auf den selben Koordinaten befinden können. Diese Anpassung bringt aber einige Probleme mit sich, wie man schnell bemerkt.

Zunächst würden Haie nun aussterben, suchten sie nur auf den benachbarten Pixeln nach Nahrung, Sie wären also sozusagen blind und würden erst dann einen Fisch fressen, wenn er sie berührt. Deshalb definieren wir eine Suchtoleranz, sodass Haie nach Fischen suchen, die sich innerhalb dieser befinden. Findet ein Hai zum gegebenen Chrone Fische, welche sich in dieser Umgebung aufhalten, so wählt er einen dieser Fische zufällig aus und macht ihn zu seiner Nahrung.

Ein weiteres Problem stellen die Bewegungsabläufe dar. Zuvor ließen wir die Bewohner zufällig in eine der vier Himmelsrichtungen schwimmen. Beim stetigen WATOR haben wir aber weit mehr mögliche Schwimmrichtungen. Man könnte als erstes versuchen für jedes Tier in jedem Schritt eine gleichverteilte Zufallszahl zu erzeugen, welche den Winkel der Schwimmrichtung bestimmt. Lässt man die Simulation nun mit diesen Änderungen laufen, so sieht das Ergebnis äußerst ernüchternd aus. Sowohl die Haie als auch die Fische schwimmen orientierungslos umher und bewegen sich eher wie die Nadel eines Kompasses, die von einem Magneten gestörte wird. Damit diese unnatürlichen Bewegungen vermieden werden, könnte man zum Beispiel die neue Schwimmrichtung so wählen, dass sie einen zufällig gewählten aber beschränkten Winkel mit der alten Schwimmrichtung einschließt.

Allerdings stellen uns all diese Nachbesserungen immer noch nicht zufrieden. Die Fische schwimmen jetzt zwar natürlicher aber dennoch sehen sie recht verloren und orientierungslos aus. Wir wollen daher einen Schritt weiter gehen und die Simulation so realistisch wie möglich machen. Also implementieren wir ein Schwarmverhalten für die Fische, um sie mit ein wenig mehr Intelligenz zu versehen.

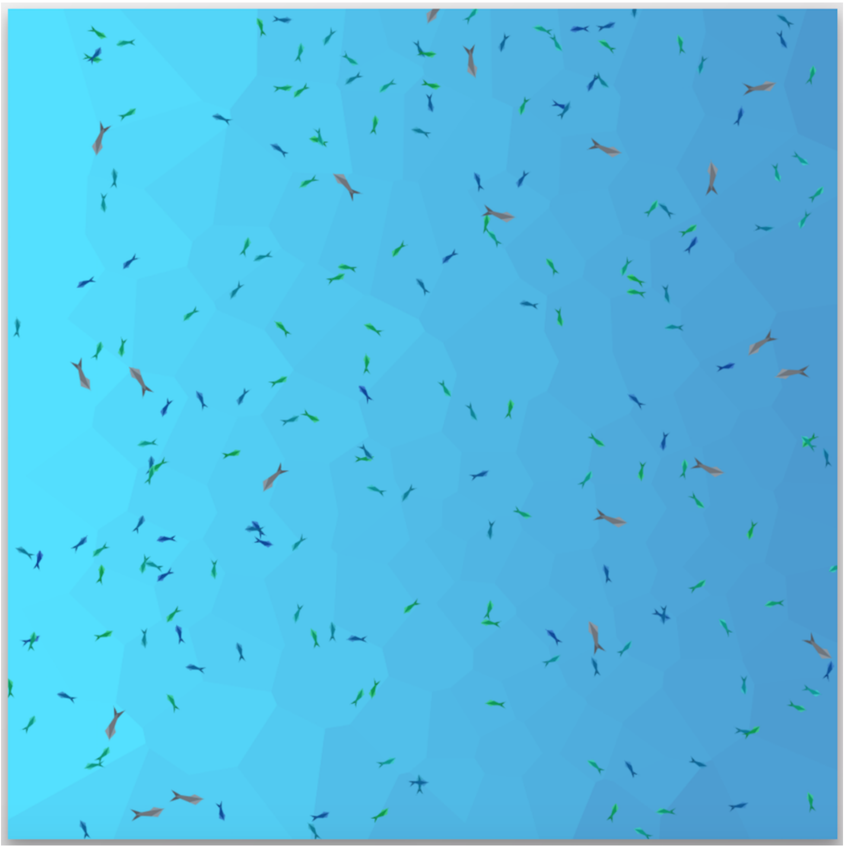

Abbildung 5: Das stetige Wator

2.1 Schwarmverhalten

Als Modell für das Schwarmverhalten nehmen wir das verfeinerte Cucker-Smale Modell nach Agueh et al. [1]. Die Bewegungsänderung für jedes Individuum wird dabei durch ein System von nichtlinearen Differentialgleichungen beschrieben.

$$\begin{aligned}
&\frac{d}{dt}x_i = v_i \\
&\frac{d}{dt}v_i = R_i^f + R_i^s + A_i + B_i + (\alpha - \beta|v_i|^2)v_i,
\end{aligned}$$

wobei R_i^f der Abstoßungsterm für das i-te Individuum von den anderen Fischen, R_i^s der Abstoßungsterm für das i-te Individuum von den Haien, A_i der Anziehungsterm für das i-te Individuum zu den anderen Fischen, B_i die Randkraft für jedes Individuum und letzter Term die Selbstantriebskraft des jeweiligen Individuums ist.
Genauer:

2.1.1 Selbstantrieb

Der Selbstantriebsterm $(\alpha - \beta|v_i|^2)v_i$ bestimmt die Geschwindigkeit des Individuums, welche es annehmen würde, wäre kein anderes Individuum in der Nähe. Der Parameter α repräsentiert den Selbstantrieb des Individuums, also den Wunsch, sich in einer gewissen Geschwindigkeit fort zu bewegen. Der Parameter β ist hingegen ein Reibungswiderstand, der abbremsend wirkt. Man sieht sofort, dass

$$|v| = \sqrt{\frac{\alpha}{\beta}} \tag{2.1}$$

eine stationäre Lösung von

$$\frac{d}{dt}v_i = (\alpha - \beta|v_i|^2)v_i \tag{2.2}$$

ist.

2.1.2 Abstoßung

Für das Verhalten der Fische wollen wir nicht nur, dass sie sich gegenseitig abstoßen, falls sie sich zu nahe kommen, sondern auch, dass sie in der Lage sind von Haien davon zu schwimmen, falls diese ihnen zu nahe kommen. Deswegen Verwenden wir zwei verschiedene Abstoßterme

$$R_i^{f,s} := \frac{\rho}{N_{f,s}} \sum_{j=1}^{N} S_0(|x_i - x_j|, k_{f,s}, d_{f,s}) \frac{x_i - x_j}{(1 + |x_i - x_j|^2)^\gamma},$$

wobei ρ und γ die Stärke der Abstoßung bestimmen und mit einer Cutoff Funktion (Abbildung 6)

$$S_0(x, k, d) := \frac{1}{2}(1 - \tanh(k(x - d)))$$

für die Distanz, wobei d die Position und k die Schärfe des Cutoffs bestimmt. Die abstoßende Kraft zwischen zwei Individuen wird dadurch abgeschnitten, wenn ihr Abstand größer als d ist.

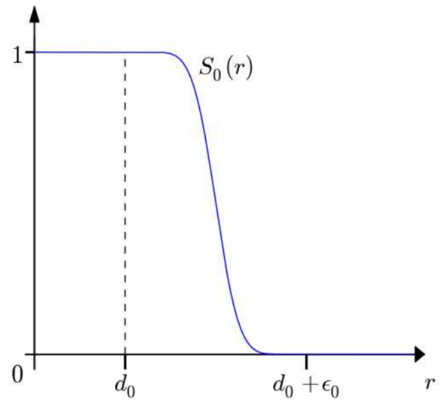

Abbildung 6: Cutoff-Funktion [1]

2.1.3 Anziehung

Der Term für die Anziehung ist folgendermaßen definiert:

$$A_i := \frac{1}{N}\sum_{j=1}^{N}(1 - S_0(|x_i - x_j|))w(x_i - x_j, v_i)(x_j - x_i),$$

mit Positions-Cutoff S_0 für die Distanz zweier Individuen und einer Funktion w für den Sichtkegel

$$w(x, v) := \frac{\gamma}{(q + |x|^2)^\sigma}\left(1 - (1 - S_0(|v|, k_1, d_1)) \cdot S_0(\frac{x \cdot v}{|x||v|}, k_2, d_2)\right),$$

wobei γ und σ die Anziehungsstärke bestimmen und die Cutoff-Funktionen die Anziehung zum einen durch die Geschwindigkeit und zum anderen durch den Sichtkegel des Individuums abschneiden. Es gilt $\cos(\phi) = \frac{x \cdot v}{|x||v|}$. Ist also $\cos(\phi)$, wobei ϕ der Winkel zwischen Schwimmrichtung v_i und dem Richtungsvektor $x_i - x_j$ zum gesehenen Individuum definiert, größer als d_2, der Winkel ϕ also klein genug, so sieht der Fisch den Anderen und der Anziehungsterm wird dazu addiert. Bei kleinen Geschwindigkeiten $|v_i| < d_1$ verschwindet die Geschwindigkeits-Cutoff-Funktion $1 - S_0(|v_i|, k_1, d_1)$ und der der Sichtkegel wird ignoriert.

2.1.4 Randkräfte

Die Randkraft bewirkt, dass Individuen, welche sich zu weit von der Gruppe entfernt haben, ihre Richtung ändern um den Schwarm wiederzufinden.

$$B_i := CS_0(\rho_i, k_3, d_3) \cdot v_i^\perp,$$

mit $\rho_i = \sum_{j=1}^{N} \frac{1}{1+|x_i - x_j|^2}$ und C die Stärke der Randkräfte bestimmt. Hierbei ist ρ_i ein Maß für die Einsamkeit eines Fisches. Je kleiner ρ_i, desto weiter befindet er sich weg vom Schwarm. Damit verschwindet B_i, falls $\rho_i > d_3$.

Somit wird nun in jedem Zeitschritt für jeden Fisch eine Differentialgleichung gelöst, was das ganze Unterfangen wesentlich aufwändiger macht. Wir müssen natürlich wieder eine künstliche Beschränkung für die Anzahl an Fischen definieren, da beim möglichen Aussterben der Haie und beim unendlichen Vermehren der Fische der Rechenaufwand und Speicher explodieren würde. Zusätzlich geben wir den Fischen ein Alter. Fische die ein gewisses Alter überstreichen, indem

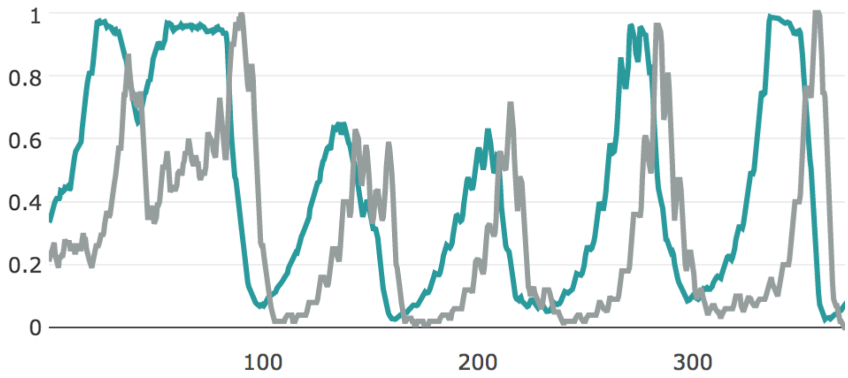

Abbildung 7: Typischer Populationsverlauf beim stetigen WATOR

sie nicht gefressen wurden, sterben. Das macht die Differentialgleichung stabiler, da Fische hoffentlich sterben, bevor durch die Fehlerfortpflanzung Geschwindigkeiten explodieren können.

Im Zuge dessen müssen wir die Haie nun ebenfalls intelligenter machen, da jetzt die Chancen zufällig einen Fisch in der Nähe zu finden geringer sind. Wir lassen sie nun jagen. Zu jedem Zeitschritt sucht sich jeder Hai jeweils den zu seiner Position nächsten Fisch aus, um ihn zu jagen. In jedem Zeitschritt wird also ein neuer Fisch ausgesucht, falls der zuvor ausgewählte nicht mehr der nächste war.

Mit diesen Erweiterungen gelingt es tatsächlich meist stabile Populationsschwankung zu erhalten, ohne dass eine Spezies ausstirbt. Dabei treten Populationsverläufe (Abblindung 7 mit zum jeweiligen Populationsmaximum relativen Bevölkerungszahlen) auf, welche stark an jene der Lotka-Volterra Gleichungen erinnern.

Abbildungsverzeichnis

Literatur

[1] Martial Agueh, Reinhard Illner und Ashlin Richardson. “Analysis and simulations of a refined flocking and swarming model of Cucker-Smale type”. In: *Kinetic and Related Models* 4.1 (2011), S. 1–16.

[2] Alexander K. Dewdney. “Computer Recreations - Sharks and fish wage an ecological war on the toroidal planet Wa-Tor”. In: *Scientific American* 12 (1984), S. 14–22.