Biomathematisches Praktikum am PC 2017

SIMULATION DES WATOR RÄUBER-BEUTE MODELLS UND ERWEITERUNGEN

31. Mai 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 WATOR swims 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,

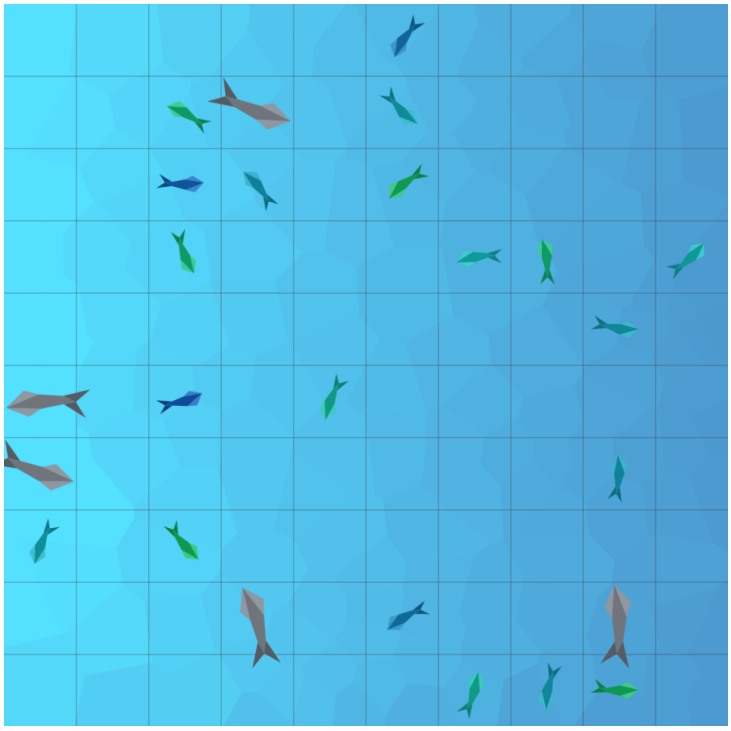


Abbildung 1: Ein typischer Tag auf WATOR

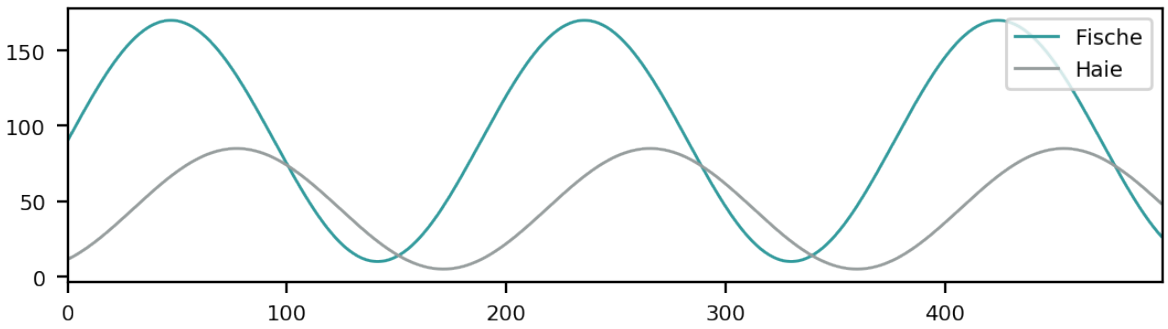
Süden oder Westen bewegen. Falls jedoch bereits alle benachbarten Zellen durch die eigene Spezies besetzt sind, so entfällt diese Regeln. Die Wahl der Bewegungsrichtung ist für Fisch ganz einfach: Wähle zufällig. Haie haben aufgrund ihrer Natur als Jäger jedoch ein wenig mehr zu beachten. Es gilt: Befindet sich in einer benachbarte 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.

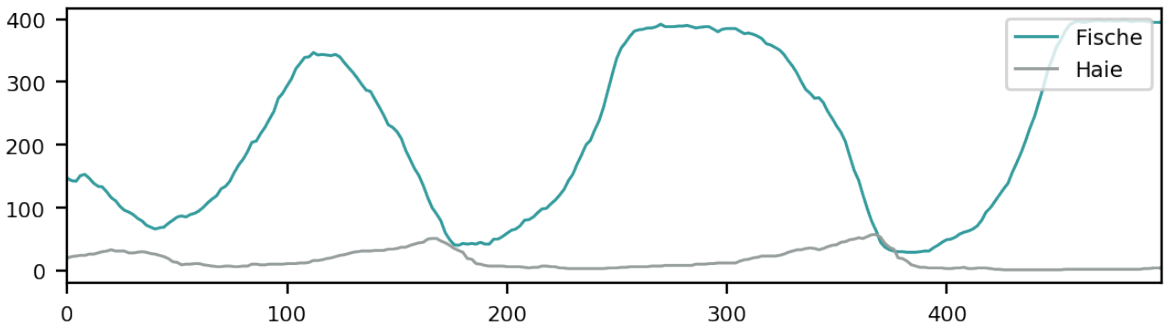
Zusätzlich zu diesen Bewegungsregeln dürfen sich unsere WATOR Bewohner unter bestimmten Voraussetzungen vermehren. Für unsere Fische führen wir dafür einen zusätzlichen Parameter `fbreed` ein, welcher die Zeit – also die Anzahl der Chronen – angibt die die Fische am Leben sein müssen bevor sie sich vermehren 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`.

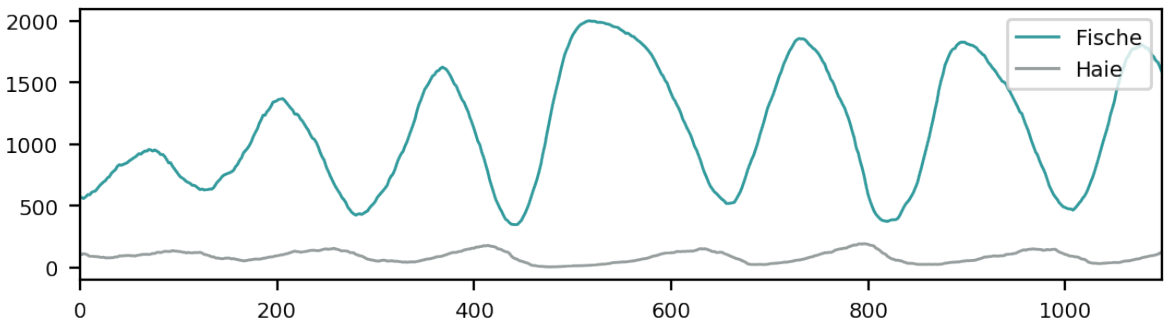
Von den idealisierten Lotka-Volterra Gleichungen würden wir uns Sinusähnliche Populationsverläufe erwarten. In der Realität sieht das ganze jedoch naturgemäß komplizierter aus, insbesondere wenn wir nur einen kleinen Planeten simulieren spielt der Zufall eine große Rolle.

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


Fische
Haie
150
100
50
0
0
100
200
300
400


Fische
Haie
400
300
200
100
0
0
100
200
300
400


Fische
Haie
2000
1500
1000
500
0
0
200
400
600
800
1000

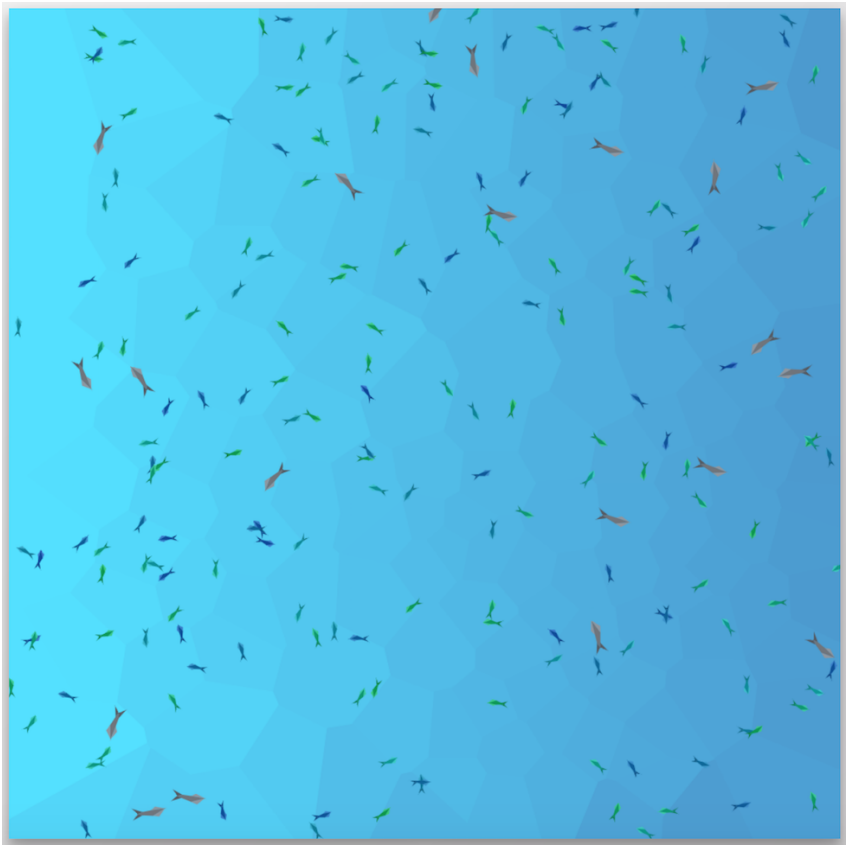


Abbildung 2: Das stetige WATOR

an Bewohnern gibt. Die Populationsschwankung ist somit 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 beschränkt wird. 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 gestörter 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 waren wir nach all diesen Nachbesserungen immer noch nicht zufrieden. Die Fische schwammen jetzt zwar natürlicher aber dennoch sahen sie recht verloren und orientierungslos aus. Wir wollten jedoch einen Schritt weiter gehen und die Simulation so realistisch wie möglich machen. Also implementierten wir ein Schwarmverhalten für die Fische, um sie mit ein wenig mehr Intelligenz zu versehen.

2.1 Schwarmverhalten

Als Modell für das Schwarmverhalten nahmen wir das verfeinerte Cucker-Smale [**agueh2011analysis**]. 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 Eigenantriebskraft des jeweiligen Individuums ist.
Genauer:

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

wobei ρ und γ die Stärke der Abstoßung bestimmen und mit einer Cutoff Funktion für die Distanz

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

wobei d die Position und k die Schärfe des Cutoffs bestimmt. Weiters ist der Anziehungsterm

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

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

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

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. Der Term für die Randkräfte

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

mit $\rho_i = \sum_{j=1}^{N}\frac{1}{1+|x_i - x_j|^2}$ und C die Stärke der Randkräfte bestimmt. Natürlich mussten wir die Haie nun ebenfalls intelligenter machen. Wir ließen sie nun jagen. Zu jedem Zeitschritt sucht sich jeder Hai jeweils den zu seiner Position nächsten Fisch aus, um ihn zu jagen.

3 Animationen

Abbildungsverzeichnis