Konstruktion ökologischer Modelle mit R

Konstruktion ökologischer Modelle mit der Open Source Software R

Thomas Petzoldt

Th. Petzoldt
Institut für Hydrobiologie
Technische Universität Dresden
01062 Dresden
GERMANY

thomas.petzoldt@tu-dresden.de

Version vom 10. Januar 2017
Beispiele und Datensätze auf können beim Autor erfragt werden:
http://www.tu-dresden.de/Members/thomas.petzoldt/

Kurze Vorrede

Dieses Tutorial entstand im Rahmen eines Workshops „Modellierung für Limnologen“ an der TU Dresden. Das Ziel war und ist das Kennenlernen ausgewählter Methoden und Techniken, die für die Konstruktion ökologischer Modelle eingesetzt werden können. Hierbei steht eindeutig der methodische Aspekt, die Vorstellung von Grundprinzipien und das Sammeln eigener Erfahrungen im Vordergrund. Alle Beispiele sind im Quelltext angegeben und laden zur kreativen Weiterentwicklung ein.

Das Tutorial wird gelegentlich aktualisiert, deshalb sind Kritik und Hinweise außerordentlich willkommen. Ich bedanke ich mich an dieser Stelle bei meinem wissenschaftlichen Lehrer, Prof. Jürgen Benndorf sowie bei Susanne Rolinski, Karsten Rinke, Jens Schmidt und Christof Bigler für Hinweise zu verschiedenen Kapiteln des Tutorials. Ganz besonderer Dank gebührt dem R-Core Team und der R-Community für ihre exzellente Arbeit, die Weiterentwicklung, die Unterstützung und den Spaß den das R-System bei der Erschließung neuen Terrains bietet.

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Vom System zum Modell

Die vorliegende Einführung verfolgt das Ziel, einige ausgewählte Techniken der ökologischen Systemanalyse und Modellierung und ihre Anwendung in Forschung und Praxis an Hand ausgewählter Beispiele vorzustellen, die zum Teil aus der Gewässerökologie stammen. Es kann hier nur ein kleiner Ausschnitt vorgestellt werden, da die Begriffe Systemanalyse und Modellierung sehr breit verwendet werden und nicht nur innerhalb der Ökologie, sondern auch darüber hinaus durch viele unterschiedliche Auffassungen, Schulen und Paradigmen gekennzeichnet sind.

Die in der Systemanalyse benutzte Denkweise ist hierbei eigentlich nichts besonders Spezielles, sondern die ganz normale Methode, die fast alle Naturwissenschaftler benutzen, um ihre Untersuchungsobjekte zu beschreiben, zu systematisieren und zu verstehen. Wenn wir hier speziell von Systemanalyse und Modellierung sprechen, dann heißt das lediglich, dass wir uns diese Methode besonders bewusst machen und einige formale Mittel kennenlernen wollen.

Der gewählte Ansatz ist relativ pragmatisch. Im Vordergrund stehen nicht spezielle mathematische Methoden oder einzelne Modellparadigmen, sondern eine Denkweise, die versucht, vorhandenes Wissen zusammenzufassen, zu quantifizieren und zum Verständnis der untersuchten natürlichen Systeme zu nutzen. Die hier vorgestellte Methodik soll am Ende einen kleinen Werkzeugkasten ergeben, der es ermöglichen soll, eigene Modelle zu konstruieren und die hinter einer Reihe von Modellen steckende Denkweise besser zu verstehen. Ich hoffe sehr, dass die erworbenen Grundkenntnisse dabei helfen, den Einstieg in weiterführende Literatur zu erleichtern.

1.1 Systeme

Unter einem System verstehen wir einen nach bestimmten Gesichtspunkten abgegrenzten Ausschnitt der Welt, der aus Elementen und Relationen (Kopplungen) besteht. Alles, was nicht zum System gehört, bezeichnen wir als Umwelt, die Grenze als Systemrand. Die Abgrenzung geschieht dabei so, dass die unter dem jeweiligen Gesichtspunkt interessierenden Beziehungen zwischen Systemelementen möglichst innerhalb des Systems und nur wenige Beziehungen nach außen (zur Umwelt) verlaufen. Die Abgrenzung kann dabei räumlich aber auch nach beliebigen anderen Kriterien erfolgen. So können wir z.B. einen Wasserfloh (*Daphnia*) als Element in einem See also räumlich betrachten oder das System *Daphnia* als genetische Einheit oder als physiologische oder systematische Einheit (*Crustacea, Cladocera*). Darüberhinaus kann man die Elemente eines Systems wiederum als Systeme (Subsysteme) auffassen, z.B. die Zellen der *Daphnia*. Betrachten wir die *Daphnia* nur als „schwimmende Filtriermaschine“ und interessiert uns die Physiologie nicht, entsteht auch hier ein Systemrand.

Diese oder vielleicht auch eine leicht modifizierte Systemsicht ist Bestandteil fast jeder Wissenschaft und somit natürlich auch ein generelles Merkmal der Ökologie allgemein, d.h. die Ökologie untersucht ökologische Systeme (Systemökologie, früher oft Synökologie) und deren Subsysteme. Hierbei betrachten wir nicht in jedem Fall gleich ganze Ökosysteme, sondern oft lediglich einzelne Lebensgemeinschaften, Populationen oder einzelne Organismen (Autökologie) mit ihrem spezifischen Verhalten, ihren physiologischen Leistungen, ihrer ontogenetischen Entwicklung oder ihrer Evolution in einer biotisch und abiotisch geprägten Umwelt.

Auch die Begriffe Element und Relation können auf ganz unterschiedliche Weise mit Leben gefüllt werden. So existieren nicht nur Stoff-, Energie- oder Informationsbeziehungen zwischen Organismen, sondern z.B. auch systematische Beziehungen (*Daphnia – Cladocera*), Beziehungen zwischen Teil und Ganzem (Filterbein einer *Daphnia*), Mengenrelationen (*Daphnia* als Element der Menge aller Wasserorganismen) oder semantische Relationen zwischen Begriffen (*Daphnia* **lebt im** Wasser).

1.2 Modelle

Ein Modell ist ebenfalls ein System, dass sich dadurch auszeichnet, dass es die im Hinblick auf eine bestimmte Fragestellung wichtigen Eigenschaften eines anderen Systems („reales System“) widerspiegelt. Hierbei wird von den für die Fragestellung unwesentlichen Eigenschaften des „realen Systems“ abstrahiert. Kurz gesagt, ein Modell ist durch die Eigenschaften Abbildcharakter, Abstraktion (oder Vereinfachung) und Zweckgebundenheit charakterisiert.

Daraus folgt, dass ein Modell ein künstlich geschaffenes oder ein natürliches Gebilde sein kann, z.B. der Bodensee als Modell für ein tiefes geschichtetes Gewässer, der Laborversuch als Modell für das Untersuchungsgewässer oder das mathematische Modell für den Laborversuch. Wir sehen daran, dass auch die Beziehung zwischen System und Modell relativ ist. Da auch Modelle ihrerseits Systeme sind, existieren auch Modelle von Modellen. Diese werden manchmal Metamodelle genannt. Darüberhinaus kann es mehrere unterschiedliche Modelle von ein und demselben System geben, so wie die unterschiedlichen Ansichten einer architektonischen Zeichnung. Das ist ein wichtiger Aspekt und heißt für uns ganz konkret, dass die Abbildung eines Systems in Modellen dem jeweiligen Zweck untergeordnet sein sollte. Ein typisches Verhalten, gerade bei Biologen, ist es, zu viele Details in einem einzigen Modell vereinen zu wollen, gewissermaßen auf Verdacht oder zur Gewissensberuhigung, weil ja die Natur so viel komplizierter ist als jedes Modell. Allerdings führt eine solche Herangehensweise fast immer in die Sackgasse, weil ein solches überkompliziertes Modell dann auch nicht leichter zu verstehen ist, als das natürliche Original. Stattdessen ist es viel sinnvoller und auch einfacher, zunächst mehrere kleine, gut überschaubare Modelle zu entwickeln.

Der Modellbegriff ist hierbei zunächst sehr weit gefasst und beinhaltet z.B. auch:

Gedankenmodelle: werden z.B. in Vorlesungen vermittelt oder in Publikationen beschrieben.

Maßstabsmodelle: z.B. Legosteine, Modelleisenbahn, Architekturmodelle

Labormodelle: Batchkultur, Chemostat, Laborkläranlage, Enclosure

Gedankenmodelle, Maßstabsmodelle oder Labormodelle sind zwar nicht Gegenstand dieser Einführung, jedoch eröffnen solche Modelle oft erst den Weg zur mathematischen Beschreibung von Ökosystemen.

1.3 Mathematische Modelle

Mathematische Modelle sind nicht gegenständlich und zeichnen sich durch einen hohen Grad an Formalisierung und Abstraktion aus. Es existieren verschiedene Methoden, um Systeme zu beschreiben, z.B. grafisch, mit algebraischen Gleichungen, mit Differentialgleichungen oder Regeln, empirisch oder analytisch, diskret oder kontinuierlich, aggregiert oder individuenbasiert, ereignisorientiert, stochastisch oder deterministisch, stationär oder instationär, statisch oder dynamisch.

Zur Formalisierung dienen nicht nur numerische Methoden, sondern z.B. auch Mengenlehre, Logik oder grafische Verfahren, z.B. semantische Netze, Flussdiagramme oder UND-ODER-Grafen.

Aufgrund der unterschiedlichen Herkunft vieler Modellansätze und der Vielfalt der Methoden existiert kein einheitlicher Stammbaum mathematischer Modelle. Stattdessen haben sich eine Reihe von Kategorien und Begriffspaaren eingebürgert, die zur Klassifikation von Modellen verwendet werden können:

Dynamische – statische Modelle: Dynamik erfordert zeitliche Variabilität. Statische Modelle hingegen beschreiben Fließgleichgewichte (*steady state*) oder das mittlere Verhalten eines dynamischen Systems. Dynamik sollte nicht mit Kinetik verwechselt werden. Kinetiken sind zwar als zeitliche Folge aufnehmbar, bezeichnen aber eine Geschwindigkeit als Momentaufnahme (Bsp. Reaktionskinetik).

Diskrete – kontinuierliche Modelle: Bei einem diskreten Modell läuft die Zeit in einem festen oder einem variablen Raster von Zeitschritten ab, bei einem kontinuierlichen Modell ohne Zeitsprünge. Zur Beschreibung kontinuierlicher Systeme verwendet man häufig Differentialgleichungen.

Deterministische – stochastische Modelle: Bei einem deterministischen Modell spielen Zufallsereignisse keine Rolle, jeder Lauf des Modells erzeugt in Abhängigkeit von den Parametern und Randbedingungen immer dasselbe Ergebnis. Deterministische Modelle beschreiben ein System *eindeutig* oder *mechanistisch* wie ein Uhrwerk. Eine bestimmte Konstellation an Eingangsgrößen hat immer denselben Systemzustand zur Folge (Bsp. Fallgesetz). In stochastischen Modellen wird der Einfluss des Zufalls (natürliche Variabilität und Fehler) explizit oder implizit berücksichtigt, wir erhalten Wahrscheinlichkeiten oder Wahrscheinlichkeitsverteilungen (Bsp. Lebensdauer einer Glühlampe als Poissonverteilung, Regenwahrscheinlichkeit bei der Wettervorhersage).

On-line – off-line-Modelle: Ein on-line Modell (Echtzeitmodell, real-time model) wirkt direkt auf das simulierte System zurück (z.B. Regler, adaptive Modelle).

Lineare – nichtlineare Modelle: Das Bezeichnungspaar linear/nichtlinear wird unterschiedlich verwendet. So kann ein lineares Differentialgleichungsmodell durchaus eine nichtlineare Kurve hervorbringen (z.B. Streeter-Phelps-Modell der Sauerstoffganglinie in Fließgewässern) und auch „nichtlineare“ Regressionsmodelle können auf einem linearen Gleichungssystem aufbauen, d.h. intrinsisch linear sein.

Empirische – analytische Modelle: Diese Einteilung berücksichtigt den Weg, wie man zu einem Modell kommt: Entweder durch bloße Gegenüberstellung von Beobachtungen (empirisch) oder durch sorgfältige Analyse und Beschreibung der inneren Zusammenhänge des untersuchten Systems (analytisch).

Beim empirischen, auch Verhaltensmodell, verhaltensbeschreibendes Modell oder „black-box“-Modell genannten Modelltyp, werden die systeminternen Zustandsvariablen und Kopplungen nicht berücksichtigt, sondern es wird nur das Übertragungsverhalten zwischen Eingangsgrößen und Ausgangsgrößen nachgeahmt (Verhaltensgültigkeit). Beispiele dafür sind statistische Regressionsmodelle, Wenn–Dann–Regeln sofern sie lediglich auf Beobachtung beruhen und sich nicht mit den zugrundeliegenden Mechanismen befassen („Bauernregeln“) oder auch die künstlichen neuronalen Netze (*artificial neural networks*, ANN).

Zur Konstruktion eines Modells vom mechanistischen oder analytischen Typ, auch verhaltenserklärendes Modell oder kurz Erklärungsmodell genannt, wird das betrachtete System in Komponenten, Subsysteme und Kopplungen (Stoff, Energie, Information) zerlegt (analysiert). Es wird versucht, die Struktur des untersuchten Systems möglichst adäquat zu beschreiben (Strukturgültigkeit). Die Form, wie diese Verhaltensbeschreibung formal umgesetzt wird, kann sehr unterschiedlich aussehen, z.B.

- Flussdiagramm für den Kohlenstoff-Flux im Nahrungsnetz (statisch, analytisch),
- System aus verhaltenserklärenden Regeln in einem Expertensystem[1] (deterministisch, statisch, analytisch)
- Differentialgleichungsmodell eines Nahrungsnetzes (dynamisch, kontinuierlich, deterministisch, analytisch),
- Individuenbasiertes Räuber-Beute-Modell (dynamisch, diskret, stochastisch, analytisch).

Zwischen der empirischen und der analytischen Methode existieren Übergänge („grey-box“). So sollte bei der Aufstellung eines Regressionsmodells eventuell vorhandene a-priori-Information über den zugrundeliegenden Funktionstyp einfließen. Umgekehrt enthalten analytische Modelle fast immer empirische Modelle für nicht weiter analysierte Subsysteme (Systemrand).

Kompartimentmodelle – individuenbasierte Modelle: Ein Kompartimentmodell besitzt stark aggregierte Zustandsgrößen (Kompartimente, *pools*), zwischen denen ein Stoff oder Energiefluss besteht. Die Kompartimente werden sowohl räumlich (Pelagial, Litoral, Zellinnenraum, Kulturmedium) als auch funktionell abgegrenzt (Phosphor, Phytoplankton, Zooplankton) und es werden nur aggregierte Eigenschaften wie Masse oder Konzentration betrachtet (z.B. Biomasse des herbivoren Zooplanktons).

[1] Man spricht hier besser von einem wissensbasierten System.

Individuenbasierte Modelle verfolgen den Ansatz, das Verhalten von Populationen oder Ökosystemen aus dem Verhalten einzelner Individuen (z.B. 100 unterschiedliche Daphnien-Individuen) abzuleiten. Das Grundelement eines individuenbasierten Modells ist das Individuum dessen individuelle Eigenschaften wie Alter, Größe, physiologische Größen und räumliche Koordinaten über die Zeit verfolgt werden. Das Verhalten der Individuen wird durch Regeln und algebraische Gleichungen beschrieben und in einem festen oder einem variablen Zeitschritt durchlaufen.

1.4 Beziehungen zwischen Systemkomponenten

Ein System besteht nicht aus isolierten Komponenten, sondern diese sind durch vielfältige Flüsse (z.B. Stoff, Energie, Information) oder durch logische oder systematische Verknüpfungen miteinander gekoppelt.

1.4.1 Kopplung und Rückkopplung

Direkte Kopplung: Eine Komponente wirkt direkt auf eine andere Komponente.

Indirekte Kopplung: Eine Komponente wirkt vermittelt durch eine zweite Komponente auf eine dritte Komponente. In ökologischen Systemen besitzen indirekte Effekte eine überragende Bedeutung (PATTEN, 1983).

Rückkopplung: Eine Komponente wirkt direkt oder indirekt auf sich selbst zurück (Rückkopplungsschleife). Die Entscheidung, ob es sich um eine Vorwärtskopplung oder Rückkopplung handelt, kann von der Betrachtungsweise abhängen. Wichtig ist also nicht die Richtung der Pfeile, sondern ob es sich um eine Rückwirkung auf das Ursprungselement handelt (Schleife).

Bei den Rückkopplungen ist es sehr wichtig, ob diese die gleiche (positive Rückkopplung) oder eine entgegengesetzte Wirkung (negative Rückkopplung) wie die ursprüngliche Kopplung aufweisen. Positive Wirkungen wirken verstärkend oder destabilierend (z.B. exponentielles Wachstum), negative Rückkopplungen dämpfend oder stabilisierend (z.B. Selbstbeschattung der Vegetation). Eine Klassifikation von direkten und indirekten Effekten in Nahrungsnetzen findet sich in MILLER and KERFOOT (1987) und BENNDORF (1992).

1.4.2 Das Aggregationsproblem

Das Zusammenfassen von Systemelementen mit ähnlichem Verhalten zu Kompartimenten (z.B. gleiche trophische Ebene) und ähnlichen Eigenschaften (z.B. Wachstumsparametern) oder das Mitteln über Zeit (z.B. Vegetationsperiode) oder Raum (z.B. das Pelagial) bezeichnet man als Aggregation. Es werden dann mittlere Parameter verwendet (z.B. Grazingrate des Zooplanktons).

Aggregation ist zum Beispiel:

- Zusammenfassen von Spezies oder Populationen,

- räumliche Aggregation (See als homogener, vollständig durchmischter Wasserkörper).

Aggregation ist einerseits eine unbedingt notwendige Abstraktion und Vereinfachung, andererseits entsteht hierdurch ein Fehler, da die zwischen den aggregierten Systemkomponenten wirkenden Kopplungen nicht berücksichtigt werden können.

Grundsätzlich ist Aggregation nicht durch ein einfaches „Mehr an Kompartimenten“ (z.B. 50 Phytoplanktongruppen in einem Gewässermodell) behebbar, da auch die Vielzahl direkter und indirekter Effekte innerhalb dieser Kompartimente beschrieben werden muss. Wichtig für die Strukturgültigkeit ist eher die korrekte und komplexe Beschreibung des Systems. Unter Komplexität verstehen wir das Verhältnis zwischen der Anzahl der Kopplungen und der Anzahl der Elemente.

1.5 Modellzweck

1.5.1 Modelle als Prognosewerkzeuge

Viele Menschen denken, wenn sie das Wort „Modell“ hören, sofort an „Prognose“. Dabei ist dies nur eine Möglichkeit des praktischen Einsatzes mathematischer Modelle. Auch die Diagnose und Klassifikation von Systemen (z.B. Trophieklassifikation) oder die Szenarioanalyse (relativer Vergleich von Varianten) sind wichtige Anwendungen. Besonders bei der Szenarioanalyse kommen die Vorteile des Abbildcharakters mathematischer Modelle zum Tragen. Gegenüber einer Manipulation an einem Originalsystem (z.B. dem Bodensee) spart man bei der Nutzung eines Modells eine Menge Zeit und Geld, kann eine höhere Anzahl Varianten prüfen und vermeidet auch juristische und politische Probleme bei einem Fehlschlag, da das Originalsystem nicht beeinträchtigt wird.

Auf der anderen Seite sind natürlich die besonders in ökologischen Modellen enthaltenen Modellfehler und die oft sehr mangelhafte Übertragbarkeit wichtige Nachteile von Modellen. Um den Modellfehler etwas abzumildern, vermeidet man es bei ökologischen Modellen meist, eine absolute Prognose abzugeben (z.B. die Phytoplanktonbiomasse erreicht im Sommer ein Maximum von $10 \mathrm{mg} l^{-1}$) und verwendet stattdessen eine Szenarioanalyse. Man vergleicht hierbei die einzelnen Varianten (Szenarien) relativ untereinander. Dies hat den Vorteil, dass sich ein möglicherweise vorhandener systematischer Modellfehler beim Vergleich der Szenarien untereinander zumindest teilweise neutralisiert.

1.5.2 Modelle als Wissensspeicher und Erklärungshilfen

Empirische Forschung hat die Eigenart, dass sich zu einem gegebenen Thema mit der Zeit immer mehr Details und Einzelergebnisse (bildlich gesprochen: große Stapel mit zu lesenden Publikationen) ansammeln. Bei dem Bestreben, Einzelergebnisse zu verallgemeinern, sind mathematische Modelle ein sehr wichtiges Hilfsmittel. Mit Hilfe mathematischer Formalismen lassen sich unterschiedliche Details zu größeren Modellen synthetisieren, auf Konsistenz prüfen und im Zusammenhang dadurch besser verstehen. Auch lassen sich aus Modellen Hypothesen ableiten und experimentell überprüfen. Im Gegensatz zur verbalen Ergebnisbeschreibung erlaubt die formale mathematische Darstellung keine Widersprüche und Ausreden. Darüberhinaus stellen Modelle einen exzellenten, weil knappen und

auf das Wesentliche beschränkten, Wissensspeicher dar. Bei der Aufstellung von Modellen werden Wissenslücken sehr viel klarer als sonst deutlich. Die Identifikation und Behebung von Wissenslücken bewirkt eine Rückkopplungsschleife und führt dazu, dass Arbeitsgruppen, die beide Wege verfolgen (empirische und theoretische Arbeiten) nur selten an Ideenmangel leiden.

1.6 Grafische Darstellung von Modellen

Es existieren unterschiedliche Möglichkeiten (Paradigmen) zur Darstellung eines Modell-Systems und zur Bezeichnung der Elemente und Kopplungen eines Systems. Als Beispiele sollen an dieser Stelle das Paradigma eines „dynamischen Systems“ (siehe Abb. 1.1) und das objektorientierte Paradigma (Klasse–Unterklasse–Objekt, Abb. 1.2) vorgestellt werden.

Zur Darstellung dynamischer Systeme werden unterschiedliche grafische Symbole benutzt, z.B. die Darstellung nach FORRESTER (1961, 1971). Wir benutzen in dieser Einführung eine vereinfachte und auf das Wesentliche beschränkte Darstellung (Abb. 1.1).

Bei der Beschreibung eines dynamischen Systems beginnt man zweckmäßigerweise mit den Zustandsgrößen. Das sind gemäß dem oben vorgestellten Systembegriff die eigentlichen Elemente (Komponenten oder auch Kompartimente) des dynamischen Systems und können z.B. mit Hilfe von Differentialgleichungen beschrieben werden. Die Umwelt wird durch Randbedingungen (Inputs, Eingangsdaten) und durch Quellen- oder Senkenterme repräsentiert. Die Randbedingungen kennzeichnen äußere Bedingungen, die auf das System einwirken aber nicht selbst im System beschrieben sind, z.B. meteorologische Einflüsse, Wasserzufluss oder Nährstoffkonzentration. Bei den Quellen und Senken kann praktisch Masse aus dem „Nichts“ oder besser gesagt unbeschränkt großen „pools“ entstehen oder dahin verschwinden.

Zwischen den Zustandsgrößen, den Quellen und Senkentermen und den Randbedingungen existieren Kopplungen, die als Informations- oder Energie und Stoffffluss wirken können. Die Geschwindigkeiten der stofflichen oder energetischen Umsätze bezeichnet man als Raten, Informationsflüsse werden in Form von Hilfsgleichungen (algebraischen Gleichungen oder auch Regeln) dargestellt. Die konkreten (und meist konstanten) Werte, mit denen die Änderungsraten, Flüsse und algebraischen Gleichungen quantitativ bestimmt werden, bezeichnen wir als Parameter, alle anderen Größen (Zustandsgrößen, Hilfsgrößen, Eingangsdaten) als Variablen.

Außer als dynamisches Sytem lassen sich Systeme und Modelle natürlich auch auf völlig andere Weise beschreiben. Als Beispiel sei hier das objektorientierte Paradigma vorgestellt (Abb. 1.2). Das Grundelement der objektorientierten Betrachtung ist das Objekt, welches aus Eigenschaften (z.B. Alter, Größe, Name) und Methoden (z.B. Wachstum, Vermehrung, Nahrungsaufnahme) besteht. Objekte, die nur als abstrakte Gebilde dazu dienen, gleichartige Objekte zu systematisieren, werden als Klassen bezeichnet (z.B. Klasse der Buchen) von denen einzelne Instanzen (die Buche vor meiner Haustür) abgeleitet werden können. Die Instanzen können hierbei materiell (die eine Buche) oder immateriell (eine spezielle Idee) sein. Klassen lassen sich ebenfalls wieder in Klassen zusammenfassen und Objekte können aus Teilobjekten zusammengesetzt sein. So ist z.B. das Schreitbein eines Flusskrebses einerseits eine Instanz aus der Klasse Extremitäten und andererseits ein Bestandteil (ein Subobjekt oder einfach eine

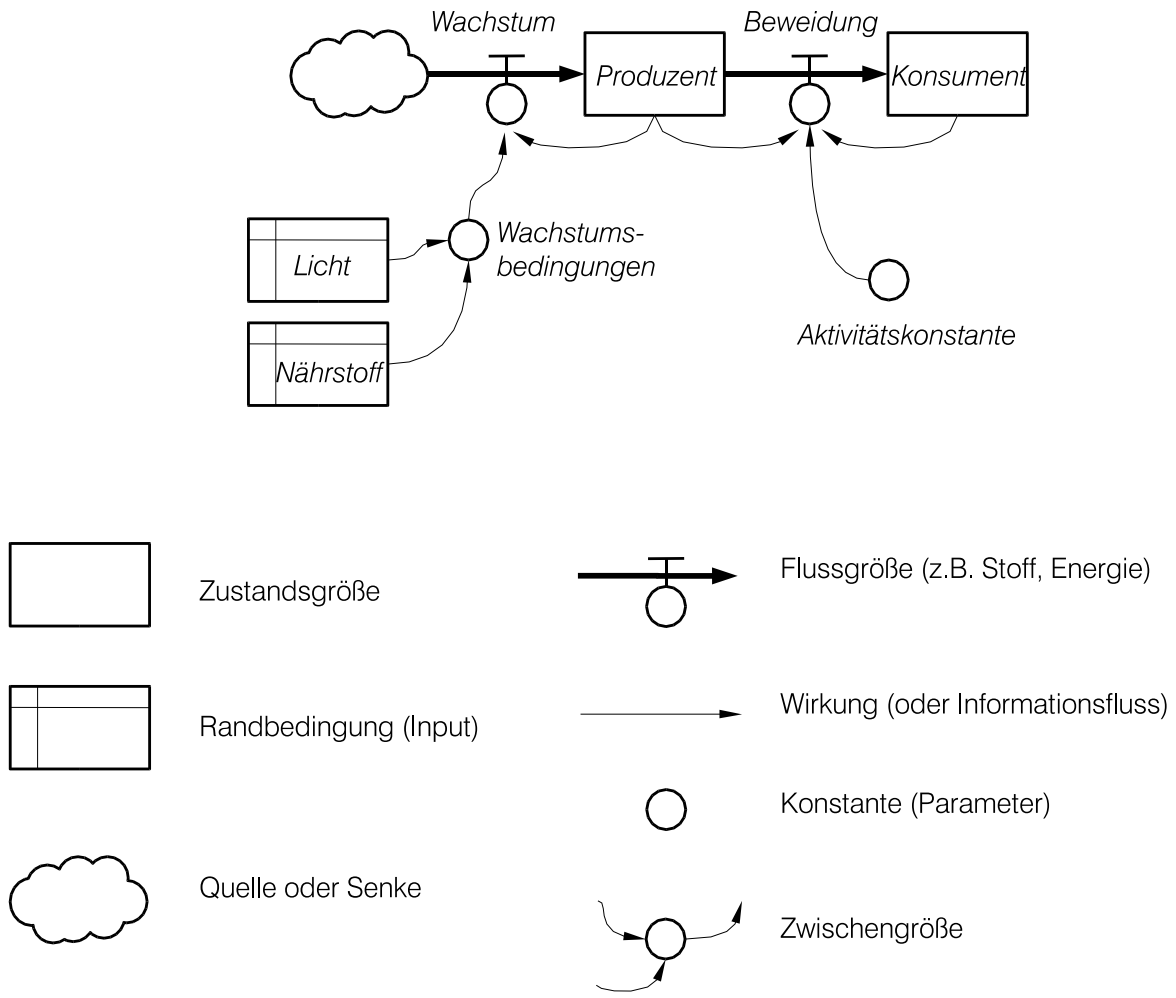

Abbildung 1.1: Elemente und Kopplungen eines dynamischen Systems.

spezielle Eigenschaft) eben dieses Flusskrebses. Somit ergibt sich eine Baumstruktur oder ein Netzwerk mit den Beziehungstypen:

- Unterklasse – Klasse („ist eine Art von“, Darstellung: Pfeil),
- Instanz einer Klasse – Klasse („ist ein“, Darstellung: normale Linie) und
- Subobjekt – Objekt („gehört zu“ - oder umgekehrt - „besteht aus“ oder einfach „hat“, Darstellung: schwarzer Rhombus).

Die objektorientierte Darstellung ist in der Biologie praktisch die Grundlage eines jeglichen Bestimmungsschlüssels und kann auch sonst sinnvoll zur Systematisierung von Wissen eingesetzt werden. Darüberhinaus versteht man unter objektorientierter Programmierung (OOP) auch eine spezielle Programmier- und Software-Entwurfstechnik, die z.B. auch für individuenbasierte Modelle erfolgreich eingesetzt werden kann. Die aus der OOP hervorgegangene grafische Sprache UML (Unified Markup

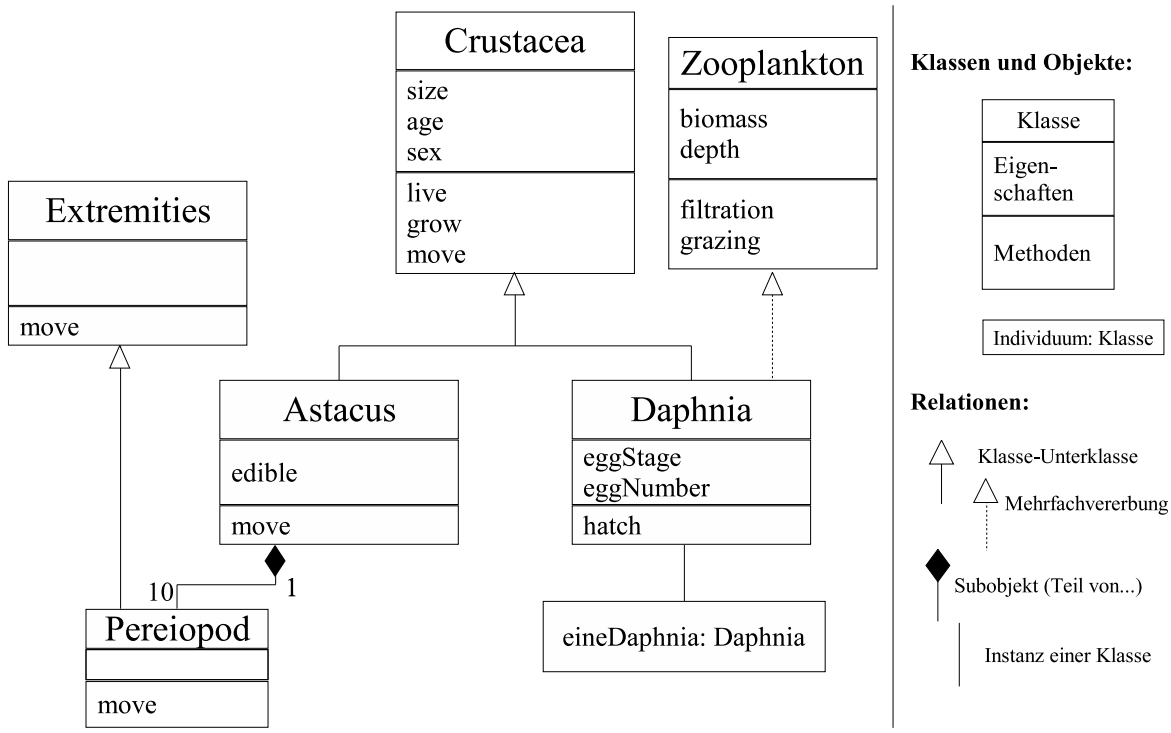

Abbildung 1.2: Mit Hilfe eines Klassendiagramms lassen sich Beziehungen zwischen Objekten formal darstellen. Das Klassendiagramm ist eines aus mehreren Diagrammtypen der grafischen UML-Sprache (Unified Markup Language).

Language, siehe z.B. FOWLER and SCOTT, 1997) definiert mehrere unterschiedliche Diagrammtypen, die auch zur Systematisierung biologischer Zusammenhänge oder zur Verständigung zwischen Biologen und „Modellierern“ nützlich sein können.

1.7 Wie komme ich zu einem Modell?

Es existieren mehr oder weniger umfangreiche schrittweise Anleitungen, wie man zu einem Modell kommt (z.B. BOSSEL, 1994). Ich möchte deshalb an dieser Stelle eine klassische Dreiteilung vorschlagen, die aus drei äußeren (Systemanalyse–Modellsynthese–Anwendung) und drei inneren Schritten besteht (Abb. 1.3).

Vor der eigentlichen Modellsynthese findet die naturwissenschaftliche Vorarbeit, die Untersuchung des Originalsystems statt (Systemanalyse des realen Systems), deren Ergebnisse in Form von Berichten und Publikationen dargestellt werden. Daran schließt sich die eigentliche Modellsynthese an, für die ich in Anlehnung FOWLER and SCOTT (1997) drei Schritte vorschlagen möchte:

1. Konzeption,

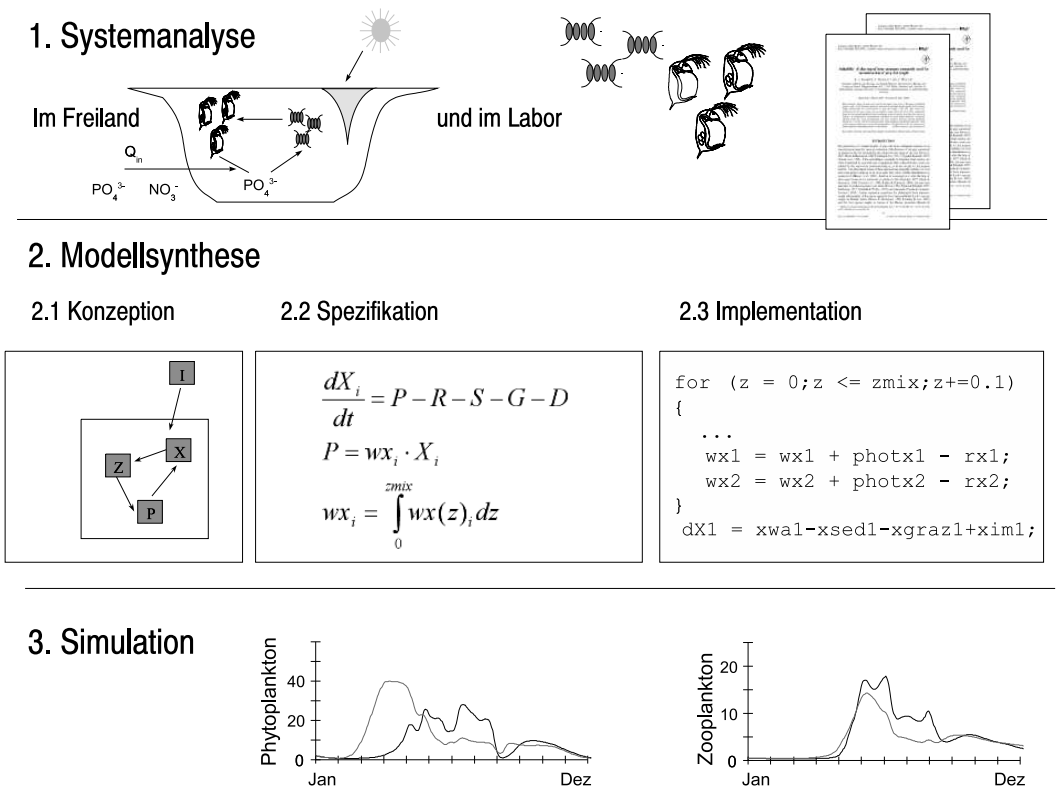

Abbildung 1.3: Systemanalyse, Modellsynthese und Simulation: An die Analyse des untersuchten Systems im Freiland und im Labor schließt sich die Modellsynthese und die Modellsimulation an.

2. Spezifikation,
3. Implementierung.

Diese Schritte stellen zugleich unterschiedliche Betrachtungsperspektiven oder Abstraktionsebenen dar. Es ist ein Unterschied, ob man ein konzeptionelles Diagramm zeichnet oder ein (vollständig simulierbares) Simulationsdiagramm bzw. ob man mit dem Begriff „Modell“ das Gleichungssystem (oder ein Simulationsdiagramm) oder das Computerprogramm meint.

Um Konfusion zu vermeiden, ist es sehr wichtig, sich diese Unterscheidung regelmäßig zu vergegenwärtigen. So ist es ein Unterschied, ob ein Modell aus der Konzeptionsperspektive, der Spezifikationsperspektive oder der Implementierungsperspektive gesehen wird. Im naturwissenschaftlichen Sinne ist das Modell strenggenommen die abstrakte Form (Gleichungssystem, Spezifikationsperspektive), die durchaus mehrere unterschiedliche Implementierungen besitzen kann.

Nicht immer gelingt es, die Schritte streng nacheinander auszuführen. So stellt man oft bei der Modellspezifikation fest, dass wichtige Elemente oder Kopplungen vergessen wurden oder umgekehrt, dass bestimmte anfängliche Vorstellungen doch nicht erfüllbar sind. In der Praxis wird deshalb die Abfolge

oft in mehreren Durchläufen ausgeführt. In schwierigen Fällen verwendet man z.B. das *rapid prototyping*, d.h. die schrittweise Verfeinerung eines anfänglich sehr einfachen Modells.

Zum Schluss schließt sich an die Modellsynthese eine mehr oder weniger umfangreiche Test- und Anwendungsphase an (Verifizierung, Validation, Sensitivitätsanalyse, Kalibrierung, Szenarioanalyse und Prognose).

1.7.1 Modellkonzeption

Bei der Modellkonzeption werden der Modellzweck, die Systemgrenzen, die Elemente und Kopplungen und die wichtigsten Eigenschaften (z.B. dynamisch, deterministisch) des zu entwickelnden Modells festgelegt. Hierzu reichen im allgemeinen Stichworte (Wortmodell) und einfache Grafiken mit nur minimalem Formalismus (z.B. Wirkungsgraf: welche Elemente wirken aufeinander ein). Darüberhinaus sollten bereits wichtige Wissensquellen (Publikationen) genannt werden.

Die wichtigste bei der Modellkonzeption zu lösende Frage ist die nach dem Zweck des Modells: Soll das Modell ein Verhaltens- oder ein Erklärungsmodell sein, welche Hypothesen sollen mit dem Modell geprüft werden, sollen Vorhersagen gemacht werden und welche Allgemeingültigkeit (beobachteter Einzelfall, Klasse ähnlicher Fälle, Prognosegültigkeit) wird angestrebt. Es ist eine wichtige Erfahrung, dass Modelle mittlerer Größe mit genau festgelegtem Einsatzzweck und eng gezogenen Systemgrenzen meist die bessere Wahl sind. Unerfahrene Modellbauer streben gern „Supermodelle“ an (BOSSEL, 1994), die jedoch in der Realität nicht erreichbar ist. Modelle sind per Definition vereinfachte Abbilder.

1.7.2 Modellspezifikation

Dieser manchmal auch „Algorithmierung“ genannte Arbeitsgang ist der wesentliche wissenschaftliche Schritt der Modellbildung. Ausgehend von Wortmodell und Wirkungsgraf werden bei der Modellspezifikation die genauen Details des Modells ermittelt (z.B. durch Literaturstudium und Experimente) und formal dargestellt. Das Ergebnis dieser Phase ist das eigentliche Modell, in dem alle Elemente und Kopplungen vollständig und detailliert ausgearbeitet und dokumentiert sind, z.B. als Regelgraf, in Form von Gleichungen und Parameterwerten, als Simulationsdiagramm oder als Algorithmus (Abfolge von Rechenschritten).

1.7.3 Implementierung

Die technische Umsetzung eines Modells in eine rechenfähige Form bezeichnet man als Implementierung. Sofern es sich bei dem Modell nicht um ein auf dem Papier lösbares Problem handelt, bedeutet das, dass das Modell mit einer entsprechenden Software umgesetzt werden muss. Es gibt Simulationssysteme (z.B. Stella, Modelmaker oder Simulink), bei denen man direkt den Wirkungsgrafen eingeben kann. Bei anderen Systemen müssen die Modelle in eine Programmiersprache oder eine spezielle Simulationssprache umgesetzt werden.

1.8 Anwendung und Testung von Modellen

Ist ein Modell implementiert, so schließt sich immer, außer bei rein theoretischen Modellen, eine mehr oder weniger umfangreiche Testung an. Man unterscheidet hierbei:

Verifikation: hierunter verstehen wir den Prozess, bei dem man das implementierte Modell auf plausibles Verhalten im Vergleich zur Spezifikation überprüft und auch eventuelle Programmier- oder Integrationsfehler beseitigt.

Validation: bezeichnet die Überprüfung des Modells an realen Daten. Man kann hierzu entweder Korrelationskoeffizienten benutzen, spezielle statistische Gütekriterien heranziehen (LEGGETT and WILLIAMS, 1981; GAYNOR and KIRKPATRICK, 1994), oder grafische Methoden verwenden, z.B. den visuellen Vergleich von Zeitreihen oder die einfache oder kumulative Gegenüberstellung von beobachteten und vorhergesagten Werten.

Kalibrierung: (oder Parameteroptimierung) ist das Anpassen eines Modells an gemessene Daten, indem man entweder einzelne Parameter von Hand justiert oder einen oder mehrere Parameter durch einen Optimierungsalgorithmus (siehe nichtlineare Regression, Abschnitt 2.4) schätzt. Man spricht dann oft von Kalibrierung oder „Fitten“ des Modells. Parameteroptimierung ist ein übliches Verfahren und besonders bei kleineren Modellen sinnvoll, um nicht direkt messbare Parameter zu ermitteln (indirekte Methode, Parameteridentifikation). Parameteroptimierung hat jedoch immer einen Verlust an Allgemeingültigkeit zur Folge, d.h. das Modell gilt dann eigentlich nur noch für den Datensatz (oder das Untersuchungsobjekt), mit dem kalibriert wurde (ein Beispiel für ein Gewässergütemodell geben PETZOLDT, 1996; PETZOLDT and HORN, 1997). Um trotzdem eine gewisse Allgemeingültigkeit zu erreichen, muss deshalb ein solches kalibriertes Modell an unabhängigen Datensätzen validiert werden. Um Selbstbetrug auszuschließen sollte man sich solche Testdatensätze vorher beiseite legen und nicht ansehen oder möglichst erst nach der Modellerstellung beschaffen.

Bei komplexeren Ökosystem-Modellen ist Kalibrierung sehr umstritten. Der Grund dafür ist, dass hier oft mehrere Prozesse in aggregierten Gleichungen zusammengefasst oder sogar gänzlich unbekannt sind sind und sich mehrere eigentlich zeitvariable Prozesse in einem einzelnen konstanten Parameter wiederfinden können. Die eigentlichen Parameter der Grundprozesse sind nicht zugänglich (verdeckte Parameter). Darüberhinaus sind nach ökologischen Prinzipien konstruierte Modelle oft „überparametrisiert“, so lässt sich z.B. die Erhöhung der Photosyntheserate durch eine Erhöhung der Respirationsrate kompensieren, so dass nach der Optimierung völlig unrealistische Parameterwerte und ein *drastischer Verlust an Allgemeingültigkeit* die Folge sind. Trotz der zuweilen gängigen Vorgehensweise und der auf den ersten Blick besseren Übereinstimmung der Kurvenverläufe mit gemessenen Daten entspricht die Parameteroptimierung komplexer Modelle oft nicht den Regeln guter wissenschaftlicher Praxis[2].

[2] siehe z.B. Empfehlungen der Deutschen Forschungsgemeinsacht unter:
`http://www.dfg.de/aktuelles_presse/reden_stellungnahmen/download/empfehlung_wiss_praxis_0198.pdf`

Eine gewisse Grauzone entsteht manchmal dann, wenn man Teilprozesse eines Modells aufgrund unabhängiger Messdaten anpasst. Ein solcher Fall liegt vor, wenn man eine indirekte Abschätzung (Parameteridentifikation) einzelner Modellparameter vornimmt, z.B. bei Ermittlung der Phosphor-Freisetzungsrate aus dem Sediment für ein spezielles Gewässer oder bei der Kalibrierung von eng begrenzten Submodellen. Eine solche Vorgehensweise ist durchaus akzeptabel, erfordert jedoch Verantwortungsgefühl, Ehrlichkeit und den Einsatz naturwissenschaftlichen Sachverstandes. Oft resultiert aus einer kritischen Bewertung der Abweichungen zwischen Modell und Realität ein weitaus größerer Erkenntnisgewinn, als durch das „zurechtfitten“ eines Modells.

Sensitivitätsanalyse: hierunter versteht man die Untersuchung des Einflusses einzelner oder mehrerer Parameterwerte oder Eingangsgrößen auf das Simulationsergebnis. Man vergleicht dann entweder komplette Kurven oder charakteristische Modellergebnisse (Einzelwerte, Muster) mit einem Standardfall und wertet diese statistisch aus. Eine wichtige Anwendung der Sensitivitätsanalyse ist auch die Auswahl von geeigneten Parametern im Rahmen einer Parameteridentifikation (siehe z.B. THORNTON *et al.*, 1979; BRUN *et al.*, 2001; OMLIN *et al.*, 2001).

1.9 Implementierung ökologischer Modelle

Grundsätzlich lassen sich manche Modelle auf analytischem Wege lösen, also mit algebraischen Verfahren, Papier und Bleistift. Die meisten ökologischen Modelle erfordern jedoch aufgrund ihrer hohen Komplexität eine numerische Lösung mit Hilfe von Computern und entsprechender Software. Für einfachere Modelle genügen hierzu bereits ein Tabellenkalkulationsprogramm und eventuell ein paar unterstützende Makros, für kompliziertere Modelle verwendet man oft spezielle Software.

Prinzipiell lassen sich natürlich alle Modelle in einer höheren Programmiersprache wie Fortran, Basic, Pascal, C, C++, Java oder Python schreiben, zum Teil unter Nutzung vorgefertigter Bibliotheken (z.B. Integrationsverfahren) oder durch die Abwandlung vorgefertigter Beispiele (*templates* oder *Frameworks*), z.B. aus den Büchern von BOSSEL (1994) oder GURNEY and NISBET (1998).

Daneben existieren spezielle, jeweils für eine bestimmte Klasse von Modellen geeignete Simulatoren, z.B. STELLA[3], DYNASYS[4] oder das Web-basierte DYNAST[5] für Differentialgleichungsmodelle oder z.B. EcoBeaker[6] für gitterbasierte individuenbasierte Modelle. Die Liste der Simulationsumgebungen ließe sich beliebig fortsetzen und im Internet finden sich dazu viele Verweise, Empfehlungen und Vergleiche[7].

Eine dritte Klasse von Simulationswerkzeugen sind spezielle Mathematik- und Simulationssprachen,

[3] High Performance Systems Inc., `http://www.hps-inc.com/`
[4] Hupfeld Software, `http://www.hupfeld-software.com/`
[5] DYNAST: `http://icosym.cvut.cz/cacsd/msa/dynast.html`
[6] BeakerWare, `http://www.ecobeaker.com/`
[7] ARGESIM Simulation Links: `http://eurosim.tuwien.ac.at/hotlinks/`

wie z.B. MAPLE[8], MATHEMATICA[9] oder MATLAB/SIMULINK[10], wobei z.B. MATLAB (Matrix Laboratory) aus einer vektororientierten Sprache, einer Sammlung hochentwickelter Simulations- und Visualisierungsalgorithmen einschließlich symbolischer Mathematik und einem grafischen Frontend zur Entwicklung von Modellen (SIMULINK) besteht.

Das im Rahmen dieser Einführung verwendete Statistiksystem R besitzt durchaus eine begrenzte Ähnlichkeit mit MATLAB, z.B. eine auf Vektor- und Matrizenrechnung ausgerichtete Interpretersprache, vorgefertigte grafische Routinen und einfache Erweiterbarkeit mit Hilfe von externen C/C++ und FORTRAN-Routinen.

R ist eine von IHAKA and GENTLEMAN (1996) entwickelte und inzwischen von einer erweiterten Gruppe von Wissenschaftlern (R Core Team) und einer sich darum scharenden Gemeinschaft ständig verbesserte und erweiterte Implementierung der vektororientierten Programmiersprache S. Neben grundlegenden Eigenschaften wie Programmierbarkeit und hoher Effizienz enthält R eine Sammlung hochentwickelter Statistik- und Grafikroutinen, die sich mit Hilfe von R auf einheitliche Weise benutzen und kombinieren lassen.

Die Sprache und das System S wurden ursprünglich von den AT&T Bell Laboratories entwickelt und sind auch Basis des kommerziellen Systems S-PLUS[11], zu dem neben dem S-Interpreter und den Statistikbibliotheken auch ein Menüsystem und hervorragende statistische Handbücher gehören.

R ist OpenSource-Software und wird unter der GNU Public License[12] vertrieben, einem Lizenzmodell das der wissenschaflicher Arbeitsweise entspricht. Alle Bestandteile des Systems sind offen und komplett einsehbar (Sourcecode in C, Fortran und R), die Quellen der Bibliotheken sind sauber zitiert und das System darf nicht nur kostenlos weiterkopiert werden, sondern es ist auch möglich, R selbst zu verändern und zu erweitern. Solche Versionen dürfen auch weitergegeben werden, müssen dann aber ebenfalls komplett offengelegt (publiziert) werden, so dass sich daraus ein Gewinn für alle Benutzer von R ergibt.

Das R-System ist auf allen wichtigen Betriebssystemen lauffähig, z.B. verschiedenen UNIX-Versionen, Linux, Microsoft Windows und Apple MacOS. Die kompletten Quellen, Binärdistributionen, die Dokumentation (z.B. VENABLES *et al.*, 2001; R DEVELOPMENT CORE TEAM, 2001) und weiteres Material kann von der R-Homepage[13] oder dem weltweit verteilten CRAN[14] (Comprehensive R Archive Network) heruntergeladen werden. Darüberhinaus existieren Bücher, die statistische Themen spzeiell mit Hilfe von von S-PLUS oder R erläutern, allen voran VENABLES and RIPLEY (1999) oder auch z.B. CRAWLEY (2002).

Obwohl R eher als ein Statistik- und noch nicht als ein Simulationspaket positioniert wurde, ist die von R zur Verfügung gestellte Umgebung zur Lösung numerischer Probleme ausgezeichnet geeignet

[8] Waterloo Maple, `http://www.maplesoft.com/`
[9] Wolfram Research, `http://www.wolfram.com/`
[10] Mathworks, `http://www.mathworks.com/`
[11] Mathsoft, `http://www.splus.mathsoft.com`
[12] Free Software Foundation, `http://www.gnu.org`
[13] R Projekt, `http://www.r-project.org`
[14] CRAN, `http://cran.r-project.org`

und universell nutzbar. Die bisher bereits vorhandenen Numerik- und Simulationspakete werden ständig ergänzt und weiterentwickelt und letztlich kann man R auch selbst durch in C oder einer anderen Programmiersprache selbst entwickelte oder einer Simulationsbibliothek entnommene Bausteine ergänzen.

Als Nebeneffekt lernen wir eines der mit Sicherheit fortgeschrittensten und leistungsfähigsten Statistikpakete überhaupt kennen und erschließen Möglichkeiten für flexible und publikationsreife Grafiken. Natürlich ist es jederzeit möglich, auch auf ein kommerzielles System mit seinen spezifischen Vorteilen umzusteigen. Der Vorteil des OpenSource-Ansatzes besteht jedoch auch darin, dass die Software bei einem in der Wissenschaft ja manchmal notwendigen Wechsel der Arbeitsgruppe mitgenommen werden darf und die eigenen Daten, Programme und Modelle problemlos weiter genutzt werden können.

2 Empirisch-statistische Modelle

Modelle, die im wesentlichen auf Erfahrung oder Beobachtungen beruhen und lediglich das Verhalten, nicht aber die Struktur des betrachteten Systems beschreiben, werden häufig als „empirische Modelle“ bezeichnet. Hierbei spielt es keine Rolle, ob die Beobachtungen rein qualitativer Art sind oder aus langen Reihen präziser Meßdaten bestehen.

Kennzeichnendes Merkmal für solche Modelle ist, dass nur das von außen sichtbare Verhalten eines Systems, d.h. nur die Beziehungen zwischen Eingangsvariablen und Ausgangsvariablen betrachtet werden und nicht die zugrundeliegenden Mechanismen. Man bezeichnet solche Modelle deshalb auch als Verhaltensmodelle oder „black-box“-Modelle. Semi-empirische oder „grey-box“-Modelle erhält man, wenn man versucht, wenigstens eine dem Grundmechanismus (z.B. Wachstumsfunktion) entsprechende Modellgleichung anzuwenden.

Zur Konstruktion solcher Modelle können unterschiedliche Verfahren angewendet werden. In vielen Fällen werden statistische Verfahren, insbesondere die Regressionsanalyse, angewendet, weshalb wir diesen Modelltyp hier als empirisch-statistische Modelle bezeichnen wollen. Dieser Begriff kann aber noch breiter aufgefasst werden und z.B. auch Eintrittswahrscheinlichkeiten, z.B. ein 1000 jähriges Hochwasser, Grenzwerte, z.B. die EC_{50} und Klassifikationssysteme, z.B. Gewässergüteklassen oder das Vollenweidermodell umfassen (z.B. VOLLENWEIDER and KEREKES, 1980).

2.1 Lineare Regression

Mittels Regressionsanalyse wird versucht, die Art des Zusammenhanges zweier Größen (*einfache* Regression) oder mehrerer Größen (*multiple* Regression) durch eine Funktion zu beschreiben[1]. Im Gegensatz zur Korrelationsanalyse, die prüft, ob überhaupt ein Zusammenhang existiert, besteht hier der Schwerpunkt in der quantitativen Beschreibung des Zusammenhanges durch eine Funktion. Einige Unterschiede bestehen auch in den zugrundeliegenden statistischen Annahmen. Während man bei der Korrelationsanalyse davon ausgeht, dass alle untersuchten Variablen fehlerbehaftete Zufallsvariablen sind (Modell II), unterscheidet man bei der Regressionsanalyse im allgemeinen unabhängige Größen (ohne Fehler) und abhängige Größen (mit Fehler) und bezeichnet das als Modell I (näheres hierzu z.B. in SACHS, 1992; ZAR, 1996).

[1] Im Gegensatz zu dieser *univariaten* Regression mit einer abhängigen Variable steht die *multivariate* Regression, wo mehrere abhängige Variablen existieren.

2.1.1 Grundlagen

Lineare Modelle allgemein bilden die Basis einer Vielzahl von statistischen Methoden, z.B. linearer Regression oder Varianzanalyse. Die Gleichung eines multiplen linearen Modells lautet

$$y_i = \alpha + \beta_1 x_{i,1} + \beta_2 x_{i,2} + \cdots + \beta_p x_{i,p} + \varepsilon_i \tag{2.1}$$

Hierbei berechnet sich jeder Wert y_i der abhängigen Variablen aus den entsprechenden Werten $x_{i,j}$ der unabhängigen Variablen, den Parametern α (Schnittpunkt mit der y-Achse) und β_j (Regressionskoeffizient oder Steigung) des Modells und einem normalverteilten Fehlerterm ε_i mit dem Mittelwert Null.

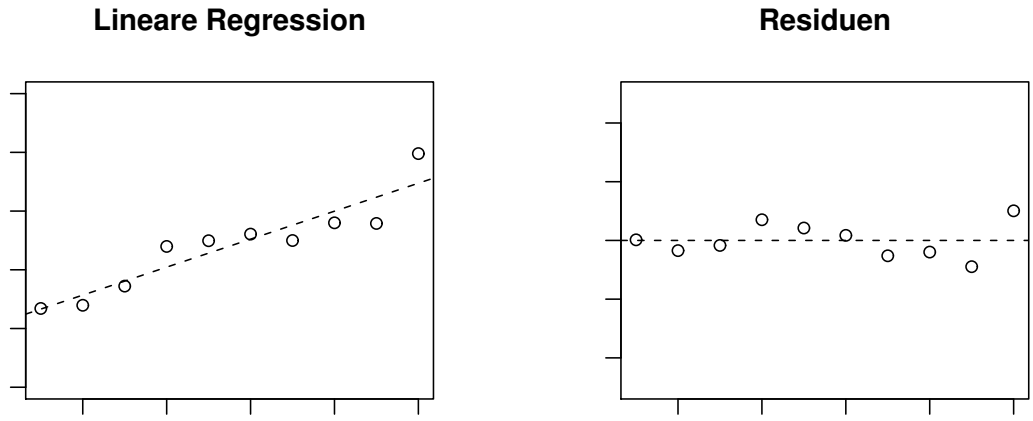

Abbildung 2.1: Lineare Regression (links) und Residuen (rechts).

Passt man z.B. ein einfaches Regressionsmodell (Ausgleichsgerade, Abb. 2.1) mit einer abhängigen Variablen y und nur einer unabhängigen Variablen x an, so schreibt man häufig:

$$\hat{y}_i = a + b \cdot x_i \tag{2.2}$$

Hierbei bezeichnet $\hat{y}$ die geschätzten Werte der abhängigen Variablen, d.h. die Werte auf der Geraden und a bzw. b die geschätzten Werte für die wahren Modellparameter α bzw. β.

Für die Ermittlung von a und b benötigt man ein Optimierungskriterium. In den meisten Fällen wird hierfür die Summe der Abweichungsquadrate $SQ = \sum(\hat{y}_i - y_i)^2$ verwendet. Um das Minimum $SQ \rightarrow Min$ zu ermitteln, setzt man die partiellen ersten Ableitungen von SQ bezogen auf die Parameter gleich Null:

$$\frac{\partial \sum(\hat{y}_i - y_i)^2}{\partial a} = \frac{\partial \sum(a + b \cdot x_i - y_i)^2}{\partial a} = 0 \tag{2.3}$$

$$\frac{\partial \sum(\hat{y}_i - y_i)^2}{\partial b} = \frac{\partial \sum(a + b \cdot x_i - y_i)^2}{\partial b} = 0 \tag{2.4}$$

Nach Lösung des resultierenden Gleichungsystems erhält man die Bestimmungsgleichungen:

$$b = \frac{\sum x_i y_i - \frac{1}{n}(\sum x_i \sum y_i)}{\sum x_i^2 - \frac{1}{n}(\sum x_i)^2} \tag{2.5}$$

$$a = \frac{\sum y_i - b \sum x_i}{n} \tag{2.6}$$

Die Ausgleichsgerade lässt sich demnach durch ein Gleichungssystem direkt aus den gegebenen Werten für x und y berechnen, man bezeichnet das deshalb als analytische Lösung.

Residuen und Bestimmtheitsmaß

Die Differenzen $\varepsilon_i = y_i - \hat{y}_i$ zwischen den gemessenen Werten der abhängigen Variablen und den mit Hilfe des Modells vorhergesagten Variablen bezeichnet man als Residuen (Abb. 2.1 rechts). Es ist offensichtlich, dass die Residuen weniger stark streuen als die ursprünglichen Messwerte y_i. Die Varianz der Residuen nennt man Restvarianz. Das bedeutet, dass ein Teil der ursprünglichen Streuung nun in der Geraden enthalten ist, d.h. durch das Modell erklärt wird. Der durch das Modell erklärte Anteil der Varianz an der ursprünglichen Varianz ist das Bestimmtheitsmaß B (Determinationskoeffizient), das bei der *linearen* Regression gleich dem Quadrat des Korrelationskoeffizienten r^2 (nach Pearson) ist.

Allgemein gilt:

$$r^2 = \frac{\text{erklärte Varianz}}{\text{ursprüngliche Varianz}} = \frac{\text{ursprüngliche Varianz} - \text{Restvarianz}}{\text{ursprüngliche Varianz}} \tag{2.7}$$

oder

$$r^2 = \frac{s_y^2 - s_{y_i - \hat{y}_i}^2}{s_y^2} \tag{2.8}$$

und im Falle der linearen Regression

$$r^2 = \frac{\sum(\hat{y} - \bar{y})^2}{\sum(y - \bar{y})^2} \tag{2.9}$$

Das Bestimmtheitsmaß hat immer einen Wert zwischen 0 und 1, ein $r^2 = 0.85$ bedeutet z.B., dass das Modell 85% der ursprünglichen Varianz erklärt.

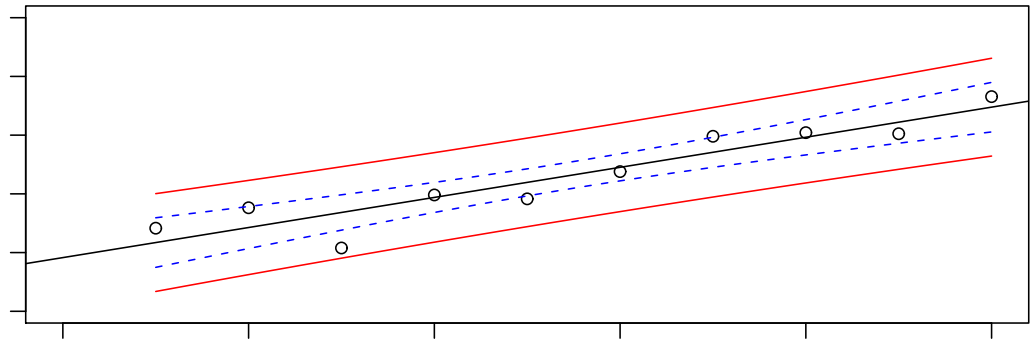

Abbildung 2.2: Lineare Regression mit Konfidenzintervall der Geraden (gestrichelt) und Vorhersageintervall (durchgezogen).

Signifikanztest

Neben der Höhe des Bestimmtheitsmaßes und der grafischen Kontrolle gehört zu jeder Regressionsrechnung ein Signifikanztest. Je größer der Stichprobenumfang n ist, umso kleinere Werte von r^2 können noch als signifikant erkannt werden. Zur Prüfung der Steigung b benutzt man hierzu einen F-Test mit:

$$\hat{F}_{1;n-2;\alpha} = \frac{s^2_{erklärt}}{s^2_{Rest}} = \frac{r^2(n-2)}{1-r^2} \tag{2.10}$$

Vertrauensintervalle

Es lassen sich Vertrauensintervalle für die Parameter (z.B. a und b), für die Regressionsgerade selbst und auch für zukünftige Beobachtungen Y_i an der Stelle X_i angeben (Vorhersageintervall). Für die Parameter a und b erhält man die Vertrauensintervalle einfach mit Hilfe ihrer Standardfehler s_a bzw. s_b und dem t-Quantil:

$$a \pm t_{1-\alpha/2,n-2} \cdot s_a \tag{2.11}$$

$$b \pm t_{1-\alpha/2,n-2} \cdot s_b \tag{2.12}$$

Oftmals werden in der graphischen Darstellung (z.B. Abb. 2.2) hyperbolische „Konfidenzintervalle“ angegeben. Hierbei handelt es sich zum einen um das eigentliche Konfidenzintervall der Regressionsgeraden. Die hyperbolische Form setzt sich aus einer Verschiebung (Vertrauensintervall des Parameters

a) und einer Drehung (Vertrauensintervall von *b*) zusammen. Bei einer angenommenen Irrtumswahrscheinlichkeit von $\alpha = 0.05$ liegt die wahre Regressionsgerade demnach mit 95%iger Wahrscheinlichkeit zwischen diesen Intervallgrenzen.

Das Vorhersageintervall hat dagegen eine völlig andere Bedeutung. Es sagt aus, in welchem Bereich sich ein für ein gegebenes x vorhergesagter Wert y befindet. Während das Konfidenzintervall demnach den mittleren Verlauf der Regressionsgeraden charakterisiert, macht das Vorhersageintervall eine Aussage über zu erwartende Einzelwerte.

Nach (SACHS, 1992) bzw. (ZAR, 1996) erhalten wir diese Intervalle wie folgt. Basierend auf der Summe der Abweichungsquadrate Q_x für die x-Werte:

$$Q_x = \sum_{i=1}^{n}(x_i - \bar{x})^2 = \sum_{i=1}^{n} x_i^2 - \frac{1}{n}\sum_{i=1}^{n} x_i \tag{2.13}$$

und des Standardfehlers der Vorhersage (also der Standardabweichung der $\hat{y}$-Werte) für ein gegebenes x ($s_{y|x}$ gelesen als „s y für x“)

$$s_{y|x} = \sqrt{\frac{\sum_{i=1}^{n}(y_i - \hat{y})^2}{n-2}} = \sqrt{\frac{\sum_{i=1}^{n}(y_i - a - b \cdot x_i)}{n-2}} \tag{2.14}$$

berechnen wir zunächst die Standardabweichung für einen geschätzten Mittelwert $\underline{\hat{y}}$ und die Standardabweichung für einen vorausgesagten Einzelwert $\hat{y}_.$ an der Stelle x

$$s_{\underline{\hat{y}}} = s_{y|x} \cdot \sqrt{\frac{1}{n} + \frac{(x-\bar{x})^2}{Q_x}} \tag{2.15}$$

$$s_{\hat{y}_.} = s_{y|x} \cdot \sqrt{1 + \frac{1}{n} + \frac{(x-\bar{x})^2}{Q_x}} \tag{2.16}$$

und erhalten das Konfidenzintervall der Regressionsgerade als:

$$\hat{y} \pm \sqrt{2 \cdot F_{(2,n-2)} \cdot s_{\underline{\hat{y}}}} \tag{2.17}$$

und das Vorhersageintervall als:

$$\hat{y} \pm t_{(n-2)} s_{\hat{y}_.} \tag{2.18}$$

wobei F und t die entsprechenden Quantile der F- und der t-Verteilung sind.

Voraussetzungen der linearen Regression

Die Parameter (a, b) werden nur dann unverzerrt geschätzt und der Signifikanztest ist nur dann zulässig, wenn die folgenden Voraussetzungen eingehalten wurden (SACHS, 1992):

1. Es gilt Modell I (x ist festgelegt, y ist eine Zufallsvariable).
2. Für jeden Wert von x ist y eine Zufallsvariable mit dem Mittelwert $\mu_{y|x}$ und der Varianz $\sigma^2_{y|x}$.
3. Die y-Werte sind unabhängig und identisch verteilt (keine Autokorrelation, überall gleiches σ^2)
4. Bei multipler Regression dürfen die x_j nicht untereinander korreliert sein (Multikollinearitätsbedingung).
5. Die Residuen e und die y-Werte müssen normalverteilt sein.

Hierbei ist es besonders wichtig, dass die Varianz der Residuen über den gesamten Bereich von x homogen ist, d.h. die Streuung der Residuen darf nicht zu- oder abnehmen und es dürfen keine systematischen Muster erkennbar sein.

Weitere Hinweise über Grundlagen und Durchführung von Regressionsanalysen sind in KÖHLER *et al.* (2002), SACHS (1992), ZAR (1996) oder anderen Statistikbüchern zu finden, z.B. zur Berechnung von Vertrauensintervallen, zur Testung der Signifikanz von a oder zu Alternativen zur Methode der kleinsten Quadrate.

2.1.2 Implementierung in R

Modellformeln in R

In R existiert eine spezielle Formelsyntax zur Beschreibung von statistischen Modellen. Ein einfaches lineares Modell (y versus x) beschreibt man z.B. mit:

```
y ~ x + 1
```

oder kurz (+ 1 kann weggelassen werden):

```
y ~ x
```

eine lineare Regression, die durch den Ursprung geht (also ohne Achsenabschnitt) mit

```
y ~ x - 1
```

und eine doppelt logarithmisch transformierte Funktion mit:

```
log(y) ~ log(x)
```

Die komplette Formelsyntax und weitere Beispiele sind in der R-Dokumentation enthalten, z.B. in VENABLES *et al.* (2001).

Ein einfaches Beispiel

Wir erzeugen zunächst einen Vektor mit 10 `x`-Werten:

```
> x <- 1:10
```

und einen davon abhängigen Vektor von `y`-Werten

```
> y <- 2 + 0.5 * x + rnorm(x, s=0.5)
```

hierbei liefert `rnorm(x)` einen Vektor mit der gleichen Anzahl von Zufallszahlen, wie `x` Werte enthält. Die Zufallszahlen entstammen einer standardnormalverteilten Grundgesamtheit mit einem Mittelwert $\mu = 0$ und einer Standardabweichung $\sigma = 0.5$. Zunächst plotten wir die Daten mit:

```
> plot(x, y)
```

Zur Anpassung eines linearen Modells kann die Funktion `lm()` verwendet werden:

```
> reg <- lm(y ~ x)
```

Das von `lm` zurückgelieferte R-Objekt (in unserem Fall `reg` genannt) enthält die kompletten Regressionsergebnisse, Signifikanztests, Residuen und weitere Informationen, die durch spezielle R-Funktionen extrahiert werden können. So liefert

```
> summary(reg)

Call: lm(formula = y ~ x)

Residuals:
    Min      1Q  Median      3Q     Max
-0.5318 -0.2818 -0.1929  0.3000  0.9199

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  2.27388    0.34886   6.518 0.000185 ***
x            0.47123    0.05622   8.381 3.12e-05 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
Residual standard error: 0.5107 on 8 degrees of freedom Multiple
R-Squared: 0.8978,     Adjusted R-squared: 0.885 F-statistic:
70.24 on 1 and 8 DF,  p-value: 3.119e-05
```

die wichtigsten Ergebnisse der Regression. Im Beispiel (die exakten Zahlenwerte können wegen des Zufallszahlengenerators natürlich abweichen) erhalten wir zunächst einige statistische Parameter der Residuen, die eine erste Regressionsdiagnostik ermöglichen. Die danach folgende Tabelle zeigt die gefundenen Regressionsparameter, den Achsenabschnitt (Parameter a, `Intercept`) und die Steigung b bezüglich der Variablen `x`. Bei der Verwendung mehrerer unabhängiger Variablen erhalten wir für jede Variable eine separate Zeile. In dieser Tabelle enthält die erste Spalte den geschätzten Parameter (a bzw. b) die zweite Spalte deren Standardfehler (s_a, s_b) und die folgenden Spalten Angaben zur Signifikanz als t-Wert als Irrtumswahrscheinlichkeit und als Signifikanzcode.

Anschließend wird der Standardfehler der Residuen, das Bestimmtheitsmaß und der F-Test nach Gleichung eg:fsimplereg zur Prüfung der Signifikanz der Regressionsgleichung ausgegeben. Der angepasste Determinationskoeffizient (`Adjuster R-squared`) berücksichtigt neben dem Verhältnis aus erklärter Varianz und Gesamtvarianz auch das Verhältnis aus Stichprobenumfang und Anzahl der benutzten Parameter (p, in unserem Fall sind das zwei: a und b):

$$r^2_{adj} = 1 - \frac{1 - r^2 \cdot (n-1)}{n - p - 1} \tag{2.19}$$

Die `abline`-Funktion ist eine Universalfunktion zum Zeichnen von Geraden (siehe online-Hilfe). Wird sie mit einem Objekt aus einem linearen Modell (lm-Objekt) aufgeerufen, erhält man die Regressionsgerade:

```
> abline(reg)
```

Mit Hilfe von `predict()` lassen sich für gegebene neue x-Werte die y-Werte berechnen

```
> x1 <- c(1.5, 2.5, 7.5)
> y1 <- predict(reg, data.frame(x=x1))
> points(x1, y1, col="green")
```

Hierbei ist zu beachten, dass `predict` die neuen Daten in einem Dataframe mit den gleichen Variablennamen erwartet, wie sie beim zugehörigen Aufruf von `lm` verwendet wurden.

Konfidenzintervalle der Regressionsparameter

Die Vertrauensintervalle für die Parameter a und b lassen sich nach Gleichung 2.11 und 2.12 ableiten. Hierbei extrahieren wie die Standardfehler der Parameter aus der 2. Spalte der Regressions-Zusammenfassung (`summary(reg)`, das t-Quantil $t_{1-\alpha/2,n-2}$ erhalten wir mit Hilfe der Funktion `qt`. Der Vektor `c(-1, 0, 1)` bewirkt, dass wir in einer Zeile das Minimum, den Mittelwert und das Maximum des jeweiligen Parameters erhalten:

```
> alpha <- 0.05
> n     <- length(x)
> tval    <- qt(1 - alpha/2, n-2)
> s.ab <- summary(reg)$coefficients[,2]
> coef(reg)[1] + c(-1, 0, 1) * tval *  s.ab[1] # a
> coef(reg)[2] + c(-1, 0, 1) * tval *  s.ab[2] # b
```

Konfidenzintervall der Geraden und Vorhersageintervall

Mit Hilfe von `predict` lassen sich auf einfache Weise die Konfidenz- und die Vorhersageintervalle in eine Regressionsgrafik einzeichnen. Zunächst erzeugen wir uns eine Anzahl von Stützstellen über den Definitionsbereich (x_{min}, x_{max}) in einem Dataframe mit einem den Daten entsprechenden Variablennamen (`x`):

```
>newdata <- data.frame(x=seq(min(x), max(x), length=100))
```

Anschließend lassen wir uns die Intervalle berechnen:

```
>conflim <- predict(reg, newdata=newdata, interval="confidence")
>predlim <- predict(reg, newdata=newdata, interval="prediction")
```

Das Ergebnis ist eine Matrix mit 3 Spalten, wobei die erste Spalte die eigentliche Vorhersage (also die Gerade) und die beiden anderen Spalten die Intervalle enthalten:

```
>lines(newdata$x, conflim[,2], col="blue", lty="dashed")
>lines(newdata$x, conflim[,3], col="blue", lty="dashed")
>lines(newdata$x, predlim[,2], col="red")
>lines(newdata$x, predlim[,3], col="red")
```

Die standardmäßig verwendete Irrtumswahrscheinlichkeit $\alpha = 0.05$ lässt sich selbst verständlich ändern. Näheres dazu enthält die Hilfe zur Funktion `predict.lm`.

Es existieren zahlreiche weitere Möglichkeiten, z.B. liefert `coefficients(reg)` die Koeffizienten für ihre weitere Verwendung und mit Hilfe von `names(reg)` oder `str(reg)` lassen sich die in `reg` enthaltenen Elemente oder eine Zusammenfassung der kompletten Datenstruktur anzeigen. Sehr wichtig ist auch die `residuals`-Funktion. Mit ihrer Hilfe lässt sich eine Diagnostik der Residuenverteilung durchführen. So erhält man z.B. mit Hilfe von:

```
> plot(x, residuals(reg))
> qqnorm(residuals(reg))
```

Informationen über eine eventuelle Abhängigkeit der Residuenvarianz von x oder über Abweichungen von der Normalverteilung. Wichtige diagnostische grafiken liefert jedoch auch ein Aufruf von:

```
> plot(reg)
```

2.1.3 Übung: Beziehung zwischen Chlorophyll und Phosphat

Problem

Die Datei `oecd.dat` enthält aus einer Grafik aus VOLLENWEIDER and KEREKES (1980) digitalisierte Jahresmittel von Gesamtphosphat (TP $\mu g l^{-1}$) und Chlorophyll a (CHLa in $\mu g l^{-1}$) von insgesamt 92 Seen[2]. Die Daten sollen grafisch dargestellt und eine geeignete Regressionsgerade angepasst werden.

Lösung

Zunächst werden die Daten aus einer Textdatei eingelesen, wobei die erste Zeile den Variablennamen enthält (`header=TRUE`). Mit Hilfe von `attach(mydata)` kann direkt auf die im Dataframe `mydata` vorhandenen Variablen zugegriffen werden.

```
> mydata <- read.table("oecd.dat", header=TRUE)
> attach(mydata)
> plot(TP, CHLa)
> reg <- lm(CHLa~TP)
> abline(reg)
> summary(reg)
```

Die Darstellung deutet darauf hin, dass die Voraussetzungen der linearen Regression grob verletzt wurden und sowohl die x- als auch die y-Achse transformiert werden muss. Wie in der Originalarbeit verwenden wir eine logarithmische Transformation. Hierbei benutzt R, wie die meisten anderen Programmiersprachen auch, `log()` für den natürlichen Logarithmus. Um Verwechslungen zu vermeiden, ist es im Regelfall besser, in Texten „ln“ anstelle von „log“ zu schreiben. Zur Anpassung einer Geraden an die transformierten Daten kann der Logarithmus direkt in der Modellformel innerhalb von `lm` verwendet werden:

```
> plot(log(TP), log(CHLa))
> reg <- lm(log(CHLa)~log(TP))
> abline(reg)
> summary(reg)
```

2.1.4 Weitere Aufgaben

1. Wandeln Sie die Gleichung $\ln(CHLa) = a + b \cdot \ln(TP)$ in die Form $CHLa = a' \cdot TP^b$ um.
2. Entfernen Sie die Daten, bei denen nicht Phosphor (P), sondern Stickstoff (N) oder Licht (I) der limitierende Faktor ist und rechnen Sie die Regression neu. Stellen Sie alle Geraden in einer gemeinsamen Abbildung dar.

[2] Da einzelne Punkte auf der Grafik übereinanderliegen, fehlen 2 der ursprünglichen 94 Seen.

3. Berechnen Sie das Bestimmtheitsmaß mittels Gleichung 2.8 und vergleichen Sie diesen Wert mit dem von R ausgegebenen Ergebnis.
4. Zeichnen Sie Konfidenz- und Vorhersageintervalle in die Grafiken ein und interpretieren Sie diese.

2.2 Polynome

2.2.1 Grundlagen

Polynome lassen sich als lineare Modelle ansehen, auch wenn die Funktion

$$y_i = \beta_0 + \beta_1 x_i + \beta_2 x_i^2 + \cdots \beta_p x_i^p + \varepsilon \tag{2.20}$$

und die dazugehörigen Kurven scheinbar nichtlinear sind. Der Grund hierfür ist, dass sich die Parameter β_j über ein lineares Gleichungssystem mit $p+1$ Gleichungen und $p+1$ Unbekannten analytisch lösen (direkt ausrechnen) lassen. Hierbei nennt man p die Ordnung des Polynoms. Dieses Gleichungssystem wird, genauso wie bei der linearen Regression, mit Hilfe der partiellen Ableitungen aufgestellt und ist in gängigen Statistikprogrammen bereits enthalten.

Mit Polynomen lassen sich in der Praxis nahezu alle nichtlinearen Zusammenhänge zwischen zwei oder auch mehreren Variablen (z.B. polynomiale Flächen) beschreiben. Hierbei ist folgendes zu beachten:

1. Obwohl die Datenpunkte meist sehr gut getroffen werden, ergeben sich insbesondere bei höheren Polynomordnungen Schwierigkeiten, da die Kurven zwischen den Stützstellen zu „Schwingungen“ neigen und außerhalb des Definitionsbereiches „weglaufen“ können. Polynome dürfen deshalb immer nur zur Interpolation und nicht zur Extrapolation verwendet werden.
2. Bei höheren Polynomordnungen treten häufig numerische Probleme auf, d.h. die numerische Genauigkeit des Computers reicht nicht aus, um die Koeffizienten der höheren Polynomgrade zu berechnen.[3] Die Polynomkoeffizienten (insbesondere die höheren Grade) müssen oft mit sehr vielen Stellen angegeben werden.
3. Bei n Datenpunkten ist eine maximale Polynomordnung von $p = n-1$ möglich. Falls keine numerischen Probleme auftreten, geht die Kurve in diesem Fall exakt durch alle Punkte.

Es existieren zahlreiche Anwendungsbeispiele. So basieren z.B. die Formeln für die Sauerstoffsättigung des Wassers oder die Dichte des Wassers auf Polynomen (FOFONOFF, 1983; AMERICAN PUBLIC HEALTH ASSOCIATION, 1992). Allerdings sind Polynome meist lediglich „empirische Interpolationsformeln“, d.h. ohne biologische, chemische oder physikalische Begründung. Wenn immer möglich, sollten anstelle von Polynomen solche Funktionstypen verwendet werden, für die eine den Daten entsprechende naturwissenschaftliche Begründung existiert (analytische Funktionen).

[3] Als Alternative können auch orthogonale Polynome (`y ~ poly(x, degree)`) verwendet werden, hierbei treten jedoch andere praktische Probleme auf, z.B. funktioniert die Funktion `predict` anders als erwartet.

2.2.2 Implementierung in R

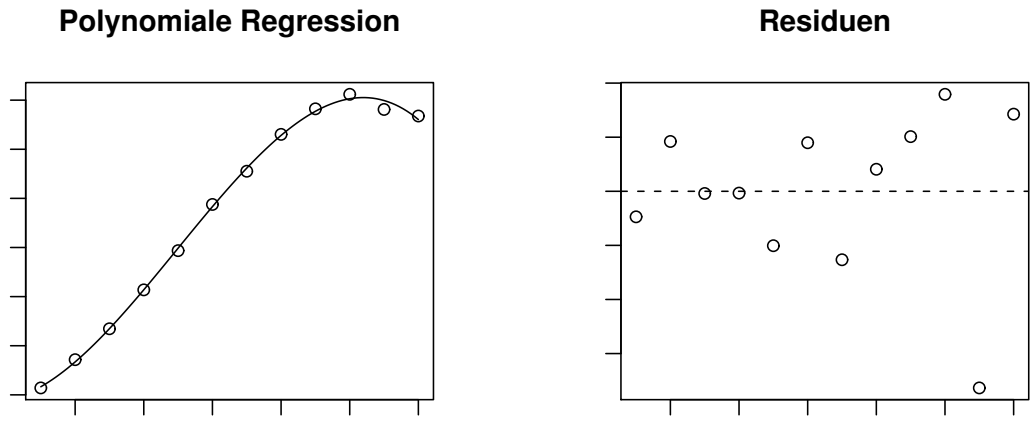

Abbildung 2.3: Polynomiale Regression (links) und Residuen (rechts).

Zunächst einmal erzeugen wir einen Testdatensatz mit Hilfe von Zufallszahlen, im Beispiel ein Polynom 3. Grades mit einem normalverteilten Zufallsfehler und einer Standardabweichung von $\sigma = 3$:

```
> x <- seq(1,12,1)
> y <- 2*x + 3*x^2 - 0.2*x^3 + 3*rnorm(x)
> plot(x, y)
```

Zur Ermittlung der Parameter („fitten“ des Modells) verwenden wir die Funktion `lm()` und die Modellformel:

```
    y ~ 1 + I(x) + I(x^2) + I(x^3)}
```

Hierbei ist `I()` eine spezielle Schreibweise (*as is*), die bewirkt, dass der in der Klammer stehende Term arithmetisch (also als Potenz) und nicht „symbolisch“ im Sinne einer R-Modellformel ausgewertet wird (siehe `help(I)`). Die `1` als Symbol für den Achsenabschnitt kann auch weggelassen werden.

```
> reg <- lm(y ~ x + I(x^2) + I(x^3))
> print(summary(reg))
> lines(x,reg$fitted.values, col="red")
```

Eine glattere Kurve erhalten wir, wenn wir eine höhere Anzahl (z.B. 100) Stützstellen verwenden

```
> x1 <- seq(min(x), max(x), length=100)
> y1 <- predict(reg, data.frame(x=x1))
> lines(x1, y1, col="green")
```

Je nach den verwendeten Zufallszahlen erhalten wir dann ein Ergebnis ähnlich Abbildung 2.3.

2.2.3 Übung

Problem

Für die Anwendung von Seenmodellen auf konkrete Gewässer oder für die Berechnung von Massenbilanzen benötigt man oft Funktionen, die die Abhängigkeit von Seefläche bzw. Volumen von der Wassertiefe oder dem Stauspiegel angeben (hypsografische Funktion oder Schlüsselkurve genannt). Es soll ein solches Polynom für die Abhängigkeit des Flächeninhaltes vom Wasserspiegel für den Datensatz `hypso.dat` gefunden werden.

Lösung

Zunächst sehen wir uns die Variablennamen in der Datendatei an und passen das obige Beispiel entsprechend an:

```
> mydata <- read.table("hypso.dat", header=TRUE)
> attach(mydata)
> plot(Level, Area)
> reg <- lm(Area ~ I(Level) + I(Level^2))
> print(summary(reg))

> lines(Level, reg$fitted.values, col="red")
> x1 <- seq(0, 80, length=100)
> y1 <- predict(reg, data.frame(Level=x1))
> lines(x1, y1, col="green")
```

Anschließend testen wir auch andere Polynomgrade (z.B. $1 \ldots 4$) und sehen uns das Bestimmtheitsmaß und den Kurvenverlauf näher an.

Hinweise

Bei der Anpassung hypsografischer Funktionen ist zu beachten, dass die Flächen- und die Volumenkurven nicht voneinander unabhängig sind, da sich das Volumen als Integral der Fläche über die Tiefe ergibt (schichtenweise Aufsummierung). Stimmt dieser Zusammenhang nicht, können Bilanzfehler auftreten, d.h. Masse entsteht aus dem Nichts bzw. verschwindet spurlos. In der Praxis ermittelt man im Regelfall die Flächen aus einer Karte oder durch Lotung und integriert dann analytisch (was bei Polynomen besonders einfach ist) oder numerisch (was zu einem gewissen Fehler führt).

Wichtig ist auch der Verlauf der Kurve in der Nähe des Nullpunktes, besonders wenn mit hypolimnischen Volumina gerechnet wird. Viele in der Praxis verwendete Kurven gehen nicht durch den Nullpunkt (sogenannter Totraum) und führen zu entsprechenden Problemen.

2.2.4 Weitere Aufgaben

1. Berechnen Sie ein Regressionspolynom für das Volumen und vergleichen Sie dieses mit der analytisch integrierten Form.
2. Bilden Sie die erste Ableitung des Volumenpolynoms und vergleichen Sie dieses mit der Flächenkurve.
3. Bestimmen Sie Regressionspolynome mit einer Ordnung von 1 bis zur maximal möglichen Ordnung für den folgenden Datensatz und interpretieren Sie das Ergebnis

   ```
   x <- c(1, 2, 3, 6, 8, 9, 10)
   y <- c(1, 3, 2, 3, 9, 8, 7)
   ```

4. Extrapolieren Sie die erhaltenen Polynome über den Definitionsbereich hinaus (z.B. von -50 bis + 50), hierbei müssen eventuell die Definitions- und Wertebereiche der Grafik angepasst werden (`plot(x, y, xlim=c(-50,50))`).

2.3 Periodische Funktionen

2.3.1 Grundlagen

In der Natur treten häufig periodische Phänomene auf, z.B. der Jahreszyklus von Globalstrahlung und Temperatur, der Entwicklungszyklus der Vegetation oder auch die die Beinschlagbewegung eines Wasserflohs (*Daphnia*). Zur Beschreibung solcher Funktionen (z.B. Abb. 2.4) können Methoden der Zeitreihenanalyse verwendet werden. Hierzu existiert eine sehr umfangreiche und ausgefeilte Methodik (z.B. BOX *et al.*, 1994; SCHLITTGEN and STREITBERG, 1989; VENABLES and RIPLEY, 1999). An dieser Stelle soll jedoch ganz pragmatisch nur die Anwendung der harmonischen Analyse (Fourieranalyse) zur Approximation periodischer Datenreihen vorgestellt werden.

Der Konvention der Zeitreihenanalyse entsprechend bezeichnen wir hier die Variablen nicht mit x und y sondern die unabhängige Variable mit t und die abhängige Variable mit x. Für das hier verwendete Verfahren müssen einige Voraussetzungen eingehalten werden. Zum einen müssen die Daten streng äquidistant vorliegen und zum anderen muss die Datenreihe stationär sein, d.h. Mittelwert, Varianz und Covarianz dürfen sich über die Zeit nicht verändern. Außerdem wird, wie bei allen hier vorgestellten Regressionsverfahren, von normalverteilten Residuen ausgegangen.

Wenn die Residuen nicht normalverteilt sind, wenn Trends vorliegen oder wenn sich die Varianz der Zeitreihe proportional zu einem Trend verändert, ist es oft angebracht, die Zeitreihe zu transformieren und gegebenenfalls den Trend vorher zu eliminieren (z.B. durch gleitende Mittelwerte oder eine

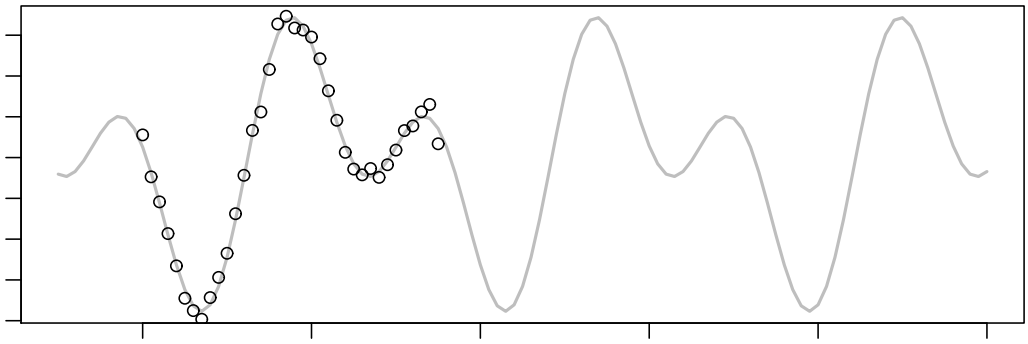

Abbildung 2.4: Anpassung einer periodischen Funktion 2. Ordnung

lineare Regression). So ist z.B. für Biomassedaten wegen der in vielen Fällen vorherrschenden linksgipfligen Verteilung oft eine logarithmische Transformation hilfreich. Eine Übersicht über praktikable Transformationen für Wassergütevariablen findet sich in HÅKANSON and PETERS (1995).

Prinzipiell läßt sich jede Zeitreihe durch eine Summe aus Sinus- und Kosinustermen mit unterschiedlicher Periode (Fourierreihe) darstellen (CHATFIELD, 1982; ENGELN-MÜLLGES and REUTTER, 1996):

$$x_t = a_0 + \sum_{p=1}^{N/2-1} \left(a_p \cos(2\pi pt/N) + b_p \sin(2\pi pt/N)\right) + a_{N/2}\cos(\pi t), \qquad t = 1 \ldots N \tag{2.21}$$

Hierbei ist a_0 der Mittelwert der Zeitreihe, a_p, b_p sind die Koeffizienten der Fourierreihe und N ist die Länge der Zeitreihe oder die Periodendauer. Ähnlich wie bei der linearen und der polynomialen Regression, lassen sich auch hier die Koeffizienten mit Hilfe eines linearen Gleichungssystems analytisch ermitteln, d.h. es sind lediglich entsprechende Summen zu bilden:

$$a_0 = \bar{x} \tag{2.22}$$

$$a_{N/2} = \sum_{t=1}^{N} (-1)^t x_t / N \tag{2.23}$$

$$a_p = 2\frac{\sum_{t=1}^{N} x_t \cos(2\pi pt/N)}{N} \tag{2.24}$$

$$b_p = 2\frac{\sum_{t=1}^{N} x_t \sin(2\pi pt/N)}{N} \tag{2.25}$$

Darüberhinaus existiert eine besonders leistungsfähige Methode, die „schnelle Fouriertransformation“ (FFT, *fast fourier transform*), mit der die Koeffizienten a_0, a_p, b_p sehr effizient mit Hilfe von komplexen Zahlen ermittelt werden können.

Die Gleichung 2.21 kann auch in eine Form mit nur einem Kosinusterm umgewandelt werden:

$$x_t = a_0 + \sum(R_p \cdot \cos(2\pi pt/N + \Phi_p) \tag{2.26}$$

mit:

$$R_p = \sqrt{a_p^2 + b_p^2} \tag{2.27}$$

$$\Phi_p = \arctan(-b_p/a_p) \tag{2.28}$$

Dies hat den Vorteil, dass die Amplituden R_i und die Phasenverschiebungen Φ_i der einzelnen Frequenzen ($2\pi/N$, $4\pi/N$, …, π) direkt ablesbar sind. Trägt man $R_p^2/2$, den Varianzanteil des p-ten harmonischen Terms, gegen die Frequenz $\omega_p = 2\pi p/N$ grafisch auf, ergibt sich das Periodogramm (welches unter Umständen noch geglättet werden muss).

Da die harmonische Analyse den untersuchten Prozeß in die Varianzanteile einzelner Frequenzen zerlegt, ist es auch möglich, eine Zeitreihe auf der Basis ausgewählter Frequenzanteile zu synthetisieren.

2.3.2 Implementierung in R

Wir benötigen zwei Hilfsfunktionen, eine Funktion zur Ermittlung der Koeffizienten a_p und b_p (Gl. 2.22 bis 2.25) und eine Funktion zur Synthese der harmonischen Funktion entsprechend Gleichung 2.21. Wegen der einfacheren Verständlichkeit sind hier zunächst die „klassischen“ Versionen angegeben, es kann jedoch auch die FFT angewendet werden. Da es recht mühsam ist, die Funktionen jedes Mal abzutippen, benutzen wir für längere Befehlsfolgen (sogenannte „Skripte“) spätestens ab jetzt immer einen externen Texteditor (siehe Anhang A). Um das Arbeiten mit einem Editor anzudeuten, wird im Folgenden des Eingabeaufforderungzeichen > weggelassen.

```
## classic method of harmonic analysis
harmonic.classic <- function(x, pmax=length(x)/2) {
 n  <- length(x)
 t  <- 0:(n-1)
 a0 <- mean(x)
 a  <- numeric(pmax)
 b  <- numeric(pmax)
 for (p in 1:pmax) {
   k    <- 2 * pi * p * t / n
```

```
    a[p] <- sum(x * cos(k))
    b[p] <- sum(x * sin(k))
  }
  list(a0=a0, a=2*a/n, b=2*b/n)
}

## fast fourier version of harmonic analysis
harmonic.fft <- function(x) {
  n  <- length(x)
  pf <- fft(x)      ## Fast Fourier Transform
  a0 <- Re(pf[1])/n ## first element = mean
  pf <- pf[-1]      ## drop first element
  a  <-  2*Re(pf)/n ## Real part of complex
  b  <- -2*Im(pf)/n ## Imaginary part
  list(a0=a0, a=a, b=b)
}


## synthesis of a harmonic function
## (classic method)
synth.harmonic.classic <- function(t, fpar, n, ord) {
  a<-fpar$a; b<-fpar$b; a0<-fpar$a0
  x <- a0
  for (p in ord) {
    k <- 2 * pi * p * t/n
    x <- x + a[p] * cos(k) + b[p] * sin(k)
  }
  x
}

## synthesis of a harmonic function
## version with amplitude (R) and phase (Phi)
synth.harmonic.amplitude <- function(t, fpar, n, ord) {
  a   <- fpar$a; b<-fpar$b; a0<-fpar$a0
  R   <- sqrt(a * a + b * b)
  Phi <- atan2(-b, a)
  x   <- a0
  for (p in ord) {
    x <- x + R[p] * cos(2 * pi * p * t/n + Phi[p])
  }
  x
}
```

Ein zur Testung geeigneter Datensatz kann mit Hilfe von Sinus- und Kosinusfunktionen erzeugt werden, ein normalverteilter Fehlerterm fügt ein gewisses „Rauschen“ hinzu:

```
n <- 36
t <- 0:(n-1)
x <- 2 + sin(t*4*pi/n+2) + sin(t*2*pi/n + 4) + rnorm(n)*0.1
```

Die Ermittlung der Koeffizienten erfolgt nun mit:

```
fpar <- harmonic.classic(x)
```

Danach zeigt der Aufruf von `fpar` die ermittelten Koeffizienten an. Die Regressionskurve und ihre grafische Darstellung erhalten wir mit:

```
t1<-seq(min(t), max(t), length=100)
x1 <-  synth.harmonic.amplitude(t1, fpar, n, ord=1:(n/2))
plot(t1, x1,col="gray", type="l")
points(t,x)
```

Hierbei ist `t1` ein willkürlich festgelegter Bereich, für den die Funktion gezeichnet werden soll, `fpar` enthält die Fourierkoeffizienten und `n` ist die Periodendauer (Anzahl der Werte des Originaldatensatzes). Mit dem Vektor `ord` werden die harmonischen Ordnungen angegeben, die zur Synthese der Funktion verwendet werden sollen. Es müssen nicht unbedingt alle Ordnungen von 1 bis $n/2$ benutzt werden, sondern es können auch einzelne Ordnungen herausgegriffen werden.

Auf ähnliche Weise wie mit `synth.harmonic`, kann die Berechnung mit Hilfe der Koeffizienten a_0, a_p, b_p und der Gleichung 2.21 auch außerhalb von R (z.B. in einer Tabellenkalkulation) erfolgen.

2.3.3 Übung

Problem

Als Antrieb ökologischer Modelle werden häufig Jahresgänge von Temperatur und Globalstrahlung benötigt. Eine einfache Möglichkeit besteht darin, diese durch eine harmonische Funktion mit einer niedrigen Ordnung darzustellen. Finden Sie eine harmonische Funktion zur Beschreibung des mittleren Jahresganges der Globalstrahlung in $\mathrm{J}\,cm^{-2}d^{-1}$. Benutzen Sie hierzu die Daten von 1981 bis 1990 der Station Wahnsdorf bei Dresden (Quelle: Täglicher Wetterbericht des Meteorologischen Dienstes der DDR oder Online-Zugriff über das World Radiation Data Center, siehe Anhang B).

Lösung

Zunächst müssen die Hilfsfunktionen für die harmonische Analyse (s.o.) eingegeben oder eingelesen werden. Anschließend werden die Daten aus einer Textdatei eingelesen, wobei die erste Zeile die Variablennamen enthält (`header=TRUE`). Wir lassen uns die Variablennamen anzeigen, kopieren die Spalte `igl` in die Variable `x` und erzeugen uns, um die nicht ganz einfache Datumsrechnung zu vermeiden, eine neue Zeitvariable `t`.

```
mydata <- read.table("igl8190.dat", header=TRUE)
names(mydata)
x   <- mydata$igl
t   <- 1:length(x)
```

Anschließend stellen wir die Daten grafisch als Zeitreihe über mehrere Jahre dar:

```
plot(t, x)
```

Alternativ können auch alle Jahre übereinander geplottet werden, wobei der Operator `%%` der Modulo-Operator (Rest einer ganzzahligen Division) ist. Das bedeutet in diesem Fall, dass der Tag 366 wieder bei 1 geplottet wird.

```
plot(t%%365, x)
```

Nach dieser ersten Orientierung folgt nun die eigentliche Analyse (dieses Mal über die FFT) und die grafische Darstellung des Ergebnisses:

```
fpar <- harmonic.fft(x)
plot(t, x)
lines(synth.harmonic.classic(t,fpar,length(x),ord=10), col="red")
```

Da der verwendete Datensatz 10 Jahre enthält, stellt `ord=10` einen jährlichen Zyklus dar. Wir experimentieren nun ein wenig mit der Fourierordnung herum und stellen z.B. die Ordnungen `ord=1`, `ord=1:10`, `ord=1:100` dar.

Lösung 2

Eine alternative Möglichkeit besteht darin, zunächst die 10jährigen Mittel für alle 365 Tage zu berechnen und anschließend direkt eine harmonische Funktion 1. Ordnung zu berechnen. Die Mittelwertberechnung kann extern in einem Tabellenkalkulationsprogramm oder mit der sehr leistungsfähigen R-Funktion `aggregate` erfolgen, die ihre Argumente als Dataframes oder Listen erwartet. Das erste Argument kennzeichnet die auszuwertenden Daten, das zweite Argument die Gruppierung und das dritte Argument die anzuwendende Funktion.

```
meanyear <- aggregate(list(x=x), list(yr=t%%365), mean)
x <- meanyear$x
t <- 1:length(x)
plot(t, x)
fpar <- harmonic.fft(x)
lines(synth.harmonic.classic(t,fpar,length(x),ord=1), col="red")
```

Zur weiteren Verwendung ist es nun notwendig, die ermittelte Funktion in einer geschlossenen Form aufzuschreiben, wobei sich die Koeffizienten ja bekanntlich in der Liste `fpar` befinden. Bei Angabe nur des Jahreszyklus ($p = 1$) ergibt sich aus Gleichung 2.21

```
> fpar$a0; fpar$a[1]; fpar$b[1]
[1] 996.746
[1] -815.988
[1] 126.741
> plot(t,x)
> x1 <- 997  - 816 * cos(2*pi*1*t/365) + 126 * sin(2*pi*1*t/365)
> lines(t,x1)
```

Als mathematische Funktion geschrieben ergibt das:

$$x = 997 - 816 \cdot \cos(2\pi \cdot t/365) + 126 \cdot \sin(2\pi \cdot t/365) \tag{2.29}$$

In unserem Beispiel ist es wegen des Jahreszyklus mehr oder weniger offensichtlich, welche Fourierordnungen benötigt werden. In der Regel ist das aber im voraus nicht bekannt und muss über eine Periodogramm- oder Frequenzanalyse ermittelt werden (siehe SCHLITTGEN and STREITBERG, 1989; BOX *et al.*, 1994).

2.3.4 Weitere Aufgaben

1. Ermitteln Sie die Amplitude und die Phasenverschiebung der Globalstrahlungs-Funktion.
2. Stimmt das Ergebnis mit Ihren Erwartungen überein (Kalender).
3. Stellen Sie die Funktion in einer Form nach Gleichung 2.26 dar und prüfen Sie das Ergebnis grafisch.
4. Vergleichen Sie die Globalstrahlungsfunktion 10. Ordnung von Lösung 1 mit der Funktion 1. Ordnung von Lösung 2.
5. Stellen Sie die Funktion und die Daten mit Ihrem bevorzugten Grafik- oder Tabellenkalkulationsprogramm dar.
6. Versuchen Sie, die Epilimniontemperatur eines Gewässers mit einer harmonischen Funktion zu beschreiben (Datensatz `t_epi7.dat`). Wieviele Fourier-Ordnungen werden benötigt?

2.4 Nichtlineare Regression

2.4.1 Grundlagen

Außer den Polynomen gibt es noch weitere Regressionsfunktionen, für die analytische Lösungen nach der Methode der kleinsten Quadrate existieren. Für einige weitere Funktionen existieren Linearisierungen, d.h. eine Transformation, um die nichtlineare Funktion in eine lineare Funktion umzuwandeln. So entspricht z.B.:

$$y = a \cdot x^b \tag{2.30}$$

der Funktion

$$\ln(y) = \ln(a) + b \cdot \ln(x) \tag{2.31}$$

Hierbei ist zu beachten, dass eine solche Transformation auch die Residuen transformiert. Das ist richtig und notwendig, wenn dadurch die statistischen Voraussetzungen für die Regressionsanalyse erst hergestellt werden, z.B. im obigen Beispiel der Abhängigkeit des Chlorophylls vom Gesamtphosphat.

In vielen Fällen (z.B. sehr oft bei der Monodfunktion) führt jedoch die Linearisierung zu einer unzulässigen Veränderung der Residuen mit dem Ergebnis, dass die Varianzhomogenität der Residuen verletzt wird und die Regressionsparameter verfälscht werden (bias).

Zur Lösung dieses Problems existieren unterschiedliche Ansätze, z.B. die Verwendung einer Wichtung (siehe ENGELN-MÜLLGES and REUTTER, 1996) oder die direkte numerische Anpassung der Funktion (numerische Optimierung).

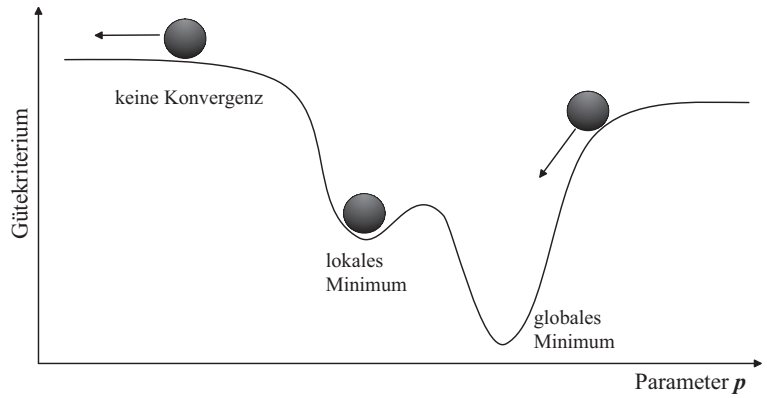

Abbildung 2.5: Numerisches Optimierungsproblem: Ziel ist das Finden des globalen Minimums einer Funktion, z.B. der Summe der kleinsten Quadrate als Gütekriterium. Sind mehrere Parameter p_i zu optimieren, ergibt sich ein mehrdimensionales „Gütegebirge“.

Numerische Optimierungsverfahren

In Fällen, in denen keine analytische Lösung verfügbar ist, wenn keine Linearisierung existiert oder wenn durch die Linearisierung die Voraussetzungen der Regressionsrechnung verletzt werden, muss ein numerisches Optimierungsverfahren angewendet werden („echt nichtlineare“ Regression)[4]. Das bedeutet, dass sich die Parameter der Regressionsgleichung nicht „direkt“, d.h. **analytisch** berechnen lassen, sondern dass sie durch ein Iterationsverfahren schrittweise, d.h. **numerisch** angenähert werden müssen.

[4]Manchmal werden Optimierungsverfahren auch aus Bequemlichkeit angewendet, obwohl eigentlich eine analytische Lösung existiert.

Wie auch bei der linearen Regression, wird die Güte der Anpassung meist durch die Summe der Abweichungsquadrate SQ gemessen:

$$SQ = \sum (y_i - f(\mathbf{x}_i, \mathbf{p}))^2 = \min! \tag{2.32}$$

wobei y die abhängige Variable, $\mathbf{x}$ die unabhängigen Variablen und $\mathbf{p}$ der Parametervektor sind.

Es existieren viele unterschiedliche Optimierungsverfahren und viele Originalpublikationen und Lehrbücher (z.B. EVERITT, 1987; BATES and WATTS, 1988; PRESS *et al.*, 1992; ENGELN-MÜLLGES and REUTTER, 1996). Die einzelnen Verfahren unterscheiden sich z.B. dadurch, ob die Ableitungen der Funktionen benötigt werden oder ob die Suche ohne Ableitungen auskommt (ableitungsfrei), nach ihrer Effizienz und ihrem Verhalten bei numerisch schwierigen Problemen.

Bei den Verfahren vom Newton-Typ (z.B. Gauss-Newton-Verfahren, Newton-Raphson-Verfahren, Quasi-Newton-Verfahren) sind die partiellen zweiten Ableitungen (Hesse-Matrix) erforderlich oder werden intern numerisch geschätzt, dadurch sind diese Verfahren sehr effizient und konvergieren schnell.

Bei den Gradienten- oder Simplexverfahren wird die Richtung des steilsten Abstieges verfolgt. Die Verfahren sind weniger effizient, empfehlen sich aber z.B. zur Gewinnung von Startwerten beim Vorliegen lokaler Minima.

In sehr schwierigen Fällen (z.B. für die Kalibrierung komplexer Modelle) werden stochastische Verfahren z.B. Monte-Carlo-Methoden oder sogenannte „Evolutionsstrategien“ angewendet.

Die allgemeine Verfügbarkeit schneller Rechner und leistungsfähiger Algorithmen in Statistikpaketen und Tabellenkalkulationen hat dazu geführt, dass Optimierungsverfahren heute in vielen Fällen recht einfach angewendet werden können. Trotzdem sind eine gewisse Vorsicht und Fingerspitzengefühl immer angebracht.

Wahl eines geeigneten Modells

Die Auswahl eines geeigneten Regressionsmodells (Regressionsfunktion) für ein gegebenes Problem oder einen gegebenen Datensatz kann nicht vom Optimierungsalgorithmus erwartet werden, sondern muss vom Anwender vorgenommen werden. Im Idealfall führen physikalische, chemische, biologische oder andere theoretische Betrachtungen zu einem mechanistischen Modell (BATES and WATTS, 1988). Die Modellauswahl ist naturgemäß die Aufgabe des jeweiligen, mit seiner Materie am besten vertrauten Anwenders und deshalb für uns als Naturwissenschafter ein wichtiger wissenschaftlicher Bestandteil der Modellbildung.

Bei der Auswahl des Regressionsmodells helfen Erfahrungswerte, ein entsprechendes Literaturstudium und geeignete Funktionsbibliotheken. Es sollten möglichst einfache, analytisch begründete Funktionen ausgewählt werden. Hinter guten Regressionsfunktionen (z.B. der logistischen Funktion) verbirgt sich oft die analytische Lösung eines Differentialgleichungsmodells. Darüberhinaus ist es möglich, auch als Differentialgleichung geschriebene Kompartimentmodelle zu optimieren.

Man beginnt immer mit einem einfachen Modell und baut dieses dann gegebenenfalls durch Hinzufügen weiterer Terme und Parameter schrittweise auf.

Ein Problem kann dadurch auftreten, dass einzelne Parameter des Modells zu stark voneinander abhängig sind, d.h. dass sie sich gegenseitig kompensieren. Im trivialen Fall der Gleichung:

$$y = a + \frac{b}{c} \cdot x \tag{2.33}$$

ist es völlig offensichtlich, dass b und c nicht gleichzeitig bestimmt werden können, da bei einer entsprechenden Vergrößerung von a und b der Quotient konstant bleibt. Man spricht in einem solchen Fall davon, dass die Parameter unbestimmbar sind.

Oftmals ist der Zusammenhang nicht so offensichtlich oder auch weniger streng. Vereinfacht gesagt, hängt die Bestimmbarkeit von der Anzahl der Parameter, der Anzahl der Datenpunkte und der Varianz der Residuen ab, sowie davon, wie streng die Parameter voneinander abhängen. Als Maß dafür dient der Korrelationskoeffizient der Parameter, der möglichst gering sein sollte. Sind einzelne oder alle Parameter schwer oder nicht bestimmbar, so muss man das Modell vereinfachen und Parameter zusammenfassen, den Datensatz vergrößern oder den Messfehler verringern oder einen der fraglichen Parameter in einem zusätzlichen Experiment separat bestimmen.

Festlegung der Startwerte

Allen numerischen Verfahren ist gemeinsam, dass zunächst Startwerte oder zu durchsuchende Parameterbereiche festgelegt werden müssen. Der Optimierungsalgorithmus versucht nun, ein globales Minimum der Gütefunktion (Abb. 2.5) zu finden und stoppt, wenn ein Minimum gefunden wurde oder wenn nach einer vorher festgelegten Rechenzeit oder Anzahl an Iterationsschritten noch immer keine Konvergenz erreicht ist. Die Konvergenz- und Toleranzparameter können vom Anwender festgelegt werden.

Da oftmals nicht nur ein globales Minimum sondern zusätzlich noch lokale Minima existieren können, gibt es grundsätzlich keine Gewähr dafür, in endlicher Rechenzeit das globale Minimum zu finden. Aus diesem Grunde sollten stets mehrere Optimierungsläufe mit unterschiedlichen Startwerten durchgeführt werden. Man erhält gute Startwerte meist durch Überlegen, durch manuelles Ausprobieren oder durch Näherungslösungen, z.B. mit Hilfe einer linearisierenden Transformation. Für einige Funktionen existieren spezielle Verfahren zur automatischen Ermittlung von Startwerten.

Bewertung der gefundenen Funktion

Bei der nichtlinearen Regression ist eine grafische Überprüfung besonders wichtig. Neben der Darstellung von Messwerten und Ausgleichskurve sollten auch die Residuen angeschaut werden. Hier darf sich kein Muster und keine systematische Abweichung abzeichnen und die Varianz muss homogen sein.

Ein wichtiges Kriterium ist auch das Bestimmtheitsmaß (r^2_{nl}), das bei der nichtlinearen Regression nicht einfach aus dem Quadrat des Korrelationskoeffizienten abgeleitet werden kann. Stattdessen gilt die allgemeine Formel nach Gleichung 2.8, also:

$$r_{nl}^2 = 1 - \frac{s_\varepsilon^2}{s_y^2} \tag{2.34}$$

wobei s_ε^2 die Varianz der Residuen ($\varepsilon = \hat{y} - y$) und s_y^2 die Varianz der abhängigen Variablen ist. In der Praxis kann es sogar vorkommen, dass das Bestimmtheitsmaß negativ wird, was bedeutet, dass die Residuen eine größere Varianz besitzen als die originalen Messwerte. Der Grund dafür ist eine fehlgeschlagene Optimierung. In der grafischen Darstellung erkennt man dies sofort, da die gefittete Kurve neben den Punkten liegt. Zur Lösung des Problems kann man versuchen, die Optimierung mit neuen Startwerten zu wiederholen oder man wählt einen anderen Funktionstyp für die Regressionsgleichung.

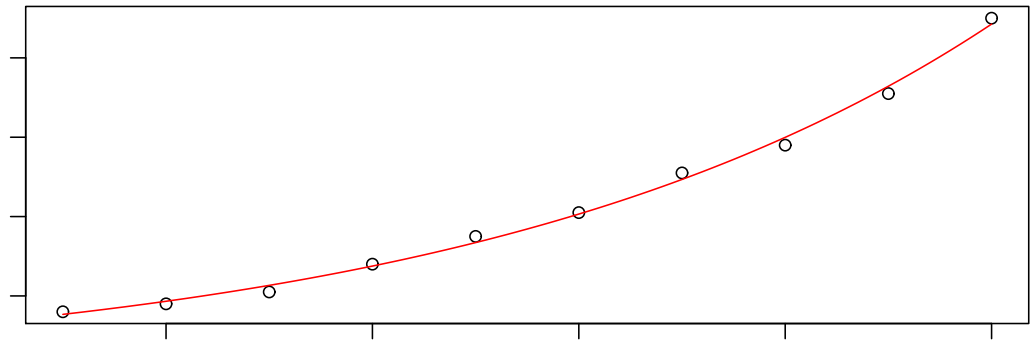

Abbildung 2.6: Exponentieller Zusammenhang.

2.4.2 Implementierung in R

In R existieren mehrere Bibliotheken zur Optimierung nichtlinearer Probleme. Für die nichtlineare Regression mit dem Verfahren der Summe der kleinsten Quadrate wird die Funktion `nls` benutzt, die standardmäßig einen Gauss-Newton-Algorithmus verwendet. Die Funktion `nls` erwartet eine Regressionsfunktion sowie die Daten und die Startwerte als Listen.

Die zu optimierende Regressionsfunktion kann in der Modellschreibweise (z.B. eine exponentielle Wachstumsfunktion als `y~a*exp(b*x)`) angegeben werden, wobei bei der nichtlinearen Regression im Unterschied zur linearen Regression alle Symbole (z.B. `^`) immer „arithmetisch“ interpretiert werden. Noch allgemeiner ist es, die Funktion als R-Funktion zu definieren:

```
f <- function(x, a, b) {
```

```
    a * exp(b * x)
}
```

Als dritte Möglichkeit können bereits vordefinierte Autostart-Funktionen genutzt werden, bei denen auch die Startwerte automatisch ermittelt werden, z.B. `SSlogis` für das logistische Modell oder `SSmicmen` für das Michaelis-Menten-Modell.

Die Vorgehensweise soll am Beispiel eines exponentiellen Modells $y = a \cdot \exp(bx)$ erläutert werden. Dieses Modell ist sehr universell und findet sich z.B. bei der Beschreibung des exponentiellen Wachstums oder als Zerfalls- oder Absorptionsgesetz wieder. Zunächst definieren wir eine R-Funktion `f`, eine Liste `mydata` mit den Daten und `pstart` mit den Startwerten für das Regressionsmodell. Mit der optionalen Angabe `trace=TRUE` erreichen wir, dass die Zwischenwerte der Optimierung angezeigt werden. Zum Schluss werden die Ergebnisse auf den Bildschirm ausgedruckt und grafisch dargestellt. Für die Funktionsauswertung (zum Beispiel für die grafische Darstellung) kann wieder die Funktion `predict` verwendet werden. Das nichtlineare Bestimmtheitsmaß berechnen wir nach Gleichung 2.34.

```
> f <- function(x, a, b) {a * exp(b * x)}

> x <- 1:10
> y <- c(1.6, 1.8, 2.1, 2.8, 3.5, 4.1, 5.1, 5.8, 7.1, 9.0)

> mydata <- list(x=x, y=y)
> pstart <- list(a=1, b=1)
> plot(x, y)

> aFit <- nls(y~f(x,a,b), data=mydata, start=pstart, trace=TRUE)

> x1 <- seq(1, 10, 0.1)
> y1 <- predict(aFit, list(x=x1))
> lines(x1, y1, col="red")

> print(summary(aFit))
> Rsquared <- 1 - var(residuals(aFit))/var(y)
> print(paste("r^2=", round(Rsquared, 4)))
```

Als Ergebnis erhalten wir die Parameterwerte mit Angaben zur Signifikanz:

```
Formula: y ~ f(x, a, b)

Parameters:
  Estimate Std. Error t value Pr(>|t|)
a 1.263586   0.049902   25.32 6.34e-09 ***
```

```
b 0.194659   0.004716   41.27 1.31e-10 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.1525 on 8 degrees of freedom

Correlation of Parameter Estimates:
        a
b -0.9682

[1] "r^2= 0.9966"
```

Der Korrelationskoeffizient zwischen a und b beträgt -0.9682, hat also einen hohen Betrag. Es besteht also eine gewisse Wechselwirkung zwischen den Parametern, die aber im vorliegenden Fall wegen der kleinen Residuen kein großes Problem darstellt. Das nichtlineare Bestimmtheitsmaß beträgt 0.9966, d.h. das verwendete Potenzmodell erklärt 99.66% der Varianz der y-Werte.

2.4.3 Übung

Problem

Anlässlich eines Praktikums[5], wurde das heterotrophe Potential (Glucose-Aufnahmerate AR in $\mu g\,\mathrm{C}\,\mathrm{l}^{-1}h^{-1}$) in Abhängigkeit vom Substratangebot (Glucosekonzentration S, in $\mu g l^{-1}$) bestimmt. Für eine Probe aus 2,5m Wassertiefe des Südwestbeckens der Fuchskuhle ergaben sich folgende Werte:

```
# Substrat ug C /l
> S  <- c(25, 25, 10, 10, 5, 5, 2.5, 2.5, 1.25, 1.25)
# Aufnahmerate ug C /(l*h)
> AR <- c(0.0998, 0.0948, 0.076, 0.0724, 0.0557,
        0.0575, 0.0399, 0.0381, 0.017, 0.0253)
```

Gesucht sind die Parameter K und V_m einer Michaelis-Menten-Kinetik:

$$AR = \frac{V_m * S}{K + S} \tag{2.35}$$

Lösung 1

Oft wird die Michaelis-Menten-Formel über Linearisierungen angepasst, meist ist für dieses Problem jedoch eine nichtlineare Optimierung vorzuziehen. Wir wandeln das `nls`-Beispiel ab und erhalten:

[5] von Studenten der TU Dresden im September 2001 am Institut für Gewässerökologie und Binnenfischerei, Abt. Geschichtete Seen

```
> f <- function(S, Vm, K) {
    Vm * S/(K + S)
}
> mydata <- list(S=S, AR=AR)
> pstart <- list(Vm=max(AR), K=5)
> aFit   <- nls(AR ~ f(S, Vm, K), data=mydata,
                  start=pstart, trace=TRUE)
> plot(S, AR, xlim=c(0, max(S)), ylim=c(0, max(AR)))
> Snew <- seq(0, 25, length=100)
> lines(Snew, predict(aFit,list(S=Snew)), col="red")
> print(summary(aFit))
> Rsquared <- 1 - var(residuals(aFit))/var(AR)
> print(paste("r^2=", round(Rsquared, 4)))
```

Damit wir eine möglichst glatte Kurve erhalten, verwenden wir zur graphischen Darstellung der Kurve wieder einen Vektor `Snew` mit 100 Werten über den Definitionsbereich von 0 bis 25.

Lösung 2

Es existiert eine noch einfachere Lösung, die Verwendung der Autostart-Funktion `SSmicmen`, bei der wir die Definition des Modells und die Angabe von Startwerten einsparen können:

```
> mydata <- list(S=S, AR=AR)
> aFit <- nls(AR ~ SSmicmen(S, Vm, K), data=mydata, trace=TRUE)
> plot(S, AR, xlim=c(0, max(S)), ylim=c(0, max(AR)))
> Snew <- seq(0, 25, length=100)
> lines(Snew, predict(aFit,list(S=Snew)), col="red")
> print(summary(aFit))
> print(paste("r^2=", round(1 - var(residuals(aFit))/var(AR), 4)))
```

2.4.4 Weitere Aufgaben

1. Linearisieren Sie das Implementierungsbeispiel (exponentielles Modell), passen Sie ein lineares Modell an und vergleichen Sie die Ergebnisse.
2. Passen Sie ein geeignetes Modell an die Daten eines Batchversuches mit dem *Microcystis aeruginosa*-Stamm PCC 7806 (JÄHNICHEN *et al.*, 2001) an:

   ```
   # Zeit (t)
   > x <- c(0, 2, 4, 6, 8, 10, 12, 14, 16, 18, 20)
   # Zellen (pro ml)
   > y <- c(0.88, 1.02, 1.43, 2.79, 4.61, 7.12,
          6.47, 8.16, 7.28, 5.67, 6.91) * 1e6
   ```

3 Kompartimentmodelle

Die bisher betrachteten empirisch-statistischen Modelle beschreiben nur die Phänomene, d.h. das Verhalten des betrachteten Systems (Verhaltensmodelle), nicht jedoch die zugrundeliegenden Ursachen, Strukturen und Funktionsmechanismen. Im Gegensatz dazu versucht man bei mechanistischen Modellen die innere Struktur in einzelne Elemente und deren Wechselbeziehungen aufzulösen. Das Gesamtsystem wird analysiert (Systemanalyse) und die einzelnen Bestandteile danach wieder zusammengesetzt (Modellsynthese). In der Ökologie werden solche Modelle oft auch als „analytische Modelle“ bezeichnet.

Zur mathematischen Beschreibung mechanistischer Modelle existieren eine Vielzahl von Methoden. Aus der Sicht des Biologen sind derzeit vor allem Differentialgleichungsmodelle, individuenbasierte Modelle (siehe Abschnitt 4) und vielleicht regelbasierte Ansätze von Bedeutung. Es gibt jedoch prinzipiell noch weitere Möglichkeiten, z.B. die Beschreibung von unscharfem Wissen mit Hilfe von Fuzzy-Logik (z.B. ZADEH, 1965, 1987; ZIMMERMANN, 1991).

Bei den Differentialgleichungsmodellen werden oft gewöhnliche Differentialgleichungssysteme 1. Ordnung verwendet und stationär (im Gleichgewichtzustand befindend) oder instationär betrachtet. Unter einem dynamischen Modell verstehen wir ein Modell, welches das Verhalten der Zustandsgrößen als Funktion der Zeit beschreibt. Auch individuenbasierte Modelle sind in diesem Sinne dynamisch.

Die hier betrachteten Differentialgleichungsmodelle haben als gemeinsames Merkmal, dass das untersuchte System in Kompartimente (z.B. Phytoplankton, Zooplankton) zerlegt wird, weshalb diese Modelle auch als Kompartimentmodelle bezeichnet werden (GODFREY, 1983; BATES and WATTS, 1988). Die Kompartimente treten im Differentialgleichungssystem als zeitlich veränderliche Zustandsgößen auf, zwischen denen ein Stoff- oder Energiefluss besteht. Der Systemrand wird durch Quellen- oder Senkenterme dargestellt. Die Transferkoeffizienten zwischen den Kompartimenten bezeichnet man als Ratenkonstanten (siehe Abb. 1.1).

3.1 Elementare Wachstumsmodelle

Nahezu jedes Buch über theoretische Ökologie oder ökologische Modellierung enthält ein oder mehrere Kapitel über elementare Wachstumsmodelle (z.B. HASTINGS, 1996; GURNEY and NISBET, 1998; ROUGHGARDEN, 1998; WILSON, 2000), darüber hinaus exisiert eine nahezu unüberschaubare Fülle von Originalpublikationen. Es soll deshalb an dieser Stelle nur ein sehr kurzer Überblick gegeben und gezeigt werden, dass die verschiedenen Modelle einem ähnlichen Grundschema folgen (TSOULARIS, 2001).

Ein weiteres Problem resultiert daraus, dass sich der Begriff Wachstum zum einen auf die Zunahme der Individuenzahl (Abundanz) einer Population (Populationswachstum), zum anderen aber auch auf die individuelle Körpermasse eines Einzelorganismus (somatisches Wachstum) oder sogar auf eine Kombination aus beiden Prozessen in Form von Biomassewachstum beziehen kann. Obwohl die hierfür verwendeten Gleichungstypen teilweise identisch sind, unterscheiden sich jedoch die den Gleichungen zu Grunde liegenden Rechtfertigungen und mechanistischen Annahmen.

Durch eine entsprechende Nomenklatur wird versucht zu verdeutlichen, wovon jeweils die Rede ist. Wir verwenden deshalb die Zustandsgröße N für die Abundanz, W für die Körpermasse eines Individuums und X für die Biomasse einer Population oder auch ganz allgemein für eine beliebige Zustandsgröße einer Differentialgleichung. Da auch die verwendeten Wachstumsparameter nicht immer einheitlich verwendet werden, benutzen wir im allgemeinen (d.h. außer bei Bezug auf eine ganz spezielle Quelle) b (*birth rate*) für die Geburtenrate, d (*death rate*) für die Mortalitätsrate und K für die Kapazitätsgrenze (*Carrying Capacity*). Der Parameter r wird oft als intrinsische Populationswachstumsrate oder intrinsische Zunahmerate bezeichnet. Wir verwenden jedoch allgemeiner den Begriff intrinsische Änderungsrate, da in r meist Geburts- und Mortalitätsprozesse zusammengefasst (aggregiert) sind.

Oftmals wird auch μ, für die Wachstumsrate verwendet, insbesondere wenn es sich um Mikroorganismen handelt. Manchmal benötigen wir auch irgendeinen oder mehrere beliebige Parameter ohne von vornherein feststehende Bedeutung. In einem solchen Fall benutzen wir den kleinen Buchstaben k oder $k_1, k_2, \ldots k_n$ oder einfach $a, b, c, d, \ldots$. Bei komplexeren Modellen, in denen viele verschiedene Prozesse beteiligt sind, hat sich die Verwendung von „sprechenden“ mehrbuchstabigen Bezeichnern oder die Verwendung von Indizes bewährt.

3.1.1 Unlimitiertes Wachstum

Zunächst einmal betrachten wir die Änderung der Abundanz oder der Biomasse einer Population (einer Zustandsgröße X) als Funktion der Zeit. Bei einem diskreten Zeitschritt und einem kontinuierlichen Wachstum (z.B. durch Immigration einer bestimmten Anzahl von Individuen pro Zeiteinheit) errechnet sich die Populationsgröße (Abundanz) zum Zeitpunkt $t + \Delta t$ aus der Populationsgröße zum Zeitpunkt t und einer allgemeinen Änderungskonstante k:

$$X_{t+\Delta t} = X_t + \Delta t \cdot k \tag{3.1}$$

und somit

$$\frac{X_{t+\Delta t} - X_t}{\Delta t} = k \tag{3.2}$$

Verkleinert man den Zeitschritt auf einen sehr kleinen Wert $\Delta t \to 0$, so wird daraus die Differentialgleichung:

$$\frac{dX}{dt} = k \tag{3.3}$$

Es zeigt sich, dass der Übergang von einer „normalen“ algebraischen Gleichung (genauer einer Differenzengleichung) hin zu einer Differentialgleichung ohne jeden „Kulturschock“ möglich ist.

3.1.1.1 Analytische und numerische Lösung

Diese Differentialgleichung lässt sich nicht nur über Gleichung 3.1 schrittweise ausrechnen, sondern sie lässt sich auch sehr leicht **analytisch** lösen, indem man die in der ersten Ableitung vorkommenden Variablen dX und dt trennt und anschließend beide Seiten integriert:

$$\int dX = \int k dt \tag{3.4}$$

$$X = kt + c \tag{3.5}$$

Die Integrationskonstante c interpretieren wir als Anfangspopulationsgröße zum Zeitpunkt X_0 und erhalten:

$$X = kt + X_0 \tag{3.6}$$

In der Realität könnte ein solches lineares Wachstum dann auftreten, wenn die Populationsgröße in einem Habitat nur durch Immigration, nicht aber durch Reproduktion, wachsen kann. Wächst die Population jedoch ausschließlich durch Reproduktion, so ist die Anzahl der Nachkommen bekanntlich von der Anzahl der reproduktionsfähigen Individuen also von X selbst abhängig (rückgekoppelt). Der Änderungsparameter k ist also keine Konstante mehr, sondern eine Funktion der Wachstumsrate μ und der Populationsgröße X, d.h. $k = \mu \cdot X$. Wir erhalten somit:

$$\frac{dX}{dt} = \mu X \tag{3.7}$$

$$\int \frac{1}{X} dX = \int \mu \, dt \tag{3.8}$$

Das Grundintegral von $\frac{1}{X}$ ist $\ln X$ (siehe Schultafelwerk). Durch Integration beider Seiten erhält man demnach

$$\ln X = \mu t + c \tag{3.9}$$

und nach Entlogarithmierung und Anwendung der Potenz- und Logarithmengesetze schließlich:

$$X = e^{\mu t + c} \tag{3.10}$$

$$X = e^c \cdot e^{\mu t} \tag{3.11}$$

Wenn wir die Zeit $t = 0$ setzen, wird $X = X_0$ und $e^{\mu t} = 1$, somit ist

$$X_0 = e^c \tag{3.12}$$

die Populationsgröße zum Zeitpunkt $t = 0$ und folglich lautet die integrierte Form der Wachstumsgleichung:

$$X = X_0 \cdot e^{\mu t} \tag{3.13}$$

Viele Differentialgleichungsmodelle lassen sich auf diese Weise oder gegebenenfalls auch auf eine andere Weise **analytisch** lösen. Allerdings, und das ist für ernstzunehmende ökologische Modelle eher die Regel als die Ausnahme, werden die Gleichungssysteme oft so kompliziert, dass sie nur sehr schwer oder überhaupt nicht analytisch lösbar sind. In einem solchen Fall gehen wir den Weg, die Differentialgleichung wieder in eine Differenzengleichung umzuwandeln und sie **numerisch** zu integrieren.

Beim einfachsten Verfahren, der Eulermethode, wird aus der Differentialgleichung 3.7 wieder eine Differenzengleichung, also

$$X_{t+\Delta t} = X_t + \mu X_t \cdot \Delta t \tag{3.14}$$

Hierbei besteht jedoch das Problem, dass sich wegen der Rückkopplung **während** des Zeitschrittes Δt ja bereits auch X wieder ändert. Das Ergebnis der numerischen Lösung unterscheidet sich also mehr oder weniger von der analytischen Lösung, es entsteht ein **Integrationsfehler**. Die einfachste Gegenmaßnahme ist, die Schrittweite soweit zu verkleinern, bis eine weitere Verkleinerung keinen Genauigkeitsgewinn mehr bringt. Der Nachteil einer kleinen Schrittweite ist natürlich eine stark verlängerte Rechenzeit, auch müssen wir bei jedem Modell aufs neue prüfen, wie groß die Schrittweite sein darf.

Zur Lösung dieses Problems wurden eine Reihe von Integrationsverfahren entwickelt (dargestellt z.B. in den Lehrbüchern PRESS *et al.*, 1992; ENGELN-MÜLLGES and REUTTER, 1996)[1]. Die Verfahren haben das Ziel, bei relativ großer Schrittweite trotzdem nur einen kleinen Fehler zu begehen und möglichst auch die Schrittweite automatisch zu ermitteln.

Ein in der Vergangenheit (und von vielen Biologen auch heute noch gern benutztes) Integrationsverfahren ist das klassische Runge-Kutta-Verfahren 4. Ordnung mit fester Schrittweite. Bei diesem Verfahren sinkt der Integrationsfehler bei einer Verkleinerung der Schrittweite um h jeweils um den Faktor h^4 (globale Fehlerordnung), d.h. die Genauigkeit steigt überproportional. Das Verfahren beruht darauf, für jeden Zeitschritt ein gewichtetes Mittel aus mehreren Zwischenschritten zu bilden:

$$F_1 = dt \cdot f(t, X) \tag{3.15}$$
$$F_2 = dt \cdot f(t + dt/2, X + 1/2F_1) \tag{3.16}$$
$$F_3 = dt \cdot f(t + dt/2, X + 1/2F_2) \tag{3.17}$$
$$F_4 = dt \cdot f(t + dt, X + F_3) \tag{3.18}$$
$$\Delta X = 1/6 \cdot (F_1 + 2F_2 + 2F_3 + F_4) \tag{3.19}$$

[1] Die Homepage von „Numerical Recipes in C“ (`http://www.nr.com`) bietet den Volltext des kompletten Buches im PDF-Format.

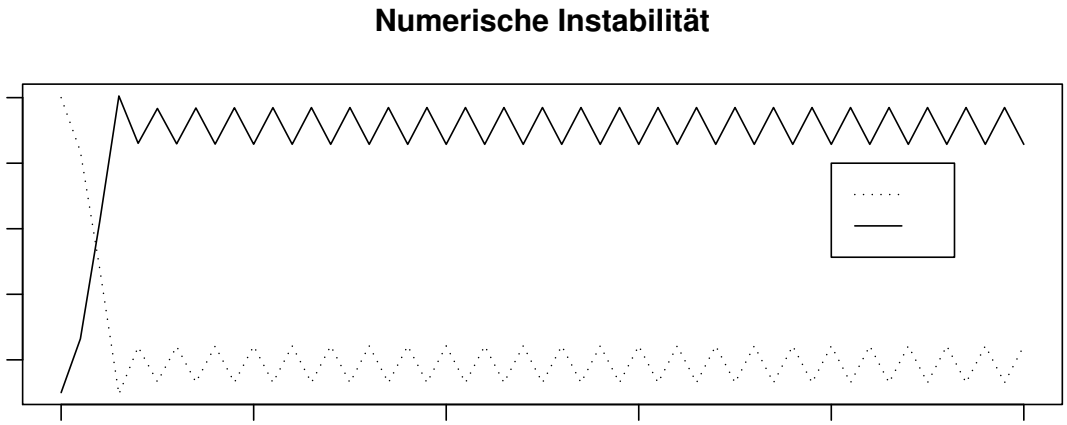

Abbildung 3.1: Auftreten von numerischer Instabilität in einem ressourcenlimitierten Wachstumsmodell bei Runge-Kutta-Integration mit fester Schrittweite. Dargestellt ist die Populationsdichte (X, durchgezogen) und die Ressourcenkonzentration (S, gepunktet).

Hierbei sind die F_i die Funktionswerte der Zwischenschritte, $f(t,X)$ ist der jeweilige Aufruf des kompletten Differentialgleichungssystems und ΔX ist die resultierende Änderung der Zustandsgröße. Da keine Iteration nötig ist und in jedem Schritt nur schon bekannte Werte verwendet werden, nennt man ein solches Verfahren explizit. Das Runge-Kutta-Verfahren 4. Ordnung ist deshalb sehr leicht zu implementieren und bei der nötigen Vorsicht durchaus für ökologische Modelle geeignet, besitzt aber eine Reihe von Nachteilen. So ist z.B. die Schrittweite fest und das Verfahren führt bei zu großer Schrittweite manchmal zu instabilen Lösungen. Bei einem Phosphor-Phytoplankton-Modell kann es z.B. sehr leicht passieren, dass das Phytoplankton in einem Zeitschritt so stark wächst, dass der Phosphor fälschlicherweise einen negativen Wert annimmt. Im nächsten Zeitschritt führt der negative Phosphor dann dazu, dass der eigentliche Wachstumsterm ebenfalls negativ und somit zum Verlustterm wird. In der Grafik ist diese *numerische Instabilität* dann als Zickzackkurve zu erkennen (Abb. 3.1).

Es existieren viele Alternativen zum Euler- oder Runge-Kutta-Verfahren, die teilweise auf ganz spezielle Probleme (z.B. steife Systeme, Algebro-Differentialgleichungen oder Differentialgleichungen mit Zeitverzögerung) zugeschnitten sind.

Verfahren mit automatischer Schrittweitensteuerung bieten neben dem Vorteil einer oft höheren Effizienz (Schrittweite immer so groß wie möglich), den viel wichtigeren Vorteil einer größeren Genauigkeit und Zuverlässigkeit (garantierter maximaler Fehler, da Schrittweite nie größer als zulässig). Einfachere Algorithmen mit Schrittweitensteuerung werden jedoch bei sogenannten „steifen Systemen“ schnell ineffizient. Ein steifes System zeichnet sich dadurch aus, dass es Umsätze mit sehr unterschiedlichen Zeitkonstanten beinhaltet. Wegen der Schwierigkeit, hier eine Schrittweite festzulegen, werden bei steifen Systemen oft sogenannte implizite Algorithmen verwendet, die die numerische Lösung mit Hilfe einer Iterationsschleife annähern. Eine gut verständliche Online-Einführung zu Integrationsverfahren und numerischer Stabilität findet sich z.B. bei PICHÉ (2000).

Für unsere Zwecke ist derzeit wahrscheinlich der „Livermore Solver for Ordinary Differential Equations“ (LSODA) von PETZOLD (1983) und HINDMARSH (1983) eine gute Wahl. Das „A“ am Ende des Kürzels steht für *automatic*, da dieser Algorithmus zwei unterschiedliche Integrationsverfahren benutzt, ein explizites Verfahren (nach Adams) für nicht-steife Systeme und ein implizites BDF-Verfahren (*backward differential formula*) für steife Systeme. Der Algorithmus verwendet zuerst den schnelleren expliziten Integrator und wechselt bei Bedarf (wenn das System steif ist) zum impliziten Verfahren. Das Livermore-Paket enthält weitere Solver für Spezialfälle, z.B. für Algebro-Differentialgleichungen oder Differentialgleichungen mit Zeitverzögerung (*time delayed differential equations*)[2].

3.1.1.2 Implementierung in R

Der bereis seit 2001 im Paket `odesolve` vorhandene Algorithmus `lsoda` ist sehr leistungsfähig, nutzerfreundlich und bei praktisch arbeitenden Modellentwicklern vieler Fachdisziplinen beliebt. Trotzdem ist auch hier Fingerspitzengefühl und gesunde Skepsis besser als blindes Vertrauen in die Automatik. Das neuere Paket `deSolve` (SOETAERT *et al.*, 2010a,b) enthält weitere Lösungsverfahren, u.a. auch für partielle Differentialgleichungen und differentialalgebraische Systeme. Es ist zu `odesolve` kompatibel und hat dieses ablgelöst.

Als Einstiegsbeispiel wollen wir einmal die exponentielle Wachstumskurve numerisch integrieren. Anschließend vergleichen wir das Ergebnis mit der analytischen Lösung nach Gleichung 3.13.

Implementierung des Modells

Zur Simulation eines Differentialgleichungsmodells benötigen wir die folgenden Bausteine:

1. das Modell, also die Differentialgleichungen in computerlesbarer Form,
2. die Parameter (Konstanten) des Modells,
3. Anfangswerte (oder Startwerte) für die Zustandsgrößen, z.B. Startabundanz oder Nährstoffkonzentration am Anfang der Simulation,
4. Randbedingungen, die die ausdrücken, wie sich die Modell-Umwelt ändert (Werden solche Umwelteinflüsse nicht betrachtet, spricht man von einem autonomen System.),
5. eine Simulationszeit (Für welche Zeitpunkte soll das Modell simuliert werden?),
6. ein Lösungsverfahren, z.B. `lsoda`.

Das zu integrierende Modell wird als R-Funktion definiert, z.B. `model()`. Es enthält die Differentialgleichungen aller zum Modell gehörigen Zustandsgrößen (im ersten Beispiel also nur `x[1]`). Als Aufrufparameter werden die Simulationszeit, der Vektor der Zustandsgrößen und der Parametervektor übergeben `(t, x, p)`.

[2] Die Original-Fortran-Quellen sind auf `http://www.netlib.org/` zu finden.

Die Zuweisung am Anfang der Funktion dient dazu, den Parameter `mu` direkt mit Namen (ohne Klammern, Anführungszeichen oder Dollar) ansprechen zu können[3]. Danach folgen eventuelle Hilfsgleichungen und danach die Differentialgleichungen, die im Programmcode fast genau so aussehen, wie in der mathematischen Schreibweise. Am Ende der Funktion werden die Ableitungen aller Differentialgleichungen in eine Liste gepackt und an die aufrufende Routine zurückgegeben.

Parameter, Startwerte und Zeitvektor

Die benannten Vektoren `parms` und `xstart` enthalten die konstanten Parameter bzw. die Startwerte für die Zustandsgrößen zum Zeitpunkt t_0. Der Vektor `times` enthält die Zeitpunkte mit der externen Schrittweite `dt`, für die Simulationsergebnisse ausgegeben werden sollen (die eigentliche Integrationsschrittweite wird bekanntlich intern von `lsoda` automatisch festgelegt).

Simulation

Die eigentliche Simulation erfolgt innerhalb einer einzigen Programmzeile `lsoda`. Hier lassen sich noch weitere Simulationsoptionen einstellen, z.B. die Integrationsgenauigkeit. Die Umwandlung in einen Dataframe dient der bequemeren Schreibweise beim Plotten der Ergebnisse.

```
library(deSolve)

model <- function(t, x, p) {
  mu <- p["mu"]
  dx1.dt <- mu * x[1]
  list(c(dx1.dt))
}

parms  <- c(mu=0.3)
xstart <- c(x1=1)
dt     <- 0.2
times  <- seq(0, 10, dt)

# Solve the differential equation
out <- lsoda(xstart, times, model, parms)
out <- as.data.frame(out)

plot(out$time, out$x1)
```

[3] Eine alternative Methode (ab R-Version 1.4) ermöglicht die Verwendung von `with`. Im Gegensatz dazu wird von der Verwendung von `attach` in diesem Zusammenhang dringend abgeraten, da dies zu Problemen führen kann.

3.1.1.3 Übung

Aufgabe

Vergleichen Sie die numerischen Verfahren LSODA und Eulerregel bei einer Schrittweite von 0.01, 0.1, 1, 2 mit der oben abgeleiteten analytischen Lösung nach Gleichung 3.13.

$$\frac{dX}{dt} = \mu \cdot X \tag{3.20}$$

Lösung

Zur grafischen Darstellung der analytischen Lösung erzeugen wir uns einen Vektor `x` als Funktion des Zeitvektors `times`

```
x0 <- 1
mu <- 0.3
x  <- x0 * exp(mu * times)
points(times, x , col="green", pch=3)
```

Zur Simulation mit `lsoda` kann das oben vorhandene Implementierungsbeispiel abgewandelt und **ergänzt** werden.

Eine Implementierung der Eulerregel zeigt das untenstehende Beispiel. Die `numeric`–Funktion bereitet zunächst einen leeren Datenvektor `x` vor, der die gleiche Länge wie der Vektor `times` hat. In der `for(){...}`–Schleife wird das Wachstum in `n` diskreten Schritten berechnet und in `x` abgespeichert. Da der erste Wert `x[1]` der Startwert ist, beginnt die Simulation mit der Berechnung des 2. Wertes.[4]

```
dt   <- 1
n    <- length(times)
x    <- numeric(n)
x[1] <- x0
for (i in 2:n) {
  x[i] <- x[i-1] + mu * x[i-1] * dt
}
points(times, x, col="red")
```

Zur Untersuchung des Einflusses der Schrittweite modifizieren wir die Variable `dt` und rechnen das Beispiel mehrmals durch. Hierbei muss gegebenenfalls auch `n` angepasst werden.

[4] In R beginnen alle Vektoren mit dem Element 1, es existiert kein Nulltes Element.

3.1.1.4 Weitere Aufgaben

1. Implementieren Sie analytische Lösung und Eulerregel in einem Tabellenkalkulationsprogramm.
2. Implementieren Sie das klassische Runge-Kutta-Verfahren 4. Ordnung in einem Tabellenkalulationsprogramm.
3. Modifizieren Sie die Modellgleichung (z.B. durch Veränderung von Parametern, Startwerten, Vorzeichenwechsel, Einführung von Potenzen, zusätzlichen Termen usw.) und schauen Sie sich das Ergebnis an.

3.1.2 Limitiertes Wachstum

Unlimitiertes Populationswachstum wird immer nur während einer sehr kurzen Zeitspanne beobachtet. Sowohl im Freiland als auch im Labor kommt es früher oder später zur Limitation des Wachstums durch Mangel an Ressourcen, Mangel an Raum, Konkurrenz oder Prädation.

Der allgemein verwendete Ansatz hierfür ist, die Wachstumsrate als eine Funktion des jeweils limitierenden Faktors aufzufassen, also $\mu = f(\text{limitierender Faktor})$. Je nachdem, wodurch diese Funktion f kontrolliert wird, ergeben sich unterschiedliche Ansätze für das limitierte Populationswachstum, z.B. das logistische Wachstum oder das ressourcenlimitierte Wachstum.

3.1.2.1 Logistisches Wachstum

Beim logistischen Modell (VERHULST, 1838, zit. in TSOULARIS, 2001) wird angenommen, dass der limitierende Faktor in der Annäherung an einen Maximalwert, der (*Carrying capacity*, K), besteht und das sich die Wachstumsrate linear von einem anfänglichen Maximalwert μ_{max} bis auf Null verringert. Bei $X = 0$ ist $\mu = \mu_{max}$, bei Annäherung von X gegen K geht der Term $(1 - X/K)$ gegen Null und das Wachstum stoppt. Man bezeichnet in diesem Fall μ_{max} als intrinsische Wachstumsrate (oder Änderungsrate) und $\frac{dX}{dt}$ als momentane Wachstumsrate (*instantaneous growth rate*):

$$\mu = f(X) = \mu_{max} \cdot (1 - X/K) \tag{3.21}$$

Daraus ergibt sich die Differentialgleichung

$$\frac{dX}{dt} = \mu_{max} \cdot X \cdot \frac{K - X}{K} \tag{3.22}$$

und deren analytische Lösung

$$X(t) = \frac{K}{(1 + (\frac{K}{x_0} - 1) \cdot e^{-\mu_{max} \cdot t})} \tag{3.23}$$

In der diskreten Form lautet die logistische Gleichung

$$X_{t+1} = X_t + \mu_{max} \cdot X_t \cdot \frac{K - X_t}{K} \cdot \Delta t \tag{3.24}$$

Die diskrete Form wird dann verwendet, wenn eine Population nicht kontinuierlich wächst (typisch für Mikroorganismen) sondern wenn die Vermehrung in Generationen oder Reproduktionszyklen verläuft (z.B. viele Fischarten).

Zuweilen wird das logistische Wachstum auch als Modell für das somatische Wachstum eingesetzt. Allerdings ist die Anpassung nicht immer zufriedenstellend. Beim somatischen Wachstum treten oft starke Abweichungen von der idealen sigmoiden Form auf, der Wendepunkt liegt nicht bei 50% des Sättigungswertes und oftmals (z.B. bei Fischen) wird überhaupt keine Sättigung erreicht, sondern die Körperlänge oder Körpermasse steigt auch im Alter noch langsam an (WEST *et al.*, 2001).

Als Ausweg wurden zahlreiche Verallgemeinerungen des logistischen Modells entwickelt. Die entsprechende Literatur und auch die verwendete Symbolik ist, wie bereits erwähnt, sehr vielfältig und manchmal auch widersprüchlich oder verwirrend[5].

3.1.2.2 Gompertz-Gleichung

Die Gompertz-Gleichung wurde bereits 1825 von GOMPERTZ zur Beschreibung von Mortalitätsraten und von Investition und Geldrückfluss angewendet. In der Version von TSOULARIS (2001) lautet diese Gleichung:

$$\frac{dN}{dt} = r \cdot N \cdot \ln\left(\frac{K}{N}\right) \tag{3.25}$$

mit der dazugehörigen analytischen Lösung:

$$N = K \cdot e^{\ln\left(\frac{N_0}{K}\right)e^{-rt}} \tag{3.26}$$

Hierbei ist N die Abundanz, K die Kapazitätsgrenze der Abundanz, r die intrinsische Wachstumsrate und N_0 die Startabundanz. Anstelle der Abundanz kann auch die Biomasse X oder das individuelle Körpergewicht W stehen.

TSOULARIS (2001) zeigt, dass die Gompertz-Kurve als Grenzfall der verallgemeinerten logistischen Funktion betrachtet werden kann. Eine andere übliche Interpretation geht von einem Modell mit zwei

[5] So findet man z.B. anstelle von μ_{max} oft auch r_0 oder sogar μ_0 mit der Begründung, dass die maximale Wachstumsrate nur bei einer sehr geringen Populationsdichte nahe Null auftritt. In allen Fällen ist mehr oder weniger dasselbe gemeint und man tut gut daran, sich jeweils den Kontext der Publikation (z.B. Mikrobiologie, Biotechnologie oder theoretische Ökologie) anzusehen.

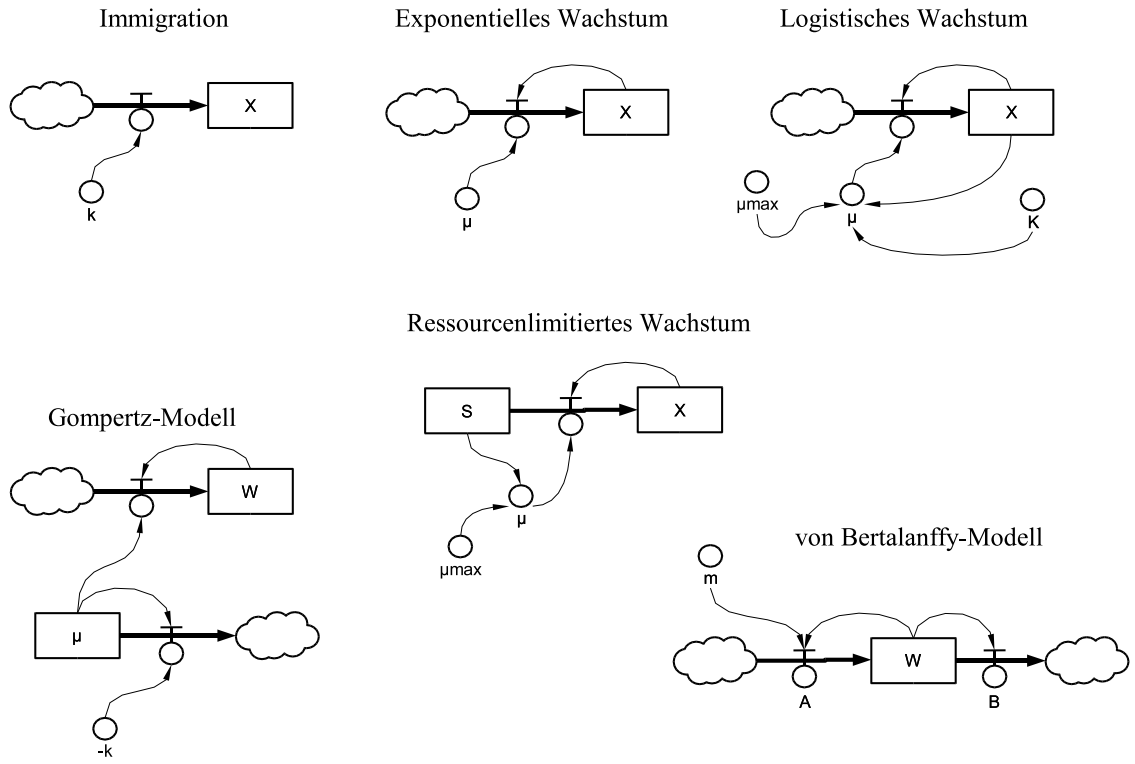

Abbildung 3.2: Schematische Darstellung elementarer Wachstumsmodelle (X=Biomasse oder Abundanz, W=individuelle Biomasse; S=Substrat (Ressource); μ_{max}=maximale (intrinsische) Wachstumsrate; μ= Wachstumsrate allgemein; A, B, k, m=spezielle Parameter des jeweiligen Modells, siehe Text).

Differentialgleichungen aus, bei der nicht nur die Körpermasse W, sondern auch die Wachstumsrate μ zeitabhängig ist, d.h. sich während des Wachstums verringert (siehe z.B. GAMITO, 1998):

$$\frac{d\mu}{dt} = -k\mu \tag{3.27}$$

$$\frac{dW}{dt} = \mu W \tag{3.28}$$

Eine analytische Lösung hierzu ist:

$$W = W_0 \cdot e^{\mu_0/k \cdot (1-e^{-k \cdot t})} \tag{3.29}$$

Die Gompertz-Gleichung wird auch heute noch häufig verwendet, obwohl flexiblere Modelle, z.B. das von Bertalanffy-Modell, das Richards-Modell (für das Wachstum von Pflanzen) oder verallgemeinerte logistische Modelle existieren.

3.1.2.3 Von Bertalanffy-Gleichung

Auch für die von Bertalanffy-Funktion existieren unterschiedliche Schreibweisen. Wir führen an dieser Stelle einmal eine Ableitung „rückwärts“ durch und beginnen mit einer der logistischen Gleichung ähnlichen Schreibweise (z.B. nach TSOULARIS, 2001)

$$\frac{dN}{dt} = rN^{\frac{2}{3}} \left[1 - \left(\frac{N}{K} \right)^{\frac{1}{3}} \right] \tag{3.30}$$

mit der analytischen Lösung:

$$N = K \left[1 + \left[1 - \left(\frac{N_0}{K} \right)^{\frac{1}{3}} \right] e^{-\frac{1}{3} r K^{-\frac{1}{3}} t} \right]^3 \tag{3.31}$$

Löst man die eckige Klammer von Gleichung 3.30 auf, so erhält man

$$\frac{dN}{dt} = rN^{\frac{2}{3}} - r\frac{N}{K^{\frac{1}{3}}} \tag{3.32}$$

Die Abundanz N setzt sich somit aus einem Gewinnterm und einem Verlustterm zusammen. Wendet man die von Bertalanffy-Gleichung auf das somatische Wachstum an und setzt statt Abundanz N die Körpermasse W ein, verallgemeinert die Potenz $2/3$ durch einen allgemeinen Wert m und setzt $A = r$ und $B = \frac{r}{K^{1/3}}$, so erhält man die häufig verwendete und z.B. in GAMITO (1998) angegebene Form

$$\frac{dW}{dt} = AW^m - BW \qquad \text{wobei } m < 1 \text{ ist.} \tag{3.33}$$

Diese Gleichung wird im allgemeinen so interpretiert, dass sich das Wachstum der Körpermasse aus einem anabolischen (Aufbau von Körperzellen) und einem katabolischen Term (Abbau von Körperzellen) zusammensetzt. Der Abbau von Körpergewebe hängt hierbei nur von der Anzahl der Zellen (bzw. deren Biomasse) ab, während der Aufbau der Zellen wegen des notwendigen Versorgungsaufwandes mit einer Potenz $2/3 \leq m < 1$ erfolgt.

Eine weitere Modifikation entsteht durch Einsetzen von $m = n - 1, B = k$ und $A = kW_{max}^n$

$$\frac{dW}{dt} = kW_{max}^n \cdot W^{1-n} - kW \tag{3.34}$$

Eine analytische Lösung hierzu ist (siehe z.B. GAMITO, 1998):

$$W(t) = (W_{max}^n - (W_{max}^n - W_0)e^{-nk(t-t_0)})^{1/n} \tag{3.35}$$

wobei hier allerdings noch ein zusätzlicher Parameter t_0 bestimmt werden muss. Der Vorteil besteht hier darin, dass W_{max} als Sättigung (Maximalwert der Körpermasse) interpretiert werden kann.

Die von Bertalanffy-Funktion wurde speziell zur Beschreibung des individuellen somatischen Wachstums von Fischen entwickelt. So haben z.B. die Parameter $1/3, 2/3$ eine physiologische Begründung. Für eine Anwendung auf das Wachstum von Populationen fehlt die mechanistische Begründung, d.h. entsprechende Anwendungen sind als „rein empirisch“ zu bezeichnen.

3.1.2.4 Verallgemeinerte logistische Modelle

Es existieren mehrere Versuche zur Verallgemeinerung, die die oben genannten Modelle als Spezialfall enthalten, z.B. die verallgemeinerte logistische Gleichung (z.B. TSOULARIS, 2001)

$$\frac{dN}{dt} = rN^{\alpha}\left[1-\left(\frac{N}{K}\right)^{\beta}\right]^{\gamma} \tag{3.36}$$

oder das u.a. von SCHNUTE (1981) verwendete Wachstumsmodell.

Eine andere Art der Verallgemeinerung besteht darin, das Wachstum auf analytische Weise von physiologischen Parametern abzuleiten. Die von WEST *et al.* (2001) vorgestellte Wachstumsgleichung ist der von Bertalanffy-Gleichung formal sehr ähnlich, geht aber in ihrer mechanistischen Begründung darüber hinaus. Sie basiert auf der Erhaltung der metabolischen Energie, der allometrischen Skalierung von metabolischer Rate und den energetischen Kosten bei der Produktion und Erhaltung der Biomasse. Die Autoren geben in ihrer Arbeit eine Reihe konkreter Parameter an und zeigen die breite Anwendbarkeit für niedere und höhere Organismen.

3.1.2.5 Das DEB-Modell

Das DEB (*dynamic energy budget*)-Modell[6] (KOOIJMAN, 1995, 2000, 2001; NISBET *et al.*, 2000) stellt eine verallgemeinerte Theorie von Wachstum und Vermehrung auf, die auf mehrzellige sich reproduzierende Organismen (höhere Pflanzen, poikilotherme und homöotherme Tiere) und auch für sich teilende Mikroorganismen anwendbar ist. Das Grundprinzip besteht in einer Aufteilung der assimilierten Energie in einen strukturbildenden und einen reproduktiven Anteil, die ihrerseits jeweils wieder in einen Erhaltungs- und einen Neubildungsanteil aufgeteilt werden. Die beiden grundlegenden Zustandsgrößen sind die strukturelle Körpermasse und die Reserven des Organismus. Ein fester Anteil der assimilierten Energie wird zunächst für den Erhaltungsstoffwechsel und das somatische Wachstum aufgewendet, der Rest steht für die Reproduktion zur Verfügung. Das Grundmodell ist in der in KOOIJMAN (2001) angegebenen Version in Form von 9 Annahmen axiomartig formuliert, weitere 3 Annahmen spezifizieren den Alterungsprozess. Die Allgemeingültigkeit ist sehr weitgehend. So zeigt KOOIJMAN (2001), dass sich eine Reihe von Modellen (z.B. von Bertalanffy-Wachstumsmodell, Michaelis-Menten-Kinetik, Holling-Gleichungen vom Typ I und Typ II) als Spezialfall aus dem DEB-Modell ergeben.

[6] Die DEB-Homepage: `http://www.bio.vu.nl/thb/deb/`

3.1.2.6 Zweistufiges Populationswachstum mit Verzögerungsphase

In vielen Mikoorganismenkulturen kann beobachtet werden, dass das Wachstum der Organismen erst nach einer mehr oder weniger langen Verzögerung einsetzt. Man teilt das Wachstum dann üblicherweise in die Phasen Lag-Phase, exponentielle Phase und limitierte Phase ein (manchmal auch noch mit weiteren Unterteilungen).

Obwohl eine solche Wachstumskurve bei linearer Darstellung auf den ersten Blick sehr stark der logistischen Wachstumskurve ähnelt (Abb. 3.3), kann dieses Verhalten mit dem einfachen logistischen Modell nicht ausreichend beschrieben werden. Stellt man die Abundanz oder Biomasse halblogarithmisch dar, so wird deutlich, dass das Wachstum in der logarithmischen Darstellung bereits zum Zeitpunkt t_0 linear ansteigt, d.h. dass sich die Kultur von Anfang an in der exponentiellen Phase befindet. Ein anfänglich verzögertes Wachstum (Lagphase) ist hier nicht zu beobachten. Wir benötigen also ein Modell, dass auch in der halblogarithmischen Darstellung einen sigmoiden Verlauf zeigt.

Zur Aufstellung dieses Modells müssen wir uns den zugrundeliegenden Mechanismus näher ansehen. Man geht davon aus, dass in der Lagphase eine Adaptation der Organismen an das Kulturmedium stattfindet. Diese Adaptation kann darin bestehen, dass zunächst Ruhestadien auskeimen müssen, dass für die Nutzung des Kulturmediums oder das Wachstum entsprechende Enzyme bereitgestellt werden oder dass zunächst der Energiestatus von „ausgehungerten“ Zellen auf ein normales Maß gebracht werden muss. Das bedeutet, dass wir im Kulturmedium zwei Typen von Organismen vorfinden, Ruhestadien und aktiv wachsende Organismen, auch wenn diese sich vielleicht äußerlich nicht unterscheiden.

Es resultiert demnach ein Modell mit zwei Zustandsgrößen, den Ruhestadien $X_{resting}$ und den aktiv wachsenden Zellen X_{active}. Von den Ruhestadien wacht pro Zeiteinheit ein bestimmter Anteil k_{wakeup} auf und wechselt vom „inaktiven“ in denn aktiven Zustand. Die jeweils aktiven Zellen wachsen dann entsprechend des logistischen Modells:

$$\frac{dX_{resting}}{dt} = -k_{wakeup} \cdot X_{resting} \tag{3.37}$$

$$\frac{dX_{active}}{dt} = k_{wakeup} \cdot X_{resting} + \mu_{max} \cdot X_{active} \cdot \frac{K - X_{active}}{K} \tag{3.38}$$

Da sich im Mikroskop die aktiven von den inaktiven Zellen meist nicht unterscheiden lassen, ergibt sich die beobachtete Populationsdichte als

$$X_{total} = X_{resting} + X_{active} \tag{3.39}$$

3.1.2.7 Ressourcenlimitiertes Wachstum

Die logistische Gleichung und die davon abgeleiteten Modelle einschließlich des zweistufigen Wachstumsmodells sind wichtige theoretische Modelle und Basis zahlreicher weiterführender Ansätze in der Populationsökologie (siehe z.B. HASTINGS, 1996). Besonders bei der Anwendung auf das Populationswachstum haben alle logistischen Modelle jedoch dieselben Nachteile, die ihre praktische Benutzung stark einschränken:

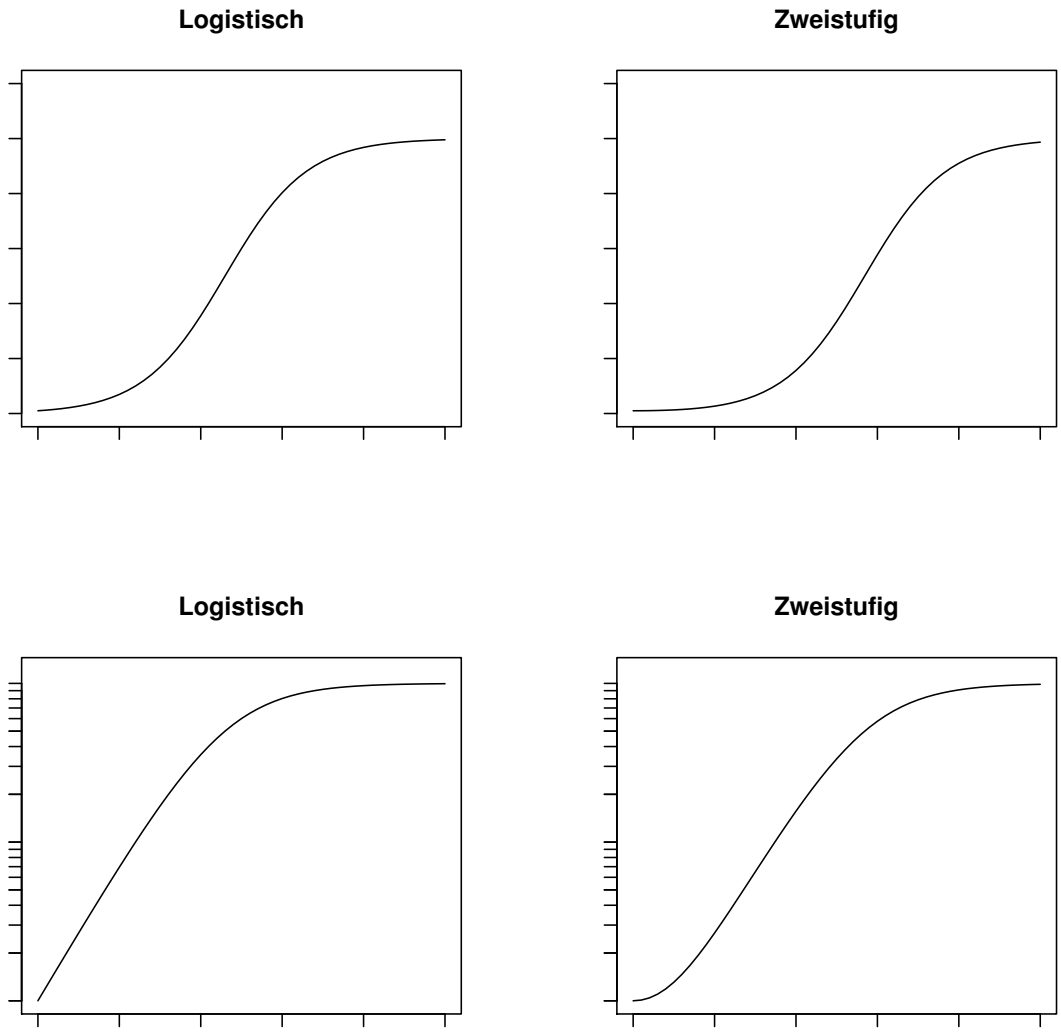

Abbildung 3.3: Logistisches Wachstum und zweistufiges Wachstum in linearer (oben) und logarithmischer Darstellung (unten).

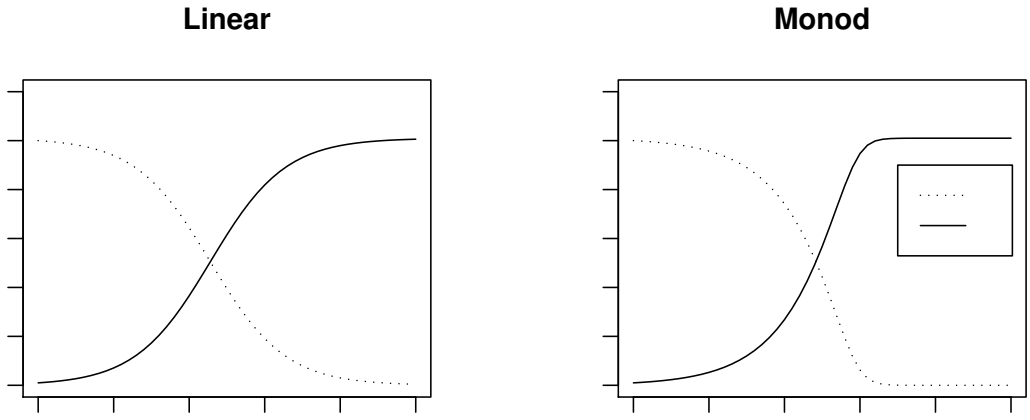

Abbildung 3.4: Ressourcenlimitiertes Wachstum mit linearer Ressourcenabhängigkeit der Wachstumsrate (links) bzw. Monod-Kinetik (rechts). Dargestellt ist die Populationsdichte (X, durchgezogen) und die Ressourcenkonzentration (S, gepunktet).

- die Carrying Capacity K muss im voraus bekannt sein,
- es ist nicht klar, welcher Faktor oder Ressource eigentlich die Carrying Capacity bestimmt,
- Verlustfaktoren (Selbstveratmung, Fraßverluste, Sedimentation, Bildung von Dauerstadien) und Interaktionen werden nicht berücksichtigt.

Der Ausweg aus diesem Problem besteht darin, die limitierenden Faktoren, die Verlustgrößen und die Interaktionen mit anderen Organismen explizit zu beschreiben.

Beim ressourcenlimitierten Wachstum werden die beschränkenden Faktoren durch eigene Zustandsgleichungen beschrieben. Die Wachstumsrate ergibt sich dann als Funktion der Ressource und wird entweder linear (bei theoretischen Modellen) oder über eine aus Wachstumsexperimenten abgeleitete Funktion (z.B. Sättigungsfunktion) abgeleitet.

Im einfachsten Fall (nur eine Ressource, z.B. Phosphor) werden 2 Zustandsgleichungen benötigt (Substrat S, Produzent X):

$$\mu = \mu_{max} \cdot \frac{S}{S_{max}} \tag{3.40}$$

$$\frac{dS}{dt} = -\mu \cdot 1/Y \cdot X \tag{3.41}$$

$$\frac{dX}{dt} = \mu \cdot X \tag{3.42}$$

Hierbei ist Y der Ertragskoeffizient (*Yield-coefficient*, ein Umrechnungsfaktor z.B. von Phosphor in Phytoplanktonbiomasse) und S_{max} dient zur Normierung.

Der Verlauf der Wachstumskurve ähnelt der logistischen Kurve (Abb. 3.4), wobei sich die Carrying Capacity hier aus der verfügbaren Ressource und ihrer Nutzung ergibt. Obwohl in der Gleichung keine explizite Carrying Capacity mehr auftritt, ergibt sich diese implizit aus dem Ressourcenverbrauch.

Das Modell des ressourcenlimitierten Wachstums lässt sich auf unterschiedliche Weise ausbauen. Oftmals wird die Abhängigkeit von der Ressource nichtlinear, in Form einer Monod-Kinetik[7] mit einer Halbsättigungsrate k_S angegeben (Abb. 3.4, rechts). Die Differentialgleichungen selbst bleiben unverändert:

$$\begin{aligned} \mu &= \mu_{max} \cdot \frac{S}{k_S + S} \\ \frac{dS}{dt} &= -\mu \cdot 1/Y \cdot X \\ \frac{dX}{dt} &= \mu \cdot X \end{aligned} \tag{3.43}$$

3.1.2.8 Implementierung in R

Als Beispiel soll die Implementierung der numerischen und der analytischen Lösung der Gompertz-Funktion vorgestellt werden. Die Implementierung baut direkt auf dem in Abschnitt 3.1.1.2 vorgestellten Beispiel auf, besitzt jedoch zwei Differentialgleichungen. Bei der Benennung der Zustandsgrößen ist zu beachten, dass diese entprechend dem in der Parameterzeile des Modells übergebenen Zustandsvektor (in unserem Fall `x`) benannt und in der richtigen Reihenfolge verwendet werden müssen. Am Ende der Funktion sind alle Ableitungen als Liste zu übergeben.

```
library(deSolve)

gompertz <- function(t, x, p) {
    k  <-p["k"]
    W  <- x[1]
    mu <- x[2]
    dx1 <- mu * W
    dx2 <- -k * mu
    list(c(dx1, dx2))
}
```

Anschließend werden die Parameterwerte festgelegt und ein entsprechender Zeitvektor, ein Parametervektor und geeignete Startwerte übergeben:

[7] Der Funktionstyp heißt in der Populationsökologie auch Holling-Typ II oder in der Biochemie Michaelis-Menten-Gleichung.

```
Wmax   <- 10                # Hilfsgröße
k      <- 0.05              # Parameter
W0     <- 1                 # Startwert
mu0    <- k * log(Wmax/W0)  # Startwert

times  <- seq(0, 100, 1)    # Zeitvektor
parms  <- c(k=k)            # Parametervektor
xstart <- c(W=W0, mu=mu0)   # Startvektor
```

Die numerische Integration erfolgt mit `lsoda`:

```
out <- as.data.frame(lsoda(xstart, times, gompertz, parms))
```

und zum Schluss folgt noch die grafische Darstellung der numerischen Lösung (Plot-Symbole) und der analytischen Lösung (durchgezogene Linie):

```
plot(times, out$W, ylim=c(0,max(out$W)))
lines(times, W0*exp(mu0/k * (1- exp(-k*times))),col="red")
```

3.1.2.9 Übung

Problem

Implementieren Sie das logistische Modell, das Zweistufen-Wachstumsmodell und ein ressourcenlimitiertes Wachstumsmodell mit Monod-Funktion.

Lösung

Als erster Schritt müssen die Gleichungssysteme für alle betrachteten Modelle als Funktionen definiert werden. Das kann sowohl in einzelnen Programmen oder auch in einem gemeinsamen Programm erfolgen. Danach folgen gegebenenfalls globale Einstellungen (2), die eigentliche numerische Lösung der Differentialgleichungssysteme mit `lsoda` (3) und die grafische Darstellung der Ergebnisse (4).

```
########################################
# Example: Growth models
########################################

library(deSolve)

#(1)======= define the required models ===
# logistic growth model
logistic <- function(t, x, p) {
  mumax  <- p["mumax"]; K <- p["K"]
  mu     <- mumax * (1-x[1]/K)
```

```
  dx1.dt <- mu * x[1]
  list(c(dx1.dt))
}
# sigmoid two step growth model
twostep  <- function(t, x, p) {
  mumax  <- p["mumax"]; K <- p["K"]; wakeup <- p["wakeup"]
  mu     <- mumax * (1-x[2]/K)
  dx1.dt <- -wakeup * x[1]              #resting stages
  dx2.dt <-  wakeup * x[1] + mu * x[2]  #active cells
  list(c(dx1.dt, dx2.dt))
}
# resource limited growth
resource <- function(t, x, p) {
  mumax  <- p["mumax"]; Y <- p["Y"]; smax <- p["smax"]
  mu     <- mumax * x[1]/smax
  dx1.dt <- -mu * 1/Y * x[2]  # Resource
  dx2.dt <-  mu * x[2]        # active cells
  list(c(dx1.dt, dx2.dt))
}
# resource limited growth with monod function
resmonod <- function(t, x, p) {
  mumax  <- p["mumax"]; Y <- p["Y"]; ks <- p["ks"]
  mu     <- mumax * x[1]/(ks + x[1])
  dx1.dt <- -mu * 1/Y * x[2]  # Resource
  dx2.dt <-  mu * x[2]        # active cells
  list(c(dx1.dt, dx2.dt))
}

#(2)======= define global constants ======
times  <- seq(0, 50, 1)

#(3)======= solve the models =============
#---------- logistic growth --------------
parms  <- c(mumax=0.2, K=10)
xstart <- c(x=0.1)
out.logistic <- as.data.frame(lsoda(xstart, times, logistic, parms))
#---------- two step growth --------------
parms  <- c(mumax=0.2, wakeup=0.1, K=10)
xstart <- c(x.rest=1, x.act=0)
out.twostep <- as.data.frame(lsoda(xstart, times, twostep, parms))
#---------- resource limited growth I-----
parms  <- c(mumax=0.2, smax=10, Y=1)
xstart <- c(s=10, x=0.1)
out.resource <- as.data.frame(lsoda(xstart, times, resource, parms))
#---------- resource limited growth II----
parms  <- c(mumax=0.2, ks=2, Y=1)
xstart <- c(s=10, x=0.1)
out.resmonod <- as.data.frame(lsoda(xstart, times, resmonod, parms))
```

```
#(4)======= plot the models ==============
par(mfrow=c(2,2))# four graphics on one page

plot(times, out.logistic$x, type="l", ylim=c(0,12),
     ylab="X", xlab="time", main="Logistic")

plot(times, out.twostep$x.res+out.twostep$x.act,
     type="l", ylim=c(0,12),
     ylab="X", xlab="time", main="Two step")

plot(times, out.resource$x, type="l", ylim=c(0,12),
     ylab="S, X",  xlab="time", main="Resource I")
lines(times, out.resource$s, col="blue")

plot(times, out.resmonod$x, type="l", ylim=c(0,12),
     ylab="S, X",  xlab="time", main="Resource II")
lines(times, out.resmonod$s, col="blue")
```

3.1.2.10 Weitere Aufgaben

1. Vergleichen Sie die numerische Lösung des logistischen Modells mit der analytischen Lösung.
2. Untersuchen Sie den Einfluss von μ_{max} und k_S beim ressourcenlimitierten Wachstum. Haben diese Parameter einen Einfluss auf die Lage des Gleichgewichtes?
3. Untersuchen Sie weitere Varianten, variieren Sie die Parameter und Startwerte.
4. Implementieren Sie das von Bertalanffy-Modell nach Gleichung 3.33.
5. Stellen Sie ein Modell für ein ressourcenlimitiertes Wachstum **mit** Lagphase auf.

3.1.3 Verlustprozesse

Verlustprozesse lassen sich als Wachstumsprozesse mit negativer Wachstumsrate beschreiben. Wichtig ist hierbei, ob die Verluste von der Populationsdichte unabhängig (z.B. Entnahme einer bestimmten Menge Fisch pro Zeiteinheit) oder von der Populationsdichte abhängig sind (z.B. natürliche Mortalität eines bestimmten *Anteils* der Fische oder Sedimentation eines bestimmten *Anteils* des Phytoplanktons).

Populationsdichteunabhängiger Verlust:

$$\frac{dX}{dt} = -d \tag{3.44}$$

Populationsdichteabhängiger Verlust:

$$\frac{dX}{dt} = -d \cdot X \tag{3.45}$$

Die Lösung der Gleichung 3.45 ist eine fallende exponentielle Funktion. Darüberhinaus kann d auch als Funktion von anderen Zustandsgrößen (z.B. Grazerdichte) oder von äußeren Randbedingungen (z.B. temperaturabhängige Respirationsverluste) dargestellt werden.

3.1.3.1 Das Resultat aus Wachstum und Verlust

Die Populationsänderungsrate r ist ein Resultat aus Wachstum und Verlust. Sie ergibt sich also z.B. als Ergebnis aus Geburtenrate b und Mortalitätsrate d. Für die momentane (*instantaneous*) Populationsänderungsrate heißt das:

$$\frac{dN}{dt} = r \cdot N = b \cdot N - d \cdot N \tag{3.46}$$

und somit

$$r \cdot N = (b - c) \cdot N \tag{3.47}$$

oder

$$r = b - c \tag{3.48}$$

daraus folgt in der integrierten Funktion:

$$N = N_0 \cdot e^{(b-d) \cdot t} = N_0 \cdot e^{r \cdot t} \tag{3.49}$$

Dieses Ergebnis ist trotz der scheinbaren Trivialität eine sehr grundlegende Regel über das Verhältnis von Wachstum und Verlust. Allerdings gilt diese Gleichung in realen Systemen strenggenommen nur für den Gleichgewichtszustand (also für $\frac{dN}{dt} = 0$), da sowohl b als auch d ihrerseits wieder von N (sie sind also rückgekoppelt) oder von anderen Größen auf unterschiedliche Weise abhängig sein können.

3.1.3.2 Durchflusssysteme

Bei den bisher betrachteten Wachstumsgleichungen wurde von einem geschlossenen System (Batchkultur) ausgegangen, bei dem die Ressourcen am Beginn einmalig zugegeben wurden und weder die Resource noch die Organismen einem Import oder Export unterliegen. Der Grundtyp eines offenen Systems kann sehr anschaulich am Beispiel einer kontinuierlichen Durchflusskultur (Chemostat, Abb. 3.5) gezeigt werden. Hierbei laufen die Wachstumsprozesse in einem konstanten Volumen V (z.B. in

m^3) ab. Pro Zeiteinheit fließt eine konstante Menge Q (in $m^3\,d^{-1}$) an Nährlösung zu und eine entsprechende Menge Kulturflüssigkeit ab. Der Quotient V/Q (in d) ist die mittlere theoretische Verweilzeit $\bar{t}$, und dessen Kehrwert $D = 1/\bar{t} = Q/V$ ist die Verdünnungsrate (*dilution rate D*).

Als Zustandsgrößen betrachten wir im einfachsten Fall ein Substrat S (z.B. Phosphor) und eine Biomasse X (z.B. *Scenedesmus acuminatus*). Die Substratkonzentration im Zulauf ist S_0. Die durch die Differentialgleichung der beiden Zustandsgrößen ausgedrückte zeitliche Änderung ergibt sich aus der Massenbilanz, d.h. der Summe der jeweiligen Gewinn- und Verlustgrößen.

Für die Biomasse X bedeutet das analog zu Gleichung 3.46 eine Bilanz aus Wachstum und Verlust:

$$\frac{dX}{dt} = \text{Wachstum} - \text{Export} \tag{3.50}$$

und für das Substrat

$$\frac{dS}{dt} = \text{Import} - \text{Export} - \text{Aufnahme durch Organismen} \tag{3.51}$$

Wenn wir eine exponentielle Wachstumsfunktion einsetzen, ergibt sich somit

$$\frac{dX}{dt} = \mu \cdot X - D \cdot X \tag{3.52}$$

$$\frac{dS}{dt} = D \cdot S_0 - D \cdot S - \mu \cdot \frac{1}{Y} \cdot X \tag{3.53}$$

Der Parameter Y (*Yield*-Koeffizient oder Ertragskoeffizient) ist wieder ein Umrechnungsfaktor von Substrat in Biomasse, der ausdrückt, wieviel Einheiten Biomasse pro Einheit Substrat gebildet werden können.

Eine kleine Stolperstelle ergibt sich daraus, dass wir definieren müssen, ob Substrat und Biomasse als **Masse** (z.B. Gramm der insgesamt im Chemostaten enthaltenen Organismen) oder als **Konzentration** (z.B. in $mg\,l^{-1}$) betrachtet werden sollen. Eine Konzentration kann bekanntlich nicht exportiert werden, beim Ausgießen der Hälfte eines Becherglases mit $10\,\mu g\,l^{-1}$ Phosphor bleibt die Konzentration gleich. Dagegen halbiert sich die Konzentration im Becherglas, wenn wir die gleiche Menge *Aqua dest.* hinzugeben. Die Konzentrationsänderung wird demnach durch die Differenz aus zugegebener und im Gefäß enthaltener Konzentration und aus dem Verhältnis der Zuflussmenge zum Kulturvolumen bestimmt.

$$\frac{dS}{dt} = Q/V \cdot (S_0 - S) \tag{3.54}$$

Beim Einsetzen von $D = Q/V$ und Auflösung der Klammer ergibt sich eine zur obigen Massenbilanz identische Gleichung. Bei über die Zeit schwankendem Volumen und damit unterschiedlichem Q_{in} (Zuflussmenge pro Zeiteinheit) und Q_{out} (Abflussmenge pro Zeiteinheit) muss bei der Konzentrationsbetrachtung die Variabilität von Q_{in} und V berücksichtigt werden. Dagegen geht Q_{out} nur indirekt über die Variation des Volumens V ein.

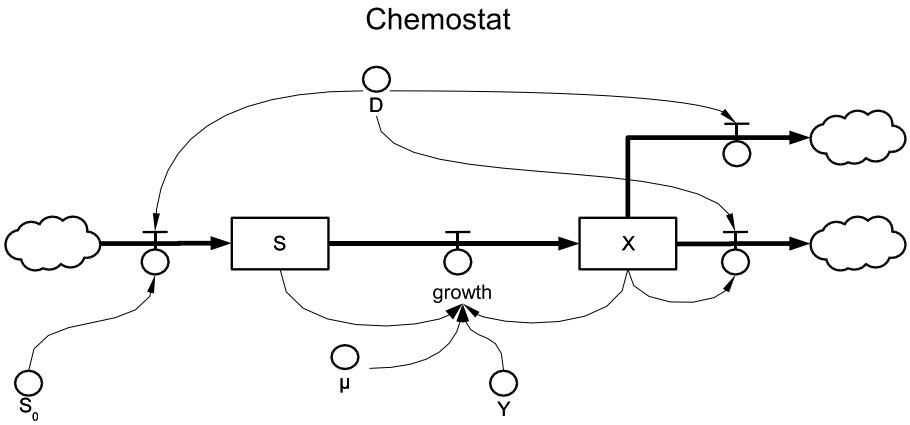

Abbildung 3.5: Schematische Darstellung eines Durchflusssystems (Chemostat) mit einer Ressource S und einer Mikroorganismenkultur X. Die zentrale Steuergröße ist die Durchflussrate D. Weitere Konstanten sind S_0 die Ressourcenkonzentration im Vorratsgefäß, μ die Wachstumsrate der Mikroorganismen und Y der Ertragskoeffizient.

3.1.3.3 Einstellung des Gleichgewichtszustandes

Am Chemostaten lässt sich eine Eigenschaft dynamischer Systeme sehr anschaulich demonstrieren, die Einstellung eines Gleichgewichtszustandes (Fließgleichgewicht, *steady state*). Dieser Zustand tritt auf, wenn sich Wachstum und Verlust die Waage halten. In der Sprache der Differentialgleichungen bedeutet das, dass die Änderungsraten aller Gleichungen des Systems Null sind, also:

$$0 = \mu \cdot X - D \cdot X \tag{3.55}$$

$$0 = D \cdot S_0 - D \cdot S - \mu \cdot \frac{1}{Y} \cdot X \tag{3.56}$$

Daraus ergibt sich ein lineares Gleichungssystem, das auf einfache Weise gelöst werden kann. Zunächst stellen wir fest, dass die erste Gleichung immer dann Null wird, wenn $\mu = D$ gilt. Das bedeutet, im Gleichgewicht kann die Wachstumsrate direkt aus der eingestellten Durchflussrate abgelesen werden.

Setzen wir nun in der zweiten Gleichung ebenfalls $\mu = D$ und stellen nach Y um, so kann aus den Größen X und S im Ablauf der Ertragskoeffizient ermittelt werden:

$$Y = \frac{X}{S_0 - S} \tag{3.57}$$

Bei einer Algenkultur und Phosphor als limitierendem Nährstoff wäre dann z.B. $1/Y$ die zellinterne Phosphorkonzentration (z.B. in $\mu g P mg^{-1}$ Frischmasse).

3.1.3.4 Implementierung

Als Implementierungsbeipiel wandeln wir das Gleichungssystem des Chemostaten dahingehend ab, dass wir die Nährstoffabhängigkeit (z.B. vom Phosphor) des Wachstums einer Phytoplanktonkultur mit Hilfe einer Monod-Funktion beschreiben, d.h. zusätzlich zu den eigentlichen Differentialgleichungen nehmen wir die Monod-Kinetik als Hilfsgleichung in das Modell auf.

```
library(deSolve)

chemostat <- function(t, x, p) {
  vm  <- p["vm"] # max growth rate
  km  <- p["km"] # half saturation constant
  D   <- p["D"]  # dilution rate
  S0  <- p["S0"] # substrate in inflow
  Y   <- p["Y"]  # yield coefficient for substrate
  X   <- x[1]    # phytoplankton cells
  S   <- x[2]    # substrate (phosphorus)

  mu  <- vm * S/(km + S)              # Monod
  dx1 <- mu * X - D * X               # algae
  dx2 <-  D *(S0 - S) - 1/Y * mu * X  # substrate
  list(c(dx1, dx2))
}
```

Bei der Festlegung der Parameter und Startwerte beachten wir streng die Maßeinheiten:

```
vm <-1.1           # 1/d
km <-5             # mug P/l
Y <- 0.800         # mg FM /mug P
D <- 0.7           # 1/d

X0 <- 0.1          # mg/l
S0 <- 20           # mug P/l
```

anschließend simulieren wir das System und stellen die Ergebnisse grafisch dar:

```
times  <- seq(0, 100, length=101)
parms  <- c(vm=vm, km=km, D=D, Y=Y, S0=S0)
xstart <- c(X=X0, S=S0)

out <- as.data.frame(lsoda(xstart, times, chemostat, parms))

par(mfrow=c(2,1))
plot(out$time, out$X, col="green", type="l",
```

```
  xlab="time (d)", ylab="algae (mg/l)", ylim=c(0,max(out$X)))
plot(out$time, out$S, col="blue",  type="l",
  xlab="time (d)", ylab="substrate (mug/l)", ylim=c(0,max(out$S)))
```

3.1.3.5 Übung

Problem

Zur Bestimmung der Wachstumsrate einer Kultur existieren am Chemostaten zwei Stellgrößen, die Substratkonzentration S_0 und die Durchflussrate D. Wir wollen dies am Beispiel untersuchen und Kennlinien für die Zustandsgrößen bei veränderlichem D aufnehmen. Hierbei wird D von Null bis auf einen Wert etwas über μ_{max} variiert. Anschließend werden die Konzentrationen von S und X, die Ausbeute $D \cdot X$ und die Wachstumsrate in Abhängigkeit von der sich im System einstellenden Substratkonzentration (also $\mu = f(S)$) dargestellt.

Lösung

Für eine „experiemtelle“ Lösung des Problems können wir mehrere Simulationen (z.B. $n = 13$) mit einem veränderlichen $D = \{0, 0.1, \ldots 1.2\}$ ausführen. Wir können hierzu das obige Programm direkt nutzen und uns jeweils die **Endergebnisse** der Simulation nach Einstellung des Gleichgewichtes notieren und mit einer Tabellenkalkulation auswerten. Wir können diesen Vorgang jedoch auch dem Computer überlassen und den Vorgang automatisieren.

Die automatisierte Variante beginnt damit, dass wir uns eine Datenstruktur (Tabelle) mit der erforderlichen Anzahl freier Stellen definieren. So hat der Dataframe `stable` 3 Spalten mit jeweils n Zeilen.

```
n<-13
stable <- data.frame(D=seq(0,1.2, length=n),
                     X=rep(0,n),
                     S=rep(0,n))
```

Wenn wir uns den Inhalt des Dataframe mit

```
> fix(stable)
```

ansehen, dann enthält die erste Spalte die vorgegebenen Durchflussraten, die zweite und dritte Spalte zunächst nur Nullen.

Für die Simulation lassen wir den Chemostaten bis zur Einstellung des Gleichgewichtes laufen, z.B. 100 Tage. Im Gegensatz zu den bisherigen Simulationen interessiert uns der Verlauf (die Zeitreihe) nicht und wir vertrauen bei der Schrittweite auf die interne Schrittweitenautomatik von `lsoda`. Somit haben wir extern nur zwei Zeitpunkte: 0 und 100. Die Variable l enthält die Anzahl der Zeitschritte (in diesem Fall also 2).

Mit Hilfe einer `for()`-Schleife rechnen wir nun n Simulationen und merken uns das Endergebnis immer in der i-ten Zeile des vorbereiteten Dataframes. Anschließend kann die Kennlinie grafisch dargestellt werden.

```
times <- c(0,100)
l <- length(times) # last element

for (i in 1:n) {
    parms["D"] <- stable$D[i]
    out <- as.data.frame(lsoda(xstart, times, chemostat, parms))
    stable$X[i] <- out$X[l]
    stable$S[i] <- out$S[l]
}
par(mfrow=c(2,2))
plot(stable$D, stable$X)
plot(stable$D, stable$S)
plot(stable$D, stable$X * stable$D)
plot(stable$S, stable$D)
```

3.1.3.6 Weitere Aufgaben

- Diskutieren Sie den Unterschied zwischen Zeitreihen und Endzustandsdiagramm und variieren Sie Parameter und Startwerte.
- Finden Sie einen Algorithmus zur automatischen Erkennung des Gleichgewichtszustandes.
- Wovon hängt es ab, wie schnell sich ein Gleichgewicht einstellt?
- Versuchen Sie, eine analytische Lösung für den Gleichgewichtszustand zu finden.
- Welche Erkenntnisse lassen sich aus dem Endzustandsdiagramm ableiten? Bei welcher Durchflussrate ist die Ausbeute am höchsten? Welchen Zusammenhang zeigt die Darstellung $D = f(S)$?
- Implementieren Sie eine stochastische Komponente, z.B. eine variable Durchflussrate (Normalverteilung) als Resultat einer unzuverlässigen Pumpe.

3.2 Interaktionen zwischen Populationen

Mit Hilfe der bisher kennengelernten Mittel lassen sich nicht nur Laborversuche nachstellen, sondern es lassen sich auf der Basis von Wachstum und Verlust eine Vielzahl von theoretischen Modellen entwickeln, um grundlegende Interaktionsmuster von Populationen verstehen zu können. Als Beispiel aus dieser Modellklasse wollen wir uns ein Konkurrenzmodell und ein Räuber-Beute-Modell etwas näher ansehen und mit Hilfe numerischer Simulationen interpretieren. Darüberhinaus ist es bei vielen derartigen Modellen möglich, auch **analytische** Lösungen zu finden (siehe z.B. WILSON, 2000).

3.2.1 Konkurrenz um eine gemeinsame Ressource

Zur Untersuchung der Konkurrenz zweier Populationen X_1 und X_2 um eine gemeinsame Ressource S, denken wir uns nun einen Batchversuch. Hierbei soll die Ressource S über einen konstanten Import S_{in} nachgeliefert werden (Abb. 3.6). Die beiden Populationen X_1 und X_2 weisen ein ressourcenabhängiges Wachstum mit der Wachstumsrate μ_i (der Einfachheit halber linear, eine Monod-Funktion ist aber auch möglich) und eine konstante Mortalitätsrate d_i auf.

$$\frac{dS}{dt} = S_{in} - \mu_1 \cdot X_1 - \mu_2 \cdot X_2 \tag{3.58}$$

$$\frac{dX_1}{dt} = \mu_1 \cdot S \cdot X_1 - d_1 \cdot X_1 \tag{3.59}$$

$$\frac{dX_2}{dt} = \mu_2 \cdot S \cdot X_2 - d_2 \cdot X_2 \tag{3.60}$$

Hierbei konkuriert der Konkurrent mit der höheren Wachstumsrate bzw. niedrigeren Sterberate den anderen Konkurrenten aus. Es können gegebenenfalls auch Einschwingvorgänge auftreten.

Derartige Modelle sind in der Literatur umfassend beschrieben (LAMPERT and SOMMER, 1993; BOSSEL, 1994; HASTINGS, 1996; GURNEY and NISBET, 1998) und befassen sich unter anderem mit dem Begriff der ökologischen Nische, der Ressourcenkonkurrenz (für ein aktuelles Beispiel siehe z.B. HUISMAN and WEISSING, 2002) mit der Nutzung mehrerer essentieller Ressourcen oder der unerwartet hohen Diversität, obwohl scheinbar nur wenige Ressourcen limitieren. vgl. Hutchinson's Paradox des Planktons (HUTCHINSON, 1961) oder Connell's *intermediate disturbance hypothesis* (CONNELL, 1978).

3.2.2 Räuber-Beute-Modelle

Das Lotka-Volterra-Modell ist wahrscheinlich das bekannteste Schulbeispiel für die Entstehung intern erzeugter Zyklen in gekoppelten nichtlinearen dynamischen Systemen. Ein einfaches Räuber-Beute-Modell kann im kontinuierlichen Fall durch die beiden gekoppelten Differentialgleichungen

$$\frac{dP}{dt} = a \cdot P - \boxed{b \cdot K} \cdot P \tag{3.61}$$

$$\frac{dK}{dt} = \boxed{c \cdot P} \cdot K - d \cdot K \tag{3.62}$$

dargestellt werden. Hierbei ist P (Produzent) die Abundanz der Beutepopulation und K (Konsument) die Abundanz der Räuberpopulation. Die Beutepopulation wächst zunächst unbeschränkt mit der Wachstumsrate a, die Räuberpopulation hat die Mortalitätsrate d. Das wesentliche am Lotka-Volterra-Modell ist, dass das Wachstum der Räuberpopulation mit der Mortalität der Beutepopulation gekoppelt ist.

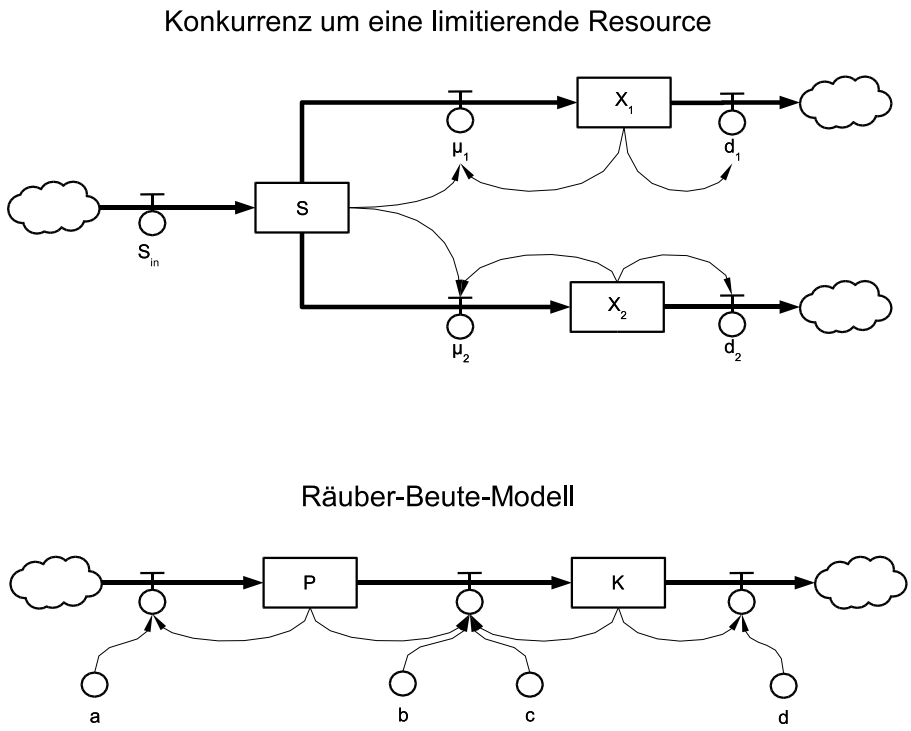

Abbildung 3.6: Schematische Darstellung eines Konkurrenzmodells und eines Räuber-Beute-Modells.

Die eingerahmten Gleichungsteile veranschaulichen diese *Ratenkopplung* der beiden Zustandsgleichungen. Da die Kopplung der Gleichungen wechselseitig ist, liegt definitionsgemäß eine Rückkopplung vor. Das Gleichungssystem ist außerdem nichtlinear, was man vereinfacht gesagt daran erkennt, dass die beiden Zustandsgrößen multiplikativ miteinander verknüpft sind.

Zur Darstellung von Zyklen und Gleichgewichtspunkten verwendet man neben den normalen Zeitreihendiagrammen häufig eine weitere Darstellungsart, das Zustandsdiagramm (Phasendiagramm, Ortskurve), bei dem jeweils zwei[8] Zustandsgrößen gegeneinander aufgetragen werden (Abb. 3.7).

Das Lotka-Volterra-Modell und eine Reihe davon abgeleiterer Modelle besitzen eine fundamentale Bedeutung für die theoretische Ökologie, z.B. zur Untersuchung

- der Konkurrenz mehrerer Populationen um eine oder mehrere Ressourcen,
- der Konkurrenz um eine variable Ressource bei nichtproportionaler Ressourcennutzung (z.B. Monod-Kinetik),
- des Einflusses zeitvarianter Parameter, z.B. $\mu = f(T, I, P, t \ldots)$,
- des Einfluss von zufälligen Effekten (z.B. durch zufällige zeitvariable Parameter),
- von Zyklen, globaler und lokaler Stabilität und der Entstehung chaotischer Phänomene.

[8] oder manchmal auch drei in perspektivischer Darstellung

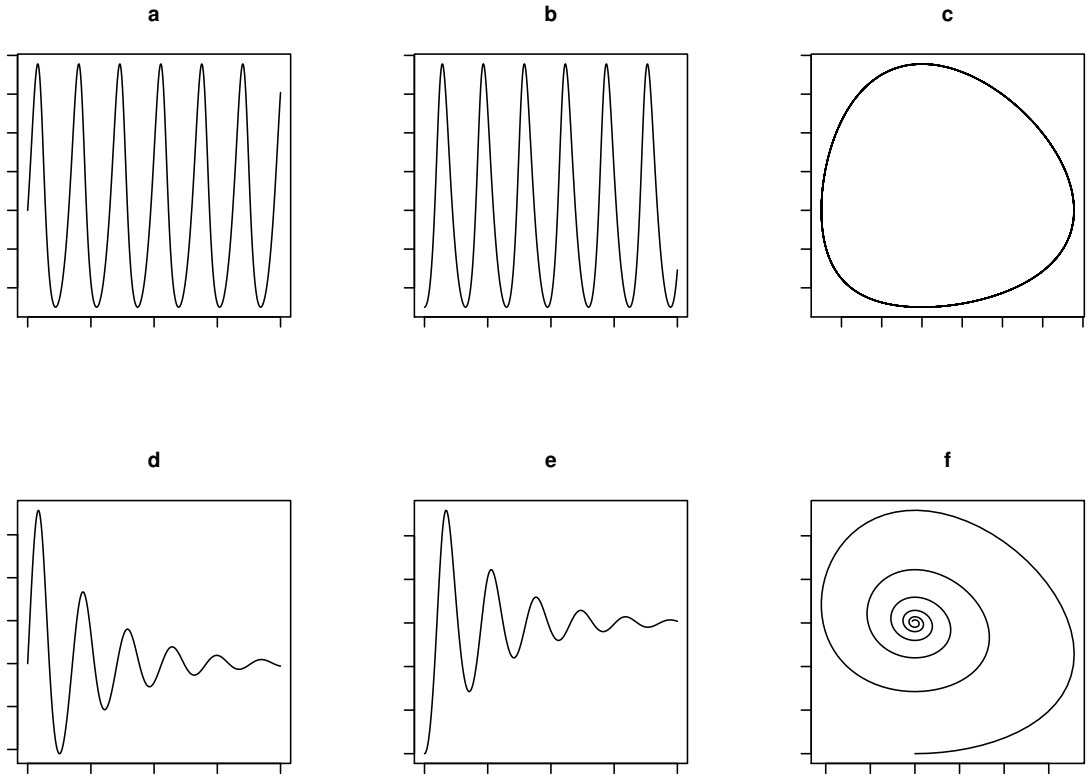

Abbildung 3.7: Lotka-Volterra-Modell: Zeitreihen (a, b) und Zustandsdiagramm (c) des ungedämpften Modells sowie Zeitreihen (d, e) und Zustandsdiagramm (f) eines gedämpften Modells. Beim gedämpften Modell enthält das Wachstum der Beutepopulation z.B. einen logistischen Term, ist also selbstlimitiert.

Näheres zu diesem Thema findet man z.B. in den bereits genannten Lehrbüchern von HASTINGS (1996), GURNEY and NISBET (1998), ROUGHGARDEN (1998) oder WILSON (2000) sowie der dort zitierten Primärliteratur. Einen Überblick über die Stabilitätsproblematik findet sich bei GRIMM and WISSEL (1997)[9], der aktuelle Stand der Diversitäts-Stabilitäts-Kontroverse wird in Band 405 von Nature diskutiert (MCCANN, 2000; PURVIS and HECTOR, 2000; TILMAN, 2000, u.a.).

Grundsätzlich helfen Lotka-Volterra-Modelle gut beim Verständnis gekoppelter nichtlinearer Systeme. Für die Beschreibung realer Räuber-Beute-Beziehungen reichen sie jedoch nicht aus. Einerseits sind dazu zusätzliche Gleichungen erforderlich und andererseits existieren auch völlig andere Erklärungen für das periodische Wechselspiel von Räuber- und Beutepopulationen, z.B. interne Fertilitätszyklen der Beute.

[9] mit online-Addendum auf `http://pinus.oesa.ufz.de/volker/`

3.2.3 Übung

Das folgende Gleichungssystem dient zur Veranschaulichung verschiedener Teilmodelle (Lotka-Volterra-Modell, ressourcenabhängiges Wachstum und Kombination aus beiden). Die Teilmodelle erhält man, wenn man einzelne Parameter oder Zustandsgrößen so auf 0 oder 1 setzt, dass ganze Terme oder Gleichungen wegfallen. Der Substratimport S_{in} ist eine zeitabhängige Größe (eine Randbedingung oder Eingangsvariable).

$$\text{Ressource (Substrat):} \quad \frac{dS}{dt} = S_{in} - b \cdot S \cdot P + g \cdot K \tag{3.63}$$

$$\text{Produzent (Beute):} \quad \frac{dP}{dt} = c \cdot S \cdot P - d \cdot K \cdot P \tag{3.64}$$

$$\text{Konsument (Räuber):} \quad \frac{dK}{dt} = e \cdot P \cdot K - f \cdot K \tag{3.65}$$

Die wechselseitigen Abhängigkeiten der Gleichungen sind hier mittels einfacher multiplikativer Kopplung realisiert, die jedoch auch durch eine Monod-Kinetik ersetzt werden können (z.B. HSU *et al.*, 2001). Generell weist das Modell natürlich erhebliche Vereinfachungen auf, eine Verallgemeinerung von Verhaltensweisen auf reale Ökosysteme ist deshalb nicht ohne weiteres möglich. Versuchen Sie jedoch in den Übungsfragen trotzdem, eine ökologische Analogie zu finden.

3.2.4 Implementierung in R

Die Implementierung erfolgt wieder nach dem Beispiel von Abschnitt 3.1.1.2. Für den zeitabhängigen Import wird eine zusätzliche zeitabhängige Größe (der Vektor `import`) eingeführt, aus dem mit Hilfe der `approx`-Funktion ein der jeweiligen Simulationszeit `t` entsprechender Wert für `s.in` durch lineare Interpolation abgelesen wird. Eine weitere Besonderheit ist die `with`-Konstruktion, die den direkten Zugriff auf die Namen innerhalb von `parms` ermöglicht (ab R-Version 1.4).

Der Parameter `g` ermöglicht die Simulation eines Substratrecyclings. Außerdem ermöglichen es die auskommentierten Zeilen (`import !!!`) eine zusätzliche impulsförmige Zugabe des Substrates (Belastungsimpuls), wobei Höhe und Zeitpunkt (oder Zeitraum) verändert werden können. Bitte lesen Sie vor der Freischaltung von `import` undbedingt den Abschnitt 3.2.7.

```
library(deSolve)
#----- The differential equations -----
lvmodel <- function(t, x, parms) {
  with(as.list(c(x, parms)),{
    # s.in <- approx(times, import, t)$y # import !!!
    ds <- s.in   - b*s*p + g*k
    dp <- c*s*p  - d*k*p
    dk <- e*p*k  - f*k
    res<-c(ds, dp, dk)
```

```
    list(res)
  })
}
# vectors of parameters, timesteps and initial values
times  <- seq(0, 200, length=201)
import <- rep(0,length(times))
#import[100] <- 0.5 # import !!!

parms  <- c(s.in=0,
            b=0.0,
            c=0.1,
            d=0.1,
            e=0.1,
            f=0.1,
            g=0
)
xstart <- c(s=1,   # substrate
            p=1,   # producer
            k=0.5  # consumer
)
# numerical solution, default solver of ode is "lsoda"
out <- ode(xstart,times, lvmodel, parms)

plot(out)
plot(out[,3], out[,4], type="l", xlab="p", ylab="k")
```

3.2.5 Weitere Aufgaben

1. gegeben: $b = 0, g = 0, c = d = e = f = 0.1, S_0 = 1, P_0 = 1, K_0 = 0.5, S_in = 0$ (Standardszenario).
   a) Schreiben Sie die Gleichungen des Modells so auf, daß alle „unnötigen“ Terme wegfallen.
   b) Verändern Sie den Startwert für den Konsumenten (K_0) auf 2 bzw. auf 1. Was passiert?
   c) Setzen Sie nochmals $K_0 = 2$. Versuchen Sie ein c (Wachstumsrate des Produzenten) zu finden, bei dem wieder ein Gleichgewicht auftritt.
2. Stellen Sie das Standardszenario wieder her! Wie müssen die Raten c, d, e, f verändert werden (Randbedingung $c = d = e = f$), damit sich die Schwingungsfrequenz:
   a) erhöht: $c = d = e = f =$.....?
   b) erniedrigt: $c = d = e = f =$.....?
   c) Warum wirken die Raten in die jeweils beobachtete Richtung?

3. Setzen Sie wieder die Standardparameterwerte ein.

   a) Verdoppeln Sie die Wachstumsrate c auf 0.2. Wie ist das Ergebnis zu erklären?

   b) Verdoppeln Sie die Verlustrate von K auf 0.2. Wie ist das Ergebnis zu erklären?

   c) Welche Folgerungen lassen sich zu Umsatz und *standing crop* in realen Systemen treffen?

   d) Verdoppeln und halbieren Sie den Umsatz zwischen P und K. Verwenden Sie dazu die Parameter: $d = e = 0.05 \ldots 0.2$.

3.2.6 Ressourcenabhängiges Wachstum

Stellen Sie zunächst wieder die Standardparameterwerte ein.

1. Setzen Sie den Parameter für Ressourcenverbrauch $b = 0.01$

   a) Schreiben Sie die nun gültigen Zustandsgleichungen auf!

   b) Was passiert hinsichtlich:

      - Amplitude?
      - Schwingungsfrequenz?

   c) Warum wirken die Parameter in diese Richtung?

   d) Erhöhen Sie die Verbrauchsrate b und protokollieren Sie die Änderungen.

   e) Warum kommt es nicht zum Aussterben des **Produzenten**? Wie müßte das Gleichungssystem aussehen, damit ein realistischeres Verhalten erzeugt wird?

3.2.7 Regenerierbare Ressource

1. Setzen Sie wieder die Standardparameterwerte ein und setzen Sie $b = 0.01$.

   a) Setzen Sie nun auch die Importrate des Substrates auf einen konstanten Wert $S_{in} = 0.01$, z.B. indem Sie den Vektor `import` auf einen konstanten Wert setzen. Was passiert?

   b) Setzen Sie nun den Import bzw. S_{in} auf 0.02! Gibt es eine Analogie in natürlichen Systemen?

2. Belastungsimpuls: (Standardszenario, $S_{in} = 0.1, b = 0.1$). Simulieren Sie einen Belastungsstoß zum Zeitpunkt $t = 100$ mit der Stärke $Import(100) = 0.5$ und beurteilen Sie das Verhalten von Produzent und Konsument.

3. Simulieren Sie folgendes Szenario: $S_{in} = b = 0.01, c = 0.1, d = e = f = 0.2$ Wie wirkt sich ein Belastungsstoß auf ein System im Gleichgewicht aus?

4. Gibt es auch eine Möglichkeit, ein schwingendes System durch einen externen Impuls in einen Zustand nahe dem Gleichgewichtszustand zu bringen? Experimentieren Sie mit der Höhe und dem Zeitpunkt des Impulses.

5. Von welchen Faktoren hängt die Wirksamkeit bzw. Wirkungsrichtung eines Störimpulses ab?

3.3 Aufbau eines einfachen Pelagialmodells

Das Modell, wir wollen es Salmo-Light (siehe Abb. 3.8) nennen, basiert zum einen Teil auf dem Modell AQUAMOD I (STRAŠKRABA and GNAUCK, 1983; GNAUCK *et al.*, 1990). Eine zweite Wurzel hat das „Spielzeugmodell“ in ausgewählten Gleichungen und Parametern des Modells SALMO (BENNDORF and RECKNAGEL, 1982), d.h. es wurden einige empirische Gleichungen durch etwas mechanistischere Ansätze ersetzt. Da hierbei die ursprünglich relativ stimmige „Kalibrierung“ verlassen wird, sinkt die Präzision des Modells zwar noch weiter, die Verständlichkeit steigt jedoch hoffentlich. Eine dritte Modifikation betrifft die Möglichkeiten zur externen Angabe von Randbedingungen. Das Modell dient zur Veranschaulichung der prinzipiellen Funktionsweise eines ökologischen Seenmodells, ist aber wegen der starken Simplifizierung nicht für praktische Vorhersagen verwendbar. Für praktische Anwendungen sollten realitätsnahere Ansätze (siehe Abschnitt 3.3.5) verwendet werden und auch von AQUAMOD existieren weiterentwickelte Versionen, z.B. mit zwei Schichten oder mit einer zusätzlichen Sedimentkomponente (STRAŠKRABA and GNAUCK, 1983).

Neben den in Salmo-Light verwendeten Submodellen werden im folgenden auch einige ausgewählte weiterführende Ansätze kurz vorgestellt.

3.3.1 Modellkonzeption

Wir beginnen mit der Definition des Modellzwecks und der Modellkonzeption:

> Das zu erstellende Modell soll grundlegende Prinzipien eines einschichtigen Pelagialmodells (Epilimnion) möglichst einfach widerspiegeln und möglichst übersichtlich sein. Als Zustandsgrößen sollen nur Phosphor, eine Phytoplanktongruppe und das Zooplankton berücksichtigt werden. Wichtige Randbedingungen sind Phosphor im Zulauf, der Jahresgang von Temperatur und Licht, die Durchmischungstiefe und die Eisbedeckung im Winter.

Nachdem wir eine Liste aller wichtigen Zustandsgrößen und Einflussfaktoren (Randbedingungen, Hilfsgrößen) erstellt haben, skizzieren wir uns ein Wirkdiagramm. Dieses soll alle Zustandsgrößen, die wichtigsten Umweltfaktoren und die wesentlichen Kopplungen zwischen diesen Elementen enthalten. Versuchen Sie einmal selbst, ein solches Diagramm zu entwerfen und diskutieren Sie das Ergebnis mit Ihren Kommilitonen oder Kollegen.

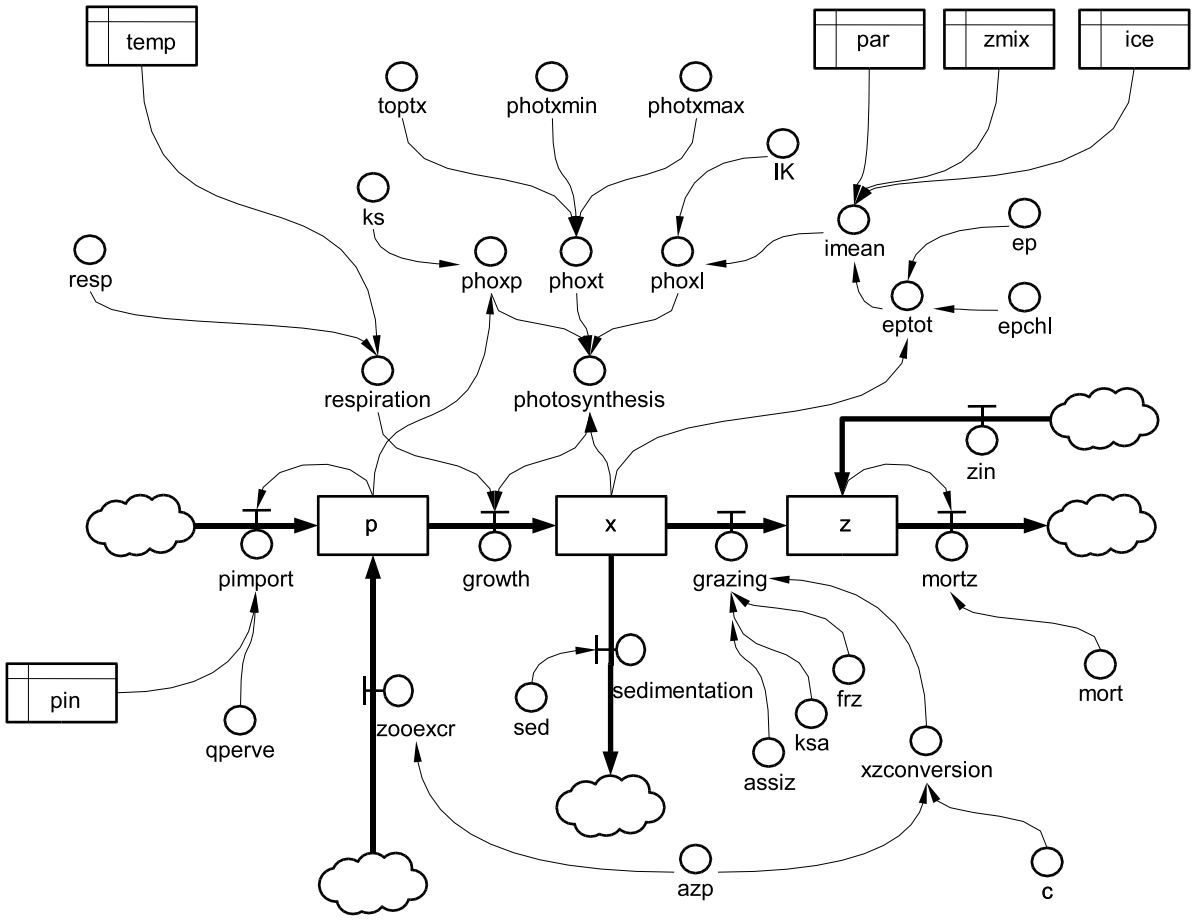

Abbildung 3.8: Salmo-Light eine Prinzipskizze für ein Standgewässer-Nahrungsnetz-Modell.

3.3.2 Modellspezifikation

Im Sinne einer einheitlichen und implementierungsfreundlichen Schreibweise für unser kleines Modell, schreiben wir alle direkt für Salmo-Light benötigten Parameter, Zustandsgrößen, Hilfsgrößen und Randbedingungen mit Kleinbuchstaben. Zusätzlich dazu wollen wir versuchen, ausgewählte in Seenmodellen enthaltene Grundmechanismen etwas umfassender zu betrachten.

3.3.2.1 Bilanzgleichungen

Wir beginnen mit dem Aufschreiben der Bilanzgleichungen für alle Zustandsgößen. Meist ist dies der einfachste Schritt. Der Konvention des Modells AQUAMOD folgend, verwenden wir für den Phosphor p die Maßeinheit $\mu g l^{-1}$P, für das Phytoplankton x die Maßeinheit $\mu g l^{-1}$Chlorophyll a und für das Zooplankton z die Maßeinheit m$g l^{-1}$Kohlenstoff. Für die Konversion P in Chl_a benutzen wir vereinfachend und der Konvention des Modells AQUAMOD folgend einen Konversionsfaktor von $Y = 1.0$.

$$\frac{dp}{dt} = pimport + zooexcr - growth \tag{3.66}$$

$$\frac{dx}{dt} = 1/Y \cdot growth - grazing - sedimentation \tag{3.67}$$

$$\frac{dz}{dt} = zin + c \cdot azp \cdot grazing - mortz \tag{3.68}$$

3.3.2.2 Nährstoffabhängigkeit des Wachstums

Für die Beschreibung der Nährstoffabhängigkeit des Wachstums existieren zwei unterschiedliche Ansätze, der einstufige Ansatz und der zweistufige Ansatz. Beim einstufigen Ansatz ist die Wachstumsrate von der externen Nährstoffkonzentration abhängig, beim zweistufigen Ansatz werden die Prozesse Nährstoffaufnahme und Wachstum in Abhängigkeit von der internen Nährstoffkonzentration unterschieden, so dass auch das Vorhandensein eines internen Speichers berücksichtigt werden kann.

Einstufen-Ansatz

Die häufigste für die Nährstoffabhängigkeit des Wachstums verwendete Funktion ist die Monod-Funktion. Hierbei handelt es sich um die selbe Gleichung wie bei der Michaelis-Menten-Kinetik. Der Unterschied besteht lediglich darin, dass die Michaelis-Menten-Kinetik die Substratabhängigkeit einzelner Enzyme beschreibt. Bei der Monod-Kinetik wurde dieser Ansatz auf komplette Organismen übertragen.

Die Bestimmung der Parameter μ_{max} und k_S kann in kontinuierlicher (Chemostat) oder diskontinuierlicher Kultur (Batch) erfolgen. Hierbei ist es bei der Batchkultur extrem wichtig, mit einer geringen Startbiomasse zu beginnen, um Konkurrenz oder allelochemische Wechselwirkungen zu vermeiden, da die Verletzung dieser Bedingung zu unrealistisch großen k_S-Werten führt.

Berücksichtigung der Phosphatspeicherung

In der Natur findet fast immer Wachstum im Ungleichgewicht statt, d.h. es sind Nährstoffspeicherung, Recycling und Übergangsphasen (transiente Phasen) zu beobachten.

Da nur im Chemostat nur die internen und die externen Substratpools (S_{intern} und S_{extern} bzw. P_{intern} und P_{extern}) im Gleichgewicht stehen, muss ein Weg zur Berücksichtigung des zellinternen Phosphats gefunden werden. Eine Möglichkeit ist das Zweistufenmodell (Droop-Modell, DROOP, 1974, 1983; LAMPERT and SOMMER, 1993):

1. Stufe: P-Aufnahme ($P_{extern} \longrightarrow P_{intern}$),
2. Stufe: Wachstum ($\mu = f(P_{intern})$).

Hierbei ist die Wachstumsrate vom zellinternen P-Gehalt pro Biomasseeinheit Q_P und einer artspezifischen minimalen Zellquote $Q_{P,0}$ abhängig:

$$\mu = \mu_m \left(1 - \frac{Q_{P,0}}{Q_P}\right) \tag{3.69}$$

Es gibt mehrere Varianten des Droop-Modells, z.B. Gl. 3.70ff. Hierbei ist P_{intern} die *Zustandsgröße* zellinterner Phosphor, also die Menge an internem P in der gesamten vorhandenen Phytoplanktonbiomasse X und $Q_P = P_{intern}/X$ die Zellquote. In Anlehnung an VARIS (1988) erhält man in etwa ein Modell in der folgenden Form:

$$\frac{dX}{dt} = \mu_{max} \cdot \left(1 - X \cdot \frac{Q_{P,0}}{P_{intern}}\right) \cdot X - \ldots \tag{3.70}$$

$$\frac{dP_{intern}}{dt} = X \cdot \mu_{max} \cdot Q_P \cdot \frac{Q_{P,max} - \frac{P_{intern}}{X}}{Q_{P,max} - Q_{P,0}} \cdot \frac{P}{k_P + P} - \ldots \tag{3.71}$$

$$\frac{dP}{dt} = \ldots \tag{3.72}$$

Der Vorteil dieser Herangehensweise ist, dass eine P-Speicherung möglich ist, d.h. das Wachstum kann zeitverzögert erfolgen. In P-limitierten Gewässern gewinnt im Frühjahr diejenige Phytoplanktonart, die den vorhandenen Phosphor am schnellsten aufnehmen und speichern kann.

Das Droop-Modell besitzt allerdings auch eine Reihe schwerwiegender Nachteile:

- Es wird eine zusätzliche Differentialgleichung benötigt.
- Die P-Aufnahmerate ist sehr schwer und nur ungenau bestimmbar. Hierbei müssen Markierungsexperimente mit einer sehr geringen Tracerkonzentration verwendet werden.
- Außer der mittleren Zellquote Q_P müssen noch die minimale und die maximale Zellquote ($Q_{P,0}$ und $Q_{P,max}$) bekannt sein.

Einen gewissen Ausweg bildet die Verwendung eines phytoplanktondichteabhängigen Selbsthemmungstermes für das Wachstum (BENNDORF, 1979) in Form einer inversen Sättigungskurve:

$$\mu(X) = \mu_{max} \cdot \frac{\frac{1}{X}}{\frac{1}{k_X} + \frac{1}{X}} \tag{3.73}$$

Somit erhält man:

$$\mu = \mu_{max} \cdot \frac{\frac{P}{X}}{\frac{k_P}{k_X} + \frac{P}{k_X} + \frac{k_P}{X} + \frac{P}{X}} \tag{3.74}$$

Dieser Ansatz ist weit weniger kompliziert als das Droop-Modell und auch einfacher als er aussieht. Ein solcher Selbsthemmungsterm besitzt natürlich auch einige Nachteile. So ist der Wirkmechanismus

in Wirklichkeit anders, der Parameter k_x ist unter Umständen seinerseits von P abhängig und es kann keine Zeitverzögerung zwischen P-Aufnahme und Wachstum dargestellt werden. Dies wird manchmal bei der Simulation der Phytoplankton-Frühjahrsentwicklung ein Problem. Auch ein *luxury uptake* P-verarmter Algen nach einer impulsförmigen P-Zugabe ist nicht darstellbar.

Auf der anderen Seite bewirkt dieser Ansatz eine relativ realistische und quantitativ gute Abbildung des Phytoplanktonwachstums. Insbesondere wird ein in anderen Modellen häufig beobachtetes „Hochschießen“ der Biomasse verhindert. In der Realität existieren neben der Konkurrenz um essentielle Nährstoffe und das Licht bekanntlich noch weitere Konkurrenzmechanismen, wie z.B. allelopathische Wirkungen oder Infektionen des Phytoplanktons mit Phagen (bei Cyanobakterien) oder Pilzen. Der auf Labor-Beobachtungen basierende Selbsthemmungsansatz vereinigt solche Mechanismen in einer aggregierten und praktikablen Form.

Stickstoff

Die Abhängigkeit der Wachstumsrate vom Stickstoff wird analog zur P-Abhängigkeit ermittelt und als Monod-Kinetik dargestellt.

Bei einer Mehrfachlimitation wird von manchen Autoren eine multiplikative Verknüpfung der Terme vorgenommen. Da es sich um essentielle Nährstoffe handelt, ist dies jedoch unzulässig, sondern es gilt näherungsweise der Liebig'sche Minimumansatz. Als Basis dient hierbei ein optimales N:P-Verhältnis von 16:1 (Redfield-Ratio, Atomverhältnis, d.h. Masseverhältnis = $(16 \cdot 14) : (1 \cdot 31) =$7.2:1). Je nachdem, welche Limitation vorherrscht, wird $\mu(P)$ bzw. $\mu(N)$ verwendet.

3.3.2.3 Temperaturabhängigkeit von Wachstum und Photosynthese

Die Abhängigkeit der Wachstumsrate von der Temperatur ($temp$) hat die Form einer Optimumkurve und fällt bei vielen Algen oberhalb von ca. 20 ° C ab. Allerdings hat es sich wenig bewährt, diese Abhängigkeit direkt zu beschreiben, da sich die beobachtete Optimumkurve aus der erst bei höheren Temperaturen abfallenden Optimumkurve der Photosynthese und einer bereits bei weniger hohen Temperaturen schnell ansteigenden Respiration zusammensetzt. Es ist deshalb sinnvoll, Photosynthese und Respiration separat zu beschreiben.

Für die Photosyntheserate lässt sich im Bereich von 0 bis 25 ° C entweder eine Potenz- oder Exponentialfunktion oder sogar noch eine lineare Näherung verwenden.

$$phoxt(temp) = photxmin + \frac{photxmax - photxmin}{toptx} \cdot temp; \qquad temp \leq toptx \tag{3.75}$$

3.3.2.4 Lichtabhängigkeit der Photosyntheserate

Die Lichtabhängigkeit der Photosyntheserate bestimmt man zweckmäßigerweise in Batchkulturen mit niedriger Startbiomasse. Als Funktionstyp kann in erster Näherung eine Monod-Beziehung verwendet werden.

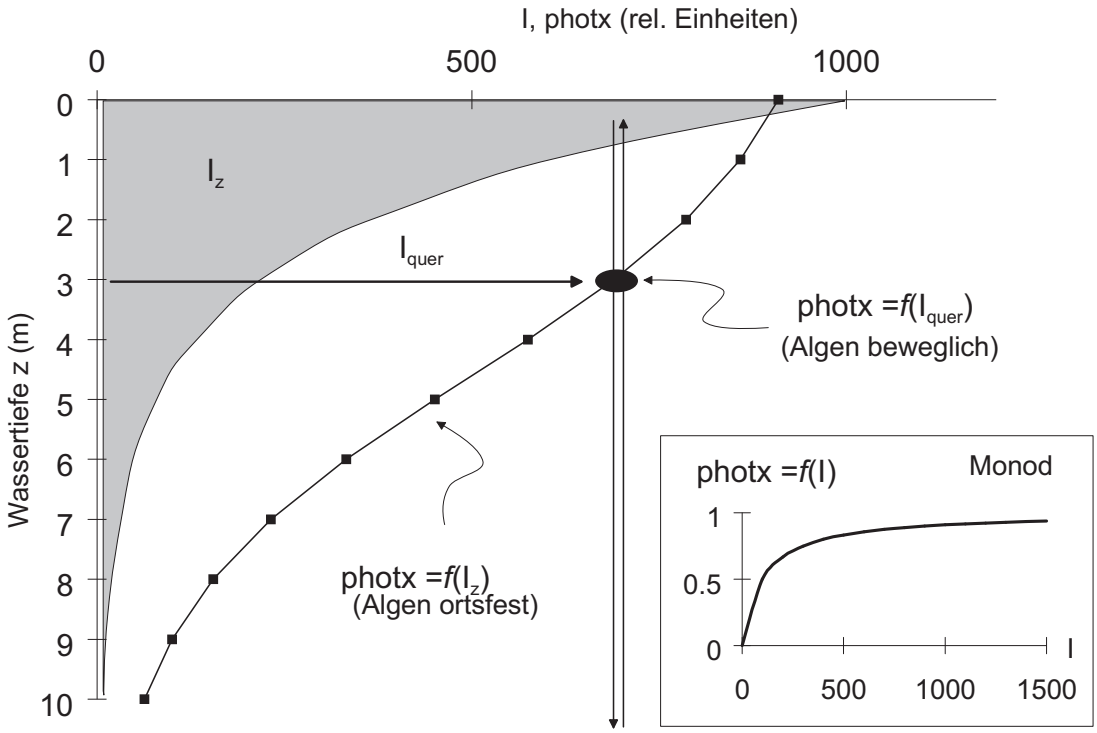

Abbildung 3.9: Das Vertikalprofil der Photosynthese (Bruttoprimärproduktion, gefüllte Quadrate) resultiert aus der vertikalen Lichtschwächung (Lambert-Beersches Gesetz, ausgefüllte Fläche) und aus der Lichtabhängigkeit der Photosynthese (Monod-Kinetik). Bei der Berechnung erhält man unterschiedliche Werte, wenn entweder die Photosynthese aus der tiefengemittelten Lichtintensität (horizontaler Pfeil) ermittelt wird (Algen beweglich, auf- und absteigende Pfeile) oder wenn die Photosynthese als tiefenabhängig berechnet wird (Algen ortsfest, gefüllte Quadrate) und man erst anschließend mittelt.

$$photx = photxmax \cdot \frac{I_z}{k_I + I_z}$$

Hierbei ist I_z die Lichtintensität in der jeweiligen Wassertiefe z. Bei einer sehr hohen Lichtintensität kann Lichthemmung auftreten, d.h. die Kurve fällt nach einem Sättigungsbereich wieder ab. Allerdings spielt dieser Prozess in der Natur meist eine geringere Rolle als zunächst angenommen:

- Die Lichthemmung tritt gewöhnlich nur an der Wasseroberfläche auf, und nur dann, wenn sehr geringe Turbulenz herrscht.
- Oftmals wachsen unter solchen Bedingungen Cyanobakterien, die gegen Lichthemmung wirksame Schutzmechanismen (z.B. Schutzpigmente) besitzen.

Die Halbsättigungskonstante k_I der Photosysntese selbst ist nur wenig temperaturabhängig, wohl aber die Respiration und somit das „k_I“ der Wachstumsrate.

Integration über die Tiefe: $\bar{I}$-Variante und I_z-Variante

Die im Freiland verfügbare photosynthetisch aktive Strahlung (400 ... 700 nm) beträgt näherungsweise 50% der einfallenden Globalstrahlung. Beim Einfall in ein Gewässer wird davon ein gewisser Anteil reflektiert (als Faustwert ca. 10%, je nach Sonnenstand und Wellenhöhe), im Winter kann der vom Eis reflektierte und absorbierte Anteil 90% betragen. Im Wasser selbst wird ein weiterer Teil des Wassers reflektiert, absorbiert oder gestreut. Diese Abnahme der Lichtintensität I mit der Tiefe z kann mit dem Lambert-Beerschen Extinktionsgesetz beschrieben werden

$$\frac{dI}{dz} = -\varepsilon \cdot I \tag{3.76}$$

das ergibt als analytische Lösung:

$$I_z = I_0 \cdot e^{-\varepsilon \cdot z} \tag{3.77}$$

wobei I_0 die Lichtintensität kurz unter der Wasseroberfläche und ε der mittlere vertikale spektrale Extinktionskoeffizient ist. Hierbei setzt sich die Extinktion als Summe aus der Extinktion des phytoplanktonfreien Wassers ε_{min}, das jedoch „Gelbstoffe“ und mineralische Trübung enthalten kann, und einem phytoplankton- oder chlorophyllspezifischen Extinktionskoeffizienten zusammen.

Bei der Berechnung der mittleren Photosynthese in einer Wasserschicht (z.B. Epilimnion) erhält man unterschiedliche Resultate, wenn man zuerst das arithmetische Mittel des verfügbaren Lichtes ermittelt und dieses in die Monod-Gleichung der Photosynthese einsetzt ($\bar{I}$ oder I_{quer}-Variante) oder wenn man zuerst für jede Tiefe die Photosynthese errechnet und die erhaltenen Werte anschließend mittelt (I_z-Variante). Dies ist mathematisch mit der Nichtlinearität der beiden verwendeten Gleichungen erklärbar oder biologisch mit der Reaktionsgeschwindigkeit des Photosystems, je nachdem ob die Algen bei ihrer Wanderung mit der Strömung in der Lage sind, die Energie über ein gewisses Lichtfeld zu integrieren oder ob die Energie praktisch augenblicklich in Photosyntheseprodukte umgesetzt wird. Die Mehrleistung kann durchaus 100% betragen und wurde sowohl theoretisch untersucht (BAUMERT and UHLMANN, 1983) als auch empirisch in Laborkulturen oder im Freiland (BENNDORF and BAUMERT, 1981; BEHRENDT and NIXDORF, 1993) beobachtet. Die Realität liegt (wie fast immer) zwischen den Extremen, in der Praxis gilt für tiefe Gewässer eher die I_z-Variante (Tabelle 3.1). Dadurch kommt durch die Tiefenabhängigkeit der Lichtintensität neben der Zeit eine zweite Dimension ins Spiel, die durch eine partielle Differentialgleichung gelöst werden kann. Für die Kombination aus Lambert-Beer'schem Gesetz und Monod-Funktion existiert eine analytische Lösung:

$$photx = \frac{photxmax \cdot \ln(I_0 + k_I)}{(\varepsilon \cdot z) \cdot (k_I + I_0 \cdot e^{-\varepsilon \cdot z})} \tag{3.78}$$

Tabelle 3.1: Varianten der vertikalen Mittelung der Photosynthese

$\bar{I}$-Variante	I_z-Variante
$\bar{I} = \frac{I_0 + I_0 \cdot e^{-\varepsilon z}}{2}$	$I_z = I_0 \cdot e^{-\varepsilon \cdot z}$
Phytoplankter sind in der Lage, das Licht über die Tiefe zu integrieren, Photosystem reagiert langsam	Phytoplankter integrieren das Licht nicht, Reaktion des Photosyntheseapparates schneller als die Verlagerung der Phytoplankter im Lichtfeld
geringe Durchmischungstiefe und starke Lichtextinktion und hohe Turbulenz	große Durchmischungstiefe oder geringe Lichtextinktion oder geringere Turbulenz

3.3.2.5 Respiration

Für das Wachstum gilt allgemein

$$growth = photosynthesis - respiration \tag{3.79}$$

Im Modell Salmo-Light benutzen wir wie in SALMO für die Temperaturabhängigkeit der Respiration einfach eine lineare Approximation:

$$respiration = x \cdot resp \cdot temp \tag{3.80}$$

Ein etwas anspruchsvollerer Ansatz beschreibt die Respirationsrate als Summe einer temperaturabhängigen Grundatmung und eines temperaturabhängigen Leistungsumsatzes. Hierbei wird die Respiration indirekt über die Photosynthese und das Wachstum ermittelt.

3.3.2.6 Grazing durch das Zooplankton

Das Grazing ist für das Phytoplankton eine wesentliche Verlustgröße. Allerdings ist die Verwertbarkeit der einzelnen Arten recht unterschiedlich und nur in Laborexperimenten zu beurteilen. Ein wichtiges Kriterium ist die Größe der Planktonorganismen, jedoch werden einerseits auch große Arten gefressen (z.B. *Asterionella formosa*), andererseits weisen kleinere Arten Fraßschutzmechanismen auf (z.B. *Microcystis*: Toxin und Filtrationshemmstoff) oder sind schwer verdaulich und überstehen die Darmpassage unbeschadet (z.B. Grünalgen mit Gallerthüllen) und sedimentieren dann mit den Kotpellets.

Bei zwei oder mehr Phytoplanktonarten ergibt sich das Problem der unterschiedlichen Präferenz durch das Zooplankton. In unserem kleinen Modell Salmo-Light benutzen wir sehr stark vereinfachend nur eine Monod-Abhängigkeit der Zooplankton-Filtrierrate von der Phytoplanktondichte und nur eine einzige Phytoplanktongruppe:

$$grazing = z \cdot x \cdot frz \cdot assiz \cdot \frac{ksa}{ksa + x} \tag{3.81}$$

Etwas anspruchsvollere Modelle berücksichtigen meist mehrere funktionelle Phytoplanktongruppen, ein oder mehrere Zooplanktongruppen und natürlich auch die Fische, entweder als eigene Zustandsgröße(n) oder zumindest über eine temperaturabhängige Aktivität.

3.3.2.7 Sedimentation

Die Sedimentation hängt von den den morphologischen (Größe, Form) und physiologischen (Dichte) Eigenschaften der Phytoplankter ab. Gemäß dem Stokes'schen Gesetz sedimentieren große, annähernd runde Zellen schneller als kleine Zellen, die gegebenenfalls noch einen Formwiderstand besitzen. Darüberhinaus spielen hydrophysikalische Prozesse (Zirkulation) eine wesentliche Rolle. In unserem kleinen Modell benutzen wir sehr vereinfacht eine konstante Nettosedimentationsrate *sed*:

$$sedimentation = x \cdot sed \tag{3.82}$$

Komplette Seenmodelle enthalten natürlich auch ein Sedimentsubmodell, entweder in Form relativ einfacher semi-empirischen Formeln oder als aufwändiges mehrschichtiges Sedimentmodell mit mehreren Zustandsgrößen.

3.3.2.8 Randbedingungen

Die Randbedingungen (Eingangsdaten, Antriebe, *inputs, forcing functions*) werden bei „realistischeren" Simulationsmodellen gewöhnlich aus einer Datei eingelesen. Es handelt sich meist um Zeitreihen äußerer Umweltfaktoren wie meteorologische und hydrologische Randbedingungen, Nährstoffbelastung und Import lebender Biomasse. Im Fall des Modells Salmo-Light sind das die Zeitreihen von Temperatur ($temp$), photosynthetisch aktiver Strahlung (par), Durchmischungstiefe ($zmix$), P-Konzentration im Zufluss (pin), Zuflussrate ($qperve$, d.h. Zuflussmenge pro Gewässervolumen qin/v) und Periode der Eisbedeckung (ice). Aus Gründen der Einfachheit werden diese Zeitreihen hier einfach als zeitabhängige Funktionen (z.B. harmonische Funktionen oder Wenn-Dann-Regeln) dargestellt (Tabelle 3.4).

Das komplette Modell (Spezifikationsperspektive) ist in den Tabellen 3.2 bis 3.5 dargestellt. Die Parameterwerte entstammen den Modellen AQUAMOD (STRAŠKRABA and GNAUCK, 1983) und SALMO BENNDORF and RECKNAGEL (1982); BENNDORF *et al.* (1985).

Tabelle 3.2: Zustandsgleichungen für das Modell Salmo-Light

$\frac{dp}{dt}$	$= pimport + zooexcr - growth$
$\frac{dx}{dt}$	$= 1/Y \cdot growth - grazing - sedimentation$
$\frac{dz}{dt}$	$= zin + c \cdot azp \cdot grazing - mortz$

Tabelle 3.3: Hilfsgleichungen für das Modell Salmo-Light

$pimport$	$= qperve \cdot (pin - p)$
$growth$	$= photosynthesis - respiration$
$photosynthesis$	$= phoxl \cdot phoxp \cdot phoxt \cdot x$
$respiration$	$= x \cdot resp \cdot temp$
$phoxl$	$= imean/(ik + imean)$
$phoxp$	$= p/(ks + p)$
$phoxt$	$= (photxmax - photxmin)/toptx \cdot temp + photxmin$
$imean$	$= (par - par \cdot exp(-eptot \cdot zmix))/eptot/zmix \cdot (1 - 0.9 \cdot ice)$
$eptot$	$= epchl \cdot x + ep$
$grazing$	$= z \cdot x \cdot frz \cdot assiz \cdot ksa/(ksa + x)$
$zooexcr$	$= grazing \cdot (1 - azp)$
$sedimentation$	$= x \cdot sed$
$mortz$	$= z \cdot mort$

Tabelle 3.4: Randbedingungen für das Modell Salmo-Light

$temp$	$= 12 + 10 \cdot \sin((time + 220) \cdot \pi/182)$	$(^\circ C)$
par	$= 0.5 \cdot (776.33 + 622.36 \cdot \sin(2 \cdot \pi \cdot (time + 284)/365))$	$(\mathrm{J}\,cm^{-2}d^{-1})$
ice	$=$ if $time < 90$ then 1 else 0	E$isbedeckung$
$zmix$	$=$ if $time < 90$ or $time > 240$ then 20 else 4	(m)
pin	$=$ if $time > 200$ and $time < 240$ then 40 else 20	(μgL^{-1})
$qperve$	$= 0.01$	(d^{-1})

Tabelle 3.5: Parameter des Modells Salmo-Light

zin	0.0005	Zooplankton-Import als Kohlenstoff	$(\mathrm{g}m^{-3}d^{-1})$
assiz	0.818	Assimilationskoeffizient des Zooplanktons	
azp	0.6	Anteil verwertbaren Futters an aufgen. Nahrung	
Y	1.00	Ertragskoeffizient (Chl:P)	
c	0.05	Kohlenstoff : Chlorophyll-Verhältnis	$(\mathrm{m}g\,\mu g^{-1})$
ep	0.2	Extinktionskoeffizient des planktonfreien Wassers	(m^{-1})
epchl	0.017	spez. Extinktionskoeff. des Chlorophylls	$(\mathrm{m}^2 mg^{-1})$
frz	0.5	Filtrationsrate Zooplankton	$(\mathrm{m}^3 g^{-1}d^{-1})$
ik	10	Halbsättigungskonstante Licht	$(\mathrm{J}cm^{-2}d^{-1})$
ks	5	Halbsättigungskonstante P	$(\mu g L^{-3})$
ksa	60	Halbsättigungskonstante Grazing	$(\mathrm{C}hl_a\ mg\,m^{-3})$
mort	0.05	Mortalitätsrate Zooplankton	(d^{-1})
photxmax	1.8	Max. Photosyntheserate	(d^{-1})
photxmin	0.17	Photosyntheserate bei 0^0 C	(d^{-1})
toptx	20	Optimumtemperatur	$(^\circ C)$
sed	0.06	Sedimentationsrate	(d^{-1})
resp	0.014	Abh. Respirationsrate von der Temperatur	$(\mathrm{d}^{-1}\,^\circ C^{-1})$

3.3.3 Implementierung

Die Implementierung des Modells folgt den obigen Beispielen, nur dass die `model`-Funktion etwas umfangreicher ausfällt. Eine Besonderheit besteht darin, dass alle Randbedingungen als Gleichung direkt im Modell aufgeführt sind. Bei größeren Modellen werden diese normalerweise als externer Input eingelesen. Wegen der treppenförmigen Randbedingungen kann es vorkommen, dass der Integrationsalgorithmus ineffektiv arbeitet, deshalb wurden die Toleranzparameter auf etwas größere Werte gesetzt. In der vorliegenden Version funktioniert `lsoda` ohne eine Schrittweitenbegrenzung (siehe Abschnitt 3.2.7), für eigene Experimente ist jedoch ein `hmax=1` durchaus angeraten.

```
###------------ An absolut simplistic lake model ----------
library(deSolve)

model <- function(time, xx, parms) {
  temp    = 12+10*sin((time+220)*pi/182)                  # (deg C)
  par     = 0.5*(776.33+622.36*sin(2*pi*(time+284)/365))# (J/(cm^2*d))
  ice     = ifelse(time < 90, 1, 0)                       # (ice cover)
  zmix    = ifelse(time < 90 | time > 240, 20, 4)         # (m)
  pin     = ifelse(time > 200 & time < 240, 40, 20)       # (mu g/L)
  qperve  = 0.01                                          # (1/d)

  with (as.list(c(xx, parms)),{

    eptot   <- epchl * x + ep
    imean   <- (par-par*exp(-eptot*zmix))/eptot/zmix*(1-0.9*ice)

    phoxl   <- imean/(ik + imean)
    phoxp   <- p/(ks + p)
    phoxt   <- (photxmax - photxmin)/toptx*temp + photxmin

    pimport       <- qperve * (pin - p)
    photosynthese <- phoxl * phoxp * phoxt * x
    respiration   <- x * resp * temp
    growth        <- photosynthese - respiration
    grazing       <- z * x * frz * assiz * ksa/(ksa+x)
    zooexcr       <- grazing * (1 - azp)
    sedimentation <- x * sed
    mortz         <- z * mort

    dp <- pimport + zooexcr            - growth
    dx <- 1/Y * growth  - grazing      - sedimentation
```

```
    dz <- zin     + c * azp * grazing  - mortz
    list(c(dp, dx, dz))
  })
}
# vectors of parameters, timesteps and initial values
parms  <- c(
      zin     = 0.0005,  # zooplankton-import (Carbon, g/(m^3*d))
      assiz   = 0.818,   # assimilation coeff. of zooplankton
      azp     = 0.6,     # part of food utilized
      c       = 0.05,    # carbon:chlorophyll-ratio (mg/mug)
      Y       = 1.00,    # yield coefficient, Chl:P ratio (-)
      ep      = 0.2,     # ext. of plankton free water (1/m)
      epchl   = 0.017,   # spec. ext. of chlorophyll (m^2/mg)
      frz     = 0.5,     # zoopl. filtration rate  (m^3/(g*d))
      ik      = 10,      # half sat. light (J/(cm^2*d))
      ks      = 5,       # half sat. phosphorus (mg/m^3)
      ksa     = 60,      # half sat. grazing (Chl_a mg/m^3)
      mort    = 0.05,    # zooplankton mortality rate (1/d)
      photxmax= 1.8,     # max photosynth. rate (1/d)
      photxmin= 0.17,    # min photosynth. rate (1/d)
      toptx   = 20,      # optimal temp. photosynthesis (deg C)
      sed     = 0.06,    # sedimentation rate (d-1)
      resp    = 0.014    # temp. dep. resp. rate (1/(d * deg C))
)
times <- seq(1, 365, 1)
xstart <- c(p = 10,  # phosphorus
            x = 0.5, # phytoplankton
            z = 0.01 # zooplankton
)

out  <- ode(xstart, times, model, parms)

plot(out)
```

3.3.4 Übungen

1. Schauen Sie sich das Modell (Simulationsdiagramm, Gleichungen) an, rechnen Sie mehrmals eine Simulation (einen Jahresgang) und versuchen Sie den Aufbau und die Funktion zu verstehen.
2. Einfluß der Randbedingungen: Entfernen Sie alle saisonal steuernden Elemente schrittweise und simulieren Sie jeweils einen Jahresgang.
   a) Setzen Sie die Randbedingung *ice* auf 0 (keine Eisbedeckung): Wann startet das Phytoplankton?
   b) Setzen Sie *zmix* (Durchmischungstiefe) auf 4m.
   c) Setzen Sie *pin* (P-Konzentration im Zufluß) auf konstant 20μg/l. Welchen Zweck könnte die erhöhte P-Konzentration im Spätsommer besitzen, d.h. welcher nicht im Modell enthaltene Prozeß wird dadurch imitiert? Wie wirkt sich das auf das Phytoplankton aus?
   d) Plotten Sie die Randbedingungen *par* und *temp* als Jahresgang. Setzen Sie nun die Temperatur *temp* auf konstant 12°C und die Globalstrahlung *par* auf konstant 370 J$cm^{-2}d^{-1}$. Wie beurteilen Sie das Endergebnis?
   e) Wodurch sind im Modell Zeitpunkt und Anzahl der Phytoplankton-Peaks gesteuert? Stellen Sie bei Bedarf für einzelne Randbedingungen die saisonale Steuerung wieder her.
3. Einfluss des Phospor-Imports: Simulieren Sie Jahresgänge mit P-Importen von (konstant) 1, 20, 50, 100 $\mu g/l$ P im Zulauf.
   a) Stellen Sie die Sensitivität des Chlorophyll-Peaks gegenüber P grafisch dar. [10].
   b) Was ist sonst noch zu beobachten?
   c) Was wäre bei Änderung der P-Belastung bezüglich der Startwerte von p, x, z zu sagen?

   Wiederholen Sie die Simulationen ggf. mit neuen Startwerten. Simulieren Sie einen mehrjährigen Zyklus.
4. Einfache Sensitivitätsanalyse: Modifizieren Sie (jeweils auf 0, auf 50% und auf 200% setzen) folgende Parameter und beobachten Sie den Einfluß auf die Zustandsgrößen. Falls 0 zu Fehlern führt, verwenden Sie einen anderen kleinen Wert.
   a) *ks*
   b) *ik*
   c) *sed*
   d) frz
   e) *resp*

[10] Hinweis: Obwohl sich das natürlich auch in R vollautomatisch programmieren lässt, geht es unter Umständen schneller, sich die Ergebnisse auf einem Zettel zu notieren und diese dann mit R oder mit einer Tabellenkalkulation grafisch darzustellen.

Diskutieren Sie jeweils die Bedeutung des Parameters sowie Art und Umfang der Wirkung auf x, z und p!

5. Biomanipulation und künstliche Destratifikation: Welche Parameter bzw. Randbedingungen müsste man modifizieren, um den Einfluß dieser Maßnahmen zu untersuchen? Simulieren Sie geeignete Szenarien!

6. Weitere Möglichkeiten für eigene Experimente: Einfluß der Respiration des Phytoplanktons, Einfluß des Grazings (frz), Einfluß der Selbstbeschattung ($epchl$). Wozu dient *zin*? Was fehlt im Modell?

3.3.5 Ausblick: Komplexe Wassergütemodelle

An dieser Stelle soll nochmals betont werden, dass „Salmo-Light“ nur ein nur ein „Spielzeugmodell“ und für praktische Szenarioanalysen völlig ungeeignet ist. Neben der geringen Anzahl von Zustandsgrößen und vielen nicht enthaltenen Prozessen fehlt z.B. ein hydrophysikalisches Submodell völlig. Zur näheren Vertiefung sei deshalb für den Standgewässerbereich die Beschäftigung mit dem „historischen“ Modell CLEANER (SCAVIA and PARK, 1976) und den aktuell verwendeten Modellen SALMO (BENNDORF and RECKNAGEL, 1982; BENNDORF, 1985; BENNDORF and PETZOLDT, 1999), CE-QUAL-W2[11], DYRESM-WQ (HAMILTON and SCHLADOW, 1997; SCHLADOW and HAMILTON, 1997), dem Modell PROTECH-C (ELLIOTT *et al.*, 2000), dem „Biochemical Model of Lake Zurich“ (OMLIN *et al.*, 2001)[12] oder den Lake-Michigan-Modellen (CHEN *et al.*, 2002a; JI *et al.*, 2002) empfohlen. Für den Fließgewässerbereich wichtige Modelle sind QUAL2E [13], das BfG-Modell QSIM (KIRCHESCH, 1992), und das darauf aufbauende ATV-Modell (ATV-ARBEITSGRUPPE 2.2.3, 1997) sowie das „River Water Quality Model no. 1“ (RWQM1, SHANAHAN *et al.*, 2001; REICHERT *et al.*, 2001; VANROLLEGHEM *et al.*, 2001).

[11] `http://www.ce.pdx.edu/w2/` und `http://www.wes.army.mil/el/elmodels/w2info.html`
[12] Das Modell heißt jetzt BELAMO.
[13] `http://www.epa.gov/OST/QUAL2E_WINDOWS/`

4 Individuenbasierte Modelle

Individuenbasierte Modelle verfolgen den Ansatz, das Verhalten von Populationen oder Ökosystemen aus dem Verhalten einzelner Individuen abzuleiten. Das Grundelement eines solchen Modells ist das Individuum dessen spezielle Eigenschaften wie Alter, Größe, physiologische Zustandsgrößen und räumliche Koordinaten über die Zeit verfolgt werden. Das Verhalten dieser Individuen wird durch Wenn-Dann-Regeln und algebraische Gleichungen oder durch einen Computeralgorithmus beschrieben. Dieses Regelwerk wird dann für jedes Individuum in einem festen oder einem variablen Zeitschritt durchlaufen. Bei festem Zeitschritt werden die einzelnen Individuen immer in der gleichen Reihenfolge oder quasi parallel simuliert. Variable Zeitschritte ermöglichen es, unterschiedlich schnell ablaufende Prozesse mit angepasster Häufigkeit abzuarbeiten (z.B. Nahrungsaufnahme häufiger als Reproduktion), erfordern aber einen zusätzlichen Verwaltungsaufwand, z.B. sortierte Listen[1].

In den meisten Fällen handelt sich also um Modelle mit diskretem Zeitschritt. Für kontinuierlich ablaufende Prozesse wie somatisches Wachstum entspricht das einer Integration nach Euler und erfordert entsprechend kleine Zeitschritte, in einigen Fällen werden solche Prozesse auch zwischen den Zeitschritten analytisch integriert.

Die meisten individuenbasierten Modelle benutzen stochastische Ansätze, so tritt z.B. an die Stelle einer Mortalitätsrate eine Mortalitätswahrscheinlichkeit. Die einzelnen Individuen können von vornherein unterschiedliche Eigenschaften aufweisen oder sich einfach durch unterschiedliche stochastische Einflüsse unterschiedlich entwickeln. Das bedeutet, dass bei einer Simulation in der Regel nur eine Stichprobe von Individuen verfolgt werden kann. Die Eigenschaften einer Population oder eines Systems ergeben sich durch eine statistische Auswertung der Einzelergebnisse und oftmals mehrerer gleichartiger Simulationen.

Die Wahl der geeigneten Stichprobengröße der zu verfolgenden Individuen erweist sich oftmals als Problem. Einerseits ist es meist prinzipiell nicht möglich, alle Individuen einzeln zu verfolgen und dies ist auch nicht notwendig (z.B. alle Daphnien in einem See). Andererseits möchte man die verschiedenen individuellen Eigenschaften der Population adäquat abbilden. Eine Möglichkeit hierzu ist es, jedem simulierten Individuum eine Wichtung mitzugeben, d.h. eine Zahl die angibt, für wie viele Individuen das simulierte Objekt steht. SCHEFFER *et al.* (1995) nennen dieses Konzept Super-Individuen-Ansatz. Umgekehrt kann es bei Populationen mit hoher Mortalität auch passieren, dass die simulierten Individuen aussterben oder zumindest eine so kleine Stichprobe ergeben, dass die abzubildende Population nicht mehr genügend repräsentiert wied. In einem solchen Fall kann man eine feste Zahl von Modellindividuen verfolgen und einen Resampling-Algorithmus benutzen. Bei Mortalität eines Individuums wird dann ein anderes zufälliges „Spender-Individuum“ ausgewählt und geteilt, d.h. seine Wertigkeit halbiert und seine Eigenschaften auf das Empfänger-Individuum übertragen (ROSE *et al.*, 1993).

[1] Hierfür gibt es in der Informatik sehr effiziente Algorithmen, z.B. balancierte Bäume.

Einen ausgezeichneten Überblick in die in der Fischereibiologie übliche Nomenklatur, sowie Empfehlungen zum Aufbau individuenbasierter Modelle geben HUSE *et al.* (2002) im „Fish and Fisheries Handbook“. Allerdings haben sich „individuenbasierte“ Simulationstechniken mehrfach unabhängig voneinander entwickelt und sind natürlich auch außerhalb der Ökologie gebräuchlich. Sie werden dort z.B. unter Bezeichnungen wie „agentenbasierte“ Simulation, *discrete event*-Simulation oder „Automaten“ geführt. Da sich das Gebiet außerdem in einer extrem stürmischen Entwicklung befindet und vielleicht auch wegen des sehr ausgeprägten kreativen Pragmatismus ist die Nomenklatur sehr vielgestaltig.

Der Pragmatismus kommt auch darin zum Ausdruck, dass es keine einheitliche Methode zur formalen Darstellung von individuenbasierten Modellen gibt. Die Darstellung besteht oft aus einer verbalen Beschreibung mit grafischen Flussdiagrammen und wenigen mathematischen Formeln, manchmal aus einer mehr oder weniger formalen Regelsyntax, die aber fast immer auf ein spezielles Problem oder einen speziellen Modelltyp zugeschnitten ist, und oft auch nur als Computercode. Manchmal werden individuenbasierte Modelle deshalb auch als „computational models“ neben die mathematischen Modelle gestellt aber jeder Computercode lässt sich mit Hilfe von Regeln auch in die Sprache der Mathematik übersetzen und es wird deshalb die Forderung erhoben, dies auch zu tun (DEANGELIS and GROSS, 1992; GROSS *et al.*, 1992). Aus meiner Sicht sind einige Diagrammtypen aus der „Unified Markup Language“ (UML, siehe z.B. FOWLER and SCOTT, 1997) sehr nützlich, da sie in ihrer Aussagekraft wesentlich vielfältiger und leistungsfähiger sind, als die klassischen Flussdiagramme.

Für die folgenden relativ einfachen Modelle möchte ich allerdings auf die Benutzung einer formalen Regelsyntax verzichten und auch aus Platzgründen eine Kombination aus verbaler Beschreibung und R-Code verwenden (Implementierungsperspektive, siehe 1.7). Für wissenschaftliche Publikationen sollte man jedoch eine von den Details einer speziellen Programmiersprache abstrahierende Darstellung (Spezifikationsebene) auf jeden Fall vorzuziehen.

Bezüglich der räumlichen Ausdehnung unterscheidet man räumlich nicht auflösende (*non-spatial*), räumlich implizite (hier hat man z.B. Metapopulationen, die aus mehreren interagierenden Patches bestehen) und räumlich explizite Modelle (*spatially explicit models*). Die räumliche Komponente kann hier entweder durch ein festes Gitter (*lattice based* oder *grid based*), z.B. Zelluläre Automaten) oder durch räumliche Koordinaten als Eigenschaften der Individuen berücksichtigt werden. Wir wollen solche Modelle ohne festes Gitter als *continuous space models* bezeichnen, allerdings besteht hier eine gewisse Verwechslungsgefahr mit Modellen, die auf partiellen Differentialgleichungen basieren (z.B. Advektions-Diffusions-Modellen) und die nicht individuenbasiert sind.

Implementierung individuenbasierter Modelle

Es existieren spezielle Simulatoren für bestimmte Klassen individuenbasierter Modelle, z.B. das in C++ entwickelte System OSIRIS (MOOIJ and BOERSMA, 1996), das am Santa Fe Institute entwickelte SWARM[2] oder das gitterbasierte System COENOCON (MAMEDOV and UDALOV, 2002).

Wegen der größeren Flexibilität und Effizienz werden viele individuenbasierte Modelle mit Hilfe allgemeiner Programmiersprachen oder auch in Tabellenkalkulationen implementiert. Hierbei kann man

[2] `http://www.swarm.org/`
Weitere Modelle und Simulatoren, siehe z.B. `http://www.red3d.com/cwr/ibm.html`

zwei grundsätzliche Implementierungsweisen unterscheiden, die objektorientierte Programmierung (OOP), z.B. mit Programmiersprachen wie SIMULA, JAVA oder C++ oder die tabellenorientierte Implementierung, bei der Matrizen und Vektoren zur Datenspeicherung benutzt werden. Die OOP ist besonders elegant, da hier jedes Individuum als Objekt mit seinen spezifischen Eigenschaften (wie Größe und Alter) und Verhaltensweisen (z.B. die Art der Futtersuche) implementiert werden kann. Bei der Simulation werden alle Individuen nacheinander aktiviert und jedes Individuum „weiß“ selbst, welche Verhaltensweisen wie ausgeführt werden müssen. So ist z.B. das Paket OSIRIS (MOOIJ and BOERSMA, 1996) objektorientiert in C++ geschrieben und auch wir benutzen manchmal gern unseren JAVA-Simulator (RINKE and PETZOLDT, 2001).

Bei der tabellen- oder matrixorientierten Implementierung sind die Eigenschaften jedes Individuums jeweils in einem Datensatz (einer Tabellenzeile) dargestellt, die Verhaltensweisen werden durch entsprechende Unterprogramme beschrieben. Eine Liste aller gleichartigen Objekte in einem objektorientierten Programm oder alle Zeilen (Datensätze) einer Individuentabelle stellen eine Stichprobe oder eine „Population“ dar. Da die großen Vorteile der OOP erfahrungsgemäß erst nach einem gewissen Umdenken sichtbar werden, begnügen wir uns im folgenden mit der tabellenorientierten Darstellung. Prinzipiell lassen sich aber mit beiden Implementierungsweisen ähnlich leistungsfähige Modelle erstellen.

4.1 Populationsdynamik

Zur Konstruktion eines individuenbasierten Modells muss zunächst einmal das spezielle Verhalten oder der gesamte Lebenszyklus des zu modellierenden Individuums mit Regeln und Wahrscheinlichkeiten oder mit Gleichungen beschrieben werden. Hierzu können Ansätze verwendet werden, die direkt aus Labor- oder Freilandversuchen abgeleitet worden sind, z.B. elementare Wachstumsmodelle aus Abschnitt 3.1, Regressionsmodelle aus Abschnitt 2 oder physiologische und bioenergetische Modelle. Solche nicht räumlichen physiologischen oder populations- und verhaltensbiologischen Modelle finden sich als Submodelle auch bei den räumlich expliziten Ansätzen wieder.

4.1.1 Wachstum einer Population

Das Wachstum einer Population lässt sich ganz einfach so beschreiben, dass jedes Individuum während eines Zeitschrittes Δt mit einer bestimmten Wahrscheinlichkeit p Nachkommen erzeugt. Die Wahrscheinlichkeit kann für alle Individuen gleich sein oder von Alter und Fitness abhängen. Die jeweiligen Nachkommen werden dann in die Population aufgenommen, d.h. an die Individuentabelle angehängt. Mortalität wird dadurch dargestellt, dass gestorbene Individuen aus der Tabelle gestrichen werden.

4.1.2 Implementierung in R

Die Vektororientierung von R vereinfacht die Darstellung individuenbasierter Modelle ganz erheblich, da die Behandlung aller Individuen einer Population in einem Vektor, einer Matrix oder einem Dataframe quasi zeitgleich (quasiparallel) ausgeführt werden kann. Ein weiterer ganz entscheidender Vorteil

besteht darin, dass mit Hilfe des in R möglichen datenbankartigen Zugriffs sehr einfach Teilmengen gebildet werden können.

Wir beginnen mit einem simplen Beispiel, einer Bakterienpopulation. Die Vermehrung beschreiben wir mit Hilfe der Teilungsrate, die in unserem Beispiel $0.4h^{-1}$ beträgt, d.h. pro Stunde teilen sich im Mittel 40% der Individuen. Zunächst definieren wir uns einen Vektor mit 5 Startindividuen. Jedes Vektorelement steht für ein Individuum. Da wir zunächst nur die Zahl der Bakterien berücksichtigen, ist es im Prinzip gleich, welchen Wert dieses Vektorelement enthält, wir wählen einmal die Zahlen 1 bis 5 um zu sehen, von welchen Vorfahren die neu entstandenen Nachkommen abstammen:

```
> individuals <- 1:5
```

Im zweiten Schritt erzeugen wir uns für jedes Individuum eine gleichverteile Zufallszahl zwischen 0 und 1 und vergleichen diese Zufallszahl mit einer vorgegebenen Wahrscheinlichkeit (in unserem Fall der Teilungsrate von 0.4, da 40% aller Zufallszahlen zwischen 0 und 1 kleiner als 0.4 sind):

```
> random <- runif(individuals)
> divide <- ifelse(random < 0.4, TRUE, FALSE)
```

Nun sehen wir uns die Zufallszahlen und den Vektor `divide` an und sehen, dass die Individuen 1 und 3 zufällig für die Teilung ausgewählt wurden. Je nach Initialisierung des Zufallszahlengenerators kann auch ein anderes Ergebnis herauskommen:

```
> random
[1] 0.3541511 0.6399804 0.2090023 0.8684789 0.5571617
> divide
[1]  TRUE FALSE  TRUE FALSE FALSE
```

Es müssen jetzt also zwei neue Individuen in den Individuenvektor aufgenommen werden. Prinzipiell könnten wir einfach zwei beliebige Elemente an `indivuals` anhängen. Wegen des „Zellteilungscharakters“ gehen wir jedoch so vor, dass wir uns eine Kopie der Individuen 1 und 3 anlegen

```
> newindividuals <- subset(individuals, divide)
> newindividuals
[1] 1 3
```

und diese an die Liste anfügen, so dass der neue Individuenvektor nun 7 Individuen enthält:

```
> individuals <- c(individuals, newindividuals)
> individuals
[1] 1 2 3 4 5 1 3
```

Sollen mehrere Zeitschritte verfolgt werden, muss dieser Vorgang mehrfach wiederholt werden, z.B. in Form einer Schleife. Zur Abspeicherung der Zellzahlen führen wir einen zusätzlichen Vektor `counter` ein, der für jeden Durchlauf die Zellzahl registriert.

Die elementare individuenbasierte Populationsdynamik benötigt einschließlich grafischer Darstellung lediglich 14 Zeilen R-Code:

```
individuals <- 1:5
counter <- length(individuals)
for (i in 1:10) {
    random <- runif(individuals)
    divide <- ifelse(random < 0.4, TRUE, FALSE)
    newindividuals <- subset(individuals, divide)
    individuals <- c(individuals, newindividuals)
    n<-length(individuals)
    counter <- c(counter, n)
}
par(mfrow=c(1,2))
print(counter)
plot(counter)
plot(counter, log="y")
```

Wir simulieren nun mehrere Durchläufe und schauen uns die Ergebnisse und Grafiken an. Bei komplexeren Modellen sind weitere Auswertungen möglich. Im obigen einfachen Fall könnten wir lediglich testen, wieviele Nachkommen jeweils von einem der 5 ursprünglichen Individuen abstammen:

```
> table(individuals)
```

und bekommen eine Tabelle mit der Anzahl der Nachkommen jeder der ursprünglich vorhandenen fünf Bakterien:

```
individuals
 1  2  3  4  5
19 27  7 41 55
```

Mortalität

Die Beschreibung der Mortalität erfolgt analog zur Reproduktion. Im Unterschied zur Reproduktion müssen hier jedoch keine neuen Individuen an eine Tabelle angefügt werden, sondern es wird einfach eine Teilmenge der überlebenden Individuen gebildet, d.h. bei einer Mortalitätswahrscheinlichkeit p_m von 0.1 ist das:

```
> individuals <- subset(individuals, runif(individuals > 0.1))
```

Anstelle der Mortalitätswahrscheinlichkeit kann auch die Überlebensrate $p_s = 1 - p_m$ verwendet werden, wobei sich das Relationszeichen umkehrt.

4.1.3 Populationsdynamik am Beispiel von *Daphnia*

Beim vorhergehenden „Modell“ waren alle Individuen bis auf die Abstammung einheitlich, d.h. es gab keine unterschiedlichen Eigenschaften wie Alter oder Länge. Obwohl auch mit Bakterien durchaus

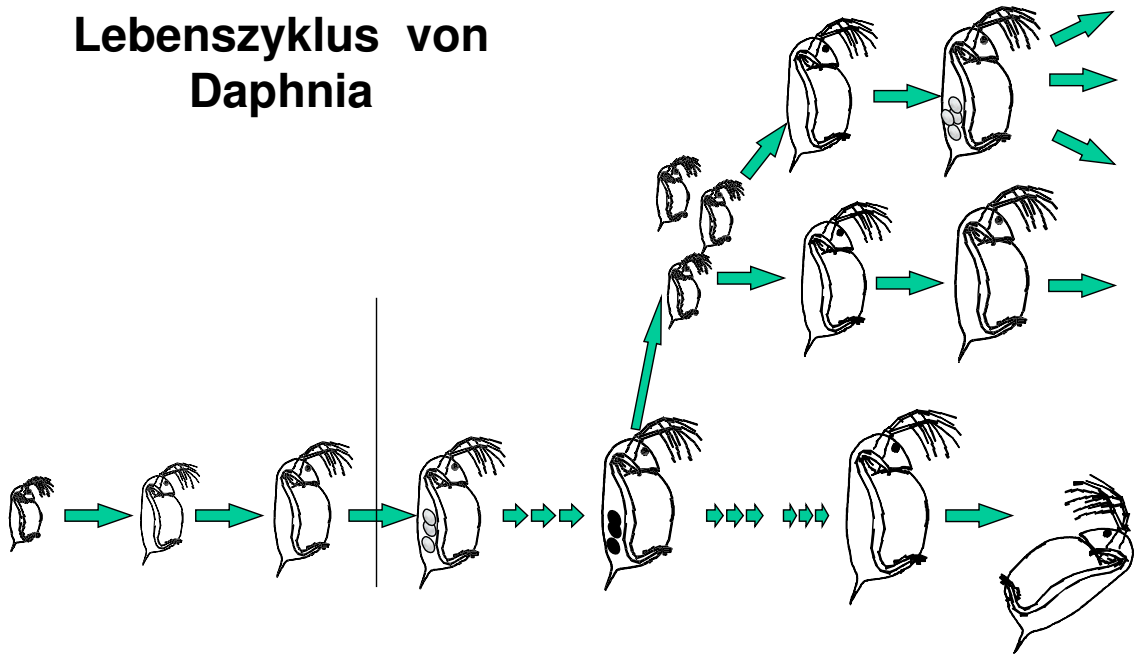

Abbildung 4.1: Lebenszyklus einer *Daphnia* (parthenogenetische Phase).

sinnvolle Demonstrationsmodelle und Anwendungen möglich wären, wollen hierzu die Populationsdynamik von *Daphnia* verwenden (Abb. 4.1).

Zunächst definieren wir den Modellzweck: Das Modell soll die Populationsdynamik und die Längen- und Altersklassenverteilung einer Daphnienpopulation in Abhängigkeit von der Temperatur reproduzieren. Primäres Ziel ist die Implementierung von Individuen mit unterschiedlichen individuellen Eigenschaften. Wir benötigen dafür ein paar Informationen und Randbedingungen, die wir wie folgt festlegen:

- Die Ei-Entwicklungszeit t_E ist eine Funktion der Temperatur (BOTTRELL *et al.*, 1976).
- Die Länge der frisch geschlüpften Daphnien (der Neonaten) soll $510\mu m$ betragen, das ist der Mittelwert für *Daphnia galeata* aus der Talsperre Bautzen vom 29.04.1999 (HÜLSMANN and WEILER, 2000; HÜLSMANN, 2000).
- Die juvenile Wachstumsrate betrug während der Untersuchungsperiode im Mittel $83\,\mu m d^{-1}$ (Mittelwert vom Frühjahr 1999).
- Wir nehmen lineares Längenwachstum bis zur Primiparagröße (Größe, in der erstmals Eier abgelegt werden) an,
- setzen die Primiparagröße fest auf $1250\mu m$ (Mittelwert in der Untersuchungsperiode) und

- die adulte Wachstumsrate soll 80% der juvenilen Wachstumsrate betragen (vereinfacht abgeleitet aus Laborversuchen von VOIGT und HÜLSMANN).
- Die maximale Lebenserwartung legen wir auf 30 d fest (Annahme),
- und verwenden die Biovolumen-Längen-Beziehung von KOROBOZOWOI (1974).

Die Anzahl der Nachkommen (Gelegegröße) kann als Häufigkeitsverteilung beschrieben werden. Diese Verteilung hängt von unterschiedlichen Faktoren (z.B. Futterangebot, Länge der Elterntiere) ab, zur Vereinfachung benutzen wir hier jedoch eine feste Verteilung der Gelegegröße adulter Tiere (vom 29.04.1999):

Gelegegröße	0	1	2	3	4	5	6	7
relative Häufigkeit	39%	9%	12%	14%	12%	8%	4%	1%
kumulative Häufigkeit	39%	49%	61%	74%	86%	95%	99%	100%

Es fällt auf, dass viele adulte Tiere keine Eier tragen. Um eine solche Verteilung dennoch reproduzieren zu können verwenden wir einen kleinen Trick. Auch bei Null Eiern muss das Tier einen kompletten Eientwicklungszyklus durchlaufen.

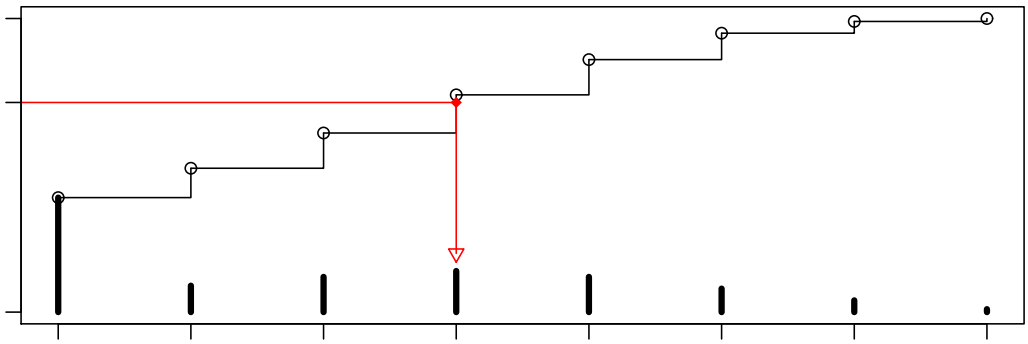

Abbildung 4.2: Auswahl aus einer empirischen Verteilung: Dargestellt ist die relative (Balken) und die kumulative Häufigkeitsverteilung (Treppen) der Eizahlen. Eine zufällige Eiverteilung wird ausgewählt, indem man für eine gleichverteilte Zufallszahl auf der y-Achse den dazugehörigen Wert auf der x-Achse sucht. Bei einer diskreten Verteilung wird eine Treppenkurve, bei kontinuierlichen Verteilungen eine lineare Interpolation verwendet.

Bei der Eiablage muss für jedes ausgewählte Individuum eine zufällige Eizahl entsprechend der empirisch beobachteten Häufigkeitsverteilung ausgelost werden. Zu diesem Zweck müssen wir eine mit

`runif()` erzeugte gleichverteilte Zufallszahl so umwandeln, dass wir eine Zufallszahl mit einer Verteilung erhalten, die der beobachteten Häufigkeitsverteilung der Gelegegröße entspricht.

Hierbei zieht man zunächst eine gleichverteilte Zufallszahl für die Häufigkeitsachse und sucht die zugehörige Eizahl auf der x-Achse durch Abzählen oder durch lineare Interpolation (Abb. 4.2).

Die Startpopulation verdient ebenfalls eine besondere Beachtung. Soll eine reale Gewässersituation nachvollzogen werden, so muss die anfängliche Populationsstruktur die gleiche Alters- und Längenverteilung wie im Freiland aufweisen (die für das Beispiel vorliegt). Für das Beispiel sparen wir uns jedoch den Aufwand und legen die Anfangslängenverteilung willkürlich fest.

4.1.4 Implementierung in R

Wir beginnen mit der Funktion `clutchsize` zur zufälligen Auslosung einer Gelegegröße. Zunächst legen wir die kumulative Häufigkeit in einem Vektor ab.

```
eggfreq <- c(0.39, 0.49, 0.61, 0.74, 0.86, 0.95, 0.99, 1)
```

Eine einfache Variante der `clutchsize`-Funktion besteht einfach darin, eine im Intervall $(0, 1)$ gleichverteilte Zufallszahl `rrr` zu ziehen, die kumulative Verteilung hochzuzählen und zu stoppen, wenn die Zufallszahl nicht mehr größer als der entsprechende y-Wert der Treppenkurve (`eggfreq`) ist:

```
clutchsize <- function() {
    rrr <- runif(1)
    n <- 1
    while (rrr >  eggfreq[n]) {
      n <- n+1
    }
    n - 1 # first element is for zero eggs!
}
```

Eine wesentlich schnellere Variante, die außerdem den Vorteil hat, vektorisiert zu arbeiten, d.h. gleich mehrere Gelegegrößen zu liefern, arbeitet mit der `approx`-Funktion, der Funktion für die lineare oder stufenweise Interpolation (`method = "constant"`):

```
clutchsize <- function(nn) {
    approx(c(0,eggfreq), 0:(length(eggfreq)),
           runif(nn), method="constant", f=0)$y
}
```

Der Programmcode des Daphnien-Populationsdynamikmodells besteht aus fünf einzelnen Blöcken:

1. der Festlegung von globalen Parametern,

2. Hilfsfunktionen (z.B. Botrell-Formel, Biomasse Formel und die oben vorgestellte Zufallsauswahl der Gelegegröße),
3. den eigentlichen Lebensprozessen (Wachstum, Mortalität, Eiablage, Freisetzung der Juvenilen),
4. der Festlegung einer Startpopulation,
5. der eigentlichen Simulationsschleife, in der die Lebensprozesse nacheinander aufgerufen werden und der
6. Auswertung, mit statistischen und grafischen Mitteln.

Die Individuen selbst sind in dem Dataframe `inds` abgelegt, wobei jedes Individuum durch eine Zeile dargestellt ist, welche die individuellen Eigenschaften Alter (`age`), Länge in μm (`size`), Eizahl (`eggs`) und Eialter (`eggage`) enthält. Weitere gewünschte Eigenschaften lassen sich beliebig zum Dataframe hinzufügen.

Die Lebensfunktionen werden für die alle Daphnien einmal pro Zeitschritt abgearbeitet. Wegen der Vektororientierung von R arbeitet dieser Aufruf jeweils mit der gesamten Population (quasiparallel).

In der `grow`-Funktion werden Alter, Körpergröße und Eialter erhöht, eine Regel (`ifelse`) bewirkt, dass für die adulten Individuen eine andere Wachstumsrate als für die Juvenilen angewendet wird.

Die `die`-Funktion erzeugt eine Teilmenge (`subset`) der überlebenden Individuen. Im Moment ist hier lediglich ein festes Sterbealter implementiert, es lassen sich jedoch ohne weiteres spezifische oder unspezifische Mortalitätsmuster mit Hilfe von Wahrscheinlichkeiten nachrüsten, z.B. eine größenabhängige Mortalität, wenn die Prädatoren entweder große (z.B. planktonfressende Fische) oder kleine (z.B. *Chaoborus*-Glasmückenlarven) Daphnien bevorzugen.

In der `spawn`-Funktion wird für jede Daphnie eine Gelegegröße entsprechend der Eizahlen-Verteilung ermittelt. Hierbei wird die zunächst für alle Tiere bestimmte Gelegegröße mit Hilfe einer `ifelse`-Regel nur in die Individuen (Zeilen des Dataframe) eingetragen, die adult sind:

```
inds$size > primipara
```

und sich augenblicklich nicht in einem Ei-Entwicklungszyklus (Häutungszyklus) befinden. Da ein Häutungszyklus ja bekanntlich auch stattfinden kann, wenn das Tier gar kein Gelege besitzt, benutzen wir die Eigenschaft `eggage` (Zeit innerhalb eines Häutungszyklus, gemessen an der normalen Eientwicklungszeit) als Erkennungsmerkmal:

```
inds$eggage==0
```

Bei den Individuen, auf die diese Regel nicht zutrifft, wird der alte Wert aus `inds$eggs` beibehalten.

Die `hatch`-Funktion ist etwas komplizierter. Hier muss für Individuen, deren Eientwicklungszyklus abgeschlossen ist, eine entsprechende Anzahl von Nachkommen erzeugt und in die Individuentabelle eingetragen werden. Die Funktion arbeitet hier nicht vektororientiert sondern es werden alle Individuen in einer Schleife (`for (i in 1:ninds)`) einzeln bearbeitet. Für jedes Individuum, bei dem das aktuelle Ialter größer als die temperaturabhängige Eientwicklungszeit ist und bei dem Eier angelegt sind wird eine entsprechende Anzahl Nachkommen mittels `rbind` an eine neue Populationstabelle

`newinds` angefügt[3]. Anschließend werden Eizahl und Ialter auf Null zurückgesetzt. Dies gilt auch für die pausierernden Individuen ohne Eier, falls die Ei-Entwicklungszeit (`edt`) um ist. In der letzten Zeile gibt die `hatch`-Funktion schließlich die mit Hilfe von `rbind` zusammengesetzte Population, bestehend aus den alten und den neu geborenen (neonaten) Individuen zurück.

Nachdem nun alle notwendigen Funktionen (Unterprogramme) definiert sind, kann mit der eigentlichen Simulation begonnen werden. Zunächst muss das Modell initialisiert, das heißt mit einer gewissen Anzahl von Startindividuen „besetzt“ werden (Schritt 4). Da die Populationsdynamik nicht unerheblich vom Zustand der Startpopulation abhängt, sollte der Initialisierung immer mit Bedacht vorgenommen werden.

In unserem Beispiel kann die Startpopulation z.B. aus synchronisierten neonaten Individuen bestehen (auskommentierte Variante). Wir können aber auch eine bestimmte Populationsstruktur vorgeben, z.B. basierend auf Messungen aus der Talsperre Bautzen. Als Test können wir uns die Startpopulation nach der Initialisierung einfach mit `fix(inds)` anschauen.

Im fünften Schritt, der eigentlichen Simulation wird die *life loop* eine vorgegegene Anzahl von Durchläufen ausgeführt. Innerhalb der *life loop* werden die einzelnen Lebensfunktionen Wachstum, Mortalität, Eiablage, Schlupf nacheinander aufgerufen und jeder Aufruf einer Lebensfunktion ergibt eine neue Individuentabelle (`inds`). Da meist nicht nur das Endergebnis der Simulation, sondern auch der Verlauf wichtig sind, werden in der *life loop* auch Zwischenergebnisse gesammelt.

Im Beispiel werden dafür die Vektoren `sample.n`, `sample.size` und `sample.age` verwendet, in denen die momentane Abundanz die mittlere Körperlänge und das mittlere Alter der Individuen abgespeichert werden. Darüber hinaus sammelt `sample.eggs` die Eizahlen der adulten Tiere. Bei `sample.agedist` und `sample.sizedist` handelt es sich um Listen, in denen für jeden Zeitschritt das Alter bzw. die Körperlänge aller Individuen gespeichert werden.

Bei größeren Simulationen empfielt es sich, die Zwischenergebnisse auf die Festplatte zu schreiben, z.B. alle Individuen mit der hier auskommentierten `write.table`-Funktion. Da bei individuenbasierten Simulationen teilweise extrem große Datenmengen anfallen können empfiehlt es sich, Art und Umfang der zu protokollierenden Daten sorgfältig zu planen.

Als letzter Schritt folgen dann die notwendigen statistischen und grafischen Auswertungen, entweder gleich im Anschluss an die Simulation, mit externen Programmen oder z.B. mit einer Tabellenkalkulation.

```
#=========================================================
# (1) global parameters
#=========================================================

# cumulative egg frequency distribution
eggfreq     <- c(0.39, 0.49, 0.61, 0.74, 0.86, 0.95, 0.99, 1)
```

[3] `rbind` ist bekanntlich die Funktion, mit der an eine größere Datenstruktur (Matrix oder Dataframe) neue Zeilen (rows) hinzugefügt werden können.

```
son         <- 510           # um
primipara   <- 1250          # um
juvgrowth   <- 83            # um/d
adgrowth    <- 0.8*juvgrowth # um/d
maxage      <- 30            # d
temp        <- 15            # deg C
timestep    <- 1             # d

# template for individual
newdaphnia <- data.frame(age = 0,
                         size = son,
                         eggs = 0,
                       eggage = 0)
#=========================================================
# (2) helper functions
#=========================================================
# egg development time of Daphnia div. spc.
# (Botrell. et al. 1976)
botrell <- function(temp) {
    log.a <-  3.3956
    b     <-  0.2193
    c     <- -0.3414
    exp(log.a + b * log(temp)+c * log(temp)^2) # d
}
# sampling from an empiric distribution
clutchsize <- function(nn) {
    approx(c(0,eggfreq), 0:(length(eggfreq)),
           runif(nn), method="constant", f=0)$y
}
# biovolume of daphnia (Korobozowoi, 1973)
biov <- function(len) {
    0.074 * (len * 0.001) ^ 2.92
}
#=========================================================
# (3) life methods of the individuals
#=========================================================
grow <- function(inds){
    ninds       <- length(inds$age)
    inds$age    <- inds$age + timestep
    inds$eggage <- ifelse(inds$size > primipara,
                                inds$eggage + timestep, 0)
    inds$size   <- inds$size + timestep *
                                ifelse(inds$size < primipara,
                                juvgrowth, adgrowth)
    inds
}
die  <- function(inds) {
    inds <- subset(inds, inds$age < 30)
```

```
    inds
}
spawn <- function(inds) {
  # individuals with size > primipara
  # and eggage==0 can spawn
  ninds          <- length(inds$age)
  neweggs        <- clutchsize(ninds)
  inds$eggs      <- ifelse(inds$size > primipara
                           & inds$eggage==0,
                           neweggs, inds$eggs)
  inds
}
hatch <- function(inds) {
    # individuals with eggs
    # and eggage > egg development time spawn
    ninds          <- length(inds$age)
    newinds        <- NULL
    edt <- botrell(temp)
    for (i in 1:ninds) {
        if (inds$eggage[i] > edt) {
          if (inds$eggs[i] > 0) {
            for (j in 1:inds$eggs[i]) {
                newinds <- rbind(newinds, newdaphnia)
             }
          }
          inds$eggs[i]   <- 0
          inds$eggage[i] <- 0
        }
    }
    rbind(inds, newinds)
}
#=========================================================
# (4) start individuals
#=========================================================

#---------------------------------------------------------
# synchronised start population (only juveniles)
#---------------------------------------------------------
#inds <- NULL
#for (i in 1:10) {
#    inds <- rbind(inds,newdaphnia)
#}

#---------------------------------------------------------
# small desynchronised start population drawn from an
# empirical distribution of measured individuals
# TS Bautzen 29.04.1999 (orig. measurements S. Huelsman)
#---------------------------------------------------------
```

```
size  <- c(1091, 1339, 618, 1286, 1153,
          1557, 668, 1423, 1113, 1422)
eggs  <- c(0, 0, 0, 5, 0, 3, 0, 3, 0, 1)
eggage<- c(0, 0, 0, 1.6, 0, 1.5, 0, 1.7, 0, 1.2)
# estimated age (via growth)
age   <- c(7, 14, 1, 11, 8, 27, 2, 20, 7, 20)

inds  <- data.frame(age=age,
                    size=size,
                    eggs=eggs,
                    eggage=eggage)

#=========================================================
# (5) life loop
#=========================================================
sample.n        <- NULL
sample.age      <- NULL
sample.size     <- NULL
sample.eggs     <- NULL
sample.agedist  <- NULL
sample.sizedist <- NULL

steps <- 40

for (k in 1:steps) {
    print(paste("timestep",k))
    inds <- grow(inds)
    inds <- die(inds)
    inds <- hatch(inds)
    inds <- spawn(inds)

    ninds           <- length(inds$age)
    sample.n        <- c(sample.n, ninds)
    sample.size     <- c(sample.size, mean(inds$size))
    sample.age      <- c(sample.age, mean(inds$age))
    sample.eggs     <- c(sample.eggs,
                         inds$eggs[inds$size > primipara])
    sample.agedist  <- c(sample.agedist, list(inds$age))
    sample.sizedist <- c(sample.sizedist, list(inds$size))
    # write.table(inds, file=paste("inds", k, ".dat", sep=""))
}
#=========================================================
# (6) results and graphics
#=========================================================
par(mfrow=c(2,2))
plot(sample.n, xlab="time (d)", ylab="abundance", type="l")
plot(sample.size, xlab="time (d)",
     ylab="mean body length (µm)", type="l")
```

```
hist(sample.eggs, freq=FALSE, breaks=0:10, right=FALSE,
     ylab="rel. freq.", xlab="egg number", main="")

time <- seq(1,steps,2)
boxplot(sample.agedist[time], names=as.character(time),
        xlab="time (d)", ylab="age distribution (d)")
```

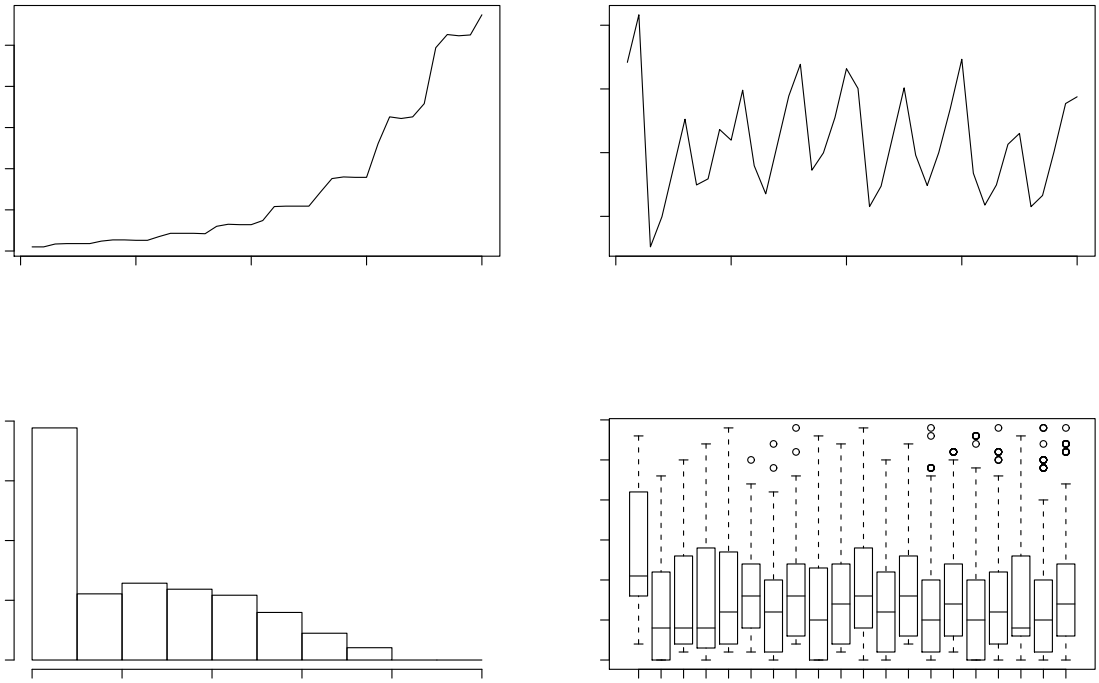

Abbildung 4.3: Individuenbasierte Simulation der Populationsdynamik von *Daphnia*.

Je nach Rechenleistung des verwendeten PCs kann die Simulation durchaus ein paar Sekunden dauern, wofür insbesondere die `hatch`-Funktion verantwortlich ist, wenn bei wachsender Abundanz immer mehr Nachkommen erzeugt werden müssen. In unserem Fall wächst die Abundanz exponentiell (Abb. 4.3). Wir erkennen außerdem an der Form der Abundanzkurve und der Schwankungen der Körperlänge, dass die Entwicklung in Kohorten verläuft. Die Ursache hierfür ist auf der einen Seite die relativ kleine und nicht zufällige Startpopulation und auf der anderen Seite das Fehlen von zufälligen Einflüssen auf die Ei-Entwicklungszeit und das somatische Wachstum. In Modellen mit zufällig schwankender oder räumlich variabler Temperatur oder mit individuellen Unterschieden tritt dagegen nach kurzer Zeit eine Desynchronisation auf. Allerdings bewegen sich die Schwankungen von Körperlänge und Alter in einem moderaten Bereich, lediglich am Anfang treten größere Ausschläge auf. Schon nach recht kurzer Zeit stellt sich eine stabile Altersverteilung ein.

Die Häufigkeitsverteilung der Gelegegröße repräsentiert alle adulten Individuen der gesamten Simulation. Wir können uns die entsprechenden Werte auch numerisch als relative kumulierte Häufigkeit ansehen

```
cumsum(table(sample.eggs)/length(sample.eggs))
```

und erhalten:

```
[1] 0.38855 0.49931 0.62791 0.74645 0.85491 0.93455 0.97940
[8] 1.00000
```

Die Verteilung stimmt bis auf einen zufälligen Fehler erwartungsgemäß gut mit der Verteilung `eggfreq` überein, die wir als Annahme für die Simulation verwendet haben. Eine weitere wichtige Größe ist die Populationswachstumsrate. Im Unterschied zur Gelegegrößenverteilung war diese nicht von vornherein bekannt, sondern ergibt sich aus den getroffenen Annahmen erst als Ergebnis der Simulation. Bei Annahme eines exponentiellen Wachstums:

$$N = N_0 \cdot e^{r \cdot t} \tag{4.1}$$

gilt:

$$r = \frac{\ln(N_t) - \ln(N_{t-1})}{\Delta t} \tag{4.2}$$

Da im Beispiel der Zeitschritt $\Delta t = 1$ war, gibt Differenz der aufeinanderfolgenden Logarithmen die Zeitreihe der Wachstumsrate wieder

```
> diff(log(sample.n))
```

die wir grafisch über die Zeit darstellen können. Der Mittelwert der Wachstumsrate

```
> mean(diff(log(sample.n)))
[1] 0.1038473
```

beträgt $r = 0.1\,\mathrm{d}^{-1}$ und ist ein sehr charakteristischer Wert der gesamten Simulation.

4.1.5 Weitere Aufgaben

1. Simulieren Sie mehrere Durchläufe des Modells und vergleichen Sie das Populationswachstum (Abundanz). Stellen Sie mehrere Simulationsläufe in einem gemeinsamen Diagramm dar.
2. Stellen Sie die Populationswachstumsrate grafisch dar und Berechnen Sie die mittlere Populationswachstumsrate b.
3. Führen Sie eine unspezifische und eine größenselektive Prädationsregel (kleinste oder größte Individuen bevorzugt) in die `die`-Funktion ein. Wie wirkt eine größenselektive Prädation auf die Populationswachstumsrate und auf die Populationsstruktur aus?
   Beispiel für unspezifische Mortalität mit Mortalitätsrate $d = 0.04\,\mathrm{d}^{-1}$:
   `inds <- subset(inds, runif(nrow(inds)) > 0.04)`

Das vorgestellte Daphnienbeispiel stellt selbstverständlich nur einen Einstieg dar, der sich beliebig ausbauen lässt (eventuell aus Effizienzgründen auch mit einem anderen System als R). Weitere Möglichkeiten wären z.B.:

- Implementierung einer *biomasseorientierten* Prädation, da es für eine Prädator einen Unterschied macht, ob er viele kleine oder viele große Daphnien konsumiert.
- Realisierung variabler Umweltbedingungen, z.B. einer variablen Temperatur, dies erfordert eine kumulative von den Individuen erfahrene Temperatur und die Einführung einer relativen Ei-Entwicklungszeit.
- Beschreibung der Temperatur- und Futterabhängigkeit der juvenilen und adulten Wachstumsrate mit einem von-Bertalanffy-Modell und Abhängigkeit der Gelegegröße von Temperatur und Futter (Rinke und Petzoldt, in Vorbereitung).
- Benutzung eines Ansatzes für Wachstum und Vermehrung nach mit einem bioenergetischen Ansatz (McCauley *et al.*, 1990a,b) oder einem DEB-Ansatz (siehe Abschnitt 3.1.2.5).

4.2 Das Matrixmodell von Leslie

4.2.1 Grundlagen

Eine besonders elegante Möglichkeit zum Umgang mit altersstrukturierten Populationen ist das Matrixmodell von Leslie (1945). Hierbei werden die Geburtenraten F_i (*fecundity*) und die Überlebensraten P_i jeder Generation in eine Matrix eingetragen, die quadratische Leslie-Matrix $\mathbf{L}$, die für jede Altersklasse eine Zeile bzw. Spalte enthält. In der ersten Zeile stehen die Geburtenraten pro Altersklasse, also z.B. für die juvenile Altersklasse 0, danach höhere Vermehrungsraten (z.B. 4, 3) und bei den alten, seneszenten Altersklassen wieder kleinere Werte, z.B. wieder 0.

Die Subdiagonale enthält für jede Altersklasse die Überlebensraten, so bedeutet z.B. ein $L_1 = 0.7$ dass von den Individuen der Altersklasse 1 ein Anteil von 70% die Altersklasse 2 erreicht.

Zusätzlich zur LESLIE-Matrix wird ein Populationsvektor $\mathbf{n_0}$ benötigt, dieser enthält in jeder Zeile die Abundanz der entsprechenden Altersklasse. Die Populationsstruktur des nächsten Zeitschritts, z.B. im nächsten Jahr enthalten wir dann einfach durch Matrixmultiplikation:

$$\mathbf{n_1} = \mathbf{L} \cdot \mathbf{n_0} \tag{4.3}$$

also:

$$\mathbf{n_1} = \begin{pmatrix} P_1 & P_2 & P_3 & P_4 \\ L_1 & 0 & 0 & 0 \\ 0 & L_2 & 0 & 0 \\ 0 & 0 & L_3 & 0 \end{pmatrix} \cdot \begin{pmatrix} N_1 \\ N_2 \\ N_3 \\ N_4 \end{pmatrix} \tag{4.4}$$

4.2.2 Implementierung in R

Die Implementierung in R gestaltet sich äußerst einfach, schließlich benutzt ja R eine matrixorientierte Programmiersprache. Wir benutzen einfach einmal eine Population mit 4 Altersklassen und definieren uns eine Matrix:

```
L <- matrix(0, nrow=4, ncol=4)
```

In die erste Zeile tragen wir die Geburtenraten für alle 4 Altersklassen ein:

```
L[1,] <- c(0,4,3,1)
```

und in die Subdiagonale für jede Altersklasse die Überlebenswahrscheinlichkeit:

```
L[2,1] <- 0.7
L[3,2] <- 0.9
L[4,3] <- 0.5
```

Natürlich hätten wir die Leslie-Matrix auch mit einem einzelnen Befehl initialisieren oder die Daten von der Festplatte einlesen können, ganz gleich, das Ergebnis ist jetzt:

```
     [,1] [,2] [,3] [,4]
[1,]  0.0  4.0  3.0    1
[2,]  0.7  0.0  0.0    0
[3,]  0.0  0.9  0.0    0
[4,]  0.0  0.0  0.5    0
```

Nun benötigen wir nur noch die aktuellen Abundanzen und speichern diese im Vektor `n0`

```
n0 <- c(9,5,4,2)
```

und erhalten die Abundanzen für den nächsten Zeitschritt, z.B. nach einem Jahr durch:

```
n1 <- L %*% n0
```

Hierbei ist `%*%` der Operator für die Matrixmultiplikation[4] in R. Die Abundanz in den Altersklassen beträgt nun also:

```
> n1
     [,1]
[1,] 34.0
[2,]  6.3
[3,]  4.5
[4,]  2.0
```

4.2.3 Weitere Aufgaben

1. Implementieren Sie eine aus 10 Altersklassen bestehende Population (z.B. Fische) und beobachten Sie diese über eine Zeit von 20 Jahren. Stellen Sie die jeweilige Altersstruktur durch ein Histogramm oder durch Boxplots dar.
2. Starten Sie nun ihre Population mit einem einzelnen Fisch in der 1. Altersklasse und beobachten Sie die Populationsentwicklung. Berechnen Sie die Populationswachstumsrate r.

4.3 Gitterbasierte Modelle

4.3.1 Grundlagen

Zellulären Automaten (*cellular automata*, CA) sind räumlich diskrete Modelle, bei denen die räumliche Ausdehnung in Form eines Gitters von aneinanderliegenden Zellen beschrieben wird. Die zeitliche Entwicklung der Gitterzellen und deren Zustand im nächsten Zeitschritt wird durch die Anwendung von Regeln auf den jeweils aktuellen Zustand der Zellen und deren Nachbarzellen ermittelt. Die Regeln bestehen ursprünglich vor allem darin, den Zustand einer Zelle aus den Zuständen der Nachbarzellen abzuleiten, wie es in streng deterministischer Form bei dem vom Mathematiker JOHN HORTON CONWAY 1970 erfundenen „Game of Life“ der Fall ist.

[4] Wer nicht mehr genau weiß, wie eine Matrixmultiplikation funktioniert, sollte jetzt bitte in einem Mathematiklehrbuch oder Tafelwerk (z.B. GÖHLER, 1987) nachschauen.

Aus der ursprünglichen Idee hat sich inzwischen eine unüberschaubare und kreative Vielfalt entwickelt. So sind z.B. die Regeln meist stochastisch, die Nachbarschaft bezieht auch weiter entfernte Zellen mit ein, innerhalb der Zellen laufen komplette physiologische Modelle ab und es werden externe Umweltfaktoren berücksichtigt.

4.3.2 Implementierung in R

Problem

Es soll das klassische „Conway's Game of Life" implementiert werden. Eine Zelle kann den Zustand lebend (= 1) oder tot (= 0) aufweisen. Die Nachbarschaft einer Zelle auf einem rechteckigen Gitter wird durch alle 8 Nachbarzellen bestimmt. Der Regelsatz ist deterministisch und lautet:

- Lebende Zellen mit genau zwei oder drei lebenden Nachbarzellen überleben.
- Tote Zellen mit genau drei lebenden Nachbarn wechseln in den Zustand „lebend".
- Zellen mit keinem, genau einem oder mehr als drei lebenden Nachbarn wechseln in den Zustand „tot".

Lösung

Bei der Implementierung von zellulären Automaten spielt die Ermittlung der Nachbarschaftsverhältnisse einer Zelle eine zentrale Rolle. Da dies mit den Hausmitteln von R nur sehr ineffizient implementiert werden kann[5], werden die entsprechenden Teilaufgaben von einer in C geschriebenen externen Routine[6] ausgeführt. Die in dieser externen Bibliothek enthaltene Funktionalität wird dann durch eine geeignete R-Funktion (im Beispiel `neighbours`) eingebunden.

Im eigentlichen Programmcode wird zunächst ein Gitter als Matrix von Zellen definiert (Matrix `z`) und mit willkürlich festgelegten Startzellen initialisert, die entweder zufällig oder regelmäßig aussehen können. Der untenstehende Programmcode demonstriert mehrere solche Initialisierungen nebeneinander auf einem Gitter.

Entsprechend des deterministischen Regelsatzes wird in jedem Durchlauf der Zustand für jede Zelle ermittelt. Im Beispiel wird mit mehreren Matrizen (`zgen`, `zsurv`) gearbeitet, die logisch verknüpft `z <- ((zgen + zsurv)>0)` und mit dem `image`-Befehl grafisch dargestellt werden.

```
#=========================================================
# (1) help functions
#=========================================================
dyn.load("simecol.dll")
eightneighbours <- function(x){
  n <- dim(x)[1]
```

[5] Zumindest ist mir keine effiziente Lösung eingefallen.
[6] unter Windows eine DLL, unter Linux eine *shared library*

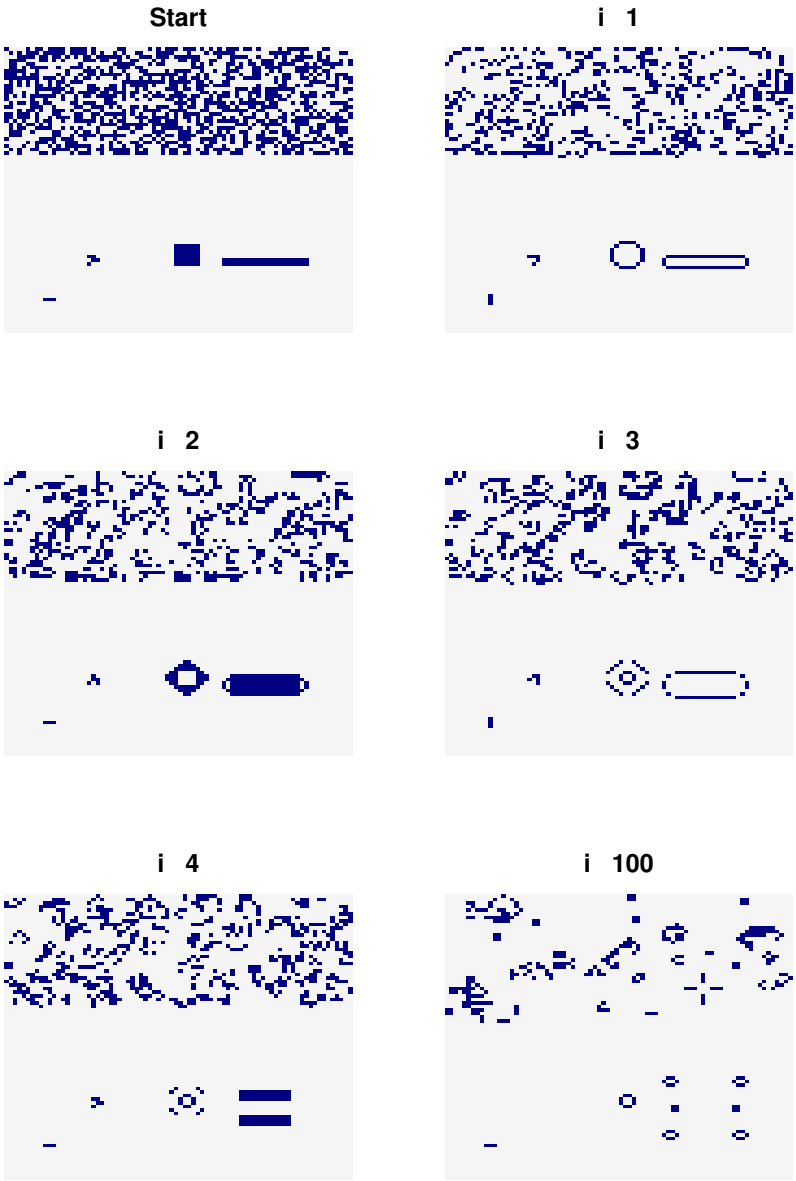

Abbildung 4.4: Conway's Game of Life. Die Teilabbildungen zeigen den Startzustand, sowie Zustände nach 1, 2, 3, 4 und 100 Schritten. Die obere Hälfte wurde zufällig initialisiert, die untere Hälfte enthält regelmäßige Strukturen, von links nach rechts: einen kurzen Balken (3x1 Zelle), einen sogenannten Gleiter, ein Rechteck mit 6x6 Zellen und einen langen Balken mit 20x2 Zellen.

```
  m <- dim(x)[2]
  y <- rep(0, length(x))
  z <- .C("eightneighbours", as.integer(n), as.integer(m),
          as.double(x), y=as.double(y))$y
  z
}
#=========================================================
# (2) define grid and start individuals
#=========================================================
n<-80
m<-80
z<-rep(0, m*n)
dim(z)<- c(n,m)

z[40:45,20:25] <-1 # filled square 6x6 cells
z[51:70,20:21] <-1 # small rectangle 20 x 2 cells
z[10:12,10]    <-1 # short bar 3x1 cells
z[20:22,20:22] <- c(1,0,0,0,1,1,1,1,0) # glider
z[1:80,51:80]  <-round(runif(2400))    # random

image(z, col=c("wheat", "navy"), axes=FALSE)
#=========================================================
# (3) life loop
#=========================================================
for (i in 1:100){
  nr <- i+100
  nb <- eightneighbours(z)
  ## survive with 2 or 3 neighbours
  zsurv <- ifelse(z > 0 & (nb == 2 | nb ==3), 1, 0)

  ## generate for empty cells with 3 neigbours
  zgen <- ifelse(z == 0 & nb == 3, 1, 0)
  z  <- ((zgen + zsurv)>0)
  dim(z) <-c(n,m)
  image(z, col=c("wheat","navy"), add=T)
}

dyn.unload("simecol.dll")
```

Die Simulation zeigt, dass sich bereits durch die Anwendung von sehr einfachen Überlebens- und Fortpflanzungsregeln regelmäßige Muster und relativ komplexe Strukturen ergeben (Abb. 4.4). So sind sogar bei einer zufälligen Initialisierung nach kurzer Zeit Strukturen erkennbar und bestimmte regelmäßige Strukturen zeigen ein festgelegtes Verhalten. Der kurze 3x1-Balken schaltet z.B. immer zwischen

2 Zuständen hin und her, der Gleiter bewegt sich über das Feld fort, aus dem 6x6-feld ergibt sich ein statisches Gebilde (siehe n=100) und der lange Balken bildet eindrucksvolle Muster. Insgesamt ist Conway's Game of Life ein Beispiel, wie aus Regeln Struktur entstehen kann und dass die Art der Struktur sehr stark von der Art der Regeln abhängt.

4.3.3 Weitere Aufgaben

1. Experimentieren Sie mit der Festlegung der Startzellen.
2. Modifizieren Sie die Überlebens- und Fortpflanzungsregeln.

4.3.4 Ausblick

Gitterbasierte Modelle mit realem ökologischem Hintergrund können sich unter anderem durch folgende Eigenschaften von „Conway's Game of Life" unterscheiden:

- Die Regeln beinhalten stochastische Elemente (Wahrscheinlichkeiten).
- Die Nachbarschaft wird nicht nur durch die unmittelbaren Gitternachbarn, sondern auch durch entferntere Nachbarn bestimmt.
- Innerhalb einer Zelle sind mehr als zwei Zustände möglich. Der Übergang zwischen den Zuständen wird durch die Nachbarn und durch die Entwicklung der Zelle selbst (z.B. Wachstum oder physiogische Prozesse), sowie durch äußere Störungen bestimmt[7].

Einige ausgewählte aktuelle Beispiele hierfür sind GRIMM (1999), CASWELL and ETTER (1999), WOOTTON (2001), und CHEN *et al.* (2002b). Zelluläre Automaten (Cellular Automata, CA) sind auch nicht unbedingt auf ein zweidimensionales rechteckiges Raster begrenzt, sondern es existieren auch eindimensionale CA (WILSON, 2000), dreidimensinale CA (GRIMM, 2001) oder sehr häufig CA mit hexagonalem Raster, also „Bienenwaben" z.B. bei LE PAGE and CURY (1997) oder auch MIDDELHOFF (1998).

4.4 Modelle mit kontinuierlichem Raum

4.4.1 Grundlagen

Bewegungsmuster

Bei den *continuous space*-Modellen ist die Bewegung der Individuen nicht an ein diskretes und fest verankertes Gitter gebunden, sondern die Individuen tragen die Koordinaten mit sich, besitzen also als

[7] Das diese Anleitung begleitende R-Package enthält ein Demonstrationsbeispiel eines stochastischen zellulären Automaten mit Altersstruktur und einer verallgemeinerten Nachbarschaftsmatrix.

Eigenschaft räumliche Koordinaten[8]. Ein weiterer Unterschied besteht darin, dass die Bewegung nicht durch Zustandsänderung von Gitterzellen in Abhängigkeit von Nachbarschaftsbeziehungen beschrieben wird, sondern durch die Veränderung der individuellen Koordinaten.

In der Ebene lässt sich eine Bewegung durch eine Bewegungsrichtung (Winkel α) und eine zurückgelegte Strecke (Radius r) beschreiben. Zur Darstellung eines solchen Bewegungsvektors in einem kartesischen (x,y) Koordinatensystem dient eine einfache trigonometrische Rechnung:

$$\Delta x = r \cdot \sin(\alpha \cdot \pi/180) \tag{4.5}$$

$$\Delta y = r \cdot \cos(\alpha \cdot \pi/180) \tag{4.6}$$

Für eine ungerichtete Bewegung (*random walk*) erzeugt man für jeden Bewegungsschritt eine zufällige Wegstrecke (normal- oder gleichverteilt) und einen zufälligen Winkel (**gleichverteilt**) im Intervall $(0, 2\pi)$ bzw. $(0°, 360°)$. Unter gewissen Voraussetzungen folgt eine solche Bewegung dem Diffusionsgesetz, die resultierenden Effekte lassen sich dann unter Umständen auch mit partiellen Differentialgleichungen (Advektions-Diffusions-Modellen) beschreiben.

Setzt sich hingegen der Winkel aus dem Winkel der vorherigen Bewegung und z.B. einem normalverteilten zufälligen Winkel (Mittelwert 0) zusammen, dann wird daraus eine gerichtete Bewegung (*correlated random walk*). In der Realität lassen sich solche Bewegungsmuster aus Beobachtungen (z.B. Videoaufnahmen) ableiten, z.B. von BAILLIEUL and BLUST (1999) für *Daphnia magna*.

Interaktion mit der Umwelt

Abgesehen von ganz einfachen theoretischen Modellen (wie die in diesem Text) beinhalten räumlich explizite Modelle meist auch eine räumlich heterogene Umwelt (z.B. HÖLKER, 1999). Diese wird auch bei den *continuous space*-Modellen meist durch ein Raster, seltener durch kontinuierliche Funktionen beschrieben. Liegt ein festes Raster vor, so werden entweder die entsprechenden Umweltfaktoren (z.B. Futterangebot, Temperatur) direkt aus dem Raster abgelesen oder es wird zwischen benachbarten Stützstellen interpoliert.

Interaktion zwischen Individuen

Der größte Rechenaufwand bei einer individuenbasierten Simulation tritt dann auf, wenn Interaktionen zwischen beweglichen Individuen berücksichtigt werden müssen, z.B. Schwarmverhalten oder Räuber-Beute-Interaktionen. Das Problem hierbei ist die Berechnung der räumlichen Abstände zwischen den Individuen. In der realen Welt ist der räumliche Abstand physikalisch vorgegeben und die Ermittlung des Abstandes zwischen einem Räuber und einem Beuteorganismus erfolgt durch die Sinnesorgane. Ein Fisch sieht z.B. nur die Daphnien in einem See, die sich direkt in seiner Nähe befinden, die Sichtweite ist von der Trübung des Wassers abhängig.

Bei einer Computersimulation helfen physikalische Gesetze wie Perspektive oder Trübung des Mediums nicht bei der Abstandsberechnung, sondern der Abstand kann nur durch computertechnische

[8] Einige Autoren merken hier an, dass wegen der begrenzten Genauigkeit von Computern auch hier ein Gitter vorhanden sei.

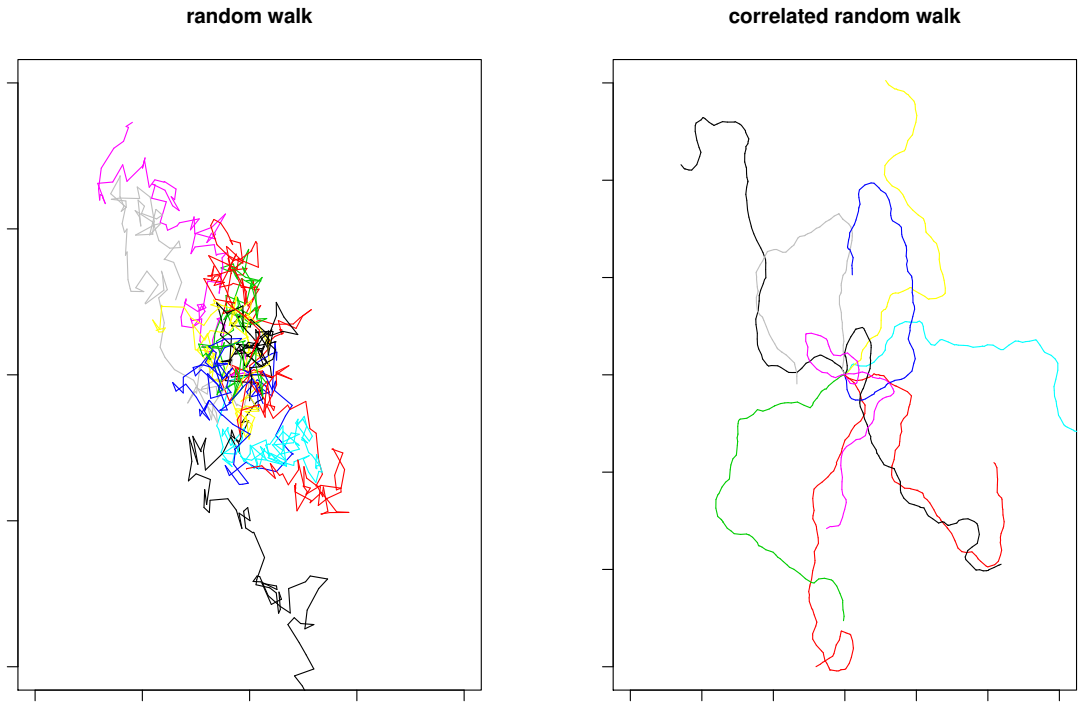

Abbildung 4.5: Jeweils 10 Bewegungsspuren mit ungerichtetem (links) und autokorreliertem Bewegungsmuster (rechts). Alle Spuren beginnen in der Mitte bei der Koordinate $(0,0)$.

(Zählen von Pixeln) oder mathematische Verfahren, z.B. dem Satz des Pythagoras (euklidischer Abstand, in der Ebene $r = \sqrt{\Delta x^2 - \Delta y^2}$), ermittelt werden. Daraus folgt, dass in jedem Simulationsschritt alle Abstände zwischen allen an der Simulation beteiligten Räuberorganismen und allen Beuteorganismen berechnet werden müssen. Bei p Räubern und q Beuteorganismen sind das also $p \cdot q$ bzw. bei $p = q$ sind es p^2 Abstandsberechnungen. Das ist ein Grund dafür, warum derartige Simulationen erst in letzter Zeit so beliebt geworden sind und es fällt sehr leicht, mit solchen Berechnungen aktuelle (und auch zukünftige) Rechner in die Knie zu zwingen, man muss einfach nur die Anzahl der simulierten Individuen erhöhen.

Es wurden unterschiedliche Algorithmen entwickelt, um dieses kombinatorische Problem zu lösen. Eine Möglichkeit zur Erhöhung der Effizienz besteht darin, eine Mischform aus gitterbasierter und freibeweglicher Simulation anzuwenden. Hierdurch steigt zwar der Verwaltungsaufwand, da jedes Individuum zusätzlich auch noch in eine Gitter-Datenstruktur eingetragen werden muss, aber bei der Abstandsberechnung müssen nur noch die Abstände zu den Individuen in den benachbarten Gitterzellen ermittelt werden.

4.4.2 Implementierung in R

Zunächst veranschaulichen wir uns einmal die Grundbewegungsmuster. In einem ersten Beispiel verfolgen wir ein sich gemäß *random walk* bewegendes Individuum (Abb. 4.5, links). Hierbei wird die Schrittweite der Bewegung (`r` im Intervall 0...1) und der Winkel (`alpha` = 0...360) zu jedem Zeitschritt aus einer Gleichverteilung ausgelost:

```
### random walk
plot(c(-10,10),c(-10,10),type="n")
x<-0; y<-0
for(i in 1:100) {
    r     <- runif(1)
    angle <- runif(1)*360
    rad <- angle*pi/180
    dx <- r * sin(rad)
    dy <- r * cos(rad)
    lines(c(x,x+dx), c(y, y+dy))
    x <- x+dx
    y <- y+dy
}
```

Der Unterschied beim *correlated random walk* besteht darin, dass sich der Winkel `alpha` aus dem Winkel des vorhergehenden Zeitschritts und einer zufälligen Drehung ergibt (relative Richtungsänderung). Das mögliche Ausmaß einer Richtungsänderung wird durch den Parameter `turn` und durch die Verteilung (hier Normalverteilung[9]) bestimmt (Abb. 4.5, rechts):

```
### correlated random walk
plot(c(-30,30),c(-30,30),type="n")

x<-0; y<-0
angle <- runif(1)*360
turn  <- 20
for(i in 1:100) {
    r     <- runif(1)
    angle <- angle + rnorm(1)*turn
    rad <- angle*pi/180
    dx <- r * sin(rad)
    dy <- r * cos(rad)
    lines(c(x,x+dx), c(y, y+dy))
    x <- x+dx
    y <- y+dy
}
```

[9] Bei Verwendung einer schiefen Verteilung erhalten wir ein spiralförmigs Suchmuster oder einen „Taumelkäfer“.

Man beachte den geänderten Maßstab der Grafik, da sich bei einer gerichteten Bewegung weitere Strecken ergeben.

4.4.3 Übung

Untersuchen Sie, wie weit die Strecke ist, die Individuen bei rein zufälliger und bei korrelierter Bewegung zurücklegen können. Mit welcher statistischen Maßzahl kann man diese mittlere Entfernung messen?

Lösung

Wie fast immer existieren unterschiedliche Lösungen. Eine Möglichkeit besteht darin, gleich mehrere sich bewegende „Individuen“ zu verfolgen, z.B. in Form eines Vektors, wobei der Bewegungsradius r bei allen Individuen gleich ist und nur der Winkel zufällig gewählt wird. Der zurückgelegte Weg lässt sich dann einfach durch die Varianz messen.

```
par(mfrow=c(1,2))
### multiple UNcorrelated random walk
plot(c(-10,10),c(-10,10),type="n")
n <- 100
x <- rep(0, n)
y <- rep(0, n)
angle <-runif(n)*360
for(i in 1:10) {
    r     <- 1
    angle <- runif(n)*360
    rad <- angle*pi/180
    dx <- r * sin(rad)
    dy <- r * cos(rad)
    for (j in 1:n) {
        lines((c(x[j],x[j]+dx[j])), (c(y[j], y[j]+dy[j])), col=j)
    }
    x <- x+dx
    y <- y+dy
}
print(var(x)); print(var(y))

### multiple correlated random walk
plot(c(-10,10),c(-10,10),type="n")
n <- 100
x <- rep(0, n)
y <- rep(0, n)
angle <-runif(n)*360
turn  <- 20
for(i in 1:10) {
    r     <- 1
    angle <- angle + rnorm(n)*turn
```

```
        rad <- angle*pi/180
        dx <- r * sin(rad)
        dy <- r * cos(rad)
        for (j in 1:n) {
            lines((c(x[j],x[j]+dx[j])), (c(y[j], y[j]+dy[j])), col=j)
        }
        x <- x+dx
        y <- y+dy
    }
    print(var(x)); print(var(y))
```

4.4.4 Weitere Aufgaben

1. Untersuchen Sie unterschiedlich stark korrelierte Bewegungsmuster.
2. Untersuchen Sie die Abhängigkeit zwischen zurückgelegter Entfernung und Anzahl der Simulationsschritte (Zeit). Durch welches physikalische Gesetz könnte man diesen Zusammenhang beschreiben?

4.4.5 Ein Individuenbasiertes Räuber-Beute-Modell

Es soll ein individuenbasiertes Räuber-Beute-Modell mit kontinuierlicher Raumrepräsentation aufgestellt und implementiert werden. Hierbei sollen Räuber und Beute entweder über Wachstums- und Mortalitätsraten wie im klassischen Lotka-Volterra-Modell oder mit Hilfe eines stark vereinfachten „bioenergetischen Prinzips“ interagieren. Die Bewegung der Individuen soll visualisiert und aggregierte Ergebnisse für eine weitere statistische Auswertung auf Festplatte gespeichert werden.

Das Modell vereinigt Grundkonzepte der Populationsdynamik (siehe Modell „Daphnienpopulationsdynamik“) mit einem korrelierten *random walk*. Hinzu kommen neben der Räuber-Beute-Interaktion einige kleinere Details, die hier nur kurz genannt werden sollen. Für eine genauere Analyse ist eine eingehende Beschäftigung mit dem Programmcode und der dort vorhandenen Kommentare unumgänglich.

4.4.6 Implementierung

Die beiden Individuenpopulationen `preds` und `preys` besitzen als Eigenschaften sowohl kartesische Koordinaten (x, y) als auch einen Bewegungsvektor (r, a) die zu Beginn der Simulation zufällig initialisiert werden. Als biologische Eigenschaften sind das Alter und eine Art „bioenergetischer“ Indikator (`energy`) vorhanden. Weitere Eigenschaften sind eine Zahl 1 oder 2 für die „Spezies“, eine Farbe für die Visualisierung und eine eindeutige Identifikationsnummer.

Während der Simulation wird allen neu erzeugten Individuen eine Identifikationsnummer zugewiesen und der Zähler `id` jeweils um die Anzahl der neuen Individuen erhöht. Hierzu verwenden wir in der

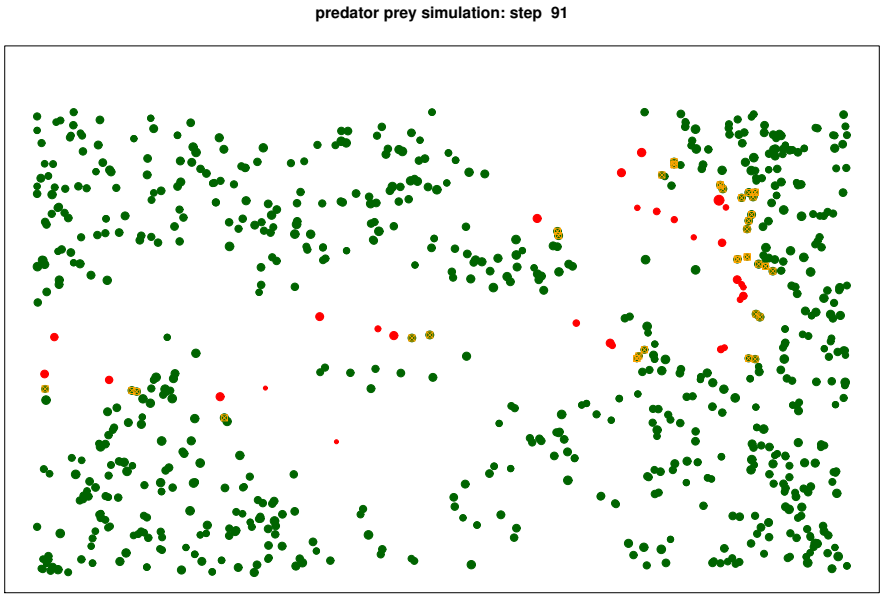

Abbildung 4.6: Individuenbasiertes Räuber-Beute-Modell: räuber: rot, Beute: grün, Interaktionen: gelb.

`generate`-Funktion ausnahmsweise den globalen Zuweisungsoperator (`<<-`), der die *globale* Variable `id` vorbei an der normalen Hierarchie verändert, so dass alle R-Funktionen auf die Variable zugreifen können.

Die `move`-Funktion folgt dem in Abschnitt 4.4.2 vorgestellten Prinzip. Hinzu kommt lediglich ein durchaus üblicher „Trick“ um den Verlust von Individuen per „Emigration“ über den Bildschirmrand zu verhindern: wir lassen die verlorengegangenen Individuen einfach auf der anderen Seite wieder in das Bild hineinwandern.

Die `generate`-Funktion besitzt zwei unterschiedliche Modi, einen (`rate`), bei dem die Vermehrung über eine Wahrscheinlichkeit gesteuert wird und einen zweiten, bei dem sich Individuen vermehren, wenn sie einen gewissen „Energie“-Inhalt überschritten haben. Im Gegensatz zum Daphnienmodell in Abschnitt 4.1.3 wird nicht zwischen Eiablage und Geburt der Nachkommen unterschieden, es entsteht also keine Zeitverzögerung.

Die `grow` Funktion simuliert eine lineare Zunahme des Alters und des Energiegehaltes (bis zu einem Sättigungswert), bei einem negativem Wert von `gain` ist auch ein Verhungern möglich.

Die `survive`-Funktion implementiert eine unspezifische Mortalitätsrate, wobei hier im Unterschied zum Daphienmodell die Überlebenswahrscheinlichkeit angegeben werden muss.

Die Interaktion zwischen Räuber und Beute ist in der `predate`-Funktion implementiert. Die Schleife wird jeweils für alle Räuber durchlaufen. Zunächst berechnet „der Räuber“ die Entfernung zu allen vorhandenen Beuteindividuen mit Hilfe des euklidischen Abstandes. Anschließend werden die Beuteindividuen in eine Teilmenge innerhalb und eine Teilmenge außerhalb der Angriffsdistanz geteilt.

Von den Individuen innerhalb der Angriffsdistanz wird ein Anteil der Energie auf den Prädator übertragen, die Individuen außerhalb der Angriffdistanz überleben.

Im Hauptprogramm werden zunächst einige Simulationsparameter gesetzt (einige Parameter stehen allerdings aus Platzgründen direkt im Programm) und die Startindividuen initialisiert.

In der Simulationsschleife durchlaufen dann alle Beute- und alle Räuber-Individuen nacheinander ihre Lebensfunktionen und werden parallel zur Simulation grafisch dargestellt, wobei der Energiestatus über die Größe der Plotsymbole visualisiert wird `cex=sqrt(inds$energy/5)`.

Alternativ lassen sich die Einzelbilder auch mit den Funktionen `bmp()` und `dev.off()` auf die Festplatte schreiben und nachträglich zu einer Videoanimation zusammenfügen. Für eine spätere Auswertung werden auch Populationsgrößen wie Abundanz und mittlerer „Energiestatus“ auf die Festplatte gespeichert.

```
#=========================================================
# an individual based predator prey interaction game
#=========================================================
euklid <- function(one, inds) {
  sqrt((inds$x-one$x)^2 + (inds$y - one$y)^2)
}
move <- function(inds, asd=0.5) {
  # asd: standard deviation of turning angle
  with(inds, {
      inds$a <- a + asd*rnorm(a)
      dx <- r * cos(a)
      dy <- r * sin(a)
      x<-inds$x + dx
      y<-inds$y + dy
      # wrap around
      x<-ifelse(x>n,0,x)
      y<-ifelse(y>m,0,y)
      inds$x<-ifelse(x<0,n,x)
      inds$y<-ifelse(y<0,m,y)
      inds
  })
}
generate <- function(inds, adult=10, p=0.01,
            mode="rate", div.energy=10, neo.energy=5) {
  # adult: age at which generation is possible
  # p:     probability to generate
  # neo.energy: energy of neonates
  # div.energy: energy at cell division

  if (mode=="rate") {
    newinds <- subset(inds, inds$energy >= adult
                            & runif(length(inds$x)) < p)
    newinds$age <- 0
    newinds$energy <- 1
```

```
  } else {                              # mode == energy
    newinds <- subset(inds, energy > div.energy)
    newinds$energy <- neo.energy
    inds$energy <- ifelse(inds$energy > div.energy,
                    inds$energy - neo.energy, inds$energy)
  }
  n <- length(newinds$age)
  newinds$id <- id0:(id0+n-1)
  id0 <<- id0 + n                       # global assignement
  inds <- rbind(inds, newinds)
}
grow <- function(inds, gain=1, saturation=20) {
  # gain:       assimilated energy per time step
  # saturation: maximum energy of adult
  inds$age <- inds$age + 1
  inds$energy <- pmin(saturation, inds$energy + gain)
  inds
}

survive <- function(inds, p=0.99) {
  # p: survival rate
  inds <- subset(inds, inds$energy > 0 & runif(inds$x) < p)
}

predate <- function(preys, preds, attack=5, assi=0.1) {
  # assi: assimulated part of prey
  # for each in predator
  for (i in 1:length(preds$x)) {
    # compute distance to all preys
    dd <-euklid(list(x=preds[i,]$x,y=preds[i,]$y),
                list(x=preys$x,y=preys$y))
    # kill individuals within attack distance
    killed <- subset(preys, dd <= attack)
    points(killed$x, killed$y, pch=7, col="orange")
    # all others survive
    preys   <- subset(preys, dd > attack)
    # gain energy from killed prey
    preds[i,]$energy <- preds[i,]$energy +
                                sum(killed$energy) * assi
  } # next predator
  # return survived preys and all predators
  list(preds=preds, preys=preys)
}
#=========================================================
# simulation parameters and start individuals
#=========================================================
id0    <- 1   # global Variable (identification number)
n <- m <- 100 # size of simulation area
```

```
nruns  <- 500 # number of simulation runs
nprey  <- 50  # number of preys
npred  <- 10  # number of predators
attack <- 5   # attack distance

preys <- data.frame(x = runif(nprey)*n,
                    y = runif(nprey)*m,
                    r = rep(1, nprey),
                    a = runif(nprey)*2*pi,
                  age = rep(0, nprey),
               energy = runif(nprey)*10,
                 type = rep(1, nprey),
                  col = rep("darkgreen", nprey),
                   id = id0:(id0+nprey-1))
id0 <- id0 + nprey # increment id number

preds <- data.frame(x = runif(npred)*n,
                    y = runif(npred)*m,
                    r = rep(2, npred),
                    a = runif(npred)*2*pi,
                  age = rep(0, npred),
               energy = runif(npred)*10,
                 type = rep(2, npred),
                  col = rep("red", npred),
                   id = id0:(id0+npred-1))
id0 <- id0 + npred # increment id number

#=========================================================
# main loop
#=========================================================
inds <- rbind(preys, preds)
write(paste("i", "npred", "nprey", "en_pred",
            "en_prey",sep="\t"), file="pp.log")

for(i in 1:nruns) {
  ## optional: write image to disc for later animation
  # bmp(filename=paste("pp", i+1000, ".bmp", sep=""))
  par(mar=c(1, 1, 4.1, 1))
  preys <- move(preys, asd  = 0.5)
  preys <- grow(preys, gain =+0.5)
  preys <- generate(preys, mode="energy")
  ## alternative:
  # preys <- generate(preys, adult=10, p=0.05)

  preys <- survive(preys, p=0.99)
  ## alternative with self limitation
  # preys <- survive(preys, p=1-length(preys$x)/50000)
```

```
  nprey <- length(preys$x)
  if (nprey < 1) {print("Extinction of prey"); break()}

  preds <- move(preds, asd  = 0.2)
  preds <- grow(preds, gain =-0.2) # starvation of preds
  preds <- generate(preds, mode="energy")
  preds <- survive(preds, p=1.0)

  npred <- length(preds$x)
  if (npred < 1) {print("Extinction of predator"); break()}

  # Visualization ========================================
  inds  <- rbind(preds, preys)
  plot(inds$x, inds$y, col=as.vector(inds$col),
       pch=16,
       cex=sqrt(inds$energy/5),
       xlim=c(0, n), ylim=c(0, m * 1.1),
       axes=FALSE, xlab="", ylab="")
  box()
  title(main=paste("predator-prey-simulation: step=",i))
  text(30,105,paste(nprey, "preys"), col="darkgreen")
  text(70,105,paste(npred, "predators"), col="red")
  # write some results to disc ===========================
  write(paste(i, npred, nprey, mean(preds$energy),
              mean(preys$energy), sep="\t"),
              file="pp.log", append=TRUE)
  # print some results to screen
  print(paste(i, npred, nprey, mean(preds$energy),
              mean(preys$energy), sep="\t"))
  # Predation ============================================
  temp  <- predate(preys, preds, attack=attack)
  preys <- temp$preys
  preds <- temp$preds
  #dev.off() # close external image file
}

#=== Evaluation of results  ============================
# xx <- read.table("pp.log", sep="\t", head=TRUE)
# plotting, statistics, ...
```

4.4.7 Weitere Übungen

- Untersuchen Sie das Räuber-Beute-Modell hinsichtlich des Aussterbens von Räuber- oder Beutepopulation. Versuchen Sie, möglichst stabile Zustände (Zyklen oder Gleichgewicht) zu erzielen. Modifizieren sie hierzu die Anzahl der Simulationsschritte, die Startabundanzen, Überlebenswahrscheinlichkeiten, bioenergetische Gewinn- bzw. Verlustparameter und Angriffsdistanz.

- Speichern Sie die Grafiken auf die Festplatte, erzeugen sie Animationen und untersuchen Sie die Populationsdynamik (Zeitreihe der Abundanz).
- Führen Sie einen Selbstlimitationsterm für Räuber oder Beute (auskommentierte Alternative) ein oder experimentieren Sie mit anderen stabilisierenden Elementen, z.B. Einführung einer maximalen Anzahl der von einem Prädator pro Zeitschritt gefressenen Beuteindividuen oder Zeitverzögerung zwischen Anlage und Geburt von Nachkommen.
- Führen Sie eine räumlich heterogene Umwelt ein (in Form eines Gitters), z.B. ein rechteckiges Refugium, d.h. einen Bereich, in dem die Räuber ihre Beute nicht „sehen“ können.

Eine ganz pragmatische Lösung für ein Refugium besteht darin, die Abstandsberechnung zu modifizieren, z.B. indem der Abstand zu allen Beuteindividuen in einem geschützten Bereich (hier für $40 < x < 60$ und $40 < y < 60$) auf einen Wert oberhalb der Angriffsdistanz (hier 1000) gesetzt wird:

```
euklid <- function(one, inds) {
  dd<-ifelse (inds$x > 40 & inds$x < 60
            & inds$y > 40 & inds$y < 60, 1000,
  sqrt((inds$x-one$x)^2 + (inds$y - one$y)^2))
  dd <- as.numeric(as.vector(dd))
}
```

5 Ausblick

Wir haben nun einige grundlegende Modellansätze und Techniken kennengelernt oder kurz gestreift und sollten nun in der Lage sein, die in jüngerer Zeit zahlreich erschienenen Bücher zur Populationsökologie, Modellierung und Simulation und viele der Originalpublikationen etwas besser zu verstehen.

Es sollte auch nicht besonders schwer fallen, für eigene Fragestellungen einfache Modelle aufzustellen oder vorhandene Modelle zu adaptieren. Wie wir hoffentlich gesehen haben, ist der Einstieg in die Welt der mathematischen Modelle nicht mit einer unüberwindbaren Hürde versehen, sondern durch eine kontinuierliche Zunahme an Umfang, Komplexität und mathematischer und technischer Schwierigkeiten gekennzeichnet. Diesen Weg kann man als Biologe durchaus einige Schritte allein beschreiten und die eigenen Grenzen ein Stück weit hinausschieben. Bei der unbedingt notwendigen Kommunikation mit Kollegen und Wissenschaftlern anderer Disziplinen kommt uns dies zu Gute. Je klarer unsere Modelle und Gedanken strukturiert sind, umso besser und tiefgründiger wird die Kommunikation.

Wie bei fast aller wissenschaftlichen Arbeit, ist das Vorhandensein eines sinnvollen Konzeptes ganz wesentlich. Es ist nach meiner Meinung sinnlos, erst am Ende einer experimentellen Arbeit über die Anwendung eines Modells nachzudenken und zu versuchen, irgendeinen Modellansatz „auf die Daten anzuwenden“. Bei der Anwendung ist darüberhinaus nicht das Computerprogramm wesentlich (Implementationsperspektive) sondern das Modell (Spezifikationsperspektive).

Bei einer guten experimentellen Arbeit steht oft eine Hypothese am Anfang (siehe z.B. das Einführungskapitel von LAMPERT and SOMMER, 1993). Warum sollte man aus einer Arbeitshypothese nicht auch gleich am Anfang ein *konzeptionelles* Modell entwickeln? Das Modell muss ja nicht unbedingt zum Selbstzweck oder zum Hauptziel der Arbeit werden, aber wir können ja die Rückkopplungschleife der Modellmethodik nutzen, als „Ideengenerator“ (Abschnitt 1.5.2).

Modelle sind wegen ihrer Abstraktion auch trotz einer gewissen Fachspezifik eine universellere Sprache als experimentelle Teilergebnisse. Hier kommt, scherzhaft gesagt, auch ein Hauptnachteil der Modelle zum tragen: Es ist möglich und auch Pflicht, alle Details lückenlos zu dokumentieren, so dass Wissenslücken, Vereinfachungen und Fehler gnadenlos sichtbar werden, besonders wenn man sich mit realen biologischen Systemen beschäftigt. Bei manchen „Modellierern“ hat das dazu geführt, sich nur noch mit „sauberen“ unangreifbaren und manchmal realitätsfernen Gleichungen zu befassen und sich verächtlich über die „Empiriker“ zu erheben.

Ich schlage einen anderen Weg vor: Wir könnten versuchen, mutig bei der Formulierung und Offenlegung unserer Ansätze zu sein, fachlich diskussionsfreudig und menschlich tolerant gegenüber Fehlern und Lücken in anderen Modellen und uns bemühen, die theoretischen Ansätze aus der Welt der Gleichungen und Mathematik ein wenig besser zu verstehen.

A Texteditoren für R

Kurze Befehlszeilen können selbstverständlich direkt in die R-Eingabeaufforderung eingetippt werden. Bei etwas größeren Befehlsfolgen oder auch zum Ansehen größerer Daten-Dateien ist es allerdings angeraten, einen Text-Editor zu benutzen. Hierzu können schon einfache Editoren, zum Beispiel der Windows-Notizblock-Editor verwendet werden. Wichtig ist, dass der benutzte Editor das „ASCII"-Format oder „Nur Text"-Format benutzt.

Für Windows können je nach persönlichen Anforderungen auch andere Editoren benutzt werden, zum Beispiel der Programmers File Editor (PFE) von Alan Phillips. Er liest im Gegensatz zum Notizblock auch große Dateien, bietet zuschaltbare Zeilennummern, Suchen und Ersetzen von Steuerzeichen, einen Makrorecorder und einigen anderen Komfort. PFE wird zwar nicht mehr weiterentwickelt, aber er ist Freeware und kann aus dem Internet[1] heruntergeladen werden.

Bei der Arbeit mit einem einfachen Editor (Notizblock, PFE) kopiert man entweder die gewünschten R-Zeilen über die Zwischenablage in R oder man speichert die Befehle in eine Datei (z.B. test.R) und startet dieses sogenannte „Skript" dann aus R unter Windows über das Menü „File - Source R Code" oder von der R-Eingabeaufforderung mit Hilfe von:

```
> source("test.R")
```

Ein sehr komfortabler, speziell auf R zugeschnittener Editor ist Tinn-R[2] von José Cláudio Faria. Er steht wie R unter der GNU General Public License und erlaubt u.a. die farbige Hervorhebung der R-Syntax, die direkte Kommunikation zwischen dem Editor und R, die zeilenweise Ausführung von R-Code oder den direkten Zugriff auf die Online-Hilfe von R.

Eine weitere Möglichkeit ist der Shareware-Editor WinEdt[3]. Dieser wurde eigentlich für das Textsatzsystem LaTeX entwickelt, kann aber auch für verschiedene andere Programmiersprachen, z.B. C++ oder in unserem Falle R konfiguriert werden.

Unter Linux und Unix gehören leistungsfähige Texteditoren normalerweise zur Grundinstallation, z.B. pico, kedit oder der komfortable Nirwana Editor (nedit).

Darüberhinaus wird vom R-Core-Team die Benutzung des für Linux, Unix, Macintosh und auch für Windows verfügbaren Editors Emacs empfohlen. Es existieren zwei Entwicklungszweige, der Original-Emacs[4] und der XEmacs[5], die beide als OpenSource unter der GNU Public License erhältlich sind.

[1] http://www.lancs.ac.uk/staff/steveb/cpaap/pfe/
[2] http://www.sciviews.org/Tinn-R/
[3] http://www.winedt.com
[4] http://www.gnu.org/software/emacs/emacs.html
[5] http://www.xemacs.org

Grundsätzlich blicken die Versionen des Emacs auf eine sehr lange Entwicklungsgeschichte zurück und sind extrem leistungsfähig und vielfältig konfigurierbar, z.B. für Fortran, C++, LaTeX, HTML oder mit Hilfe des Paketes ESS (Emacs speaks statistics) auch für R. Die Bedienung erfolgt normalerweise mit Hilfe einer unüberschaubaren Fülle an Tastenkombinationen die in viel dickeren Büchern als diesem hier beschrieben sind, es stehen aber auch Menüs zur Verfügung. Die Einarbeitung in Emacs ist selbst für fortgeschrittene Windows-Benutzer zunächst soetwas wie ein Kulturschock, dafür geht die Arbeit dann aber umso schneller von der Hand, es lohnt sich also durchaus.

B Daten des World Radiation Data Centre

Das WRDC[1] bietet Online-Zugriff auf Strahlungsdaten (Globalstrahlung, diffuse Strahlung, Sonnenstrahlung) von mehr als 1000 Messstationen über den Zeitraum von 1964 bis 1993. Der Zugriff ist öffentlich und erfolgt entweder über ein Web-Interface oder automatisiert über Email. Der resultierende Datensatz kann über das untenstehende R-Script in einen Tabelle umgewandelt werden.

Zum Download der Daten kann die folgende Vorgehensweise gewählt werden:

1. Gehen Sie auf die Homepage des WRDC, lesen Sie dort die Einführung (About) und navigieren Sie zum Download-Bereich (View Data).
2. Wählen Sie im Download-Formular die Messgröße (Global), die Code-Nummer der Messstation (z.B. Dresden Wahnsdorf = 094860 oder Egypt Aswan = 624140) und den Zeitraum (1981 bis 1990).
3. Fordern Sie die Daten mit „Submit“ an.
4. Kopieren Sie die angezeigten Daten in einen Editor, z.B. den PFE und speichern Sie diese im Textformat (ASCII) ab, z.B. als `igl_raw.txt`. Unter Umständen kann es notwendig sein, vorher einige HTML-Steuercodes manuell zu entfernen.
5. Mit Hilfe des R-Programmes `ConvertWrdc` kann der Datensatz so konvertiert werden, dass ein mit Hilfe von `read.table` als Dataframe einlesbares Format entsteht. Darüberhinaus ersetzt das Programm fehlende Daten durch linear interpolierte Werte.

[1] `http://wrdc-mgo.nrel.gov/`

```
#########################################################################
# Program: convertWrdc
# Purpose: conversion of WRDC data format 2 into a dataframe
#          and linear interpolation of missing values
# License: GPL 2.0 or above
# petzoldt@rcs.urz.tu-dresden.de
#########################################################################
library(date)

fin  <- "igl_raw.txt"
fout <- "igl8190.dat"
dat  <- readLines(fin)

## Eliminate header lines and monthly means
## good codes   : 02, 03, 04; wrong codes  : 01, ""
dat    <- dat[substr(dat,79,80)>"01"]
newdat <- matrix(0, nrow=length(dat)*11, ncol=5)

## Analysis of the data structure
j <- 0
for (i in 1:length(dat)) {
  year  <- as.numeric(substr(dat[i],  8,  9))
  month <- as.numeric(substr(dat[i], 10, 11))
  ord   <- as.numeric(substr(dat[i], 79, 80))
  k <- 14
  for (d in 1:10) {
    igl  <- as.numeric(substr(dat[i], k, k + 3))
    flag <- as.numeric(substr(dat[i], k + 4, k + 4))
    k <- k + 5
    j <- j+1
    newdat[j,] <- c(year, month, d + (ord-2) * 10 , igl, flag)
  }
  # day number 31 in last line of month with code==4
  if (ord == 4) {
    igl <- as.numeric(substr(dat[i], k, k + 3))
    flag <- as.numeric(substr(dat[i], k + 4, k + 4))
    k <- k + 5
    j <- j+1
    newdat[j,] <- c(year, month, 31 , igl, flag)
  }
}
newdat <-as.data.frame(newdat)
names(newdat) <- c("year", "month", "day", "igl", "flag")

## Subset: years 1981 - 1990
newdat <- subset(newdat, newdat$year > 80 & newdat$year < 91)
```

```
newdat$year <- newdat$year + 1900 # make complete year numbers 84 -> 1984
newdat$date <- as.date(paste(newdat$month, newdat$day, newdat$year,sep="/"))

## Elimination of nonsense dates (e.g. february 31st)
newdat <- subset(newdat, !is.na(newdat$date))

## interpolation of missing values (code != NA)
missing.date <- newdat$date[!is.na(newdat$flag)]
not.missing  <- which(is.na(newdat$flag))
missing      <- which(!is.na(newdat$flag))
newdat$igl[missing] <- round(approx(newdat$date[not.missing],
                            newdat$igl[not.missing], missing.date)$y)
newdat$flag <- ifelse(is.na(newdat$flag),0,1)

## preparation and output of results
outdat <- data.frame(
            date=paste(newdat$day, newdat$month, newdat$year, sep="."),
            igl=newdat$igl,
            interpolated = newdat$flag
          )
write.table(file=fout, outdat, row.names=FALSE)
```

C Subroutinen des Zellulären Automaten in C

Manche Dinge lassen sich in R nicht effizient genug implementieren. Zur Beschleunigung können solche Programmteile in C oder in Fortran geschrieben und eingebunden werden. Ein Beispiel hierfür ist die Nachbarschaftsberechnung im zellulären Automaten oder der Seedfill-Algorithmus zum Ausfüllen von Farbflächen in einer Pixelgrafik. Die Alpha-Version des R-Package `simecol` erhält zusätzlich eine erweiterte Funktion zur Nachbarschaftsberechnung, die mit Hilfe von in einer Nachbarschaftsmatrix enthaltenen Gewichten auch entferntere Nachbarn berücksichtigt und stochastische gitterbasierte Modelle ermöglicht. Die Funktionsdokumentation (Online-Hilfe) enthält ein entsprechendes Beispiel.

```
#include <R.h>
#include <Rinternals.h>

int is_inside(int n, int m, int i, int j, double* x) {
    int ret;
    if ((0 <= i) & (i < n) & (0 <= j) & (j < m))  ret=1; else ret=0;
    return ret;
}
double getpixel(int n, int m, int i, int j, double* x) {
    if (is_inside(n, m, i, j, x) > 0) {
      return x[i + n * j];
    } else {
      return 0;
    }
}
void setpixel(int n, int m, int i, int j, double* x, double* fcol)
{
    if (is_inside(n, m, i, j, x) > 0) {
       x[i + n * j] = *fcol;
    }
}
void seedfill(int* n, int* m, int* i, int* j, double* x, double*
fcol, double* bcol) {
  // fcol: fill color, bcol: boundary color
  int nn= *n, mm= *m,
      ii= *i, jj= *j;
  double fill_col=*fcol, boundary_col=*bcol;

    if (is_inside(nn, mm, ii, jj, x) > 0) {
      double current_col=getpixel(nn, mm, ii, jj, x);
```

```
      if( current_col!=fill_col &&
      current_col!=boundary_col ) {
          setpixel(nn, mm, ii, jj, x, fcol);
          ii=*i+1; seedfill(n, m,  &ii,   j,   x, fcol, bcol);
          jj=*j+1; seedfill(n, m,    i,   &jj, x, fcol, bcol);
          ii=*i-1; seedfill(n, m,  &ii,   j,   x, fcol, bcol);
          jj=*j-1; seedfill(n, m,    i,   &jj, x, fcol, bcol);
      }
    }
}
void eightneighbours(int* n, int* m, double* x, double* y) {
  int nn= *n, mm= *m;
  double c=0;
  for (int i=0; i < nn; i++) {
      for (int j=0; j < mm; j++) {
        c = getpixel(nn, mm, i+1, j,   x) +
            getpixel(nn, mm, i,   j+1, x) +
            getpixel(nn, mm, i-1, j,   x) +
            getpixel(nn, mm, i,   j-1, x) +
            getpixel(nn, mm, i+1, j+1, x) +
            getpixel(nn, mm, i+1, j-1, x) +
            getpixel(nn, mm, i-1, j+1, x) +
            getpixel(nn, mm, i-1, j-1, x);
        setpixel(nn, mm, i, j, y, &c);
      }
  }
}
void fourneighbours(int* n, int* m, double* x, double* y) {
  int nn= *n, mm= *m;
  double c=0;
  for (int i=0; i < nn; i++) {
      for (int j=0; j < mm; j++) {
        c = getpixel(nn, mm, i+1, j,   x) +
            getpixel(nn, mm, i,   j+1, x) +
            getpixel(nn, mm, i-1, j,   x) +
            getpixel(nn, mm, i,   j-1, x);
        setpixel(nn, mm, i, j, y, &c);
      }
  }
}
```

Literaturverzeichnis

AMERICAN PUBLIC HEALTH ASSOCIATION, 1992: Standard Methods for the Examination of Water and Wastewater. American Public Health Association, Inc., 18th edition.

ATV-ARBEITSGRUPPE 2.2.3, 1997: Einführung des ATV-Gewässergütemodells. Korrespondenz Abwasser **44**: 2058–2061.

BAILLIEUL, M. and R. BLUST, 1999: Analysis of the swimming velocity of cadmium-stressed *Daphnia magna*. Aquatic Toxicology **44**: 245–254.

BATES, D. M. and D. G. WATTS, 1988: Nonlinear Regression and its Applications. Wiley and Sons, New York.

BAUMERT, H. and D. UHLMANN, 1983: Theory of the upper limit to phytoplankton production per unit area in natural waters. Int. Rev. Ges. Hydrobiol. **68**: 753–783.

BEHRENDT, H. and B. NIXDORF, 1993: The carbon balance of phytoplankton production and loss processes based on *in situ* measurements in a shallow lake. Int. Rev. Ges. Hydrobiol. **78**: 439–458.

BENNDORF, J., 1979: Kausalanalyse, theoretische Synthese und Simulation des Eutrophierungsprozesses in stehenden und gestauten Gewässern. Habilitationsschrift, TU Dresden, Fakultät Bau-, Wasser- und Forstwesen.

BENNDORF, J., 1985: Dynamiczne modele ekologiczne jako pomoc przy podejmowaniu decyzji w sterowaniu eutrofizacja. Wladomosci Ecologiczne **31**: 335–349.

BENNDORF, J., 1992: The control of indirect effects of biomanipulation. In: SUTCLIFFE, D. W. and J. G. JONES, eds., *Eutrophication: Research and Application to Water Supply*, 82–93. Freshwater Biological Association.

BENNDORF, J. and H. BAUMERT, 1981: Modellvorstellungen zum Einfluß der Durchmischung des Wasserkörpers auf die phytoplanktische Primärproduktion. In: UNGER, K. and G. STÖCKER, eds., *Biophysikalische Ökologie und Ökosystemforschung*, 333–345. Akademie-Verlag, Berlin.

BENNDORF, J., R. KOSCHEL and F. RECKNAGEL, 1985: The pelagic zone of Lake Stechlin. an approach to a theoretical model. In: CASPER, S., ed., *Lake Stechlin. A Temperate Oligotrophic Lake.*, 443–453. Junk Publishers.

BENNDORF, J. and T. PETZOLDT, 1999: Der Einsatz eines ökologischen Modells zur Prognose der Wasserbeschaffenheit von Trinkwassertalsperren und die Wechselbeziehungen zur Wassermengenbewirtschaftung. In: *Trinkwasserversorgung aus Talsperren. ATT Schriftenreihe Band 1*, 421–438. Oldenbourg Industrieverlag, München.

BENNDORF, J. and F. RECKNAGEL, 1982: Problems of application of the ecological model SALMO to lakes and reservoirs having various trophic states. Ecological Modelling **17**: 129–145.

BOSSEL, H., 1994: Modellbildung und Simulation: Konzepte, Verfahren und Modelle zum Verhalten dynamischer Systeme; ein Lehr- und Arbeitsbuch. Vieweg Verlag, Braunschweig, second edition.

BOTTRELL, H. H., A. DUNCAN, Z. M. GLIWICZ, E. GRYGIEREK, A. HERZIG, A. HILLBRICHT-ILKOWSKA, H. KURASAWA, P. LARSSON and T. WEGLENSKA, 1976: A review of some problems in zooplankton production studies. Norvegian Journal of Zoology **24**: 419–456.

BOX, G. E., G. M. JENKINS and G. C. REINSEL, 1994: Time Series Analysis: Forecasting and Control. Prentice Hall Inc., Englewood Cliffs, NJ, third edition.

BRUN, R., P. REICHERT and H. KÜNSCH, 2001: Practical identifiability analysis of large environmental simulation models. Water Resources Research **37**: 1015–1030.

CASWELL, H. and R. ETTER, 1999: Cellular automaton models for competition in patchy environments: Facilation, inhibitation, and tolerance. Bulletin of Mathematical Biology **61**: 625–649.

CHATFIELD, C., 1982: Analyse von Zeitreihen. Eine Einführung. Teubner Verlagsgesellschaft, Leipzig.

CHEN, C., R. JI, D. J. SCHWAB, D. BELETSKY, G. L. FAHNENSTIEL, M. JIANG, T. H. JOHENGEN, H. VANDERPLOEG, B. EADIE, J. W. BUDD, M. H. BUNDY, W. GARDNER, J. COTNER and P. J. LAVRENTYEV, 2002a: A model study of the coupled biological and physical dynamics in Lake Michigan. Ecol. Model. **152**: 145–168.

CHEN, Q., A. E. MYNETT and A. W. WINNS, 2002b: Application of cellular automata to modelling competitive growths of two underwater species *Chara aspera* and *Potamogeton pectinatus* in Lake Veluwe. Ecological Modelling **147**: 253–265.

CONNELL, J., 1978: Diversity in tropical rain forrests and coral reefs. Science **199**: 1304–1310.

CRAWLEY, M. J., 2002: Statistical Computing. An Introduction to Data Analysis Using S-PLUS. Wiley, Chichester.

DEANGELIS, D. L. and L. J. GROSS, eds., 1992: Individual-Based Models and Approaches in Ecology: Populations, Communities, and Ecosystems., Knoxwille, Tennessee. Chapmann and Hall. Proceedings of a Symposium/Workshop.

DROOP, M., 1974: The nutrient status of algal cells in continuous culture. J. Mar. Biol. Assoc. United Kingdom **54**: 825–855.

DROOP, M., 1983: 25 years of algal growth kinetics. Bot. Mar. **26**: 99–112.

ELLIOTT, J. A., A. E. IRISCH, C. S. REYNOLDS and P. TETT, 2000: Modelling freshwater phytoplankton communities: An exercise in validation. Ecol. Model. **128**: 19–26.

ENGELN-MÜLLGES, G. and F. REUTTER, 1996: Numerik Algorithmen. VDI Verlag, Düsseldorf, 8th edition.

EVERITT, B., 1987: Introduction to Optimization Methods and their Application in Statistics. Chapmann and Hall, London.

FOFONOFF, R. C., N. P. ANDMILLARD, 1983: Algorithms for computation of fundamental properties of seawater. UNESCO technical papers in marine science **44**: 1–53.

FORRESTER, J., 1961: Industrial Dynamics. MIT Press, Massachusetts.

FORRESTER, J., 1971: World Dynamics. MIT Press, Massachusetts.

FOWLER, M. and K. SCOTT, 1997: UML Distilled. Applying the Standard Object Modelling Language. Addison Wesley.

GAMITO, S., 1998: Growth models and their use in ecological modelling: An application to a fish population. Ecol. Model. **113**: 83–94.

GAYNOR, P. E. and R. C. KIRKPATRICK, 1994: Introduction to Time-Series Modeling in Business and Economics. McGraw-Hill, New York.

GNAUCK, A., E. MATTHÄUS, M. STRAŠKRABA and I. AFFA, 1990: The use of SONCHES for aquatic ecosystem modelling. Syst. Anal. Model. Simul. **7**: 439–458.

GODFREY, K., 1983: Compartment Models and their Application. Academic Press, New York.

GÖHLER, W., 1987: Höhere Mathematik. Formeln und Hinweise. Leipzig: Dt. Verl für Grundstoffind 128 S.

GOMPERTZ, B., 1825: On the nature of the function expressive of the law of human mortality, and a new mode of determining the value of live contingencies. Phil. Trans. Roy. Soc. **182**: 513–585.

GRIMM, V., 1999: Ten years of individual-based modelling in ecology: What have we learned and what could we learn in future? Ecol. Model. **115**: 129–148.

GRIMM, V., 2001: Den Wald vor lauter Bäumen sehen: musterorientiertes Modellieren in der Ökologie. In: JOPP, F. and G. WEIGMANN, eds., *Rolle und Bedeutung von Modellen für den ökologischen Erkenntnisprozeß.*, Theorie in der Ökologie Band 4., 43–57. Peter Lang, Frankfurt/M.

GRIMM, V. and C. WISSEL, 1997: Babel, or the ecological stability discussions: An inventory and analysis of terminology, and a guide for avoiding confusion. Oecologia **109**: 323–334.

GROSS, L. J., K. A. ROSE, E. J. RYKIEL, W. VAN WINKLE and E. E. WERNER, 1992: Individual-based modeling: Summary of a workshop. In: DEANGELIS, D. L. and L. J. GROSS, eds., *Individual-Based Models and Approaches in Ecology: Populations, Communities, and Ecosystems.*, 213–521.

GURNEY, W. S. C. and R. M. NISBET, 1998: Ecological Dynamics. Oxford University Press, New York, Oxford.

HÅKANSON, L. and R. H. PETERS, 1995: Predictive Limnology: methods for predictive modelling. SPB Academic Publishing, Amsterdam.

HAMILTON, D. P. and S. G. SCHLADOW, 1997: Prediction of water quality in lakes and reservoirs. Part I – Model description. Ecol. Model. **96**: 91–110.

HASTINGS, A., 1996: Population Biology. Concepts and Models. Springer, New York.

HINDMARSH, A. C., 1983: ODEPACK, a systematized collection of ODE solvers. In: STEPLEMAN, R., ed., *Scientific Computing*, 55–64. IMACS / North-Holland.

HÖLKER, F., 1999: Bioenergetik dominanter Fischarten (*Abramis brama* L. und *Rutilus rutilus* L.) in einem eutrophen See Schleswig-Holsteins – Ökophysiologie und individuenbasierte Modellierung –. Dissertation, Universität Hamburg, Fachbereich Biologie, Hamburg.

HSU, S.-B., T.-W. HWANG and Y. KUANG, 2001: Global analysis of the Michaelis–Menten-type ratio-dependend predator-prey system. J. Math. Biol. **42**: 489–506.

HUISMAN, J. and F. J. WEISSING, 2002: Oscillations and chaos generated by competition for interactively essential resources. Ecological Research **17**: 175–181.

HÜLSMANN, S., 2000: Population Dynamics of *Daphnia galeata* in the Biomanipulated Bautzen Reservoir: Life History Strategies Against Food Deficiency and Predation. Ph.D. thesis, TU Dresden Fakultät Forst-, Geo-, Hydrowissenschaften, Institut für Hydrobiologie.

HÜLSMANN, S. and W. WEILER, 2000: Adult, not juvenile mortality as a major reason for the mid-summer decline of a *Daphnia* population. Journal of Plankton Research **22**: 151–168.

HUSE, G., J. GISKE, A. GRO and V. SALVANES, 2002: Individual based modelling. In: HART, P. and J. REYNOLDS, eds., *Fish and Fisheries Handbook: The Biology, Conservation and Management of Exploited Species*, volume 2, chapter 11. Blackwell Science, Oxford.

HUTCHINSON, G. E., 1961: The paradox of the plankton. American Naturalist **95**: 137–146.

IHAKA, R. and R. GENTLEMAN, 1996: R: A language for data analysis and graphics. Journal of Computational and Graphical Statistics **5**: 299–314.

JÄHNICHEN, S., T. PETZOLDT and J. BENNDORF, 2001: Evidence for control of microcystin dynamics in Bautzen Reservoir (Germany) by cyanobacterial population growth rates and dissolved inorganic carbon. Archiv für Hydrobiologie **150**: 177–196.

JI, R., C. CHEN, J. W. BUDD, D. J. SCHWAB, D. BELETSKY, G. L. FAHNENSTIEL, T. H. JOHENGEN, H. VANDERPLOEG, B. EADIE, J. COTNER, W. GARDNER and M. BUNDY, 2002: Influences of suspended sediments on the ecosystem in Lake Michigan: A 3D coupled bio-physical modeling experiment. Ecol. Model. **152**: 169–190.

KIRCHESCH, V., 1992: Ein gütemodell zur simulation und prognose des stoffhaushaltes von fließgewässern. In: *Tagungsband 4. Magd. Gewässerschutzseminar, 22.-26.9.1992*, volume 357. Spindlermühle.

KÖHLER, W., G. SCHACHTEL and P. VOLESKE, 2002: Biostatistik. Eine Einführung in die Biometrie für Biologen und Agrarwissenschaftler. Springer-Verlag, Berlin, Heidelberg, second edition.

KOOIJMAN, S. A. L. M., 1995: The stoichiometry of animal energetics. Journal of Theoretical Biology **177**: 139–149.

KOOIJMAN, S. A. L. M., 2000: Dynamic Energy and Mass Budgets in Biological Systems. Cambridge University Press, Cambridge.

KOOIJMAN, S. A. L. M., 2001: Quantitative aspects of metabolic organization: A discussion of concepts. Philos. Trans. R. Soc. Lond., B **356**: 331–349.

KOROBOZOWOI, E. W., 1974: Symposium zu Problemen der ökologischen Physiologie von Süßwasserinvertebraten (in Russisch). Gidrobiol. Zh. **10**: 127–130.

LAMPERT, W. and U. SOMMER, 1993: Limnoökologie. Georg Thieme Verlag, Stuttgart.

LE PAGE, C. and P. CURY, 1997: Population viability and spatial fish reproductive strategies in constant and changing environments: an individual-based modelling approach. Canadian Journal of Fisheries and Aquatic Sciences **54**: 2235–2246.

LEGGETT, R. and L. WILLIAMS, 1981: A reliability index for models. Ecol. Model. **13**: 303–312.

LESLIE, P. H., 1945: On the use of matrices in certain population mathematics. Biometrica **33**: 183–212.

MAMEDOV, A. and S. UDALOV, 2002: A computer tool to develop individual-based models for simulation of population interactions. Ecol. Model. **147**: 53–68.

MCCANN, K. S., 2000: The diversity-stability debate. Nature **405**: 228–233.

MCCAULEY, E., W. W. MURDOCH and R. M. NISBET, 1990a: Growth, reproduction, and mortality of daphnia pulex leydig: life at low food. Functional Ecology **4**: 505–514.

MCCAULEY, E., W. W. MURDOCH, R. M. NISBET and W. S. GURNEY, 1990b: The physiological ecology of daphnia: development of a model of growth and reproduction. Ecology **71**: 703–715.

MIDDELHOFF, U., 1998: An object-oriented model developing competing root systems of black aldertrees. ASU Newsletter **24, Supplement**: 65–74.

MILLER, T. and W. KERFOOT, 1987: Redefining indirect effects. In: KERFOOT, W. and A. SIH, eds., *Predation: Direct and Indirect Impacts on Aquatic Communities.*, 33–37. University Press of New England, Hanover and London.

MOOIJ, W. M. and M. BOERSMA, 1996: An object oriented simulation framework for individual-based simulations (OSIRIS): *Daphnia* population dynamics as an example. Ecol. Model. **93**: 139–153.

NISBET, R. M., E. B. MULLER, K. LIKA and S. A. L. M. KOOIJMAN, 2000: From molecules to ecosystems through dynamic energy budget models. Journal of Animal Ecology **69**: 913–926.

OMLIN, M., R. BRUN and P. REICHERT, 2001: Biogeochemical model of Lake Zürich: Sensitivity, identifiability and uncertainty analysis. Ecol. Model. **141**: 105–123.

PATTEN, B., 1983: On the quantitative dominance of indirect effects in ecosystems. In: Analysis of ecological systems: State of the art in ecological modeling. Lauenroth, W.K. Skogerboe, G.V. Flug, M. (eds.). Amsterdam, Oxford, New York:Elsevier 27–37.

PETZOLD, L. R., 1983: Automatic selection of methods for solving stiff and nonstiff systems of ordinary differential equations. Siam J. Sci. Stat. Comput. **4**: 136–148.

PETZOLDT, T., 1996: Möglichkeiten zur Vorhersage von Phytoplanktonmassenentwicklungen: Von der statischen Betrachtungsweise zur Kurzfristprognose. Dissertation, TU Dresden, Fakultät Forst-, Geo- und Hydrowissenschaften.

PETZOLDT, T. and H. HORN, 1997: Sind Algenmassenentwicklungen in Talsperren kurzfristig vorhersagbar? In: *Technische Universität Ilmenau, Proceedings des 42. Internationalen wissenschaftlichen Kolloquiums: Informatik und Automatisierung im Zeitalter der Informationsgesellschaft, 22. – 25.09. 1997*, 261–266. ISSN 0943-7207.

PICHÉ, R., 2000: Teaching unit 2: Numerical differential equation solvers. http://iac.tut.fi/ piche/.

PRESS, W. H., S. A. TEUKOLSKY, W. T. VETTERLING and B. P. FLANNERY, 1992: Numerical Recipes in C. Cambridge University Press, second edition. http://www.nr.com/.

PURVIS, A. and A. HECTOR, 2000: Getting the measure of biodiversity. Nature **405**: 212–219.

R DEVELOPMENT CORE TEAM, 2001: R Data Import/Export, Version 1.4.0, 1–34. www.r-project.org.

REICHERT, P., D. BORCHARDT, M. HENZE, W. RAUCH, P. SHANAHAN, L. SOMLYÓDY and P. VANROLLEGHEM, 2001: River water quality model no. 1 (RWQM1): II. Biochemical process equations. Water Sci. Technol. **43**: 11–30.

RINKE, K. and T. PETZOLDT, 2001: Vertikalwanderung von Daphnien: Ein Modell der Proximatfaktoren. In: *Tagungsbericht der DGL, Magdeburg 2000*, 618–622. Tutzing.

ROSE, K. A., S. W. CHRISTENSEN and D. L. DEANGELIS, 1993: Individual-based modeling of populations with high mortality: A new method based on following a fixed number of model individuals. Ecol. Model. **68**: 273–292.

ROUGHGARDEN, J., 1998: Primer of Ecological Theory. Prentice Hall.

SACHS, L., 1992: Angewandte Statistik. Springer-Verlag, Berlin, 7th edition.

SCAVIA, D. and R. A. PARK, 1976: Documentation of selected constructs and parameter values in the aquatic model CLEANER. Ecol. Model. **2**: 33–58.

SCHEFFER, M., J. BAVECO, D. DEANGELIS, K. ROSE and E. VAN NES, 1995: Super-individuals a simple solution for modelling large populations on an individual basis. Ecol. Model. **80**: 161–170.

SCHLADOW, S. G. and D. P. HAMILTON, 1997: Prediction of water quality in lakes and reservoirs: Part II –Model calibration, sensitivity analysis and application. Ecol. Model. **96**: 111–123.

SCHLITTGEN, R. and B. H. J. STREITBERG, 1989: Zeitreihenanalyse. R. Oldenbourg Verlag, Wien. 3. Auflage.

SCHNUTE, J., 1981: A versatile growth model with statistically stable parameters. Can. J. Fish. Aquat. Sci. **38**.

SHANAHAN, P., D. BORCHARDT, M. HENZE, W. RAUCH, P. REICHERT, L. SOMLYODY and P. VANROLLEGHEM, 2001: River water quality model no. 1 (RWQM1): I. Modelling approach. Water Sci. Technol. **43**: 1–9.

SOETAERT, K., T. PETZOLDT and R. W. SETZER, 2010a: Solving Differential Equations in R. The R Journal **2**: 5–15. URL `http://journal.r-project.org/archive/2010-2/RJournal_2010-2_Soetaert~et~al.pdf`

SOETAERT, K., T. PETZOLDT and R. W. SETZER, 2010b: Solving differential equations in r: Package desolve. Journal of Statistical Software **33**: 1–25. URL `http://www.jstatsoft.org/v33/i09`.

STRAŠKRABA, M. and A. GNAUCK, 1983: Modellierung limnischer Ökosysteme. Gustav Fischer Verlag, Jena.

THORNTON, K. W., A. S. LESSEM, D. E. FORD and C. A. STIRGUS, 1979: Improving ecological simulation through sensitivity analysis. Simulation **32**: 155–166.

TILMAN, D., 2000: Causes, consequences and ethics of biodiversity. Nature **405**: 208–211.

TSOULARIS, A., 2001: Analysis of logistic growth models. Res. Lett. Inf. Math. Sci. **2**: 23–46.

VANROLLEGHEM, P., D. BORCHARDT, M. HENZE, W. RAUCH, P. REICHERT, P. SHANAHAN and L. SOMLYÓDY, 2001: River water quality model no. 1: III. Biochemical submodel selection. Water Sci. Technol. **43**: 31–40.

VARIS, O., 1988: Temporal sensitivity of *Aphanizomenon flos-aquae* dominance – A whole-lake simulation study with input perturbations. Ecol. Model. **43**: 137–153.

VENABLES, W. N. and B. D. RIPLEY, 1999: Modern Applied Statistics with S-Plus. Springer, New-York, third edition.

VENABLES, W. N., D. M. SMITH and THE R DEVELOPMENT CORE TEAM, 2001: An Introduction to R: A Programming Environment for Data Analysis and Graphics, Version 1.4.0. www.r-project.org.

VERHULST, P., 1838: Notice sur la loi que la population suit dans son accroissement. Correspondence Mathématique et Physique **10**: 113–121.

VOLLENWEIDER, R. A. and J. KEREKES, 1980: OECD cooperative programme for monitoring of inland waters. (Eutrophication control). Synthesis report, Organisation for Economic Co-operation and Development, Paris.

WEST, G., J. H. BROWN and B. J. ENQUIST, 2001: A general model of ontogenetic growth. Nature **413**: 628–631.

WILSON, W., 2000: Simulating Ecological and Evolutionary Systems in C. Cambridge University Press, Cambridge.

WOOTTON, J. T., 2001: Local interactions predict large-scale pattern in empirically derived cellular automata. Nature **413**: 841–844.

ZADEH, L., 1965: Fuzzy Sets. Information and Control. Academic Press, New York.

ZADEH, L., 1987: Fuzzy sets and applications. In: Yager,R.R. et al. (eds): Selected Papers by L.A.Zadeh. John Wiley & Sons,New York .

ZAR, J. H., 1996: Biostatistical Analysis. Prentice-Hall, Upper Saddle River, NJ, third edition.

ZIMMERMANN, H.-J., 1991: Fuzzy set theory and its applications. Kluwer Academic Publishers, 2nd Edition .