Software-unterstützte Erzeugung von mathematischen Modellen zur biotechnologischen Prozessführung

Norman Violet

Software-unterstützte Erzeugung von mathematischen Modellen zur biotechnologischen Prozessführung

vorgelegt von
Dipl.-Ing.
Norman Violet
geb. in Berlin

von der Fakultät III - Prozesswissenschaften
der Technischen Universität Berlin
zur Erlangung des akademischen Grades

Doktor der Ingenieurwissenschaften
- Dr.-Ing. -

genehmigte Disseration

Promotionsausschuss:

Vorsitzender: Prof. Dr.-Ing. Matthias Kraume
Gutachter: Prof. Dr.-Ing. Rudibert King
Gutachter: Prof. Dr.-Ing. Peter Götz

Tag der wissenschaftlichen Aussprache: 09. März 2016

Berlin 2016

Impressum:

Druck: epubli ein Service der neopubli GmbH, Berlin, www.epubli.de

Vorwort

Die Grundlagen zu dieser Arbeit entstanden während meiner Zeit als wissenschaftlicher Mitarbeiter am Fachgebiet für Mess- und Regelungstechnik des Instituts für Prozess- und Verfahrenstechnik bei der Bearbeitung verschiedener Projekte. Im Rahmen des Forschungsvorhabens „Entwicklung eines Rapid Prototyping Ansatzes zum schnellen und effizienten Aufbau optimaler Prozessführungsstrategien für biotechnologische Produktionsprozesse“[1] wurden Methoden zur Beschleunigung des Aufbaus von Modellen gesucht. Neben der Implementation vieler Programmfunktionen zur Verbesserung vorhandener Algorithmen, wurde auch die Idee einer automatischen Modellierungshilfe entwickelt, auf sie konzentriert sich die vorliegende Arbeit. Die Grundzüge eines solchen Software-Werkzeugs wurden in obigem Projekt programmiert, allerdings war der Einsatzbereich sehr beschränkt auf den konkreten Anwendungsfall. Darauf folgte der Forschungsverbund „Autotrophic production of fine chemicals and specifically labeled biocompounds in *Ralstonia eutropha*“[2] mit dem Teilprojekt „Process control for the synthesis of SI-labeled biomolecules with *R. eutropha*“. In dieser Zeit wurde die Software „erwachsen“ und erhielt einen Namen: *Modellstrukturgenerator*. Durch viele Änderungen wurde eine breitere Einsatzfähigkeit ermöglicht. Zuletzt bearbeitete ich ein Teilprojekt im Forschungsverbund „BioIndustrie von Morgen – Plattformtechnologien für automatisierte Bioprozessentwicklung – AutoBio“[3]. Während der letzten Projektlaufzeit wurde vor allem die Bedienbarkeit des Modellstrukturgenerators verbessert und Sonderfälle berücksichtigt, aber auch Fehler korrigiert, die sich durch das dritte Projekt und seine Besonderheiten offenbarten.

Dem Leiter des Fachgebiets, Herrn Prof. Dr.-Ing. R. King, danke ich sehr für die wissenschaftliche Unterstützung und Betreuung, aber auch für den eingeräumten Freiraum bei der Durchführung meiner Forschungstätigkeit. Ich danke ihm auch dafür, dass er es mir ermöglichte, an der Technischen Universität Berlin zu promovieren. Seine kritischen und inspirierenden Hinweise waren besonders wertvoll für mich. Ebenso danke ich Prof. Dr.-Ing. P. Götz für sein Interesse und die Übernahme des Mitberichts, sowie Prof. Dr.-Ing. M. Kraume für den Prüfungsvorsitz.

Mein Dank richtet sich auch an alle gegenwärtigen und ehemaligen Kollegen aus den beiden Arbeitsgruppen am Fachgebiet, zusammen haben wir viele fachliche und unfachli-

---

[1] Gefördert von der Arbeitsgemeinschaft industrieller Forschung (IGF-Nummer 14775N).

[2] Gefördert vom Bundesministerium für Bildung und Forschung (BMBF).

[3] Ebenfalls vom BMBF gefördert.

che Herausforderungen angenommen und gemeistert. Danken möchte ich auch Dipl.-Ing. Joachim Kraatz für seine technische Unterstützung sowie Laborleiterin Dipl.-Ing. Margrit Valentin.

Schließlich danke ich auch meiner Familie: Meiner Frau Bianca für ihre Geduld und Unterstützung bei der Fertigstellung dieser Arbeit, und meiner Tochter Fiona dafür, dass sie mich oft zum Lachen gebracht hat.

Berlin, 16. Januar 2016 Norman Violet

Inhaltsverzeichnis

Abkürzungen, Symbole und Textmarkierungen

Verwendete Abkürzungen und Begriffe

ABC	Advanced Batch Control - Das Prozessleitsystem am FG MRT
AProS	Automatisches Probennahmesystem
atline	Zeitnah zur Verfügung stehende Messdaten
BTM	Biotrockenmasse
CSTR	Continuous ideally stirred-tank reactor
DCU	Digital Control Unit des Fermenters
DOE	Design of Experiments
FG MRT	Fachgebiet Mess- und Regelungstechnik
GUI	Graphical User Interface
HTML	Hypertext Markup Language
in silico	"in Silizium", in einem/als Computerprogramm
in vivo	"im Leben", im lebenden Organismus
IUPAC	International Union of Pure and Applied Chemistry
LabView	IDE zur grafischen Programmentwicklung
LS	Least squares
Matlab	Programm für numerische Berechnungen
MFCS	Multi-Fermenter-Control-System
MiMe	Michaelis-Menten-Gleichung
MLE	Maximum Likelihood Estimation
MySQL	My Structured Query Language
NLopt	Nonlinear Optimization (open-source library)
NMPC	Nonlinear Model Predictive Control
ODE	Ordinary Differential Equation
OLE	Object Linking and Embedding
OPC	OLE for Process Control, Software-Schnittstelle der Automatisierungstechnik
PC	Personal Computer
pH	potentia Hydrogenii
PHP	Personal Home Page (tools)
PLS	Prozessleitsystem
SQP	Sequential Quadratic Programming
TomLab	AddOn für Matlab mit Optimierungsverfahren
WLS	Weighted Least Squares

Verwendete Symbole

$\mathbf{C}$	Kovarianzmatrix
c	Messgröße, Konzentration eines Zustands bezogen auf das Gesamtvolumen
$\mathbf{C}_y$	Kovarianzmatrix des Messrauschens
f_q	Formalkinetik
$\underline{f}$	Zustandsgleichungen
$\overline{h}$	Messgröße, Konzentration eines Zustands bezogen auf das Zellvolumen
$\underline{h}$	Messgleichungen
h	Schrittweite
i	Laufindex, auch Index eines Zustands, das als Edukt in einer Reaktion auftritt
j	Laufindex, auch Index eines Zustands, das als Produkt in einer Reaktion auftritt
k	allgemeiner Laufindex, (als tiefergestellter Index) auch (zeitlich)diskreter Datenwert
m	Masse eines Zustands bzw. einer Substanz
$\mathbf{M_R}$	Matrix-Schema der einzelnen Reaktionsraten
μ	Spezifische Reaktionsrate
$\mu_{\max}$	maximale Wachstums- bzw. Reaktionsrate
N	Anzahl bzw. Länge
P	Wahrscheinlichkeitsdichtefunktion
p	(als Präfix) Partialdruck gelöster Gase
Φ	Gütewert, -funktion
q	Laufindex
r	Reaktion, Reaktionsrate
t	Zeit
t_0	Anfangszeit
$\underline{\theta}$	Modellparametervektor
T	(als hochgestellter Index) transponierte Matrix
$\mathbf{U}$	Stellgrößenmatrix
u	Stellgröße
V	Volumen
V_x	Zellvolumen
$\mathbf{W}$	Allgemeine Gewichtungsmatrix
$\underline{x}$	Zustandsvektor
X_0	Anfangswertparameter
$\underline{x}_0$	Anfangszustandsvektor des Differentialgleichungssystems
Y	Ausbeutekoeffizient
$\underline{y}$	Messwertvektor
$\dot{()}$	Zeitliche Ableitung
$*$	(als hochgestellter Index) optimaler (wahrer) Wert

Hinweise zu Textmarkierungen

- Variablenbezeichnungen in **Fettdruck** werden für Matrizen verwendet.
- In *kursiver* Schrift werden Fachworte oder Begriffe eingeführt.
- Text in `nicht-proportionaler` Schrift charakterisiert Programmfunktionsaufrufe, Variablen oder Befehle in Matlab.
- Eine KAPITÄLCHEN-Schrift wird zur Darstellung der logischen Werte WAHR oder FALSCH verwendet.

Even the simpliest living cell is a system of such forbidding complexity that any mathematical description of it is an extremely modest approximation.

(James E. Bailey, 1998)

Abstract

The aim of this study is to automate the process of finding an adequate model for fermentations of microorganismen and hence to facilitate it. With the aid of the developed methods and the resulting program *Modellstrukturgenerator* (model structure generator) little effort is needed to generate models to be implemented in an existing process development and control system. The Modellstrukturgenerator focusses on two main features: The first step includes the automatic transformation of given reaction formulas into standardized model equations. The creation of executable programs is the second step. An essential part of this thesis is therefore the design and creation of information-theoretical structures for achieving the above objectives.

Additionally, the automation of the model programming allows to create numerous variants of a given model without extra effort. These can automatically be adjusted to existing measurement data and thus being evaluated according their suitability. Thus, users don't have to refrain to a single model implementation in the first place, instead they can select the most useful variant after evaluation. The situation may occur that no clear decision is possible about which model should be used to work with. Therefore an existing program for planning processes (trajectory planning) was extended to use several models at the same time. Furthermore, experiments can be especially designed so that differences between models will become more significant (model discriminating design of experiments). The application of these two multi-model planning procedures will also be presented in this work.

Kurzfassung

Das Ziel der vorliegenden Arbeit ist es, die Modellerzeugung für Fermentationen mit Mikroorganismen zu automatisieren und somit zu erleichtern. Mit Hilfe der hierfür entwickelten Methoden und des Programms *Modellstrukturgenerator* können mit wenig Aufwand Modelle für ein bestehendes Prozessentwicklungs- und -führungssystem erstellt werden. Der Modellstrukturgenerator arbeitet dafür zwei Schritte ab: Zunächst werden aus gegebenen Reaktionsformeln standardisierte Modellgleichungen formuliert, danach werden daraus ausführbare Programme erzeugt. Wesentlicher Teil dieser Dissertationsschrift ist daher der Aufbau informationstheoretischer Strukturen zur Erreichung des obigen Ziels.

Die Automatisierung der Modell-Programmierung ermöglicht es außerdem, ohne zusätzlichen Aufwand, zahlreiche Modellvarianten erstellen zu lassen. Diese können mit weiteren Funktionen automatisch an vorhandene Messdaten angepasst und somit ihre Eignung beurteilt werden. Der Benutzer muss sich somit nicht von vornherein auf eine konkrete Modellimplementierung beschränken, stattdessen kann er nach der Evaluierung die geeignetste Variante auswählen. Dabei kann die Situation auftreten, dass nicht klar entschieden werden kann, mit welchem Modell weitergearbeitet werden soll. Daher wurde für die Planung von Prozessen (Trajektorienplanung) die vorhandene Funktion so erweitert, dass mehrere Modelle parallel berücksichtigt werden. Auch können Versuche eigens so geplant werden, dass Unterschiede zwischen Modellen herausgearbeitet werden (modelldiskriminierende Versuchsplanung). Die Anwendung dieser beiden Multimodell-Planungsverfahren wird in dieser Arbeit ebenfalls vorgestellt.

1 Einleitung

Bakterien und Archaeen beherrschen den Planeten. Uns Menschen sowieso.

(Clemens Makanay, 2012)

Diese Sätze aus [49] sind eine der möglichen Antworten auf die Frage „Warum ist es wichtig, sich mit Mikroorganismen zu beschäftigen?“. Allerdings zieht diese Antwort gleich die nächste Frage nach sich; warum besitzen denn Mikroorganismen einen so großen Einfluss auf das menschliche Leben? Hierauf soll im Folgenden ein wenig detaillierter, trotzdem auch nur exemplarisch anhand von drei Aspekten eingegangen werden:

Historisch betrachtet begann die Entwicklung des Lebens auf unserem Planeten mit Einzellern. Sie stellen somit die Urahnen der Menschheit dar. Heutige Mikroorganismen unterscheiden sich vielleicht in einigen Details von den damaligen, aber im Großen und Ganzen wirken noch immer dieselben Mechanismen in ihnen und auch in uns.

Zweitens lebt der Mensch auch heute noch durch und mit ihnen: Es waren Mikroorganismen, die die Atmosphäre unseres Planeten soweit veränderten, dass wir heute Sauerstoff atmen können. Jetzt leben wir in einer engen Symbiose mit ihnen, tatsächlich handelt es sich sogar teilweise um eine „obligatorische Endosymbiose“: Zum Beispiel ist das menschliche Verdauungssystem auf Mikroorganismen angewiesen. Nach [73] besteht der Mensch nur zu 10 % aus menschlichen Zellen; die restlichen Zellen sind Mikroben und leben auf oder in uns. Ihre Gesamtmasse beträgt allerdings nur einige, wenige Kilogramm pro Mensch. Umfassende Untersuchungen werden jedoch erschwert, da viele Mikroorganismen sich nicht außerhalb ihres gewöhnlichen Habitats kultivieren lassen. Doch es ist nicht nur schwer, Mikroorganismen zu untersuchen. Man kann sie auch nicht vollständig beseitigen, was insbesondere bei pathogenen Einzellern wünschenswert wäre. Selbst an Bord der Internationalen Raumstation ISS wurde eine lebendige und sich dynamisch verändernde mikrobielle Population nachgewiesen, trotz aller Bemühungen um Sterilität [87]. Das Leben und die Vielfalt der Mikroorganismen werden daher noch auf absehbare Zeit hinaus Gegenstand intensiver Forschung sein.

Die Hauptmotivation für diese Arbeit basiert auf einem weiteren Aspekt: Abgesehen von der Abstammung und der Symbiose besitzen Mikroorganismen einen erheblichen wirtschaftlichen Einfluss – positiven als auch negativen – auf die heutige menschliche Gesellschaft: Zu den negativen Folgen gehören Krankheiten durch pathogene Erreger oder Schäden durch Pilze an Gebäuden, worunter die Menschheit[1] nicht erst seit der Neuzeit zu leiden hat. Seit einigen Jahrzehnten beginnt der Mensch, sich auch das positive Potenzial der Mikroorganismen gezielt und insbesondere technisch nutzbar zu machen[2].

Heute werden die verschiedensten Entwicklungen in diesem Bereich unter dem Begriff Biotechnologie zusammengefasst. Zwei der ökonomisch wichtigsten Gebiete sind die sogenannte rote (Medizin) und weiße (industrielle Anwendung) Biotechnologie: Pharmaka sind meist sehr hochpreisige Produkte, während industrielle Produkte von der Masse beziehungsweise dem Produktionsvolumen her einen hohen Stellenwert einnehmen. Auf diese beiden Bereiche fokussiert sich auch die vorliegende Arbeit.

Tabelle 1.1 führt einige bekanntere Produktbeispiele aus verschiedenen Bereichen auf, die im großtechnischen Maßstab mit Hilfe biotechnologischer Prozesse hergestellt werden. Dort wird unter anderem auch Penicillin aufgeführt, das als ein Vertreter einer besonderen Substanzgruppe steht. Die Penicilline werden durch verschiedene Mikroorganismen gebildet. Die positive medizinische Wirkung einiger Arten von Penicillin war schon in der Antike bekannt, auch wenn man die Ursache nicht verstand [66]. Es dauerte bis in das späte 19. Jahrhundert, bevor der Zusammenhang zwischen Schimmelpilzen und ihrer therapeutischen Wirkung systematisch untersucht wurde, doch die Verfügbarkeit von Penicillin war noch sehr gering. Erst Mitte des 20. Jahrhunderts wurde erkannt, dass Penicillin auch industriell durch einen Fermentationsprozess[3] hergestellt werden kann [79]. Ein möglicher Grund für diese späte Erkenntnis mag sein, dass Penicillin aus

[1] Genaugenommen gilt dies nicht nur für den Menschen.

[2] Natürlich gibt es auch einige historische Beispiele, die sehr viel weiter zurückreichen: Die Herstellung von Alkohol, Sauerteig, Bier, Essig und Käse [31, 84].

[3] Nach den internationalen Empfehlungen der IUPAC bezeichnet das Wort *Fermentation* allgemein eine Kultivierung von Zellen (Mikroorganismen, Pflanzen oder tierischen Zellen) in einem Bioreaktor [53]. Der heutige deutsche Sprachgebrauch ist aber noch abhängig vom jeweiligen wissenschaftlichen Fachgebiet. Während in der Mikrobiologie und (Bio-)Verfahrenstechnik die empfohlene Bedeutung üblich ist, hat das Wort *Fermentation* beispielsweise in der Metabolomik noch die ursprüngliche Bedeutung von Louis Pasteur (1822 – 1895, Chemiker und Mikrobiologe, Frankreich) beibehalten und ist dort ein Synonym für *Gärung* oder anaerobe Kultivierungen.

einem Sekundärstoffwechsel der entsprechenden Mikroorganismen stammt, also nur ein fakultatives Produkt darstellt, einen sogenannten Sekundärmetabolit. Produkte dieses Typs stellen die industrielle Biotechnologie vor besondere Herausforderungen. Passende Prozessführungen zu ihrer Herstellung sind nicht zuletzt deswegen ein interessantes Forschungsgebiet. Einige Eigenschaften werden zum besseren Verständnis im Folgenden kurz zusammengefasst.

Produkt	Weltjahresproduktion 2012 [t/a]	Anwendung
Bioethanol	71.000.000	Energieträger, Lösungsmittel
Zitronensäure	1.600.000	Lebensmittel, Waschmittel
Essigsäure	1.400.000	Lebensmittel
Penicillin	45.000	Medizin
Polymilchsäure	120.000	Verpackung
Vitamin C	100.000	Pharma, Lebensmittel
L-Glutamat	2.500.000	Geschmacksverstärker
L-Arginin	10.000	Medizin, Kosmetik

Tabelle 1.1: Beispiele für im industriellen Maßstab hergestellte Biotech-Produkte [15]

Lange Zeit wurden Sekundärmetabolite als Substanzen mit vergleichsweise geringen Molekulargewichten definiert, die nicht Produkt des Primärstoffwechsels des Mikroorganismus waren und auch keinen Einfluss auf diesen hatten. Zellen hätten, so nahm man weiter an, keinerlei Nutzen davon, sie zu produzieren. Tatsächlich ging man sogar davon aus, dass sie keinerlei Funktion hätten und eher Nebenprodukte im Sinne von Abfallstoffen wären [80, 88]. Heutzutage ist man vom genauen Gegenteil überzeugt: Die Produktion der individuellen Sekundärmetabolite stellt eine Investition der Zelle dar, die einen Wettbewerbsvorteil gegenüber anderen, insbesondere fremden Zellen bietet [75]. Bei einer bestimmten Gruppe von Produkten – den Antibiotika – erlangen die Mikroorganismen diesen Vorteil, weil die Substanzen toxisch auf andere Organismen wirken. Zusätzlich wird in der Literatur berichtet, dass diese Sekundärmetabolite Produkte aus speziellen Stoffwechselwegen sind, die sich eben nur zu diesem Zweck im Laufe der Evolution des jeweiligen Organismus entwickelt haben [18, 19]. Für den Menschen ist dabei relevant, dass diese Inhibitoren mehr oder weniger spezifisch wirken und daher als Abwehrstoffe gegenüber unerwünschten Schädlingen und Krankheitserregern genutzt werden können, daher auch ihr Name „Antibiotika“ [76].

Bereits vor Jahrhunderten wurden erste biotechnologische Prozesse, die nicht zur Lebensmittelherstellung dienten, durchgeführt. Dazu gehörte beispielsweise die Salpeterherstellung [65]. Ungeachtet dessen begann die eigentliche industrielle Entwicklung der Biotechnologie erst im letzten Jahrhundert. Viele Bio-Prozesse sind seit etlichen Jahren zum Teil sogar seit Jahrzehnten in der Industrie etabliert. Technisch und wirtschaftlich sinnvolle Prozessführungsstrategien sind daher ebenfalls seit längerer Zeit bekannt. Viele Fortschritte gibt es im Bereich der Pharmazeutik, weil unter anderem die Molekularbiologie zusammen mit der Gentechnik fortwährend neue Möglichkeiten und Produkte schafft. Das zukünftige ökonomische Potenzial der gesamten Biotechnologie wird als „unüberschaubar groß“ eingeschätzt [67].

Aufgrund der Kosten, die eine Fermentation grundsätzlich verursacht, werden bislang hauptsächlich Feinchemikalien (engl.: „*fine chemicals*“) auf diesem Wege hergestellt. Sie zeichnen sich im Gegensatz zu den Grundchemikalien (engl.: „*bulk chemicals*“) darin aus, durch komplexere, mehrstufige chemische (oder biochemische) Prozesse mit einer vergleichsweise hohen Reinheit und meist nur in geringen Mengen hergestellt zu werden. Diese Bedingungen führen meist automatisch dazu, dass Feinchemikalien durch rein chemische Verfahren nur zu deutlich höheren Herstellungskosten produziert werden können. Aus diesem Grund kann die Fermentationstechnik wirtschaftlich lohnenswert sein. Insbesondere können Fermentationen den Vorteil bieten, dass mehrstufige chemische Prozesse vermieden werden können, wenn Mikroorganismen das entsprechende Produkt direkt aus billigen Rohstoffen herstellen.

Um ein paar konkrete Zahlen zum Wachstum der Branche zu nennen: In [14] wird aufgeführt, dass der erwirtschaftete Umsatz innerhalb der Jahre 2006 bis 2010 von 80 Mrd. Euro weltweit auf etwa 115 Mrd. US-Dollar allein in den USA anstieg. Der Gesamtanteil der Produkte und Verfahren, die Biotechnologie einsetzen, beläuft sich aktuell (2013) auf rund 5 %. Innerhalb der nächsten zehn Jahre wird mit einer Verdopplung bis Vervierfachung dieses Anteils gerechnet.

Diese sehr optimistische Tendenz wird in [10] prognostiziert, obwohl die Entwicklung eines technisch und auch wirtschaftlich erfolgreichen Prozesses nicht einfach ist. Bislang verhindert häufig der hohe Aufwand verbunden mit dem Risiko, dass ein bestimmtes, vorgegebenes Produktionsziel technisch, biochemisch oder wirtschaftlich nicht erreicht werden kann, den umfassenden Einsatz. Aber die Prognose berücksichtigt auch, dass bei der Entwicklung vieler bereits etablierter Prozesse noch keine effizienten und effektiven Verfahren zur Verfügung standen. Ihre Potentiale zur Verbesserung sind daher groß.

Für Fermentationen gibt es verschiedene Arten der Prozessführung. Statische, kontinuierliche Prozesse sind für viele Prozessen geeignet, bei denen beispielsweise die Syntheserate des entsprechenden Produkts proportional zum Wachstum der Mikroorganismen ist. Die Umgebungsbedingungen der Zellen sind bei dieser Prozessart konstant. Fed-Batch-Prozesse hingegen beeinflussen die Umgebungsbedingungen aktiv. Die Prozessentwicklung und -führung sind meist deutlich komplexer, sie ermöglichen dafür aber die gezielte Einstellung von Bedingungen, wie sie beispielsweise für die Produktion von Antibiotika benötigt werden. Auch reine Batch-Prozesse führen zu unterschiedlichen Bedingungen. Auf diese kann aber nur durch die Wahl der Anfangsbedingungen Einfluss genommen werden; der weitere Verlauf wird durch das Wachstum und Verhalten der Mikroorganismen bestimmt. Eine effiziente und optimale Prozessführung für Sekundärmetabolit-Produkte kann so nicht erreicht werden.

Die Prozessentwicklung von effizienten Fed-Batch-Verfahren ist wie erwähnt aufwendig, aber für viele Produkte gibt es bislang keine alternativen chemischen Herstellungsverfahren. Häufig sind die Produktionskosten vergleichsweise hoch und Fehlschläge durch Fehler in der Prozessführung reduzieren einen Gewinn zusätzlich. Die Optimierung (und auch die Robustifizierung) von Fed-Batch-Prozessen ist daher ökonomisch von ständig zunehmender Bedeutung für die Industrie. Der Fokus dieser Arbeit liegt in der Erzeugung von mathematischen, dynamischen Modellen in Form von Differentialgleichungssystemen zur Beschreibung von Bio-Prozessen. Für diese Modelle existieren viele moderne numerische Verfahren, die beispielsweise zur Optimierung eines Prozesses genutzt werden können.

Der Einsatz moderner Prozessentwicklungsverfahren ist nicht nur für die aufwendige Neuentwicklung von Prozessen sinnvoll. Durch Veränderung äußerer Rahmenbedingungen, wie Konkurrenz, Ressourcenknappheit und steigenden Energiepreisen, aber auch durch notwendige Maßnahmen zum Umweltschutz, können eine Anpassung bislang geeigneter Prozessführungen erfordern. Dabei wird verständlicherweise meist eine Optimierung eines vorhandenen Prozesses angestrebt, anstatt nach vollkommen neuen Verfahren zu suchen. In der Vergangenheit hat der Mensch in vielen Fällen durch jahrelanges Ausprobieren herausgefunden, wie ein bestimmter Prozess verbessert werden konnte. Jedoch führt dieses intuitive Vorgehen weder in der kurzen Zeitspanne, die die heutige Schnelllebigkeit der Marktsituation vielfach verlangt, noch bei komplexen Prozessen, die vom Menschen nicht mehr in ihrer Gesamtheit überblickt werden können, zum Ziel. Abgesehen davon muss das intuitive Vorgehen nicht unbedingt zu einem tatsächlichen Optimum führen. Daher ist die Try-and-Error-Methode vor allem bei neuen, komplexen Prozessen

und besonders bei Fed-Batch-Prozessen häufig nicht mehr angebracht, sondern sollte durch eine mathematische Optimierung ersetzt werden.

Moderne Optimierungsverfahren eröffnen ein großes Potential zur Prozessverbesserung. Für ihren Einsatz werden allerdings mathematische Modelle benötigt. Durch solche Modelle kann auch ermittelt werden, wie effizient oder effektiv ein gedachter Prozess erwartungsgemäß sein wird. Ganz unabhängig von der technischen Umsetzbarkeit kann so durch ein Modell die Wirtschaftlichkeit eines Prozesses bestimmt werden. Voraussetzung hierfür sind gute Modelle, die gefunden beziehungsweise aufgestellt werden müssen. Die Güte eines Modells ist ein wesentlicher Aspekt bei der Prozessentwicklung. Der zu ihrer Erzeugung notwendige Aufwand ist nicht unerheblich, aber ebenso groß ist der mögliche Nutzen für einen Prozess.

Obwohl die hier im Weiteren behandelten Modelle im Vergleich zu den Modellen der Systembiologie sehr vereinfacht und abstrakt mit biologischen Mechanismen umgehen, erfüllen sie zwei wesentliche Voraussetzungen für Prozessmodelle: Sie sind in der Lage Voraussagen über einen Prozessverlauf auch bei sich verändernden Bedingungen, wie zum Beispiel durch eine einsetzende Zufuhr oder bei einer Limitation von Nährstoffen, zu berechnen, und diese Berechnung ist in kurzer Zeit möglich, was wichtig für eine Prozessregelung ist. Für die Entwicklung von (gegenüber Störeinflüssen) robusten Prozessführungen sind ebenfalls dynamische Simulationen notwendig. Dies gilt für robuste Trajektorienplanungen ebenso wie für robuste Regelungsverfahren. Diese Robustheit fehlt bei vielen biotechnologischen Prozessen gegenwärtig noch [34]. Störungen mögen sich unter günstigen Umständen nur in einer leicht verminderten Ausbeute des Produkts äußern, aber sie können auch zum Totalverlust der Einsatzstoffe führen. Die zum Teil nicht unerheblichen Kosten der Einsatzstoffe in biotechnologischen Prozessen heben die Bedeutung von guten Prozessmodellen deutlich hervor.

Trotz vieler Bemühungen ist das Auffinden eines geeigneten dynamischen Modells für biotechnologische Prozesse eine große Herausforderung. Die gesuchten Modelle müssen das Verhalten von Mikroorganismen voraussagen können, doch die Informationsmenge zu Beginn der Modellentwicklung ist meist eher gering. Zudem ist es gegenwärtig – und wahrscheinlich auch mittelfristig – nicht möglich, alle Substanzen, die eine Rolle im Metabolismus der Zellen spielen, hinreichend häufig und präzise genug zu messen, um die Dynamik der zellinternen Prozesse vollständig zu erfassen. Daher ist schon das reine Beschreiben von Messdaten aus Bioprozessen ein nicht einfaches oder gar intuitiv gelingendes Unterfangen, da es meist um unbekannte nichtlineare Zusammenhänge geht.

Abgesehen von dem Unvermögen, jede denkbare Messgröße messen zu können, gibt es auch Gründe, nicht alle messen zu wollen: Beispielsweise kosten neue Sensoren in der Anschaffung Geld, verursachen einen erhöhten Aufwand bei der Wartung, stellen zusätzliche Fehlerquellen dar und sind in vorhandene Anlagen nicht immer nachträglich einbaubar.

Für viele Zusammenhänge sind eine große Zahl von Varianten der mathematischen Beschreibung möglich. In der Praxis werden daher oft Annahmen oder Entscheidungen über Modelleigenschaften getroffen, die später nicht mehr auf ihre Gültigkeit oder Richtigkeit überprüft werden (können). Diese Annahmen werden oft beibehalten, selbst dann, wenn die Datenbasis im späteren Verlauf der Prozessentwicklung umfangreicher geworden ist: Der Zeitaufwand der manuellen Modellbildung, selbst durch Experten mit den verfügbaren Modellbildungswerkzeugen – siehe Abschnitt 1.3 – ist nicht zu unterschätzen [29, 68]. Vor diesem Hintergrund wird deutlich, dass eine effektive industrielle Biotechnologie unter Ausnutzung modellgestützter Ansätze noch am Anfang ihrer Entwicklung steht.

Der Modellbildungsprozess für biotechnologische Anwendungen stellt somit ein klassisches Beispiel für eine Gratwanderung dar: Einerseits ist eine hohe Komplexität gewünscht, um die vielfältigen Prozesse im Inneren von Zellen abzubilden und vorauszusagen. Dies gilt insbesondere für die Bildung von Sekundärmetaboliten. Andererseits stehen insbesondere zu Beginn der Prozessentwicklung nur wenige Daten über die Dynamik des Prozesses zur Verfügung, was die sinnvolle Modellkomplexität begrenzt. In der Praxis stellen die gesuchten Prozessmodelle daher immer starke Vereinfachungen der Realität dar, und sie müssen zu den jeweils verfügbaren Daten passen. Außerdem besteht eine Wahlfreiheit darüber, welche Einflussfaktoren bei der Modellierung berücksichtigt werden sollen. Zusammen führt dies dazu, dass es kein einzelnes „richtiges“ Modell gibt.

1.1 Ziele dieser Arbeit

Ein Lösungsansatz, um die (industrielle) Einführung neuer oder optimierter Prozessführungen dennoch zu vereinfachen und zu beschleunigen, besteht darin, erstens den Aufwand der Modellierung zu verringern und zweitens die Güte von Modellen zu verbessern. An diesen beiden Punkten setzt die vorliegende Arbeit an. Es wird eine Möglichkeit vorgestellt, um den Zeitaufwand zu reduzieren und gleichzeitig auf das frühzeitige Festlegen auf eine willkürliche, möglicherweise letztendlich ungeeignete Modellimplementation

zu verzichten: Die automatische Generierung, Codierung und Bewertung von Modellkandidaten. Hierzu sind Methoden und, neben anderen Hilfsfunktionen, ein Programm *Modellstrukturgenerator* zu entwickeln. Insbesondere durch den Modellstrukturgenerator soll der Aufwand zur Modellerzeugung soweit sinken, dass eine große Anzahl von Modellen untersucht werden kann.

Auf die Beschreibung der Methoden und dieses Programms, aber auch auf die weitere Verwendung der so erzeugten Modelle konzentriert sich die vorliegende Arbeit. Erst mit einsatzfähigen, validen Modellen eröffnen sich die vielfältigen Möglichkeiten der modernen Prozessentwicklung und Prozessführung. Außerdem können mit ihrer Hilfe die Kosten und der Zeitaufwand für eine Prozessentwicklung gesenkt werden, wodurch unteren anderem auch kleinen und mittelständischen Unternehmen die Möglichkeit geboten wird, eigene Projekte zu entwickeln und voranzutreiben.

1.2 Die Arbeitsumgebung

Die hier vorgestellten Methoden und Programmteile sind eingebettet in Arbeiten des Fachgebiets Mess- und Regelungstechnik der TU Berlin. Es verfügt über ein eigenes Biolabor, damit hier entwickelte modellbasierte regelungstechnische Verfahren nicht nur mittels Simulationsstudien, sondern auch in der Realität getestet werden können. Zu diesem Zweck wurde ein sehr umfangreiches Prozessleitsystems entwickelt, das im nächsten Kapitel beschriebene *ABC-System*, an dessen Entwicklung der Autor mitwirkte. Die Durchführung realer Experimente ermöglicht es zudem, die praktische Relevanz der Methoden zu überprüfen. Viele davon wurden und werden speziell auf die Fermentationen mit Sekundärmetabolit-Produkten angewendet. Dies bezieht sich teilweise auch auf Fermentationen, die im Rahmen der hier vorgestellten Methodenentwicklung durchgeführt wurden und die unten exemplarisch gezeigt werden. Die Sekundärmetabolit-Produktion stellt eine besondere Herausforderung aus modell- und prozesstechnischer Sicht dar. Bei ihnen spielen unter anderem messtechnisch nicht dynamisch erfassbare Vorgänge im Inneren der Zellen eine wichtige Rolle. Gleichzeitig sind viele der Sekundärmetabolite von besonderem Interesse für die Pharmazeutik.

Um den Prozess der Verfahrensmodellierung zu unterstützen und zu beschleunigen, wurde wie erwähnt ein automatischer Modellstrukturgenerator entwickelt. Um die Möglichkeiten des bestehenden ABC-Systems nutzen zu können, mussten einige Annahmen beziehungsweise Einschränkungen des ABC-Systems übernommen werden, die nachfol-

gend zusammengefasst sind. Es soll gleich betont werden, dass eine Reihe von Einschränkungen im Laufe der Arbeit teilweise aufgehoben wurden:

- Synthetische (chemisch definierte) Minimalmedien
  Aus Vorläuferarbeiten wurde übernommen, dass bis zu drei essentielle Grundsubstrate in den Fermentationen verwendet und für die Modellierung berücksichtigt werden: Jeweils eine Stickstoff-, eine Phosphor- und eine Kohlenstoff-Quelle. Die chemische Zusammensetzung synthetischer Medien kann exakt angegeben werden. Anteile von beispielsweise Hefe- oder Soya-Extrakten mit veränderlichen Inhaltsstoffen sind nicht enthalten. Minimalmedien schließen diauxische Effekte (Übergangseffekt bei der Metabolisierung von Primär- und Sekundärsubstratquellen) aus und führen somit zu einer Vereinfachung der Modelle. Die Verwendung chemisch definierter Medien führt außerdem zu einer besseren Reproduzierbarkeit der Anfangsbedingungen. Weitere essentielle Stoffe, wie Sauerstoff und Spurenelemente, werden nicht in die Modelle aufgenommen. Diese modelltechnische Vereinfachung ist sinnvoll, solange prozesstechnisch sichergestellt werden kann, dass keine Limitation bezüglich dieser Stoffe auftritt.

- Konstante Versuchsbedingungen bezüglich pH und Temperatur
  Weitere Prozessbedingungen wie pH-Wert und Temperatur werden durch geeignete Regeleinrichtungen während einer Fermentation und für alle einzelnen Fermentationen (eines Mikroorganismus) konstant gehalten.

- Schnelle Probennahmen
  Die für die Durchführung einer Probennahme erforderliche Zeitspanne ist verglichen mit einer typischen Fermentationszeit sehr kurz. Deswegen kann eine Probennahme einem Zeitpunkt zugeordnet werden, statt einen Zeitraum dafür anzugeben. Für die Lösung der Differentialgleichungssysteme müssen Algorithmen verwendet werden, die die daraus resultierenden diskreten Zustandsänderungen berücksichtigen. Diese Funktionalität ist bereits Teil des ABC-Systems und hat für den Modellstrukturgenerator den vereinfachenden Effekt, dass ein zeitlicher Verlauf von Probennahmen nicht modelliert werden muss.

- Ideale Durchmischung eines Rührkesselreaktors (CSTR)
  Die ideale Durchmischung des Reaktorinhalts wird durch eine vorgegebene minimale Rührerdrehzahl ($300\ \mathrm{min}^{-1}$), sowie auch durch eine Begasungsrate von min-

destens 0.5 vvm[4] angestrebt. Unter der Annahme der idealen Durchmischung sind keine partiellen Differentialgleichungen notwendig, da die Zustandsbeschreibung nur von einer Variablen, der Zeit, und nicht zusätzlich vom Ort abhängig ist.

- Homogene Zusammensetzung der Mikroorganismen
  Zur Vereinfachung der Prozessbeschreibungen wird erstens davon ausgegangen, dass bei den Fermentationen nur Mikroorganismen einer Art vorhanden sind. Dies schließt auch das Auftreten von Mutationen aus. Zweitens wird auf die Modellierung segregierter Populationen verzichtet, die unterschiedliche Zustände der einzelnen Zellen darstellen können.

- „Idealer“ Stofftransport
  Reine Transportvorgänge werden als nicht zeitbestimmend für die zellulären Reaktionen angenommen und somit nicht modelliert. Dazu zählen Transporte durch Zellwände und über Phasengrenzen (fest, flüssig, gasförmig) hinweg.

Die Funktionen beziehungsweise Fähigkeiten des Modellstrukturgenerators, unterschiedliche Modelleigenschaften automatisch verarbeiten zu können, wurden während verschiedener Projekte und der Entstehung dieser Arbeit zunehmend erweitert. Während der Entwicklung des Modellstrukturgenerators gab es außerdem Veränderungen bei einigen der oben genannten Beschränkungen des ABC-Systems. Diesen Änderungen konnte die Entwicklung des Modellstrukturgenerators aus Zeitgründen nicht immer folgen. Dies trifft beispielsweise für Fermentationen mit kontrollierten Begasungen[5] zu; dazu passende Modelle müssen auch ein physikalisches Gasmodell beinhalten. Die Entwicklung eines allgemein gültigen Gasmodells ist jedoch in Planung. Andererseits führten Funktionen des Modellstrukturgenerators auch dazu, dass zuvor geltende Limitationen des ABC-Systems aufgehoben wurden. Beispielsweise besteht nun die Möglichkeit, pH-Wert- oder temperatur-abhängige Reaktionsgeschwindigkeiten zu definieren. Auch kann der Modellstrukturgenerator (in einem begrenzten Umfang) Modelle mit Mehrfachquellen für einen Substrat-Typ erstellen (beispielsweise mit zwei verschiedenen Zuckern als C-Quelle). Zuletzt können Transportvorgänge auch relativ einfach simuliert werden, in dem ein neuer Modellzustand als Zwischengröße eingeführt wird. Auf einige dieser Erweiterungen wird in Kapitel 4 näher eingegangen.

[4] Einheit „vvm“: Gasvolumenstrom (in Minuten) pro Flüssigkeitsvolumen

[5] Begasungen mit unterschiedlichen Volumenanteilen verschiedener Gase.

1.3 Stand der Technik

Wie in den letzten Abschnitten erwähnt, ist die Modellierung von Prozessen heute eine Schlüsseltechnologie zum Erfolg. Beispielsweise in [39] wird jedoch ausführlich hervorgehoben, dass das Modellieren besonders für biotechnologische Anwendungen eine Herausforderung für sich darstellt.

In der chemischen Industrie stehen inzwischen viele unterstützende Software-Werkzeuge bereit. Schon 1994 wurde in [85] ein Vergleich zwischen verschiedenen Programmen vorgenommen: Auf der einen Seite dienen Matlab mit SimuLink (MathWorks) und ACSL (Advanced Continuous Simulation Languages, Mitchell & Gauthier Associates, Inc.) als Simulationsumgebungen für nahezu beliebige Probleme. Auf der anderen Seite stehen mit SpeedUp (Aspen, vormals Imperial College London) und DIVA (Universität Stuttgart) zwei Werkzeuge speziell für verfahrenstechnische Aufgaben zur Verfügung. Später kamen Produkte wie AspenPlus (Aspen), ChemCAD (ChemStations) oder gProms (PSE) hinzu, die über eine Vielzahl von Funktionen verfügen. Mit Ausnahme von AspenPlus beschränken sie sich aber auch heute im Wesentlichen nur auf die chemische Verfahrenstechnik.

Bei den Anwendungen für biochemische Prozesse hängt die Entwicklung einige Jahre hinterher. Allgemein kann man sagen, dass die Tendenz besteht, dass die meisten kommerziellen Programme eher für bestimmte Anwendungsfälle hochspezialisiert sind und nur dazu passende, etablierte Verfahren einsetzen. Auch Software aus akademischen Institutionen konzentriert sich häufig auf bestimmte Aspekte, allerdings seltener nur auf bestimmte Anwendungsfälle, und geben zusätzlich durch neue Methoden neue Impulse für die Branche. Beispielsweise beschäftigten sich [4, 9, 11, 12, 50, 51] mit der Entwicklung neuer Methoden zur Unterstützung der Modellierung von biochemischen Prozessen.

Es ist vielleicht verfrüht, von einem Durchbruch zu sprechen, aber im letzten Jahrzehnt begannen viele Projekte zur Entwicklung von Modellierungswerkzeugen. Dieser Trend ist vermutlich auch dadurch zu begründen, dass die Unterstützung durch Computer eine immer größere Selbstverständlichkeit ist. Daher ist zunächst keine Stagnation bei der Entwicklung neuer Methoden und Programme zu erwarten.

Im Folgenden werden einige erfolgreiche Software-Werkzeuge präsentiert, die zwei zukunftsträchtige Kriterien erfüllen. Erstens soll die Entwicklung der Software nicht schon vor Jahren eingestellt worden sein, so dass damit zu rechnen ist, dass einerseits auch neue Impulse und Methoden berücksichtigt werden und andererseits die Anzahl der Anwendungsfälle groß ist. Zweitens werden nur solche Programme ausgewählt, die eine offene

Schnittstelle aufweisen, wodurch ermöglicht wird, dass die Software auch im Zusammenspiel mit anderen Programmen genutzt werden kann. Hier hat sich in den letzten Jahren die Systems Biology Markup Language (SBML) durchgesetzt. Bei dieser handelt es sich um ein auf der Extensible Markup Language (XML) basierendes Dokumentenformat. Durch offene Standards wird die Plattform- und Programmunabhängigkeit ermöglicht [1, 2]. Obwohl das Dokumentenformat hauptsächlich zum Austausch von komplexen Informationen zwischen Computern und Programmen entwickelt wurde, und dementsprechend ein maschinenlesbares Format ist, werden die Informationen in den meisten Fällen durch Textzeichen kodiert und bleiben somit prinzipiell auch menschenlesbar.

Trotz der Filterung durch die beiden genannten Kriterien werden in einer Liste für SBML-Programme noch etwa 300 Programme aufgeführt [3]. Daher können im Folgenden nur einige exemplarische Programm(pakete) aufgeführt und kurz beschrieben werden.

Das Programm CellDesigner ist ein grafischer Modell-Editor, mit dessen Hilfe genregulierende und biochemische Netzwerke erstellt werden können. Es ist ein gutes Beispiel für den Nutzen der SBML-Schnittstelle, denn es verfügt über keine eigenen Simulations- oder Analysefunktionen, sondern speichert stattdessen das erzeugte Modell im SBML-Format. Ein solches Modell kann durch andere SBML-Programme, die auf diese Funktionen spezialisiert sind, geladen werden [24, 25]. Ein Beispiel für ein derartiges Programm ist SloppyCell. Es verfügt über keine eigenen Funktionen zum Erstellen von Modellen, sondern liest vorhandene SBML-Modelle ein und simuliert und analysiert diese [28, 59].

OMIX ist ein weiteres Beispiel für einen grafischen SBML-Modelleditor, der sich speziell für metabolische Netzwerke und deren Darstellung eignet. Hierfür bietet das Programm viele grafische Funktionen. Zusätzlich können Messdaten in einer frei vom Benutzer wählbaren Darstellungsart angezeigt werden werden [20].

SimBiology ist eine Matlab-Toolbox, die ebenfalls eine grafische Oberfläche zur Modellierung zur Verfügung stellt und darüber hinaus über umfangreiche Analyse- und Simulationsfunktionen verfügt. Die Möglichkeiten zur Modellierung sind umfangreich, aber die Unterstützung des SBML-Formats ist nicht vollständig; insbesondere kinetische Informationen (Gleichungen zur Beschreibung des dynamischen Verhaltens) werden nicht in das SBML-Format übertragen. Dieses Defizit betrifft allerdings nicht nur SimBiology.

Sycamore ist als schnelle Allround-Lösung zum Aufbau von Modellen metabolischer Netzwerke und für deren ersten Simulationen konzipiert [86]. Es bedient sich hierzu der Informationen aus öffentlichen Datenbanken.

VCell bietet sich einerseits für die Erzeugung unterschiedlich komplexer Modelle an

(grafisch oder durch mathematische Formulierung), und beinhaltet andererseits auch umfangreiche Funktionen zur Simulation der Modelle. Abgesehen von der SBML-Schnittstelle ist es als verteilte Anwendung konzipiert worden, und bietet daher die Möglichkeit, zeitintensive Berechnungen auszulagern [58].

ProMoT ist ursprünglich nur für verfahrenstechnische Anwendungen entworfen worden, hat sich aber zu einem Entwicklungswerkzeug auch für systembiologische Modelle weiterentwickelt. Es zeichnet sich durch einen starken objekt-orientierten Modellierungsansatz aus und verfügt zusammen mit dem Software-Tool Diana auch über verschiedene Analyse- und Simulationsfunktionen [27, 55, 71].

Ein besonders weitverbreitetes Programm ist COPASI. Nicht zuletzt durch seine lange Entwicklungshistorie[6] wird es heute als Referenz-Simulationsprogramm für sehr viele SBML-Modelleditoren angegeben. Darüber hinaus verfügt es auch über eigene Möglichkeiten, Modelle zu erstellen, Simulationen zu berechnen und darzustellen sowie verschiedene Analysen durchzuführen [35].

In Abgrenzung zu den genannten Programmen verfolgt der in dieser Arbeit vorgestellte Modellstrukturgenerator einen eher pragmatischen Ansatz bei der Modellierung und verfügt als herausragendes Leistungsmerkmal über die Fähigkeit zur automatischen Erstellung von zahlreichen Modellvarianten. Das (SBML-freie) TAM-B verfolgte ähnliche Ziele [45, 46].

Allerdings kann der Modellstrukturgenerator nicht direkt mit den anderen Programmen verglichen werden, da er eine Ergänzung des ABC-Systems darstellt. Zusammen stellen sie jedoch aufgrund des modularen Aufbaus des ABC-Systems ein sehr mächtiges Werkzeug zur Prozessentwicklung und -führung dar, siehe Kapitel 2.

1.4 Gliederung

Der Modellstrukturgenerator ist zwar prinzipiell ein eigenständiges Programm, wurde aber wie viele weitere Hilfsprogramme des Autors für das ABC-System entwickelt. Das ABC-System wird daher wenigstens in Ansätzen zunächst im Kapitel 2 beschrieben. Die Aufgabe des Modellstrukturgenerators ist die Erstellung von Modellen, diese liegen in Form von Differentialgleichungssystemen vor, die ihrerseits auf Massenbilanzen beruhen. Sie werden im Kapitel 3 vorgestellt. Abschnitt 3.1 stellt einige der Grundtypen von Modellen vor, die sich für die Prozessführung in der Bioverfahrenstechnik eignen. Im

[6] Über den Vorläufer GEPASI wurde bereits 1993 publiziert.

Rahmen dieser Arbeit wird dabei auf eine besondere Modellformulierung eingegangen, die sich für die Automatisierung gut eignet und im weiteren Verlauf verwendet wird.

Darauffolgend führt Abschnitt 3.2 allgemeine Aussagen und Hinweise auf typische, dem Autor zur Verfügung stehende Messgrößen auf, und liefert Erklärungen bezüglich Messgenauigkeit und Nutzbarkeit der Messwerte.

Abschnitt 3.3 beschreibt die Grundlagen der Parameteridentifikation zur Bestimmung unbekannter Modellparameter mit Hilfe von Messdaten. Abschnitt 3.4 stellt einige Besonderheiten des ABC-Systems zur Berücksichtigung spezieller, versuchsabhängiger Parameter vor.

Die Möglichkeiten, die sich mit einer passenden mathematischen Beschreibung eines Prozesses eröffnen, sind vielfältig, jedoch ist den regelungstechnischen Methoden häufig eines gemeinsam: Die Modelle werden in der einen oder anderen Form mit unterschiedlichen Zielsetzungen in Optimierungen eingesetzt. Abschnitt 3.5 gibt eine Übersicht verschiedener Optimierer und erläutert genauer die im Rahmen dieser Arbeit verwendeten Algorithmen.

Die Grundzüge der automatischen Modellbildung werden in Kapitel 4 vorgestellt. Eine wesentliche Voraussetzung für die weitgehende Automatisierung stellen effektive informationstheoretische Strukturen dar. Zusammen mit Einzelheiten der Umsetzung werden diese im Abschnitt 4.2 erläutert.

Kapitel 5 zeigt die Anwendung und die Ergebnisse von Methoden, die speziell für den Einsatz mit Multimodellen modifiziert wurden. Dies sind die Verfahren zur Beschleunigung der notwendigen Parameteridentifikationen, die Optimierung von Prozessen am Beispiel der Multimodell-Trajektorienplanung und der modelldiskriminierenden Versuchsplanung.

Eine Zusammenfassung der Arbeit und ein Ausblick auf weitere Entwicklungsmöglichkeiten werden im letzten Kapitel 6 gegeben.

Im Anhang A werden weitere Einzelheiten der Programmierung aufgeführt, darunter ein vollständiges, automatisch erstelltes Modell, das mit Hilfe der hier vorgestellten Methoden erzeugt wurde.

2 Advanced-Batch-Control-System

Alles sollte so einfach wie möglich gemacht sein, aber nicht einfacher.

(Albert Einstein)

In den letzten 14 Jahren wurde am Fachgebiet Mess- und Regelungstechnik (FG MRT) ein umfangreiches Software-Paket entwickelt, das *ABC-System*, dessen Methoden auch für andere Labore und Einsatzgebiete interessant ist. Da es häufig im Rahmen dieser Arbeit Erwähnung findet, soll es im Folgenden kurz beschrieben werden.

Das ABC-System zielt auf eine Vereinheitlichung und Automatisierung vieler wiederkehrender Tätigkeiten in einem Biolabor ab, insbesondere wenn regelungstechnische Arbeiten im Vordergrund stehen. Zunächst betraf es nur einzelne Schritte, im Laufe der Zeit wurden aber immer weitere Aufgaben mit seiner Hilfe automatisiert.

Mit dem ABC-System wurde einerseits ein Prozessleitsystem (PLS) entwickelt, das alle dafür typischen Aufgaben in einem biotechnologischen Labor übernimmt: Hierzu gehören die Messdatenerfassung und Steuerung oder Regelung des laufenden Prozesses, ebenso wie die Prozessüberwachung, Datenvisualisierung und -aufbereitung. Die dazu notwendigen Schnittstellen zu einem Fermentersystem liegen vor. Konkret betrifft dies die Schnittstellen zum Multi-Fermenter-Control-System (MFCS) der Firma Sartorius, ein Hardware-Treiber für die Ansteuerung der Fermenter-eigenen Digital Control Unit des Fermenters (DCU). Im Prinzip sind Schnittstellen für andere Hersteller integrierbar.

Andererseits ist das ABC-System ein Software-Framework zur Entwicklung neuer, modellbasierter Methoden aus dem Bereich der Regelungstechnik: Es ist modular aufgebaut und verfügt über viele Schnittstellen, um neue Programme und Funktionen einzubinden oder auszutauschen. Beispielsweise können (fertig programmierte) Optimierer aus verschiedenen Quellen (Matlab [52], TomLab [77] und NLopt [37]) genauso für verschiedene Funktionen des ABC-Systems ausgewählt werden, wie auch einige selbst programmierte. Als Entwickler-System konzipiert, erfährt es fortwährend Änderungen, Erweiterungen und Verbesserungen.

Die Entwicklung des ABC-Systems begann mit den in [30] beschriebenen Software-Methoden, beispielsweise Parameteridentifikation, Optimale Versuchsplanung und verschiedenen Zustandsschätzern. Seitdem kamen beständig Erweiterungen hinzu. Zu den Funktionen des ABC-Systems gehört unter anderem auch der für diese Arbeit wichtige Themenkomplex Modellierung. Der Modellstrukturgenerator automatisiert das Erzeugen von Modellen für das ABC-System wie später beschrieben wird.

Die beiden Funktionsteile, PLS und Methoden-Framework, sind über eine zentrale Datenbank miteinander verbunden. Durch die Kombination ist es möglich, neu entwickelte Methoden (zum Beispiel Regler, Beobachter, etc.) im Einsatz unter realen Bedingungen zu überprüfen. Abbildung 2.1 stellt die Zusammenhänge zwischen den einzelnen Teilen des Gesamtsystems dar.

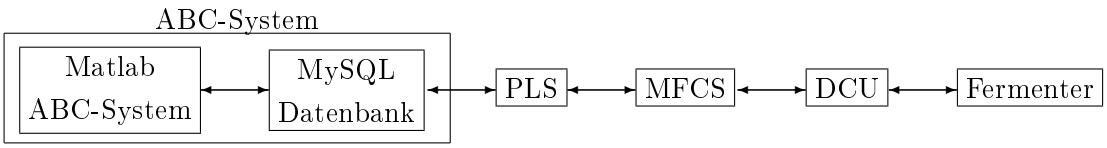


Abbildung 2.1: Schematisches Datenflussschema

Die meisten Funktionen wurden in Matlab entwickelt, parallel kommt aus Geschwindigkeitsgründen auch die Programmiersprache C zum Einsatz. Andere Teile wurden aufgrund der einfachen Entwicklung grafischer Benutzeroberflächen in LabView [60] oder im Rahmen einer Web-Anwendung mit HTML und PHP programmiert. Das ABC-System ist auf viele Rechner verteilt; teilweise aus praktischen Gründen (zum Beispiel für eine Eingabemöglichkeit für Atline-Messdaten direkt im Labor), teilweise aus Gründen der Ausfallsicherheit und Redundanz (Beispiel: der Datenbank-Server ist ein eigenständiger PC und wird mit keinen anderen Aufgaben betraut).

Die PLS-Funktionen, die das ABC-System enthält, umfassen unter anderem:

- Kommunikation mit dem MFCS
  - Erfassen der Messdaten vom MFCS
  - Senden von neuen Steuerungsbefehlen (notwendig für eine Prozessregelung)
- Kommunikation mit zusätzlichen Mess- und Steuerungsgeräten
- Datensicherung in die ABC-eigene MySQL-Datenbank
- Alarmierung per E-Mail bei Grenzwertüberschreitung von Messdaten
- Responsiveness Test (WatchDog-Funktion).

Die zentrale MySQL-Datenbank ist im ABC-System die Schnittstelle für alle beteiligten Computer: Die PLS-Funktion speichert in ihr alle online anfallenden Messdaten. Diese müssen nicht unbedingt vom MFCS kommen, sondern können auch aus anderen Quellen stammen. Alle Messgeräte und Anlagen werden soweit möglich so konfiguriert oder umgerüstet, dass eine Kommunikation mit einem PC erfolgen kann, der wiederum Teil des ABC-Systems ist. Andere Programme tragen atline oder auch offline verfügbare Daten ein. Computer, auf denen Regler-Algorithmen laufen, lesen die aktuellen Werte aus und schreiben neue Steuerbefehle in die Datenbank, die dann wiederum vom PLS gelesen und per OPC-Verbindung an das MFCS gesendet werden. Entsprechendes gilt natürlich auch für im voraus berechnete Trajektorien (Feeding-Profil), die beispielsweise im Rahmen einer Trajektorienplanung oder einer Optimalen Versuchsplanung berechnet wurden. Hier entfällt lediglich die Online-Aktualisierung der Steuerbefehle in der Datenbank.

Ein Beispiel für eine weitere Funktion des ABC-Systems ist ein selbst entwickeltes Automatisches Probennahmesystem (AProS): Es liest Informationen über die jeweils nächste geplante Probennahme aus der Datenbank, führt diese entsprechend aus und bestätigt dies in der Datenbank. Eine Speicherung erfolgt auch für manuell durchgeführte Probennahmen, sie muss allerdings händisch ausgelöst werden. Auf diese Weise werden geplante und tatsächlich durchgeführte Probennahmen dokumentiert, und im Falle des AProS auch mit einem exakten Zeitstempel versehen.

Die „höheren“ Funktionen des ABC-Systems beinhalten neben der in dieser Arbeit erläuterten automatisierten Modellentwicklung unter anderem die folgenden Funktionsbereiche:

- Steuerungen (Trajektorienplanung und Optimale Versuchsplanung (OVP))
- Regelungen (Nichtlineare modellprädiktive Regelung (NMPC), Online-OVP,...)
- Zustandsschätzer und -beobachter
- Modellbibliothek
- Messdatenanalyse mit biologischer Phänomenerkennung

Die Entwicklung des ABC-Systems ist noch nicht abgeschlossen; auch gegenwärtig wird das Software-Paket weiter ausgebaut. Ein paar der aktuellsten Erweiterungen sind:

- Simulation mit Unsicherheitsbeschreibungen auf Basis von Sigmapunkten von Kawohl [40].

- Darauf aufbauend hat Rossner eine Erweiterung für nicht-Gauß-verteilte Unsicherheiten programmiert, indem diese durch mehrere Gauß-Kurven approximiert werden [69].
- Für die Modellstrukturerkennung hat Herold die biologische Phänomenerkennung entwickelt und als Methode ins ABC-System integriert [32].

Die einzelnen Funktionen sind Matlab-Programme innerhalb des ABC-Systems und können durch erweiterte Versionen ausgetauscht werden, sofern Rück- und Übergabewerte beibehalten werden. Ebenso können neue Funktionen hinzugefügt werden. Insgesamt ist das ABC-System durch seinen Aufbau und seine umfassende Einbindung sowohl in den praktischen Laborbetrieb (Fermentationen) als auch in den theoretischen Methodenentwicklungsbereich eine enorm hilfreiche Umgebung geworden.

Abbildung 2.2 zeigt schematisch die inneren Zusammenhänge eines Teils des ABC-Frameworks und den sich daraus ergebenden Arbeitsablauf, ausgehend von einem Modellordner, dessen Inhalt später näher erläutert wird.

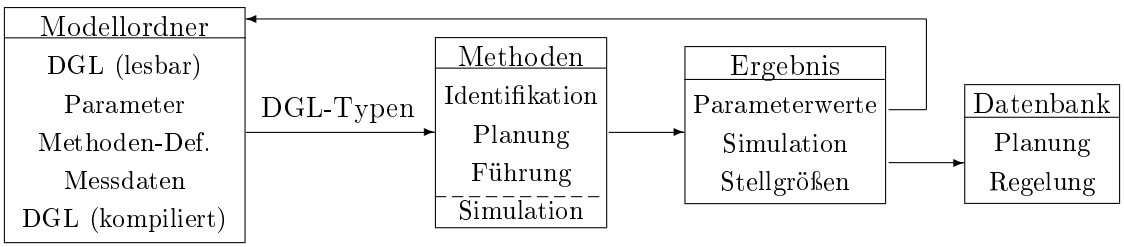


Abbildung 2.2: Matlab ABC-System: Schematischer Arbeitsablauf

Zu einem Modellordner gehören unter anderem die das betrachtete biotechnologische System beschreibenden Differentialgleichungen in lesbarer Form, die Modell-Parameter sowie die Methoden-Definitionen. Durch letztere wird in einer Datei `SystemInit` festgelegt, welche Arten von kompilierten Differentialgleichungssystemen erzeugt werden (siehe nächsten Abschnitt), wenn der Befehl `ABC_Build` ausgeführt wird, siehe Abbildung 2.3. Die kompilierten Dateien werden in einem Unterordner `auto` gespeichert. Der Konfiguration entsprechend können die verschiedenen Methoden anschließend eingesetzt werden, wobei die normale Simulation ein Bestandteil vieler Methoden ist. Je nach Methode erhält der Benutzer ein unterschiedliches Ergebnis, beispielsweise durch eine Planung eine Trajektorie und im Rahmen einer Regelung aktualisierte Stellgrößen. Beide Ergebnisse

werden in der Datenbank gespeichert. Wird eine Parameteridentifikation durchgeführt, erhält man neue Parameterwerte, die im Modellordner gespeichert werden und nach erneuten Aufruf von `ABC_Build` auch für andere Methoden zur Verfügung stehen.

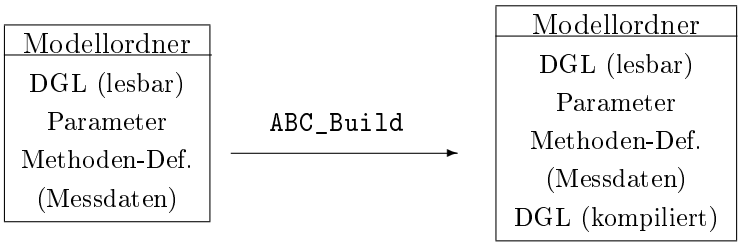


Abbildung 2.3: Erzeugen der kompilierten Modellgleichungen

Der Ausgangspunkt für alle Funktionen des ABC-Systems stellt daher der Modellordner und sein Inhalt dar, der im nächsten Abschnitt beschrieben wird. Es ist daher hilfreich, dessen zeitaufwendige und fehleranfällige manuelle Erstellung zu automatisieren. Hierzu wurde der Modellstrukturgenerator geschaffen.

2.1 Aufbau und Inhalt eines „Modellordners“

Im ABC-System werden Programme (=Methoden) und Modelle voneinander getrennt, das heißt in verschiedenen Ordnern gespeichert. Auf diese Weise wird erreicht, dass Programme wie zum Beispiel jenes für die Trajektorienplanung, mit einem einheitlichen Satz von Übergabe- und Rückgabewerten für beliebige Modelle eingesetzt werden können.

Der Fokus dieser Arbeit liegt jedoch nicht in der Beschreibung der vielseitigen ABC-Funktionen, sondern auf der Beschreibung der Methoden des Modellstrukturgenerators und in Teilen seiner informationstheoretischen Implementierung. Mit seiner Hilfe werden alle notwendigen Dateien für ein Modell automatisch erstellt. Der Benutzer muss zunächst keine zusätzlichen Angaben machen, sondern kann das Modell für Standardaufgaben gleich nutzen. Für einige Programme gibt es Konfigurationsdateien, deren Inhalte nicht automatisch erzeugt werden können, jedoch stellt der Modellstrukturgenerator dokumentierte Grundgerüste zur Verfügung, die der Benutzer verwenden kann.

Der typische Inhalt eines einzelnen Modellordners ist in Abbildung 2.4 dargestellt. Die aufgeführten Dateien sind als Matlab-Skriptdateien ausführbare Programme, doch im Wesentlichen handelt es sich um Konfigurationseinstellungen, die beim Starten des ABC-Systems und der Initialisierung eines Modellordners durchgeführt werden.

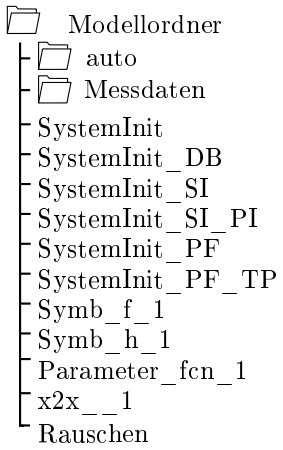


Abbildung 2.4: Grundbestandteile eines ABC-Modellordners

Durch die Verwendung der Skriptsprache ergibt sich der Vorteil, dass die Dateien für Menschen lesbar sind. Das ABC-System erstellt aus der gesamten Modellkonfiguration mehrere Programmvarianten, mit denen ein Prozess mit unterschiedlichen Zielen simuliert werden kann. Beispielsweise wird für reine Simulationen ein Differentialgleichungssystem verwendet, bei dem die Parameterwerte fest einprogrammiert sind, um die höchste Rechengeschwindigkeit zu erreichen. Für eine Parameteridentifikation hingegen müssen die Parameterwerte flexibel sein, daher wird ein Programm mit dem Differentialgleichungssystem erzeugt, bei dem diese als zusätzliche Funktionsargumente übergeben werden. Die ebenfalls automatisch erstellten Simulationsprogramme werden im Ordner `auto` gespeichert. Außerdem werden die verschiedenen Simulationsprogramme nicht nur in der Matlab-Skriptsprache verfasst, sondern auch in C. Letztere Dateien können nach ihrer Kompilierung, die ebenfalls automatisch durchgeführt wird, auf die gleiche Weise wie ihre Matlab-Variante verwendet werden. Sie eignen sich allerdings nicht mehr für Debugging-Zwecke, da sie für Menschen nicht mehr in lesbarer Form vorliegen. Im Gegenzug ist ihre Ausführungsgeschwindigkeit erheblich höher[1]. Über einen „Geschwindigkeitsschalter“ wird im ABC-System festgelegt, ob die Matlab- oder die C-Variante verwendet wird, wenn ein Modell aufgerufen wird.

Zu einigen der Dateien folgen allgemeine Erläuterungen, die für das Verständnis des Modellstrukturgenerators im Rahmen des ABC-Systems nützlich sind. Mit dem Unterordner `Messdaten` und Messdaten im ABC-System allgemein befasst sich der übernächste Abschnitt genauer.

- Die zentrale Datei `SystemInit` enthält allgemeine Informationen zu dem Modell, zum Beispiel Modellname, Anzahl der Modellkandidaten und die Einheit der Fermentationszeit, üblicherweise „Stunde“. Außerdem beinhaltet sie Anzahl, Namen und Typen der Zustände, Messgrößen und Stellgrößen (siehe nächsten Abschnitt), eine Definition der Probennahmetypen, typische Anfangswerte, Berechnungsformeln für X_0-Parameter (Abschnitt 3.4) und Einstellungen für Optimierer und Lö-

[1] Die Rechengeschwindigkeit ist von vielen Faktoren abhängig. Nach Untersuchungen von M. Kawohl ist jedoch die C-Variante gegenüber der Matlab-Version 20- bis 100-fach schneller (Mündliche Mitteilung).

ser, wie zum Beispiel die Auswahl eines speziellen Algorithmus, der standardmäßig eingesetzt werden soll. Ein kurzer, selbsterklärender Ausschnitt der `SystemInit` soll dies verdeutlichen. Die vollständige Datei findet sich im Anhang A.1.4.

SystemInit (Ausschnitt)

```
SYSTEM.ModellName   = 'P.polymyxa';
SYSTEM.Info         = 'Beste Anpassung für P.p., veröffentlicht CAB2010';
SYSTEM.ModellAnzahl = 72;
SYSTEM.tEinheit     = 'h';
% Anzahl der Zustandsgrößen
SYSTEM.n = 12;
% Anzahl der Messgrößen
SYSTEM.m = 8;
% Anzahl der Stellgrößen
SYSTEM.p = 10;
% Anzahl der diskreten Zustandsänderungsarten
SYSTEM.q = 4;

%% Def. der Anfangsbed.
SYSTEM.ZDGL.t0 = 0;
SYSTEM.ZDGL.x0 = [0;0;0;0;0;0;0;0;0;0;0;10];

SYSTEM.MGL.MessdatenFile = 'Messdaten_file';
SYSTEM.MGL.Cy            = @Rauschen;

SYSTEM.ZDGL.ParameterFile = ['Parameter_FCN_' num2str(mod_id)];
SYSTEM.ZDGL.fSymb         = ['symb_f_' num2str(mod_id)];
SYSTEM.MGL.hSymb          = 'symb_h_1';
SYSTEM.ZDGL.zFile         = 'x2x__1';
```

In den Unterdateien `SystemInit_<···>` werden Einstellungen zur Datenbank oder einzelnen ABC-Funktionen festgelegt. Teilweise können diese vom Modellstrukturgenerator automatisch nur als Grundgerüst, ohne funktionalen Inhalt, angelegt werden. Beispiele und Kommentare erleichtern dem Benutzer dennoch das korrekte Ausfüllen.

- Die Zustandsgleichungen des Differentialgleichungssystems stehen in den Dateien `Symb_f_<n>`, wobei der Namensbestandteil `_<n>` die vom Modellstrukturgenera-

tor vergebene Nummer des Modellkandidaten beinhaltet. Bei Modellordnern mit nur einem Modellkandidaten entfällt dieser Bestandteil. Um die Lesbarkeit zu erhöhen, werden automatisch sinnvolle Abkürzungen und Kommentare eingefügt. Dabei werden zum Beispiel die in der `SystemInit` definierten verständlichen Namen für Zustandsgrößen („DNA“ anstelle von „x(1)“) verwendet oder bei der Definition der Reaktionsraten auch die jeweils passende Reaktionsgleichung hinzugefügt.

Symb_f (Ausschnitt)

```
m_DNA          =  x(1);
...
% rDNA  (Nu --> DNA)
rDNA =  MuMax.rDNA ...
     * MiMe([g_As],[MiMe_.k.As.rDNA]) ...
     * MiMe([g_Nu],[MiMe_.k.Nu.rDNA]);
...
%% DGL-Gleichungen
 % m_DNA
xdot(1) = rDNA * m_DNA - rzDNA * m_DNA;
```

- Analog werden die Modellmessgleichungen in den Dateien `Symb_h_<n>` hinterlegt:

Symb_h (Ausschnitt)

```
m_X       =  x(1);
...
V         =  x(end);

c_X           =  m_X / V;
...

...
y(6) = c_X;
```

- In `Parameter_fcn_<n>` werden alle Informationen über die Parameterwerte der einzelnen Modellkandidaten gespeichert. Aufgeführt werden unter anderem die

Namen der einzelnen Parameterwerte sowie deren Anfangswerte für eine Parameteridentifikation, gefolgt von der Auswahl der Parameter, die in einer Parameteridentifikation variiert werden können[2], und für diese auch der jeweils erlaubte Gültigkeitsbereich. Des Weiteren wird festgelegt, welche Parameter als versuchsabhängige X_0-Parameter anzusehen sind (Abschnitt 3.4).

Parameter_Fcn (Ausschnitt)

```
MiMe.k.As.rDNA                          =      0.10000000;
...

%% Definition der freien Parameter
ParNames = { ...
'MiMe.k.As.rDNA'
...
};

%% CONS
Parameter.CONS.min = [ ...
    0.01000000          ,  ... % MiMe.k.As.rDNA
...
];

Parameter.CONS.max = [ ...
   50.00000000          ,  ... % MiMe.k.As.rDNA
...
];
```

- Die Definitionen für die diskreten Zustandsänderungen (siehe Abschnitt 2.4) werden in der Datei `x2x__1` gespeichert. Sie legen die Auswirkungen von Probennahmen auf die Systemzustände fest.

  Die Zustandsänderungen gelten für alle Modellkandidaten innerhalb einer Modellfamilie, da alle diese Kandidaten dieselben Zustände beinhalten. Für zukünftige Erweiterungen, durch die dies nicht mehr gilt, wird das Namensschema `_<n>` für diese Datei bereits verwendet.

[2] Die restlichen Parameter sind echte Konstanten.

- Für einige Methoden der Regelungstechnik sind Informationen über die Genauigkeit der Messgrößen notwendig, beispielsweise zur Gewichtung der Messgrößen bei einer Parameteridentifikation. Für jede Messgröße wird die entsprechende Varianz in der Datei `Rauschen` eingetragen. Für häufig verwendete Messgrößen sind Standardwerte vordefiniert. Während der Modellerzeugung durch den Modellstrukturgenerator gibt es eine Kontroll- und Eingabemöglichkeit für die einzelnen Einträge. Wurden manuell virtuelle Messgrößen, beispielsweise zur Modellanalyse Teile des Differentialgleichungssystems, definiert, muss die Datei entsprechend ergänzt werden. Auch für Messgrößen, zu denen es prinzipiell keine Messwerte gibt, muss aus programmtechnischen Gründen ein Eintrag vorhanden sein, auch wenn er später nicht verwendet wird. Bei den zellinternen Konzentrationen wird beispielsweise automatisch der Wert eins eingetragen. Eine null ist hingegen aus Kompatibilitätsgründen nicht gestattet.

2.2 Zustands- und Messgrößen-Typen

Die ausschließliche Verwendung von Namen für die verschiedenen Größen in Modellen bringt viele Probleme mit sich. Alle Programme und Funktionen, die innerhalb des Rechnerverbunds des ABC-Systems Informationen austauschen, sind davon abhängig, dass überall die gleichen Bezeichnungen verwendet werden.

Zur Lösung dieses Problems wurden für die einzelnen Modellgrößen definierte Typencodes eingeführt. Ein solcher Typ besteht aus einem oder mehreren alphanumerischen Zeichen, deren Reihenfolge beliebig ist. Jedes einzelne Zeichen kodiert eine Eigenschaft. Somit entfällt der (ausschließliche) Vergleich von Klarnamen. Für Zustandsgrößen wird der jeweilige Typ in einer Variable `xTyp` gespeichert, für Messgrößen in `yTyp`, Stellgrößen in `uTyp` und für Probennahmen in `zTyp`. Beispielsweise erhält eine Zustandsgröße, die für ein Substrat steht, den Code `S`. Handelt es sich genauer um ein Kohlenstoff-Substrat, zum Beispiel Glucose, so lautet der Code `SC`, wobei die Reihenfolge der einzelnen Zeichen beliebig ist. Benutzer müssen sich somit nicht mehr an genaue Schreibweisen halten und Programme müssen nicht mehr nach vielen Varianten von möglichen Schreibweisen für den Namen dieses Zustands suchen, sondern nur nach einer Zustandsgröße mit dem `xTyp SC` (oder `CS`). Andere, gängige Zustandsgrößen, die in den verschiedenen Kapitels dieser Arbeit auch als Beispiel verwendet werden, und deren `xTyp`en sind Biomasse (`X`), Kompartimente der Biomasse (`K`) und das Volumen (`V`).

Die gefundene Lösung stellt gleichzeitig einen wichtigen Schritt in der Automatisierung dar, wie anhand eines Beispiels im folgenden Abschnitt 2.3 näher erläutert wird. Auch das Programm des Modellstrukturgenerators nutzt Typencodes in großem Umfang.

Eine vollständige Liste der aktuell definierten Codes für Messgrößen (`yTyp`) und Zustandsgrößen (`xTyp`) zeigen Tabellen 2.1 und 2.2. Die Liste der Zustandstypen ist fast identisch mit der Liste der Messgrößentypen. Tabelle 2.2 ergänzt die Zustandstypen. Die Stellgrößentypen (`uTyp`) und Probennahmetypen (`zTyp`) finden sich im Anhang A.2.

Messgröße	**Typ**	**Messgröße**	**Typ**
Substrat	S	Biomasse	X
Kompartiment	K	Interner Speicher	I
Zielprodukt	Z	Gas	G
Volumen	V	pH-Wert	H
Rührerdrehzahl	M	Luftzufuhr	L
Antischaummittel	F	Temperatur	T
Druck	d	OD-Wert	o
Aktive Biomasse	A	DNS	D
RNS	R	Proteine	p
Stickstoffquelle	N	Phosphorquelle	P
Kohlenstoffquelle	C	Energiequelle	z
Enzymaktivität	E	Masse	m
# der Substratquelle	[1-9]	Virtuelle Messgröße	-
Waagenfehler	e	Konzentration	c
Zellinterne Konz.	g	Molmenge	n
Gas	v	Flüssigkeit	l

Tabelle 2.1: Messgrößentypen (yTyp) und Zustandstypen (xTyp)

Zustandsgröße	**Typ**	**Zustandsgröße**	**Typ**
Enzymaktivität	E	Masse	m
Energiequelle	z	Gesamtzellmasse	X
Aktive Zellmasse	KAX		

Tabelle 2.2: Zusätzliche Zustandsgrößentypen (xTyp), ergänzend zu Tabelle 2.1

2.3 Messdaten

Ein Modellordner enthält auch einen Unterordner namens `Messdaten`. Hier werden Dateien gespeichert, die jeweils die Messdaten eines Experiments enthalten. Programme können auf die darin enthaltenen Messwerte zugreifen. Zusätzlich werden Metadaten gespeichert, die weitere Details zu den Versuchen enthalten. Beispielsweise können Informationen darüber gespeichert werden, welcher Zeitraum der Datenaufzeichnung zum Beispiel für eine Parameteridentifikation genutzt werden soll. Die Dateien werden durch ABC-Funktionen erzeugt, die die Versuchsdaten direkt aus der MySQL-Datenbank lesen und in Form von Matlab-Skriptdateien abspeichern.

Die Messdaten, die aus der Datenbank stammen, können durch spezielle Dateien in diesem Ordner ergänzt und verändert werden. Auf diese Weise können Messwerte, die von Geräten ohne Verbindung zur Datenbank stammen, schnell genutzt werden. Langfristig sollten diese Messdaten allerdings in die Datenbank mit Hilfe von Eingabeprogrammen überführt werden. Die vorhandenen Daten können auch temporär verändert werden, ohne die originalen Datenbankeinträge zu verändern.

Messdaten, die im ABC-System verwendet werden, sind in einer Strukturvariablen zusammen mit Metadaten, die den einzelnen Versuch beschreiben, angeordnet, siehe unten. Die Rohdaten sind darin in der Feldvariablen `.Messdaten` gespeichert, die Modellmessdaten sind in `.y` als Matrix angeordnet, deren Spalteneinträge für jeweils einen Zeitpunkt gelten und deren Zeilen die verschiedenen Messgrößen darstellen. Die Größe der Matrix ist somit abhängig von der Anzahl und Reihenfolge der in `SystemInit` definierten Modellmessgrößen.

Messdatenstruktur

```
Identifikationsversuch: 1
             Messdaten: [1x39 struct]
    Messdatenbewertung: [1x1 struct]
                  Name: 'SCdef_AH22_1'
             PlotStyle: 'ko'
           Probennahmen: [1x2 struct]
              Rohdaten: [1x38 struct]
         Stellgroessen: [1x10 struct]
  Verifikationsversuch: []
```

```
Versuchsgewicht: 1
              W: [10x721 double]
      globalInf: [1x1 struct]
              t: [1x721 double]
              u: [11x320 double]
         uOrder: 0
              x: [10x721 double]
             x0: [10x1 double]
    x0Parameter: [1x1 struct]
           xTyp: {'X' 'SN' 'SP' 'SC' 'Z' '' '' 'G' '' 'V'}
              y: [10x721 double]
       yGewicht: [10x1 double]
    y_Parameter: []
              z: [5x84 double]
```

Die Rohdaten aus der Datenbank können daher nicht direkt in die y-Matrix überführt werden, da sich die Anzahl und die Reihenfolge der Messgrößen im Modell vom Inhalt der Datenbank typischerweise unterscheidet. Bereits im letzten Abschnitt wurde erläutert, dass eine Zuordnung nach Namen sich in der Praxis als umständlich und fehleranfällig erwiesen hat: Jeder Modellordner benötigte für die Messdaten eine eigene Zuordnungstabelle. Daher hat der Autor eine Funktion programmiert, die diese modellabhängige Zuweisung automatisch vornimmt: `Messdatenfile_template`

Die Grundlage für die Automatisierung ist, dass die Messwertbezeichnungen in der Datenbank vom PLS eingetragen werden und diese sich nicht oder zumindest sehr viel seltener ändern, als in den Modellen. So wird beispielsweise der Feed einer Kohlenstoffquelle als „Vdot_soll_C“ bezeichnet, und ist unabhängig davon, ob es sich um Glucose, Saccharose oder andere kohlenstoffhaltige Substanzen handelt. Diese Vereinfachung ist möglich, da sich diese Arbeit auf Fermentationen mit Minimalmedien mit einzelnen Quellen konzentriert. Weiterhin sind den Bezeichnungen im Modell charakterisierende Typen zugeordnet, siehe letzten Abschnitt, so dass für die Zuordnung nicht mehr die Modellnamen verwendet werden. Stattdessen wird ein automatisierter Typenvergleich durchgeführt. Neben der reinen Zuordnung übernimmt die Automatik auch die Konvertierung des Datenformats von den Rohdaten aus der Datenbank zu der im ABC-System verwendeten Datenstruktur für Messdaten.

In einigen Fällen ist eine automatische Zuordnung nicht möglich oder nicht eindeu-

tig, dann greift in `Messdatenfile_template` ein Fall-Back-Mechanismus ein, der nach ähnlichen Namen sucht und sie dem Benutzer zur Auswahl vorschlägt.

Abgesehen von der Reduzierung fehlerhafter Zuordnungen, erlaubt es diese Automatisierung die Anzahl der Zustände oder Messgrößen im Modell ohne größeren Aufwand zu ändern: Die Automatik sucht im nächsten Messdatenladevorgang nur die jeweils benötigten Werte aus dem Messdaten-File ein und ordnet sie korrekt an.

Die Automatik zum Einlesen von Messdaten ist ein zentraler Bestandteil des ABC-Systems geworden. Daher wurden in das Programm im Laufe der Zeit weitere Funktionen integriert: Plausibilitätstest für die Messdaten (Vergleich zwischen Startwerten und ersten Messwerten), die Berechnung der Gewichtungsmatrix **W** für Parameteridentifikationen und eine Datenvorverarbeitung zur optional automatischen Reduzierung sehr großer Stellgrößen-Matrizen, um die Simulation zu beschleunigen (siehe Abschnitt 3.1.3).

2.4 Diskrete Zustandsänderungen durch Probennahmen

Eine Besonderheit des ABC-Systems sind die diskreten Zustandsänderungen bei der Modellsimulation. Anstatt eine Probennahme einer Stellgröße zuzuordnen, um die Volumenänderung wiederzugeben, wurden diskrete Zustandsänderungen eingeführt. Die Idee dabei ist, dass die Zeit, die während der Probenentnahme vergeht, vernachlässigbar gering im Vergleich zur Prozesslaufzeit ist. Zwei Fälle müssen dabei jedoch unterschieden werden: Bei einer Vollprobe ändern sich das Volumen und die Massen aller Substanzen gleichermaßen. Eine Konzentrationsänderung tritt also nicht auf. Wird eine Überstandsprobe (ohne Biomasse) entnommen, so ändert sich die Masse der Mikroorganismen im Gegensatz zu den übrigen Bestandteilen des Mediums nicht. Da das Volumen sinkt, steigt sogar die Biomassen-Konzentration. Ebenso sind zellinterne Zustände, wie sie bei strukturierten Modellen auftreten, zu behandeln. Um diese Besonderheiten zu berücksichtigen, wird daher im ABC-System zwischen diesen zwei Probennahmearten unterschieden[3]. Der Modellstrukturgenerator erstellt die Dateien für die diskreten Zustandsänderungen automatisch. Mit den Volumina für ungefilterte Vollproben (V_{Voll}) und

[3] Genaugenommen sogar vier, da für automatische Probennahmesysteme zusätzlich zwischen automatischen und manuellen Proben unterschieden wird.

(gefilterten) Überstandsproben ($V_{\text{Über}}$) und der Masse x_i der i-ten Größe lauten diese:

$$x_{i,\text{neu}} = x_{i,\text{alt}} \cdot (1 - (V_{\text{Voll}})/V_{\text{alt}}) \qquad i : \text{Biomasse-Zustände} \tag{2.1}$$

$$x_{j,\text{neu}} = x_{j,\text{alt}} \cdot (1 - (V_{\text{Voll}} + V_{\text{Über}})/V_{\text{alt}}) \qquad j : \text{sonstige Massenzustände inkl. Volumen} \tag{2.2}$$

Bei der Planung von Versuchen werden die gewünschten oder technisch vorgegebenen Probenvolumina in die Datenbank eintragen. Der vorgegebene Zahlenwert für eine manuelle Probennahme wird durch das tatsächlich entnommene Volumen nachträglich aktualisiert. Massenfreie Zustände, die beispielsweise eine Enzymaktivität oder den pH-Wert abbilden, werden automatisch erkannt und von den diskreten Zustandsänderungen ausgeschlossen (sie sind nicht in i oder j in Gleichung 2.1 und 2.2 enthalten).

Durch die Einführung diskreter Zustandsänderungen können übliche Lösungsalgorithmen für Differentialgleichungen nicht mehr direkt verwendet werden. Stattdessen wird ein Zwischenschritt eingeführt, in welchem der Gesamtzeitraum der Simulation in einzelne Zeitintervalle unterteilt wird. Jedes Intervall wird einzeln gelöst. Nach jedem Intervall wird der erhaltene Lösungsvektor durch die oben angegebenen Formeln modifiziert, bevor er wieder als Startwertvektor für das nächste Intervall verwendet wird. Wie später erläutert, benötigt dieses Neustarten der Löser-Algorithmen allerdings zusätzliche Rechenzeit. Dem (einmaligen) Programmieraufwand zur Berücksichtigung der diskreten Zustandsänderungen steht allerdings der Vorteil gegenüber, dass die unterschiedlichen Probennahmen nicht als zusätzliche Stellgrößen in die Modelle eingeführt werden müssen. Die Modelle sind daher einfacher.

2.5 Stellgrößen

Die Stellgrößen werden im ABC-System standardmäßig als Treppenfunktion dargestellt. Daneben sind auch Rampen (stückweise lineare Verläufe) und quadratische Splines möglich. Der Vorteil der Treppenfunktion liegt in ihrer einfachen Implementierung und Überwachung. Zu jedem Zeitpunkt ist der Wert jeder Stellgröße eindeutig bestimmt (und für Menschen leicht überprüfbar). Auf die Vorteile der anderen beiden Darstellungsarten wird weiter unten eingegangen.

Alle Stellgrößeninformationen werden in einer Matrix $\mathbf{U}$ vereint. Der Aufbau der Matrix ist unabhängig von der gewünschten Ordnung des Interpolationsverfahrens. Gleichung 2.3 zeigt ein Beispiel für eine Stellgrößenmatrix $\mathbf{U}$. Die erste Zeile enthält die Zeitpunkte (angegeben als Fermentationszeit in der in `SystemInit` definierten Zeitein-

heit), von denen ab die neuen Stellgrößen gelten sollen. Darunter folgen in diesem Beispiel für zwei Feeds jeweils zwei Zeilen mit der Zudosierkonzentration und -rate (c_{feed} und u_{feed}). Für andere Stellgrößen wie Temperatur, Rührerdrehzahl oder pH-Wert wird nur eine Zeile benötigt. Für die Auswertung im Löser wird bei der Treppenfunktion („zero-order-hold“) die jeweils aktuelle Spalte ermittelt und die ab dort gültigen Werte übernommen. Bei Rampen werden die beiden umliegenden Spalten bestimmt und eine lineare Interpolation entsprechend der aktuellen Zeit berechnet. Die Ermittlung bei Splines erfolgt analog.

Die Einbindung in das Modell erfolgt wie in Gleichung 3.19 auf Seite 43 dargestellt. Die Definition für die Stellgrößen erfolgt in der `SystemInit`, die Zustandsgleichungen mit den Einflüssen der Stellgrößen sind in `Symb_f` festgelegt.

$$\mathbf{U} = \begin{array}{c} \begin{matrix} t_1 & t_2 & t_3 & \end{matrix} \\ \begin{bmatrix} 0 & 1 & 2 & \dots \\ 10 & 10 & 10 & \dots \\ 0.2 & 0.4 & 0.5 & \dots \\ 50 & 50 & 50 & \dots \\ 0 & 0 & 1.2 & \dots \end{bmatrix} \end{array} \begin{matrix} t \\ c_{\text{feed1}} \\ u_{\text{feed1}} \\ c_{\text{feed2}} \\ u_{\text{feed2}} \end{matrix} \tag{2.3}$$

Wie auch bei den diskreten Zustandsänderungen, unterbricht das ABC-System den Löser künstlich auch zu jedem Zeitpunkt, der in der Stellgrößenmatrix **U** aufgeführt wird. Da die Berechnungsdauer einer Modellsimulation somit von der Anzahl der Unterbrechungen abhängt, ist es von Interesse die Anzahl der Spalten zu beachten. Sowohl bei der Parameteridentifikation als auch beispielsweise bei der Trajektorienplanung sind häufig viele Simulationen eines Modells notwendig. Somit wirken sich bereits kleinere zeitliche Änderungen signifikant auf die Gesamtdauer der Berechnung aus.

Die Anzahl der Stellgrößenänderungen (=Optimierungsparameter) bei einer Trajektorienplanung ist normalerweise eher gering, wenn sie dem Prozess angemessen ist, und stellt die Algorithmen zur Simulation der Modelle nur vor geringe Herausforderungen. Bei der Treppenfunktion sind typischerweise 30 Stufen noch unproblematisch. Geht die Anzahl darüber hinaus, ist möglicherweise eine der anderen Darstellungsarten zur Beschreibung der Zufütterungsprofile besser geeignet. Sind beispielsweise ansteigende oder fallende Zudosierraten über einen langen Zeitraum gewünscht oder notwendig, kann durch den Einsatz von Rampen oder Splines die Anzahl der jeweils für die Festlegung

notwendigen Spalteneinträge deutlich geringer ausfallen. Da bei der Trajektorienplanung die Einträge der Stellgrößenmatrix **U** genau die Optimierungsparameter sind, hat dies nicht nur Einfluss auf die Rechenzeit einer Simulation, sondern generell auf die Größe des Optimierungsproblems.

Bei einer Parameteridentifikation hingegen sind nicht die geplanten, sondern die tatsächlich umgesetzten Stellgrößen relevant. Handelt es sich nämlich um einen geregelten Versuch, können (meist minimale) Stellgrößenänderungen mehrfach pro Minute auftreten und zu Stellgrößen-Matrizen mit vielen tausend Spalten führen. Dies führt zu einer signifikanten Vergrößerung des zeitlichen Rechenaufwands pro Simulation. Um die benötigte Rechenzeit für eine Simulation bei einer Parameteridentifikation (oder anderen Methoden) wieder zu senken, existiert im ABC-System eine Optimierungsroutine für die Treppenfunktion, die die Anzahl der Spalten von **U** verlustarm reduziert. Hierbei werden jeweils mehrere aufeinander folgende Einträge für Zudosierraten durch einen einzelnen Wert zusammengefasst, sofern keine weiteren Änderungen in den übrigen Elementen der betroffenen Spalten auftreten. Der neue Wert wird durch eine Optimierung so bestimmt, dass die im jeweils betrachteten Zeitraum zudosierten Mengen nur minimal von den tatsächlichen Werten abweichen. In der Praxis konnte so die Anzahl der Spalten verschiedener Matrizen von über 4000 auf unter 500 reduziert werden – mit einem Fehler $\ll 1$ ‰ [4]. Als Kontrollmöglichkeit für den Benutzer wird automatisch der numerische Fehler ausgegeben und zusätzlich die originalen Stellgrößenverläufe gemeinsam mit den modifizierten grafisch dargestellt.

Der rechnerische Aufwand für die zusätzlich notwendigen Zwischenberechnungen bei der Simulation steigt mit der Ordnung der Interpolation an. Da in der Praxis keine Vorteile beim Einsatz von Ordnungen größer zwei zu erkennen waren, wurde im ABC-System auf die Implementation höherer Ordnungen verzichtet. Es können nur maximal quadratische Splines ausgewählt werden.

Zusätzlich kann der Berechnungsaufwand für eine Simulation speziell bei Verfahren zur Stellgrößenoptimierung reduziert werden, wenn beachtet wird, dass eine Stellgrößenänderung zu einem Zeitpunkt t_n zu keiner Änderungen der Zustandsverläufe vor diesem Zeitpunkt führen kann. Ist der Verlauf bis t_n durch einen vorherigen Aufruf bereits bekannt, muss durch den ODE-Löser nur noch der Verlauf ab t_n berechnet werden. Insbesondere bei Gradientenberechnungen im Rahmen zum Beispiel einer Trajektorienplanung kann diese Situation häufig auftreten, so dass bei gradientenbasierten Optimie-

[4] Die Angabe bezieht sich auf das gesamte Zufüttervolumen.

rungsverfahren durch die vorgeschlagene Modifikation der größte Zeitgewinn zu erreichen ist. Allerdings verwenden die meisten Optimiereralgorithmen integrierte, hochoptimierte Methoden zur Bestimmung von Gradienten. Häufig können diese nicht modifiziert werden, stattdessen muss die gesamte Gradientenberechnung neu programmiert werden. In der Vergangenheit ist die vorgeschlagene Modifikation anhand einiger Beispiele erfolgreich getestet worden. Zukünftig sollte aktuell verwendete Optimierer im ABC-System ebenso modifiziert werden.

Zuletzt kann hinzugefügt werden, dass die Matrix **U** auch negative Werte für die Zudosierraten enthalten darf. Wie im letzten Abschnitt beschrieben, werden die Volumen- und Massenverluste durch Probennahmen im Modell durch zeitdiskrete Zustandsänderungen berücksichtigt. Wenn die Zeitdauer, in der solche Änderungen stattfinden, hingegen nicht vernachlässigt werden darf, können negative Volumenströme in die Stellgrößenmatrix für die betreffenden Zeiträume eingetragen werden. Tatsächlich wird von dieser Möglichkeit bereits für offene Fermentersysteme[5] Gebrauch gemacht. Da in offenen Systemen ständig Wasserdampf als Bestandteil des Abgases austritt, reduziert sich das Volumen des Fermentermediums, abgesehen von Substratzufütterungen und Probennahmen, kontinuierlich. Daher wird automatisch durch den Modellstrukturgenerator eine Stellgröße „Abdampf“ zu dem Modell mit einem negativen Massenstrom hinzugefügt.

Auf die im Modellstrukturgenerator betrachteten Modelle, die theoretischen Grundlagen der Identifikation der Parameter und der Optimierung wird im folgenden Kapitel eingegangen.

[5] Offen ist in dem Sinne gemeint, dass Abgas kontinuierlich den Fermenter verlässt.

3 Modelle, Parameteridentifikation und Optimierung

Essentially, all models are wrong, but some are useful.

(George E. P. Box, 1978)

Technische Prozesse in der Verfahrenstechnik beschreiben Umwandlungsvorgänge von Edukten zu Produkten. Die Umwandlung kann dabei chemisch-physikalischer oder biologischer Natur sein. Es ist nicht Ziel der Verfahrenstechnik, mögliche neue Umwandlungen zu entdecken, sondern bekannte technisch zu realisieren. In der Regel müssen hierfür bestimmte Bedingungen geschaffen werden, die einen solchen Prozess ermöglichen. Für die technische Umsetzbarkeit und Wirtschaftlichkeit eines Prozesses ist die Kenntnis solcher Bedingungen wichtig und wird hier vorausgesetzt. Stattdessen konzentriert sich diese Arbeit auf die Optimierung von insbesondere Fed-Batch-Prozessen und auf die dafür erforderlichen mathematischen Modelle.

In der Biotechnologie werden Fed-Batch-Prozessführungen meist aus zwei Gründen eingesetzt. Erstens, wenn eine Zufütterung von flüssigen Substraten notwendig ist. Dies ist beispielsweise der Fall, wenn es gilt, Eduktkonzentrationen unterhalb eines toxischen Bereichs zu halten. Oder, zweitens, wenn beispielsweise für die Bildung eines Sekundärmetaboliten die Umgebungsbedingungen (in Form der Nährstoffkonzentrationen) des Mikroorganismus kontrolliert verändert werden müssen. Um diese Ziele zu erreichen, sind öfters komplexe Feeding-Strategien wünschenswert, die es zunächst aufwendig zu ermitteln gilt. In der Regel ist hierzu ein Optimierungsprozess durchzuführen, für welchen dynamische Modelle erstellt werden müssen, die wiederum auch für die Regelung von Fed-Batch-Prozessen eingesetzt werden können. Um die Parameterwerte in den Modellen bestimmen zu können, sind häufig umfangreiche Messungen notwendig.

Die vorliegende Arbeit leistet einen unterstützenden Beitrag speziell zur Modellbildung bei Fed-Batch-Fermentationen. Wie noch erläutert wird, ist bei diesen ein hoher

Modellierungsaufwand notwendig, um das Potential biotechnologischer Prozesse ausnutzen zu können. Im Folgenden beziehen sich daher die Aussagen vor allem auf Fed-Batch-Fermentationen. Bevor auf die eigentliche Modellbildung in Abschnitt 3.1 eingegangen wird, sollen zunächst ein paar allgemeine Grundlagen zur Prozessoptimierung erörtert werden, die später vertieft werden.

Der Betreiber einer Anlage oder der Benutzer legt die Kriterien fest, bezüglich derer ein Prozess zu optimieren ist. Diese k Optimierungskriterien müssen ausgewählt und in einem (skalaren) Gütefunktional Φ formuliert werden. Die verschiedenen möglichen Prozessführungen werden so quantifiziert miteinander vergleichbar.

$$\Phi : \mathbb{R}^k \to \mathbb{R} \tag{3.1}$$

In dieser Arbeit konzentriert sich die Prozessoptimierung auf die Prozessführung, genauer auf die Wahl der Anfangsbedingungen ($\underline{x}_0$) und das Zufütterungsprofil ($\mathbf{U}$) von Nährstoffen. Diese werden in den nächsten Abschnitten genauer erläutert und zunächst zusammengefasst in der Optimierungsvariablen $\underline{\Theta}$. Die Güte Φ eines (modellierten) Prozesses ist somit abhängig von $\underline{\Theta}$.

Im einfachsten Fall ist das Prozessziel und damit das Optimierungskriterium die Masse eines Produktes (m_{Produkt}) am Ende der Fermentation (t_{end}), welche zu maximieren ist. Der Wert für m_{Produkt} ist natürlich ebenfalls von $\underline{\Theta}$ abhängig, was aus Gründen der Übersichtlichkeit zunächst hier nicht explizit in den Gleichungen aufgeführt wird. Der Zusammenhang wird im nächsten Abschnitt 3.1 genauer erläutert. Allgemein formuliert ist die Güte die Differenz zwischen Nutzen und Aufwand. Letzteres könnte beispielsweise die Masse der verbrauchten Substrate sein:

$$\Phi = \Phi(\underline{\Theta}) = -\left(m_{\text{Produkt}}(t_{\text{end}}) - \sum m_{\text{Substrate}}\right) \tag{3.2}$$

Um beispielsweise verschiedene Kosten beziehungsweise Preise der einzelnen Stoffe zu berücksichtigen, gehen üblicherweise die Massen der einzelnen Stoffe gewichtet in das Gütefunktional ein, was in Gleichung 3.2 der Einfachheit halber nicht dargestellt wurde.

Die numerische Optimierung sucht nach der besten Güte, um die optimale Prozessführung $\underline{\Theta}^*$ zu finden. Wie später in Kapitel 3.5 ausgeführt wird, suchen Optimierungsalgorithmen häufig nach einem Minimum. Die Gütefunktionale werden daher mit einem negativen Vorzeichen versehen. Das allgemeine Optimierungsproblem lautet daher:

$$\underline{\Theta}^* = \arg\min_{\underline{\Theta}} \Phi(\underline{\Theta}) \tag{3.3}$$

Prinzipiell gibt es zwei Arten der Prozessoptimierung. Die erste basiert auf einer statistischen Beschreibung, die zweite auf Modellen. Im Folgenden werden beide beschrieben:

Bei der **statistischen Versuchsplanung** (engl.: *Design of Experiments (DOE)*) wird eine Anzahl von Versuchen mit unterschiedlichen Prozessführungen durch eine geeignete Interpolation ausgewertet. Mit Hilfe dieser Interpolation werden die optimalen Werte für die einzelnen Faktoren der Prozessführung ermittelt[1]. Die Anzahl v der notwendigen Versuche steigt mit der Anzahl der Faktoren f exponentiell: $v = s^f$, wenn s die Anzahl der Versuche für jeden Faktor ist. Für die Auswahl der Versuche gibt es optimierte Methoden, die sogenannten Teilfaktorpläne, bei denen die Anzahl der notwendigen Versuche reduziert wird. Trotzdem steigt die Anzahl exponentiell mit der Faktorzahl. Daher wären für die Optimierung einer Fed-Batch-Prozessführung mit ihrer großen Zahl von Faktoren sehr viele Versuche notwendig. Dies ist einerseits aus Zeitgründen nicht praktikabel, andererseits aus ökonomischen Gründen nicht vertretbar, da die Durchführung jedes einzelnen Versuchs an einer realen Anlage in der Regel mit hohen Kosten verbunden ist. Auch steht die Anlage in dieser Zeit nicht für die eigentliche Produktion zur Verfügung. Daher verlieren statistikbasierte Optimierungsverfahren im industriellen Maßstab für die Optimierung von Fed-Batch-Prozessen an Bedeutung. Dies gilt besonders für biotechnologische Prozesse, weil die Reaktion im Inneren von Mikroorganismen nicht durch die in diesen Verfahren implizit angenommenen linearen, quadratischen (oder teils auch polynomialen) Zusammenhänge beschrieben werden können. Die Anzahl der Versuche s pro Faktor müsste stark erhöht werden, um eine Verbesserung durch stückweise Approximation zu erreichen.

Der offensichtlichen Lösung, die Vielzahl notwendiger Experimente aus ökonomischen Gründen in kleinerem Maßstab und zur Verringerung des Zeitaufwands hoch parallel durchzuführen, sind ebenfalls Grenzen gesetzt. Das notwendige Scale-Up auf den Produktionsmaßstab zeigt in der Praxis nicht selten, dass optimale Bedingungen und Prozesse im Labormaßstab zu nicht befriedigenden Ergebnissen im Produktionsmaßstab führen. Gleiches gilt für das Scale-Down, um den Industrieprozess im Labormaßstab nachstellen zu können.

Trotz der kritischen Einschränkungen für Fed-Batch-Prozesse finden DOE-Verfahren weiterhin sinnvolle Anwendungen in der Prozessoptimierung. Sie helfen bei der Wahl konstanter Versuchsbedingungen wie Temperatur oder pH-Wert, oder der Suche nach

[1] Faktoren sind die einzelnen Einflussgrößen, so ist beispielsweise jeder einzelne Anfangswert (pH-Wert, Temperatur, Konzentrationen, etc.) ein Faktor. Bei Fed-Batchen mit treppenförmigen Zufütterungsprofilen ist jede einzelne „Stufe“ ein einzelner Faktor (pro Substrat).

optimalen Startkonzentrationen für Batch- und Fed-Batch-Prozesse. Im Rahmen dieser Arbeit waren sie jedoch nicht notwendig, da die Informationen als bekannt vorausgesetzt wurden[2].

Bei der **modellbasierten Optimierung** werden individuelle mathematische Modelle eingesetzt, um die Nachteile der statistikbasierten Optimierung zu umgehen. Mit Hilfe der Modelle, sowie entsprechender Algorithmen, wird der Prozessverlauf unter Berücksichtigung der gewählten Prozessbedingungen einfach *in silico* simuliert. Änderungen in der Prozessführungsstrategie können somit mit vergleichsweise geringem Aufwand auf ihre Eignung hin überprüft werden. Die Änderungen des Gütefunktionalwerts sind direkt berechenbar. Darüber hinaus stehen auch viele mathematische Methoden der Regelungstechnik zur Bestimmung optimaler Prozessführungsstrategien zur Verfügung, die auch automatisiert werden können.

Um die Vorteile modellbasierter Verfahren nutzen zu können, müssen die Modelle zunächst entwickelt werden. Dies ist ein aufwendiger und fehlerträchtiger Entwicklungsprozess – unter anderem deshalb sind modellbasierte Verfahren noch nicht Standard in der Industrie. Dabei ist das Aufstellen in der (anorganischen) chemischen Industrie vielfach eine einfachere Aufgabe: Viele Prozesse sind durch Einzelschritte modellierbar, die ihrerseits gut durch physikalisch fundierte Gleichungen beschrieben werden können. So lässt sich die Richtung reversibler chemischer Reaktionen durch das Massenwirkungsgesetz ermitteln[3]. Einzelne Elementarreaktionen lassen sich einfach beschreiben. Bei Reaktionen zweiter Ordnung sind zum Beispiel die Reaktionsgeschwindigkeiten r abhängig vom Produkt der beiden Eduktkonzentrationen c_1 und c_2 und von der Temperatur T. Der Einfluss der Temperatur wird durch die Arrhenius-Gleichung[4] beschrieben:

$$r = k \cdot c_1 \cdot c_2, \quad \text{mit} \quad k = A \cdot \exp\left(-\frac{E_A}{R \cdot T}\right) \tag{3.4}$$

A ist in der ursprünglichen Arrhenius-Gleichung ein temperaturunabhängiger Vorfaktor, E_A die Aktivierungsenergie und R die universelle Gaskonstante.

Technisch interessante Umsetzungen laufen auch in der Chemie vielfach über eine Vielzahl von Elementarreaktionen, deren genaue Struktur und deren Parameter nicht

[2] Tatsächlich wurden sie von Projektpartnern zur Verfügung gestellt.

[3] Gleichgewichtskonstante $K = \prod_{i=1}^{n} a_i^{v_i}$, darin sind a_i die Aktivitäten und v_i die stöchiometrischen Koeffizienten der Edukte und Produkte.

[4] benannt nach Svante August Arrhenius (1859 – 1927, Physiker und Chemiker, Schweden)

bekannt sind. Die biochemischen Prozesse, auf die sich diese Arbeit konzentriert, erscheinen zunächst einfacherer Natur, da es nur wenige offensichtliche Prozessschritte gibt. Im Detail laufen jedoch auch hier sehr viele Reaktionen im Inneren von Mikroorganismen ab, die sich vielfach einer genauen Untersuchung entziehen. Zudem laufen nicht zu jedem Zeitpunkt die gleichen Reaktionen ab. Auch kann schon eine moderat erhöhte Reaktionstemperatur zum Zerfall eines Produkt-Moleküls führen und nicht zu einer erhöhten Produktbildungsrate (wie nach Arrhenius). Die Fortschritte der letzten Jahre in der Systembiologie haben zwar einen Großteil der unterlagerten Reaktionsnetzwerke aufgedeckt, jedoch ist über die Regulationen und damit über die dynamischen Reaktionsgeschwindigkeiten der Einzelreaktionen weiterhin nur wenig bekannt.

Stattdessen werden die Reaktionsnetzwerke abstrahiert, und beobachtete oder postulierte Effekte approximiert. Die Reaktionen in diesen vereinfachten Modellen können aber nicht durch ähnlich simple Zusammenhänge wie bei Elementarreaktionen beschrieben werden. Die Ansätze zur Beschreibung der Reaktionsrate der „fiktiven“ Reaktionen sind in der Mehrheit der Fälle willkürlich gewählte Funktionen, die formal-mathematisch den gewünschten oder beobachteten dynamischen Verlauf darstellen können. Aus diesem Grund werden die nichtlinearen Funktionen auch Formalkinetiken genannt, Näheres folgt in Abschnitt 3.1.1.

Auf absehbare Zeit werden Modelle in der Biochemie und -technologie selten rein rigoroser Natur[5] sein. Eine wichtige Feststellung ist daher, dass im Detail eines praktikablen Modells immer Black-Box-Ansätze und Abstraktionen der Realität enthalten sind. Ein solches Modell ist deswegen in der Regel nicht geeignet, alle realen Details in der Biochemie einer Zelle zu erklären. Aus der Willkürlichkeit der mathematischen Formulierung folgt außerdem, dass viele mathematische Ansätze verwendet werden könnten. Eine Lösung zu diesem Auswahl-Problem wird in den folgenden Kapiteln dargestellt.

Zusammengefasst stellen alle Modelle, nicht nur (aber besonders) in der Biotechnologie, Vereinfachungen der Wirklichkeit dar. Es gilt, jenes Modell zu finden, das für einen bestimmten Zweck die besten Ergebnisse liefert. Ein Modell beschreibt also nicht die gesamte Wirklichkeit, sondern nur eine Auswahl von Aspekten. Diese Auswahl wird getroffen, um einzelne relevante Eigenschaften wiederzugeben – manchmal erfolgt die Auswahl auch, um das Modell zu vereinfachen. Beispielsweise muss die Temperatur eines Prozes-

[5] Rigorose Modelle beinhalten eine Modellierung vollständig durch physikalisch oder chemische Gesetzmäßigkeiten. Im Gegensatz dazu verwenden empirische Modelle eine Beschreibung auf der Grundlage von Messdaten. Ein Beispiel dafür sind die hier verwendeten Modelle mit Formalkinetiken.

ses nicht in das Modell aufgenommen werden, wenn sie während der Fermentationsdauer (und für alle Fermentationen) konstant gehalten wird.

3.1 Modellansatz

Die hier betrachteten Modelle weisen einige Merkmale und Eigenschaften auf, die wichtig für den sinnvollen Einsatz in der Verfahrens- beziehungsweise Bioverfahrenstechnik sind. Daher sollen im Folgenden die Unterschiede zwischen verschiedenen Modellansätzen erläutert werden.

Statische Modelle können zum Beispiel nur Gleichgewichte oder Endwerte bestimmen, aber mit ihrer Hilfe ist es nicht möglich, den zeitlichen Verlauf eines Prozesses zu beschreiben. Hierfür werden dynamische Modellansätze benötigt, mit denen man die Änderungen über der Zeit darstellen kann. Diese Eigenschaft ist eine erste Voraussetzung für den Einsatz von Prozessführungsstrategien und vielen weiteren regelungstechnischen Methoden. Nachfolgend wird daher nur auf dynamische Modelle Bezug genommen.

Es gibt Modelle, die die Messdaten mit Hilfe von Gleichungen (oft Polynome) interpolieren und somit kompakt zusammenfassen können. Hierbei handelt es sich im weitesten Sinne um eine reine Datenkompression. Die ermittelten Gleichungen können sinnvoll für eine bessere Darstellbarkeit von Messdaten verwendet werden. Aus dem Verlauf der Funktionsgraphen sinnvolle Erkenntnisse zu gewinnen ist hingegen, insbesondere wenn der zeitliche Abstand zwischen den Messdaten groß ist, nicht möglich.

Mit Modellen speziell für die Prozessführung sollen sich jedoch auch Vorhersagen berechnen lassen. Im Unterschied zu den Interpolationsmodellen müssen hier Wirkzusammenhänge modelliert werden, die eine Wiedergabe des Verhaltens der Mikroorganismen anstreben. Die Parameterwerte dieser Modelle müssen zunächst mit Hilfe von Messdaten in einem Optimierungsprozess identifiziert werden (siehe Abschnitt 3.3). Außerdem ist es sinnvoll, diese Modelle anhand weiterer, neuer Daten zu überprüfen, bevor sie eingesetzt werden. Der Aufwand ist also deutlich höher als bei den interpolierenden Modellen, aber ihr Nutzen ebenso.

3.1.1 Zustandsraummodelle

In dieser Arbeit werden ausschließlich Zustandsraummodelle betrachtet, welche die Anforderungen aus dem letzten Abschnitt erfüllen können. Im Folgenden wird das Grundgerüst vorgestellt, mit der ein solches Modell formuliert und wie es im ABC-System und in späteren Abschnitten auch verwendet wird.

Leitet man Modelle prinzipiell aus Massen- und Stoffbilanzen her, so erhält man Differentialgleichungssysteme, die die zeitlichen Änderungen der einzelnen Massen der beteiligten Substanzen beschreiben. Die Massen können als m innere Zustände oder Zustandsgrößen des zu modellierenden Systems bezeichnet werden. Für Zustandsraummodelle werden sie üblicherweise im Zustandsvektor $\underline{x}$ zusammengefasst:

$$\underline{x} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{pmatrix} \tag{3.5}$$

Wie Abbildung 3.1 zeigt, wirken auf ein System – charakterisiert durch $\underline{x}$ – Einflussgrößen ein, die Stellgrößen $\underline{u}$ genannt werden. Die einzelnen Differentialgleichungen des Modells beschreiben jeweils den zeitlichen Verlauf eines Zustands. Diese Zustandsdifferentialgleichungen $\underline{f}$ können kompakt als

$$\dot{\underline{x}} = \underline{f}(t, \underline{x}, \underline{\theta}, \mathbf{U}) \tag{3.6}$$

in der allgemein üblichen Form zur Beschreibung nichtlinearer Differentialgleichungssysteme dargestellt werden. In der Gleichung ist t die Zeit und $\underline{\theta}$ der Vektor mit Modellparametern[6]. Fermentationen sind Anfangswertprobleme, daher muss zur Lösung des Differentialgleichungssystems der Vektor der Anfangswerte $\underline{x}_0 = \underline{x}(t_0)$ gegeben sein. Da hier Fed-Batch-Prozesse beschrieben werden, hängt die zeitliche Änderung einer Masse x_i auch davon ab, ob und wie die betreffende Substanz hinzu gefördert wird. Diese Informationen werden durch die Stellgrößenmatrix $\mathbf{U}$, siehe Seite 30, gegeben, auf die auch am Ende dieses Abschnitts noch kurz eingegangen wird. Der Verlust von Masse durch Probennahmen muss hier nicht beachtet werden, da diese durch diskrete Zustandsänderungen berücksichtigt werden (siehe Abschnitt 2.4). Im ABC-System werden die Zustandsgleichungen $\underline{f}$ in der `Symb_f` definiert.

Tatsächlich ist ein Zustandsraummodell nicht auf Massenzustände beschränkt. Es können auch Hilfsgrößen eingebaut werden, die keine Massen darstellen. Die Aktivität eines Enzyms (mit einem Wertebereich von 0 – 100 %) ist ein Beispiel dafür. Auch diese Größen stellen Zustände des Systems dar und gehören ebenso wie die Massen zum Zustandsvektor $\underline{x}$.

Vervollständigt wird ein Zustandsraummodell durch die Ausgangsgrößen des Systems, die man mit dem Messgrößenvektor $\underline{y}$ beschreibt.

[6] Nicht zu verwechseln mit den Parametern der Prozessführung $\underline{\Theta}$.

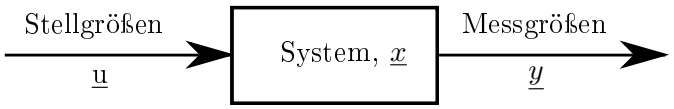


Abbildung 3.1: Schematische Darstellung der Systembeschreibung

Sehr oft wird auch das Volumen der Flüssigphase bilanziert, damit der Zusammenhang zwischen den internen Massezuständen $\underline{x}$ und den messbaren (Konzentrations-) Ausgangsgrößen $\underline{y}$ über Messgleichungen $\underline{h}$ hergestellt werden kann. Mit der Annahme, dass sich die Dichte der Flüssigphase nicht ändert, kann das Volumen wie eine Masse behandelt werden. Die allgemeine Formulierung der Messgleichungen lautet:

$$\underline{y} = \underline{h}(t, \underline{x}, \underline{\theta}, \mathbf{U}) \tag{3.7}$$

Sie können in vielen Anwendungsfällen vereinfacht dargestellt werden, da selten eine direkte Abhängigkeit von t, $\underline{\theta}$ oder $\mathbf{U}$ gegeben ist:

$$\underline{y} = \underline{h}(\underline{x}) \tag{3.8}$$

Ein Beispiel für eine Messgleichung ist die Berechnung einer Konzentration eines Massezustands aus dem Quotienten seiner Masse x_i und dem Flüssigvolumen V: $c_i = x_i/V$, siehe Beispiel in der Beschreibung von `Symb_h` (Seite 22). Beispielsweise im Rahmen einer Parameteridentifikation können so die berechneten Konzentrationen mit den in Realität gemessenen Werten aus einem Experiment verglichen werden.

In der Literatur werden die Funktionen f für massebehaftete Zustände aus Gleichung 3.6 üblicherweise in eine funktionale *Reaktionsrate* r und in das Reaktionsvolumen V unterteilt. Gleichung 3.9 zeigt diese Aufteilung für den Zustand x_i, $i \in \{1, \ldots, m\}$. Darin gibt die erste Summe die Produktion von x_i wieder, die zweite den Verbrauch durch Reaktionen, in denen x_i Edukt ist. Der jeweilige Reaktionspartner ist x_j, mit $j = 1...m, i \neq j$.

$$\dot{x}_i = \left(\sum_{j=1, i \neq j}^{m} r_{ji}(t, \underline{x}, \underline{\theta}, \mathbf{U}) - \sum_{j=1, i \neq j}^{m} r_{ij}(t, \underline{x}, \underline{\theta}, \mathbf{U}) \right) \cdot V(t) \tag{3.9}$$

Ein Element r_{ij} oder r_{ji} ist in dieser Darstellungsweise nur dann $\neq 0$, wenn eine Reaktion $x_i \underset{r_{ji}}{\overset{r_{ij}}{\rightleftharpoons}} x_j$ tatsächlich modelliert wird. Alle Reaktionsraten r_{ij} können in einer Matrix $\mathbf{M_R}$

mit der Größe $i \times j$ angeordnet werden. Dies vereinfacht die Indizierung, auch wenn die Matrix typischerweise viele Nullen enthält. Für eine bessere Lesbarkeit wird im Weiteren auf die Angabe der Abhängigkeit der Variablen $(t, \underline{x}, \underline{\theta}, \mathbf{U})$ verzichtet.

Weiterhin soll zunächst nur eine einzige, irreversible Reaktion $r_{ij} \neq 0$ betrachtet werden, daher folgt für das Produkt x_j, welches durch die Reaktion r_{ij} aus dem Edukt x_i entsteht:

$$\dot{x}_i = -r_{ij} \cdot V \tag{3.10}$$

$$\dot{x}_j = \quad r_{ij} \cdot V \tag{3.11}$$

Jede Reaktion tritt somit in Form eines entsprechenden Terms r_{ij} einmal in der Differentialgleichung für das Edukt und einmal in der für das Produkt auf. Bei Reaktionen mit mehr als zwei Reaktionspartnern treten die Terme entsprechend häufiger auf.

Um das Massenverhältnis der Reaktionspartner wiederzugeben, ist zusätzlich ein Ausbeutekoeffizient Y analog zum stöchiometrischen Koeffizienten in der Chemie notwendig. Er gibt für eine Reaktion das Verhältnis von verbrauchter Eduktmasse zu produzierter Masse an. In der Literatur wird für den Ausbeutekoeffizienten meist eine Bezeichnung nach dem Schema $Y_{j/i}$ verwendet, worin i das Edukt und j das Produkt meint. Die automatische Benennung durch den in dieser Arbeit vorgestellten Modellstrukturgenerator ist abhängig von der durch die Anzahl der Edukte und Produkte charakterisierten Reaktion. Beispielsweise wird der erste Ausbeutekoeffizient für die Reaktion $A + B \longrightarrow C$ mit `Y_A.C` bezeichnet. Abschnitt 4.2.4 beschäftigt sich eingehender mit der Situation bei Reaktionen mit mehr als einem Edukt und Produkt. Von der konkreten Reaktionsgleichung ist auch abhängig, ob Y in der Eduktgleichung (3.10) oder in der Produktgleichung (3.11) auftritt. In dieser Arbeit werden Ausbeutekoeffizienten, abweichend von ihrer tatsächlichen Benennung im Modell, entsprechend der Indizierung der Reaktionsraten, mit Y_{ij} bezeichnet. Zur vereinfachten Darstellung wird auf die explizite Erwähnung eines Ausbeutekoeffizienten Y in den meisten folgenden Gleichungen allerdings verzichtet.

Die Reaktionsrate wird weiter unterteilt in eine (ebenfalls funktionale) *spezifische Reaktionsrate* μ_{ij} und eine Konzentration $c_{P_{ij}}$.

$$r_{ij} = \mu_{ij} \cdot c_{P_{ij}} \tag{3.12}$$

Diese Konzentration $c_{P_{ij}}$ muss keiner der Konzentrationen der an der Reaktion beteiligten Stoffe (hier c_i oder c_j) entsprechen, sondern kann zu einem dritten Zustand oder Stoff gehören. Dieser Zusammenhang soll kompakt durch die folgende Umindizierung definiert werden:

$$P_{ij} \in \{1, \ldots, m\} \mid i, j \in \{1, \ldots, m\} \tag{3.13}$$

Für eine vollständige Beschreibung muss also angegeben werden, welchem Stoff die einzelnen P_{ij} entsprechen.

Gleichung 3.12 eingesetzt in Gleichung 3.10 ergibt dann mit $x_{P_{ij}} = c_{P_{ij}} \cdot V$:

$$\dot{x}_i = -\mu_{ij} \cdot c_{P_{ij}} \cdot V \tag{3.14}$$

$$= -\mu_{ij} \cdot x_{P_{ij}} \tag{3.15}$$

Aus historischen Gründen wird der Multiplikator $x_{P_{ij}}$ in dieser Arbeit als (zeitlich variabler) P-Faktor einer Reaktion bezeichnet.

Ist x_i Bestandteil von mehreren Reaktionen, werden die jeweiligen Terme wie oben in Gleichung 3.9 eingeführt addiert, wobei dieselben Elemente in r_{ij} auch in μ_{ij} null sind:

$$\dot{x}_i = -\sum_{j=1, i \neq j}^{m} \mu_{ij} \cdot x_{P_{ij}} \tag{3.16}$$

In einem letzten Schritt kann die spezifische Reaktionsgeschwindigkeit μ_{ij} noch weiter in eine maximale Reaktionsrate $\mu_{ij_{\max}}$ und in eine Formalkinetik f_q aufgetrennt werden:

$$\mu_{ij} = \mu_{ij_{\max}} \cdot f_q \tag{3.17}$$

In Gleichung 3.17 ist nur der letzte Term f_q eine Funktion: Eine Formalkinetik, die auch das Produkt aus einzelnen Formalkinetiken sein kann. Sie beschreibt in einem Modell die Abhängigkeit einer Reaktionsgeschwindigkeit von den aktuellen Milieubedingungen, das heißt zum Beispiel von den Konzentrationen der verschiedenen Substrate. Auf die Formalkinetiken wird unten genauer eingegangen. Ist in einer Reaktionsrate r_{ij} die Anzahl der Formalkinetiken $n_{F_{ij}}$, lautet die spezifische Reaktionsrate:

$$\mu_{ij} = \mu_{ij_{\max}} \cdot \prod_{k=1}^{n_{F_{ij}}} f_{q_k} \tag{3.18}$$

Die Formalkinetiken f_q bzw. f_{q_k} sind zusätzlich abhängig von i und j, aber zur besseren Lesbarkeit wird hier auf eine weitere Mehrfachindizierung mit i, j verzichtet.

Zuletzt sind Zufütterungen oder Feedings der Substrate zu berücksichtigen. Um die volle Flexibilität im ABC-System zu erhalten, werden hierfür zwei Stellgrößen für jeden

Massen-Feed eingeführt; die Konzentration des Feeds $c_{\text{feed,i}}$ [$\text{g}\,\text{l}^{-1}$] und die zeitlich veränderliche Zudosierrate u_{feed} [$\text{l}\,\text{h}^{-1}$]. Das Produkt aus beiden ergibt den Massenstrom der jeweiligen Zufütterung.

Unter Berücksichtigung aller Ersetzungen und Elemente ergibt sich zusammengefasst für die zeitliche positive Änderung eines Zustands x_i, also die Produktion von x_i, die folgende Differentialgleichung. Analog ergibt sich der erste Summand in der Gleichung für die Eduktreaktionen.

$$\dot{x}_i = \sum_{j=1, i \neq j}^{m} Y_{ij} \cdot \mu_{ij_{\max}} \cdot x_{P_{ij}} \cdot \prod_{k=1}^{n_{F_{ij}}} f_{q_k} + u_{\text{feed,i}} \cdot c_{\text{feed,i}} \tag{3.19}$$

Dieser Formalismus gilt für eine große Anzahl von Differentialgleichungssystemen. Er ermöglicht die Automatisierung der Modellierung und ist damit eine Grundlage dieser Arbeit.

Modellparameter

In den vorangegangenen Abschnitten wurden viele Modellvariablen eingeführt, darunter $\mu_{\max}$ und Y_{ij}. Dazu gehören auch die im nächsten Abschnitt beschriebenen Kinetikparameter. Gemeinsam werden diese Modellparameter in einem Vektor $\underline{\theta}$ zusammengefasst. Die Parameter sind zunächst Variablen, bis ihnen durch eine Parameteridentifikation Werte zugewiesen werden. Obwohl man dann von Modellkonstanten sprechen könnte, ändern sich die Werte häufig erneut, wenn weitere Experimente durchgeführt werden und erneut eine Parameteridentifikation durchgeführt wird. Die Bezeichnung Parameter wird daher beibehalten.

Formal-Kinetiken

Um ein Differentialgleichungssystem für einen Bioprozess zu vervollständigen, sind in Gleichung 3.19 die konkreten Formalkinetiken f_{q_k} festzulegen, um die dynamische Reaktionsgeschwindigkeit bestimmen zu können. In der Biochemie handelt es sich dabei meist um nichtlineare Funktionen. Sowohl die Funktion als auch die sie regulierende(n) Größe(n) müssen während der Modellierung gewählt werden. Die Wahl ist dabei zumindest in Teilen willkürlich und durch die Erfahrung des menschlichen Modellbildners bestimmt, denn eine Formalkinetik soll nur einen formalen, mathematischen Zusammenhang zwischen Reaktionspartnern beschreiben und hat nicht den Anspruch, biochemische Reaktionen exakt beschreiben zu können. Die Konzentrationen der an der jeweiligen Reaktion beteiligten Stoffe sind normalerweise automatisch auch Regulationsgrößen.

Zusätzlich können weitere regulierende Größen auftreten, die unabhängig von den Reaktionspartnern sind, beispielsweise die Temperatur oder die Konzentration eines Enzyms. Welche Regulationsgrößen als relevant angesehen und in das Modell aufgenommen werden, wird wieder entweder durch den menschlichen Modellbildner festgelegt oder sind das Ergebnis einer automatischen Modellbildung, deren Methoden ebenfalls im ABC-System umgesetzt sind, siehe [32, 33].

Die realen einzelnen Funktionen im Inneren einer Zelle könnten prinzipiell meist mit sehr einfachen Formalkinetiken beschrieben werden. Der gesamte Stoffwechsel umfasst jedoch hunderte oder tausende von Reaktionen, die nicht von einem Prozessmodell abgebildet werden können. Das Reaktionsnetzwerk kann jedoch aufgrund der Tatsache vereinfacht werden, dass in einer Kette von aufeinander folgenden Reaktionen eine der Reaktionen am langsamsten abläuft und somit geschwindigkeitsbestimmend für den gesamten Reaktionsweg ist. Dieser Reaktionsschritt – auch Flaschenhals oder im Englischen *bottleneck* genannt – kann sich aber abhängig vom Zustand oder den Umgebungsbedingungen der Zellen auch verschieben. Um diese Variabilität ansatzweise korrekt darstellen zu können, sind komplexere Formalkinetiken notwendig.

Es gibt jedoch keine Regel, die besagt, wann welche Kinetik verwendet werden soll, oder wie eine solche Formalkinetik für einen bestimmten Fall formuliert werden muss. In der abstrakten Modellbeschreibung gibt es keine richtigen oder falschen Kinetiken, sondern nur Kinetiken, mit deren Hilfe die Messdaten gut oder schlecht beschrieben werden können. Dies erklärt auch, warum in der Literatur sehr viele Formalkinetiken zu finden sind, zum Beispiel in [7].

Im Folgenden werden Formalkinetiken $f_q(t, \underline{x}, \underline{\theta}, \mathbf{U})$ und ihre automatisierte Kodierung durch den Modellstrukturgenerator anhand von zwei einfachen Beispielen näher erläutert. Der Einfachheit halber wird hier auf die doppelte Indizierung der Formalkinetiken mit f_{q_k} verzichtet.

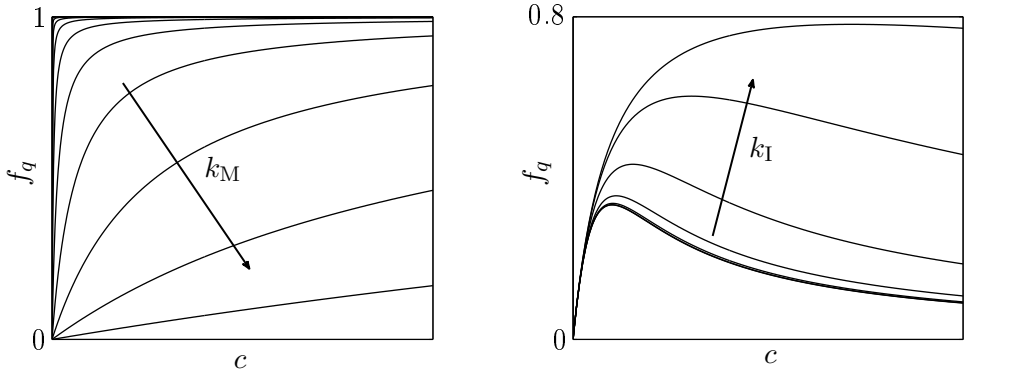
In den Formalkinetiken treten Kinetikparameter auf, für die teilweise eigene Variablenbezeichnungen üblich sind. Beispielsweise wird in der Monod-Kinetik[7] oder der mathematisch identischen Michaelis-Menten-Gleichung (MiMe)[8], siehe Gleichung 3.20, der Kinetikparameter oft mit k_M bezeichnet. Er ist ein Element des Modellparametervektors $\underline{\theta}$ und muss durch eine Anpassung an Messdaten im Rahmen einer Parameteridentifika-

[7] Jacques Monod (1910 – 1976, Biologe, Frankreich)

[8] Leonor Michaelis (1875 – 1949, Biochemiker und Mediziner, Deutschland/USA) und Maud Menten (1879 – 1960, Medizinerin, Kanada)

tion, siehe Abschnitt 3.3, bestimmt werden. Die regulierende Konzentration ist in der Gleichung mit c bezeichnet. Abbildung 3.2(a) zeigt einige exemplarische Funktionsverläufe. Mit sinkender Konzentration c sinkt auch der Funktionswert und damit die Reaktionsgeschwindigkeit. Die MiMe-Funktion gehört daher zur Gruppe der limitierenden Formalkinetiken.

$$f_q(c, k_\mathrm{M}) = \frac{c}{c + k_\mathrm{M}} \qquad \text{„Michaelis-Menten-Gleichung“} \tag{3.20}$$



(a) Monod-Kinetik mit Parameter k_M (b) Haldane-Kinetik mit k_I und k_M (const)

Abbildung 3.2: Einige Verläufe zweier Formalkinetiken in Abhängigkeit von Konzentration c und ihren Parametern.

Ein zweites Beispiel für eine Formalkinetik ist die Haldane-Kinetik[9], Gleichung 3.21, deren zwei Parameter in der Regel mit k_M und k_I bezeichnet werden. Sie ist ein Beispiel für eine komplexere Kinetik, die limitierend aber auch inhibierend wirken kann, siehe Abbildung 3.2(b).

$$f_q(c, [k_\mathrm{M}, k_\mathrm{I}]) = \frac{c}{k_\mathrm{M} + c + \frac{c^2}{k_\mathrm{M}+k_\mathrm{I}}} \qquad \text{„Haldane-Kinetik“} \tag{3.21}$$

$f_q(c, [k_\mathrm{M}, k_\mathrm{I}])$ stellt die verwendete Matlab-Notation dar: Eckige Klammern vereinen die beiden Parameter zu einem Vektor, der von dem Haldane-Programm erwartet wird.

Einige weitere Beispiele für andere Formalkinetiken im ABC-System sind in Tabelle 3.1 aufgelistet. Die Einflussgröße ist eine Konzentration c, alle Variablen k kennzeichnen Kinetikparameter.

[9] John Burdon Sanderson Haldane (1892 – 1964, Genetiker, England)

Die einzelnen Formalkinetiken sind nach einem vereinheitlichten Schema implementiert und in einer jederzeit erweiterbaren Kinetikbibliothek zusammengefasst, auf welche wiederum dynamisch zugegriffen werden kann. Im Kapitel 4 wird diese Kinetikbibliothek im Rahmen der automatischen Programmierung mit weiteren Kinetikbeispielen näher erläutert.

Kinetikname	**Formel** $f_q =$
Aiba	$\exp(-k \cdot c)$
Jerusaliwski	$k/(k+c)$
Ming	$c^2/(c^2+k)$
Tessier	$1-\exp(-\frac{c}{k})$

Tabelle 3.1: Weitere Beispiele aus der Kinetikbibliothek

Das auf Seite 22 gezeigte Code-Fragment beschreibt daher die Produktion und den Zerfall von DNA, wobei die DNA-Bildung als Produkt zweiter Michaelis-Menten-Terme angeschrieben ist, die von den intrazellulären Konzentrationen (g) von Aminosäuren (As) und Nukleotiden (Nu) abhängen.

3.1.2 Begriffserklärung: Modelltyp, -struktur, -kandidat, -familie und Reaktionsschema

Im Folgenden werden einige Begriffe kurz zusammengefasst, da sie in den nächsten Kapiteln regelmäßig verwendet werden.

Der *Modelltyp* bezeichnet grob die „Komplexität“ eines Modells oder genauer den modellierten Detailgrad des Zellinneren wie unstrukturiert oder strukturiert. Nähere Erläuterungen folgen in Abschnitt 3.1.4. Die Anzahl der Zustände $\underline{x}$ nimmt ebenso wie die Anzahl der Reaktionen prinzipiell mit höherer Komplexität zu.

Ein *Reaktionsschema* beinhaltet den zu modellierenden Satz von Reaktionsgleichungen, der wiederum die *mathematische Struktur* eines Modells grundlegend festgelegt. Hierunter ist die generelle Zusammensetzung der rechten Seiten $\underline{f}$ der einzelnen Differentialgleichungen zu verstehen. Die mathematische Struktur definiert also bereits alle Stoffwechselwege, ohne dass eine Festlegung auf konkrete Formalkinetiken erfolgt. Die Möglichkeit, sehr unterschiedliche Stoffwechselsysteme in eine kompakte, einheitliche mathematische Struktur zu bringen, ist die Grundlage für die Automatisierung, wie sie in Kapitel 4 erläutert wird.

Ist die Festlegung auf bestimmte Formalkinetiken erfolgt, ergibt dies einen konkreten *Modellkandidaten*. Die Menge aller erzeugten Modellkandidaten mit den verschiedenen Kombinationen von Formalkinetiken ergibt die *Modellfamilie*.

3.1.3 Lösen von Differentialgleichungen

Bei den im Abschnitt 3.1.1 vorgestellten Gleichungen handelt sich um gewöhnliche Differentialgleichungen (engl.: *Ordinary Differential Equation (ODE)*). Sie sind meist nicht analytisch lösbar und erfordern die Behandlung durch vergleichsweise aufwendige numerische Integrationsverfahren, den ODE-Lösern. Davon stehen in Matlab verschiedene zur Auswahl. Verwendet wurden in dieser Arbeit die Löser `ode45` und `ode15s`. Ersterer basiert auf dem Runge-Kutta-Verfahren 4. und 5. Ordnung, letzterer ist ein Löser mit variabler Ordnung speziell für steife Differentialgleichungssysteme.

Auf die genauen Details der Algorithmen wird an dieser Stelle nicht eingegangen, aber für das weitere Verständnis soll die Arbeitsweise stark vereinfacht am Beispiel des Euler-Verfahrens dargestellt werden. Ausgehend von einem Startwert x_0 und der über einen Differenzenquotienten bestimmten lokalen Ableitung $\dot{x}_0$ kann ein Schritt nach der folgenden Iterationsvorschrift berechnet werden:

$$x_{k+1} = x_k + h \cdot \dot{x}_k \qquad\qquad k = 0, 1, \ldots, k_{\text{end}} \tag{3.22}$$

In der Gleichung ist h die Schrittweite. Sie ist für die Genauigkeit der Lösung entscheidend, bestimmt aber auch die Anzahl der Rechenschritte, bis das Ende der Berechnung ($k = k_{\text{end}}$) erreicht wird.

Die einzelnen Löser-Verfahren unterscheiden sich in der Iterationsvorschrift. Bei den meisten kann die Schrittweite h zwischen den einzelnen Schritten angepasst werden, um ein vorgegebenes Maß an Genauigkeit einzuhalten. Vielfach muss dabei zunächst eine Anfangsschrittweite gewählt werden. Dies ist ein wichtiger Einstellwert, wie im Folgenden gezeigt wird.

Ist eine Schrittweite zu klein, ist die Iterationsfortschritt unnötig gering, die Rechengeschwindigkeit daher langsam. In der nächsten Iteration wird die Schrittweite deswegen automatisch ein wenig vergrößert. Ist sie hingegen zu groß, das heißt, die Berechnung überschreitet festgelegte Fehlertoleranzen, muss das aktuelle Berechnungsergebnis verworfen werden. Der Löser beginnt an der ursprünglichen Stelle mit einer verkürzten Schrittweite erneut.

Durch eine geschickte Wahl der Anfangsschrittweite kann daher viel Rechenzeit eingespart werden. In absoluten Zeiten gemessen, erscheint das Einsparpotential nicht groß. Wie aber in den Abschnitten 2.4 und 2.5 erklärt wurde, unterteilt das ABC-System das zu berechnende Zeitintervall in viele, teilweise tausende von Einzelintervallen. Für jedes einzelne Intervall wird der Löser neu gestartet. Wenn dieser dabei jedes Mal mit einer ungeeigneten Anfangsschrittweite beginnt, ergibt sich für eine Simulation des gesamten Zeitraums eine deutlich messbare Zunahme der Rechenzeit. Verstärkt wird der Effekt durch die vielfache Wiederholung von Simulationen, wie es beispielsweise für die Parameteridentifikation oder Trajektorienplanung notwendig ist. Im Folgenden werden Simulationsberechnungszeiten anhand von drei verschiedenen Modellen, die echte Experimente beschreiben, miteinander verglichen. Für die Untersuchung wurde der `ode15s` von Matlab eingesetzt. In der nachstehenden Liste sind jeweils der Modelltyp, die Anzahl der Zustände (=Differentialgleichungen) und die Anzahl der Modellparameter angegeben. Außerdem ist die Anzahl der Einzelintervalle aufgeführt, in die eine Simulation des jeweiligen Modells unterteilt wird.

- Modell 1: Unstrukturiert, 7 Zustände, 16 Parameter, ca. 4000 Einzelintervalle
- Modell 2: Unstrukturiert, 10 Zustände, 20 Parameter, ca. 500 / 700 Einzelintervalle
- Modell 3: Strukturiert, 11 Zustände, 60 Parameter, ca. 30 Einzelintervalle

Die deutlichen Unterschiede in der Anzahl der Einzelintervalle werden hauptsächlich durch die Anzahl der Messzeitpunkte verursacht. Das Modell 1 beschreibt einen Versuch, in dem einzelne Messwerte automatisch im Abstand weniger Sekunden gemessen wurden, woraus sich die hohe Anzahl von Intervallen ergibt. Aus dem Versuch zu Modell 3 standen nur Messwerte von manuellen Probennahmen zur Verfügung. Bei Modell 2 wurden zwei verschiedene Versuche mit einer unterschiedlichen Anzahl von automatischen Messungen nachgerechnet, um unabhängig vom Modell den Einfluss der Intervallanzahl untersuchen zu können.

Die Abbildungen 3.3 bis 3.5 zeigen in verschiedenen Vergrößerungen die Rechendauer der oben aufgeführten Simulationen aufgetragen über der Startschrittweite. Für den Vergleich wurden rund 90 verschiedene Anfangsschrittweiten zwischen etwa 10 Mikrosekunden und 3 Sekunden gewählt, dargestellt ist jeweils der Mittelwert der benötigten Rechenzeit aus mindestens vier Wiederholungen, um Einflüsse des Betriebssystems zu minimieren.

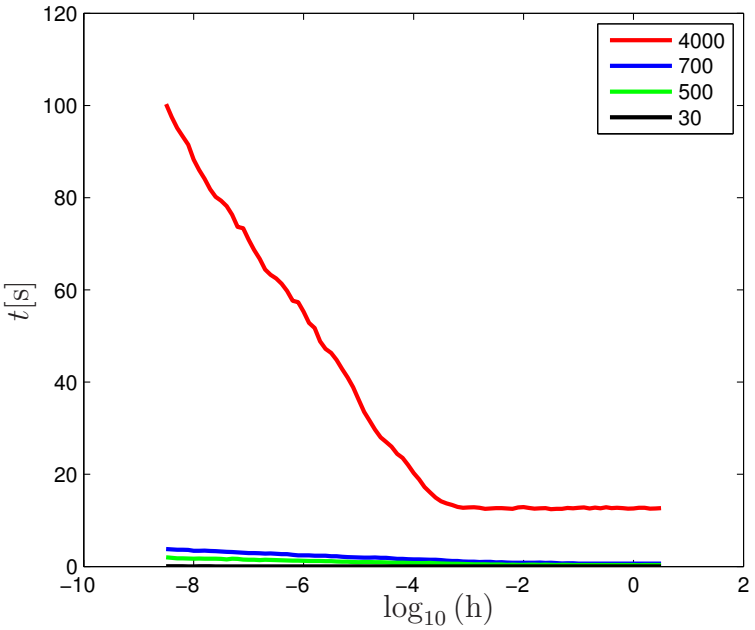


Abbildung 3.3: Von der Anfangsschrittweite h abhängige Simulationsdauer für verschiedene Intervallanzahlen (30 – 4000)

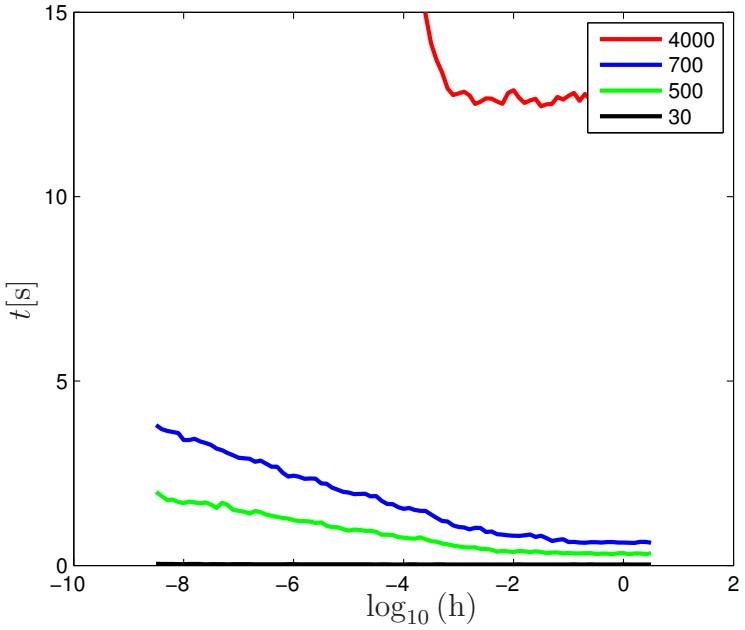


Abbildung 3.4: Ausschnitt aus Bild 3.3

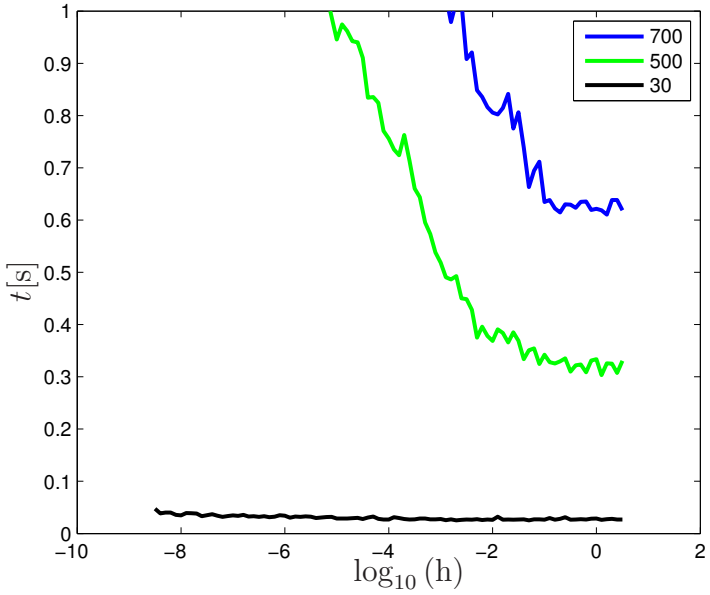


Abbildung 3.5: Ausschnitt aus Bild 3.4

Aus den Vergleichen sind zwei Erkenntnisse zu gewinnen. Erstens ist die Komplexität eines Modells aus den gewählten Beispielen nicht ausschlaggebend für die Berechnungsdauer. Im Gegenteil; in diesem Vergleich nahm das komplexeste Modell mit den meisten Parametern und Gleichungen die geringste Rechenzeit in Anspruch. Zweitens gibt es bei allen vier Fällen im betrachteten Bereich eine Grenzschrittweite unterhalb derer sich die Simulationsdauer deutlich erhöht. Oberhalb bleibt die Dauer nahezu konstant, und auch die Genauigkeit der Berechnungen ändert sich nicht, was separat untersucht wurde, aber hier nicht dargestellt wird. Offenbar ist die Anpassung an kleinere Schrittweiten deutlich effizienter als die Vergrößerung derselben, so dass kein signifikanter Einfluss auf die Rechenzeit festgestellt werden konnte. Die exakte Grenze ist vermutlich modellabhängig, da sie für die beiden Gruppen von Simulationen des Modells 2 sehr ähnlich ist, sich aber wieder um ein bis zwei Größenordnungen von den beiden anderen Modellsimulationen unterscheidet. Für eine endgültige Aussage müssten umfangreichere Untersuchungen durchgeführt werden. Die Vermutung liegt nah, dass die Rechendauer auch von der jeweiligen Prozessführung abhängt. Generell lässt sich festhalten:

1. Es ist sinnvoll, für ein Modell eine passende Startschrittweite zu ermitteln. Hilfsweise kann auch ein vergleichsweise großer Wert (zum Beispiel eine Zehntelsekunde)

gewählt werden, da kein negativer Einfluss auf die Rechenzeit für Schrittweiten in dieser Größenordnung festgestellt werden konnte.

2. Die Anzahl der Einzelintervalle sollte so weit wie möglich reduziert werden. Da die Funktionsweise des ABC-Systems Ursache für die hohe Anzahl ist, kann der Benutzer keinen direkten Einfluss darauf nehmen. Abschnitt 2.5 stellte jedoch ein Verfahren vor, durch das die Einzelintervalle optimiert und in ihrer Anzahl reduziert werden können.

Eine weitere Möglichkeit besteht darin, die Funktionsweise der verschiedenen Löser so zu verändern, dass die jeweils vorletzte Schrittweite[10] des vorangegangenen, bereits berechneten Intervalls als Startschrittweite für das aktuell zu berechnende Intervall verwendet wird. Der Aufwand für die notwendigen Änderungen ist allerdings zu hoch, als dass sie im Rahmen dieser Arbeit hätten durchgeführt werden können.

3.1.4 Modelltypen für unterschiedliche Anforderungen

Es wurden unterschiedliche Ansätze zur automatisierten Modellprogrammierung auf ihre Programmierbarkeit und ihren praktischen Nutzen hin untersucht. In einem „Universalmodell“ werden beispielsweise alle definierten Zustände durch alle Kombinationen von Reaktionen verknüpft. Die Modellstruktur ist daher prinzipiell für alle Mikroorganismen geeignet. Durch eine Messdatenanalyse müsste das verallgemeinerte Modell nur an den jeweiligen Mikroorganismus angepasst werden. Die Idee des Universalmodells wurde jedoch verworfen, da es zwar theoretisch einfach aufzustellen wäre, aber die Anpassung in der Praxis große Herausforderungen verursacht hätte: Die Anzahl aller möglichen Reaktionen wird bereits bei Modellen mit wenigen Zuständen sehr groß, dementsprechend viele zu identifizierende Parameter wären notwendig und ihre Identifizierbarkeit wäre nicht sichergestellt. Hinzu kommt noch die Vielfalt der Formalkinetik-Kombinationen für derart viele Reaktionen, die zu einer Gesamtanzahl von Modellkandidaten führen würde, die auch von modernen Computern nicht mehr handhabbar ist.

Der umgekehrte Ansatz einer rein modularen Modellerzeugung, wie sie in [45] beschrieben ist, erscheint auf den ersten Blick sinnvoller. Allerdings erfolgt die Auswahl der Modellbausteine dort ausschließlich auf Basis von Messdaten. Die verwendeten Bausteine beschreiben meist Reaktionswege, sind aber ihrerseits nicht ausreichend flexibel,

[10] Die letzte Schrittweite kann nicht verwendet werden, da diese so gewählt wird, dass das Ende des aktuellen Einzelintervalls erreicht wird. Sie hat daher nichts mit der aktuell notwendigen Schrittweite zu tun, sondern ist im Allgemeinen kleiner.

um insbesondere bei der Definition der Reaktionsgeschwindigkeiten die Formalkinetiken beliebig austauschen zu können. Durch das Rauschen von Messdaten ergeben sich auch bei diesem Ansatz viele Varianten von Modellen mit einer sehr unterschiedlichen Anzahl von Reaktionen. Außerdem ist das System darauf ausgelegt, nur mit jeweils einem Modell weiterzuarbeiten. Dass dies einen prinzipiellen Nachteil darstellt, wurde in der Einleitung angesprochen und wird in Abschnitt 4.1 nochmals vertieft.

Der Ansatz, der durch den Modellstrukturgenerator ermöglicht wird, geht von der Überlegung aus, dass grundsätzlich solange mit vielen Modellkandidaten gearbeitet werden muss, bis neue Informationen aus Experimenten die Menge der Kandidaten eingrenzen. Mit beliebig vielen Formalkinetiken zur Auswahl würde dies jedoch wieder zu ähnlichen Problemen wie beim Universalmodell führen. Daher wurde zunächst das Konzept des „Mastermodells“ verfolgt, um die Anzahl der Modellstrukturen und -kandidaten einzuschränken: Ein biologisch motiviertes Grundmodell dient als Basis für die Modellierung. Das Grundmodell soll durch weitere strukturelle Bausteine modifiziert oder ergänzt werden, im Gegensatz zur modularen Modellerzeugung aber bereits grundsätzlich das Verhalten von Mikroorganismen beschreiben können. Um außerdem die Modelle nach den jeweiligen Prozessanforderungen entwerfen zu können, wurden für die Modellkomplexität unterschiedliche Kategorien festgelegt. In jeder davon sollte ein Mastermodell zur Verfügung stehen.

Letztlich konnte das Konzept der Mastermodelle nicht weiter verfolgt werden, da die Software zur automatischen Modellerzeugung nicht ausreichend flexibel war: Sehr viele zu berücksichtigende Sonderfälle hätten zu einer Vielzahl von Mastermodellen geführt. Anstatt Mastermodelle zu verwenden, wertet der Modellstrukturgenerator, der in Kapitel 4 detailliert vorgestellt wird, zur Erzeugung von Modellen nur eine „Modellstrukturdefinition“ aus. Darin werden Listen von Zuständen und Reaktionen gespeichert, die eine kompakte und eindeutige Darstellung eines Modells im ABC-System erlauben. Die sehr flexible Modellstrukturdefinition kann manuell oder auch durch andere Programme wie in [32] erzeugt werden. Die Aufgabe, sinnvolle Modellstrukturen aufzustellen, liegt daher außerhalb des Modellstrukturgenerators, wobei dieser durch fertige Modellbausteine mit biologischen Hintergrund den Entwicklungsprozess unterstützt.

Die Einteilung nach Modellkomplexität blieb erhalten, da für den Modellstrukturgenerator eine solche Einteilung hilfreich bei der Erzeugung der Modelle ist. So können automatisch bestimmte Modelleigenschaften berücksichtigt oder ausgelassen werden. Die einzelnen Typen, ihre Anwendungsgebiete und die erwähnten Konsequenzen für die Modellerzeugung werden in den folgenden Abschnitten erläutert.

Unstrukturiert Modelle

Unstrukturierte Modelle behandeln das Innere von Mikroorganismen als *Black Boxes* (siehe Abbildung 3.6); das Innenleben der Zellen wird nicht betrachtet. Modelliert werden somit in der Regel lediglich der Verbrauch an relevanten Nährstoffen[11], das Zellwachstum und gegebenenfalls die Bildung eines Produkts. Ein unstrukturiertes Modell beinhaltet daher grundsätzlich neben dem Volumen nur die Massen der genannten Stoffe als Zustände.

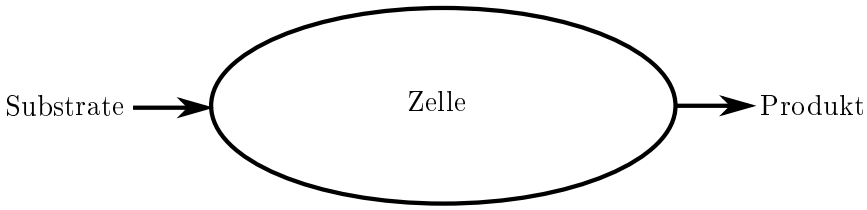


Abbildung 3.6: Unstrukturiertes Modell: Das Innere der Zelle wird nicht modelliert.

Der einzige Freiheitsgrad bei der Erzeugung der Modellkandidaten (abgesehen von der Wahl der Formalkinetiken und der sie regulierenden Einflüsse) ist das Edukt bei der modellierten Produktbildung. Dies können ein oder mehrere Substrate oder auch die Biomasse selbst sein. Man kann ein Edukt bei der Reaktion sogar vernachlässigen, wenn die Produktmasse um mehrere Größenordnungen kleiner ist als die übrigen Modellmassen und der Unterschied messtechnisch nicht erfasst werden kann. Diese Situation tritt insbesondere bei Sekundärmetaboliten ein. Die Grundgleichungen des unstrukturierten Modells lauten:

$$\begin{aligned} \text{Wachstum:} \quad & \sum \text{Substrate} \xrightarrow{r_\text{X}} x_\text{BTM} \\ \text{Produktbildung:} \quad & (?) \xrightarrow{r_P} x_\text{P} \end{aligned}$$

Hier ist x_BTM die Zell(trocken)masse, r_X die Wachstumsgeschwindigkeit, x_P das Produkt und r_P die Produktbildungsrate.

Substrate werden nicht nur für das Wachstum verbraucht. Mit ihnen werden auch beschädigte beziehungsweise denaturierte DNS/RNS, Proteine und andere Moleküle erneuert. Zusätzlich benötigt jeder Organismus Energie, die aus dem Stoffwechsel gewonnen wird, um zu überleben. Entsprechend wird dieser Vorgang auch (Lebens)Erhaltung

[11] Unter den hier betrachteten flüssigen Substraten sind dies je eine C-Quelle, N-Quelle und P-Quelle.

genannt. Ist der Verbrauch der Substrate signifikant, kann dieser im Modell durch eine weitere Reaktion dargestellt werden. In diesem speziellen Fall als eine Reaktion ohne Produkt (dargestellt durch „(-)“). Die Erhaltung ist hauptsächlich von der Kohlenstoff(C)-Quelle als Energie-Lieferant abhängig, daher vereinfachen die verwendeten Modelle die Erhaltung auf diese eine Quelle. Sie wird in einer weiteren Reaktion mit der Rate r_M verbraucht.

$$\text{Erhaltung:} \qquad (\text{C-Quelle}) \xrightarrow{r_M} (-)$$

Der unstrukturierte Modelltyp wird insbesondere in zwei Situationen eingesetzt. Zunächst, wenn keinerlei Messwerte zu den inneren Zuständen einer Zelle zur Verfügung stehen oder wenn bislang keine Informationen für einen Modellentwurf vorliegen. Dann ist ein unstrukturiertes Modell ein sinnvoller erster Ansatz. Dieser ist besonders dann nützlich, wenn in dem zu modellierenden Prozess das Produkt immer proportional zum Zellwachstum gebildet wird („wachstumsassoziierte Produktion“). Ist die Produktmaximierung eines solchen Prozesses relevant, kann die Problemstellung auf die Maximierung des Zellwachstums reduziert werden. Diese Maximierung wird in der exponentiellen Phase erreicht, wodurch sich der optimale Prozessverlauf auf diese eine Phase beschränkt. Eine Änderung des Stoffwechsels der Zellen ist somit nicht notwendig oder wünschenswert und muss nicht modelliert werden. Für dieses Beispiel bringt unter Umständen eine spätere Modellerweiterung in Richtung strukturierter Modelle daher auch keine Vorteile.

Einfache Modelle mit Speicher

In vielen Fällen kann beobachtet werden, dass das Wachstum der Mikroorganismen nicht abrupt, sondern erst zeitlich verzögert endet, nachdem eines der essentiellen Substrate nicht mehr zur Verfügung steht. Um entsprechende Beobachtungen zu erklären, können beispielsweise für jeden betroffenen Nährstoff zellinterne Speicher postuliert und als Zustände in einem einfachst-strukturierten Modell aufgenommen werden, siehe Abbildung 3.7. Solange noch Nährstoffe im Medium vorhanden sind, werden die zellinternen Speicher bis zu einem Maximum gefüllt. Tritt im Medium ein Nährstoffmangel auf, bedient sich die Zelle aus ihren internen Speichern und kann so begrenzt weiter wachsen. Die Ein- und Auslagerungsvorgänge können in Form normaler Reaktionen mit Formalkinetiken beschrieben werden. Im Modell können diese mit unterschiedlichen Reaktionsraten dargestellt werden, aber die Identifikation der betreffenden Modellparameter gestaltet sich in der Regel als schwierig.

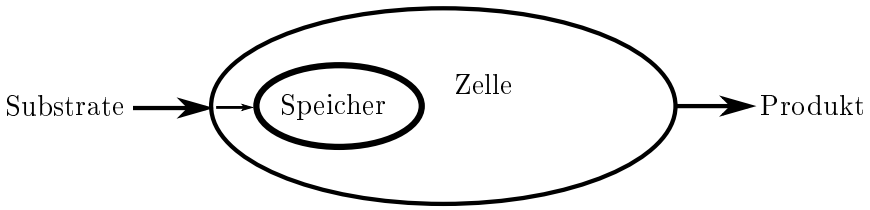


Abbildung 3.7: Einfaches Modell mit Speicher

Solche Speicher existieren tatsächlich in Zellen, beispielsweise kann Phosphat als Polyphosphat akkumuliert werden. Allerdings gibt es nur für einige dieser Speicherzustände (einfache) Messmethoden, so dass der aktuelle Füllstand oder auch die maximale Kapazität des internen Speichers häufig nicht bestimmt werden kann. Außerdem handelt es sich bei den Modell-Speichern um „funktionale“ Kompartimente: Es geht bei ihrer Modellierung nicht um bestimmte Moleküle, sondern um ihre Funktion. Die Identifikation der entsprechenden Modellparameter ist im Wesentlichen auf die Messdaten des jeweiligen Nährstoffs im Medium und der Biomasse angewiesen.

Das Reaktionsschema bei diesem Modelltyp ist davon abhängig, wie der Stoffffluss vom und zum Speicher modelliert wird. Drei verschiedene Implementationen sind denkbar, siehe Abbildung 3.8:

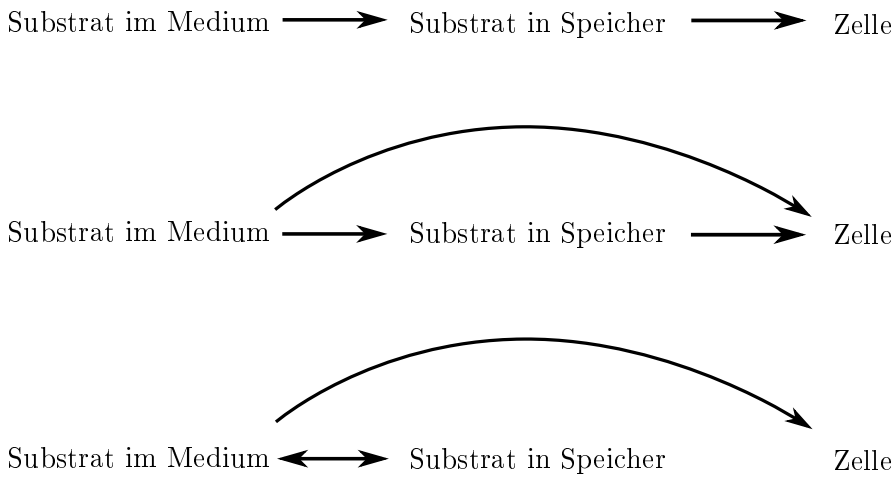


Abbildung 3.8: Mögliche Stoffffluss-Implementationen eines zellinternen Speichers

Für die meisten Anwendungsfälle wird die mittlere dargestellte Implementation sinnvoll sein. Das Zellwachstum kann normal beschrieben werden und nur zu den Zeiten, in denen bei der Fermentation eine Limitation auftritt, soll die alternative Versorgung

durch den betreffenden Speicher aktiv sein. Sofern Limitationen nicht maßgeblich für die gesamte Fermentation sind, verursacht die vereinfachte Beschreibung nur geringe Fehler bei der Gesamtsimulation.

Die obere Variante aus der Abbildung beinhaltet zwar nur zwei Reaktionen und somit potentiell weniger Modellparameter, aber sie hat den Nachteil, dass auch das normale Wachstum über den Zustand des Zellspeichers laufen muss, über den wenig bekannt ist. Dennoch ist die Variante geeignet, um folgendes biologische Phänomen zu modellieren: Nach einem Wechsel in der Zufütterung oder einer Hungerphase kann nach erneuter Substratzufuhr häufig eine zeitliche Verzögerung beobachtet werden, bevor das Zellwachstum wieder voll einsetzt. Dieses Phänomen kann unter anderem durch eine langsame Substrataufnahme verursacht werden, die mittels Zwischenspeicher auf eine einfache Art und Weise modelliert werden kann. Allerdings ist anzunehmen, dass die meisten Zellen sehr effektive Transportmechanismen entwickelt haben, und eine zeitliche Verzögerung kaum signifikant sein dürfte. Werden allerdings keine synthetischen Minimalmedien beziehungsweise nicht die Primärquellen des Organismus als Nährstoffe eingesetzt, gilt dies nicht unbedingt. Beispielsweise kann die Umwandlung langkettiger Kohlenhydrate in kleinere Moleküle der zeitbestimmende Schritt sein.

Wenn eine Umstellung des Stoffwechsels die beobachtete Verzögerung verursacht, wird diese als Anlauf- oder Latenzzeit oder *Lag-Phase* bezeichnet. Ist die Modellierung durch einen Substratspeicher nicht ausreichend, können andere, später vorgestellte Modellierungsansätze unter Umständen bessere Ergebnisse erzielen. Eine weitere Ursache für eine entsprechende Beobachtung zu Beginn einer Fermentation ist ebenfalls möglich: Ein heterogener Anfangszustand der Biomasse, insbesondere bezüglich ihrer Zellteilungsfähigkeit. Ein modelltechnisch zu empfehlender Ansatz sind variable Anfangswerte für die Parameteridentifikation. Abschnitt 3.4 erläutert diese und gibt auch ein anschauliches Beispiel für die Situation mit einer inhomogenen Biomasse.

Die dritte Variante der Stoffffluss-Implementation in der Abbildung 3.8 erscheint zunächst unnötig, unterscheidet sich jedoch in einem wesentlich Punkt von der zweiten. Sie ist auch geeignet, um zellexterne Speicher zu beschreiben. Hierbei entfällt auch die Notwendigkeit eine maximale Kapazität für den Speicherstoff innerhalb der Zellen bestimmen zu müssen. Zur Vereinfachung ist es zusätzlich möglich, nur einen der beiden Speicherungsvorgänge zu modellieren: Beispielsweise kann die Auslagerung als einfacher Zerfall des Speicherstoffs betrachtet werden (Reaktion erster Ordnung). Somit ist die Zerfallsgeschwindigkeit proportional zum Speicherinhalt. Dadurch entfällt die Notwendigkeit für eine Formalkinetik und reduziert die Anzahl der Modellparameter.

Grundsätzlich können biologische Wachstumsreaktionen mit diesem Modelltyp weiterhin nur stark vereinfacht dargestellt werden. Wie die Ausführungen außerdem exemplarisch zeigen, gibt es viele mögliche Modellstrukturen, um einen beobachteten Effekt zu beschreiben. Sie von Hand zu erstellen und miteinander zu vergleichen ist aufwendig. Die automatisierte Modellbildung ermöglicht es, die beste Struktur einfach durch einen Vergleich zu ermitteln.

Die Gesamtbiomasse ist bei strukturierten Modellen kein eigener Zustand. Stattdessen gibt es die Masse der einzelnen Speicher und eine „Rest-Biomasse“. Aus der Summe dieser Zustände ergibt sich dann die Gesamtbiomasse. Wie erwähnt, sind Speicher unter Umständen messtechnisch nicht zu erfassen, ebenso gilt dies für die „Rest-Biomasse“[12]. Nur die Gesamtmasse ist bestimmbar, nicht deren Aufteilung. Für die Lösung der Modellgleichungen werden jedoch Anfangswerte für alle Zustände benötigt. Die in diesem Fall messtechnisch nicht ermittelbaren Anfangswertparameter müssen erst durch eine Parameteridentifikation bestimmt werden.

Um die Identifikation zu erleichtern, ist darauf zu achten, dass die Vorkulturführung für die einzelnen Fermentationen identisch durchgeführt wird. Dann sollten die identifizierten Startwerte auch in verschiedenen Experimenten sehr ähnlich sein. Werden trotzdem starke Unterschiede identifiziert, kann dies darauf hindeuten, dass die Vorkulturführung nicht wie geplant durchgeführt wurde, weil zum Beispiel ein beeinflussender Umweltfaktor bislang nicht erkannt und berücksichtigt wurde.

Zusammengefasst sind Modelle mit Speicher nützlich, um Fermentationen zu beschreiben, bei denen eine Limitation von Nährstoffen auftritt. Mit ihnen können sowohl Wachstumsunterbrechungen und eine reduzierte Wachstumsgeschwindigkeit, als auch eine nicht-wachstumsassoziierte Produktion besser modelliert werden als mit den unstrukturierten Modellen des letzten Abschnitts. Dennoch kann auch ihr Nutzen begrenzt sein, wenn sich innerhalb einer Limitationsphase der Metabolismus eines Mikroorganismus signifikant ändert und dies im Rahmen dieses einfachen Modelltyps nicht modelliert werden kann. Die Stoffwechselumstellung auf eine sekundäre Substratquelle während der Fermentation ist ein Beispiel für eine derartige Änderung. Für eine bessere Modellierung bieten sich Kompartiment-Modelle der folgenden Abschnitte an.

[12] Es ist durchaus möglich, über die Messung von bestimmten Zellwand-Proteinen zumindest einen Teil der restlichen Biomasse zu erfassen. Im Rahmen der einfachen Modelle mit Speicher ist eine derartige Modellierung aber meist zu detailliert.

Einfach strukturierte Kompartiment-Modelle

Einen Schritt weiter gehen Kompartimentmodelle. Im Rahmen dieser Arbeit bezeichnet ein Kompartiment nicht einzelne, konkrete Molekülarten, sondern eine funktionelle Gruppe von Substanzen in einer Zelle, wodurch einige dieser Kompartimente generell nicht direkt messbar sind. Durch die Aufteilung in Kompartimente wird eine Anlehnung an reale biologische Vorgänge angestrebt. Abbildung 3.9 zeigt die Kompartimente eines einfach strukturierten Modelltyps, wie er in dieser Arbeit definiert ist.

Die Speicher des Modelltyps aus dem letzten Abschnitt für Phosphat und Stickstoff werden durch die Kompartimente „Nukleotide“ und „Aminosäuren“ ersetzt, was zunächst nur eine Umbenennung darstellt, aber einen sinnvollen Zwischenschritt zum nächstkomplexeren Modelltyp, siehe unten, darstellt. Die entsprechenden Modellzustände können in beliebigen Reaktionen verwendet werden, während die Speicher eben nur Zwischenlager für die Substrate darstellen. Beide Molekülgruppen sind nur mit erhöhtem Aufwand zu messen, und werden – soweit es für den jeweiligen Anwendungsfall möglich ist – zusammengefasst zu jeweils einem Modellkompartiment.

Die dargestellte aktive Zellmasse vereinigt die biologischen Funktionen von DNS, RNS und Proteinen. Zur passiven Masse zählt neben den Speichern in diesem Modelltyp auch die Zellwand[13]. Somit besteht auch das Kompartiment „passive Zellmasse“ aus vielen unterschiedlichen Stoffen, so dass es in der Praxis nicht direkt gemessen werden kann.

Die Abbildung zeigt auch die typischen Stoffflüsse zwischen den Kompartimenten, die sich aus biologischem Verständnis ergeben. Das Reaktionsschema für das Wachstum ist damit grundsätzlich festgelegt.

Zu den dargestellten Kompartimenten wären zusätzlich auch Substrat-Speicher wie beim Modelltyp aus dem vorherigen Abschnitt möglich. Dies ist im Allgemeinen nicht notwendig, da die Kompartimente „Nukleotide“ und „Aminosäuren“ wie bereits erwähnt die Speicher ersetzen. Außerdem ist es auch nicht empfehlenswert, da mit diesen Speichern weitere nicht messbare Größen in das Modell aufgenommen würden. Dies kann zu Identifikationsproblemen der Modellparameter führen. Insbesondere ist dies der Fall, wenn in einem Stoffffluss (z. B.: Phosphat → Nukleotide → Aktive Zellmasse → Produkt) mehrere nicht messbare Zustände unmittelbar hintereinander aufgeführt werden. Daraus

[13] Ihr Aufbau verbraucht Ressourcen, sie ist aber in der vereinfachten Darstellung – im Gegensatz zu den Speichern – nicht an weiteren Reaktionen beteiligt. Die Zellwand und weitere Bestandteile mit dieser Eigenschaft werden im Folgenden auch Strukturelemente genannt.

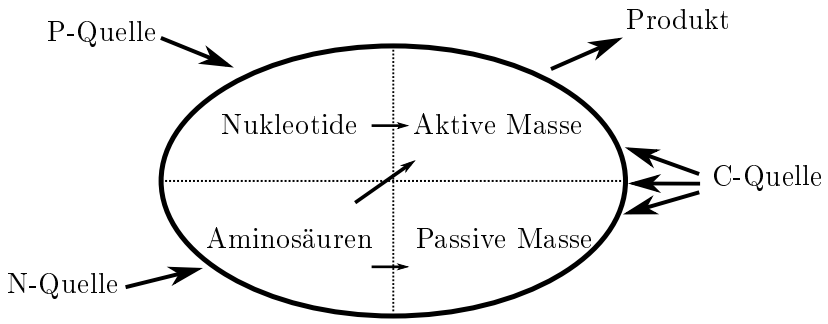


Abbildung 3.9: Einfach strukturiertes Kompartimentmodell

folgt, dass zumindest die „aktive Zellmasse“, bestehend aus DNS, RNS und Proteinen, gemessen werden sollte, um die Modellparameter identifizieren zu können. Tendenziell verstärkt sich das Problem einer fehlenden Identifizierbarkeit mit einer steigenden Anzahl von nicht messbaren Zuständen.

Wesentliches Merkmal für den einfach strukturierten Modelltyp ist das bereits erwähnte Kompartiment „aktive Biomasse“. Durch die Trennung in eine aktive Masse und den Rest wird eine für die Modellierung hilfreiche Unterscheidung zwischen Zellzahl und Zellmasse angestrebt. Diese Unterscheidung ist relevant, da sich die Masse einer einzelnen Zelle verändern kann, ihre Syntheseleistung (Wachstum, Produktion) aber nicht unmittelbar von ihrer Masse abhängt. Diese Differenzierung im Modell ist besonders dann hilfreich, wenn sich die Masse einer einzelnen Zelle während einer Fermentation stark verändert.

Dieser Modelltyp sollte verwendet werden, wenn die Speicher der Zellen über ausgeprägte Kapazitäten verfügen und daher bei ausreichender Nährstoffversorgung deutlich größer werden und somit die Gesamtmasse der einzelnen Zelle beeinflussen. Außerdem wird durch die gewählten Kompartimente näherungsweise der innere Zustand der Zellen charakterisiert, so dass beispielsweise durch das Modell ein Wechsel von Wachstum auf Produktion dargestellt werden kann.

Mittelstrukturierte Kompartimentmodelle

Bei diesem Modelltyp wird eine größere Anzahl von Kompartimenten eingesetzt, um das Verhalten der Zellen zu beschreiben, siehe Abbildung 3.10. Mit ihm wird die modellhafte Annäherung an die komplexen systembiologischen Modelle vollzogen, ohne aber das Ziel der Prozessmodell-Bildung zu vernachlässigen. Die Aussagen bezüglich der Identifizier-

barkeit von Modellparametern aus dem letzten Abschnitt gelten weiterhin. In dieser wie auch in Vorläuferarbeiten ([13],[41]) werden DNS, RNS und Gesamtprotein sowie Aminosäuren, Nukleotide und Strukturelemente betrachtet, wobei die ersten drei gemessen werden, so dass nur die Verkettung Aminosäuren und Strukturelemente unbeobachtet bleibt. Das Reaktionsschema ist, wieder mit Ausnahme der Produktbildung, festgelegt und wieder aus vereinfachtem, biologischen Verständnis abgeleitet.

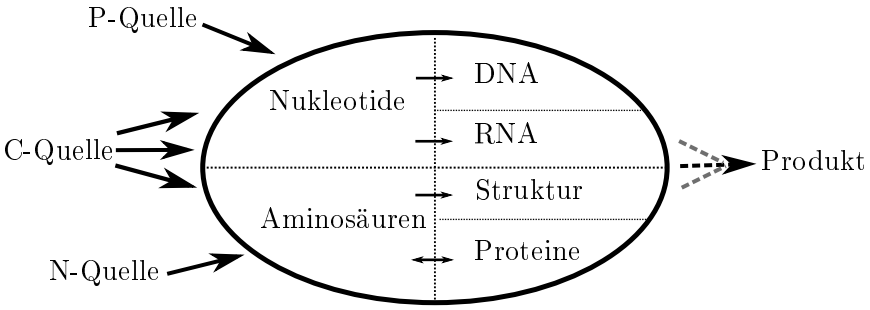


Abbildung 3.10: Mittelstrukturiertes Kompartimentmodell

Dadurch, dass die inneren Zustände der Zellen detaillierter modelliert werden, kann die Zellaktivität besser beschrieben werden. Zugleich ermöglichen die Messungen auch eine bessere Identifizierbarkeit der Kompartimente der wichtigen biochemischen Vorstufen Nukleotide und Aminosäuren. Limitationen und Stoffwechseländerungen können somit genauer beschrieben werden. Insbesondere die Reaktionen zur Produktbildung können mit diesem Modelltyp besser modelliert werden: Die Produktbildungsgeschwindigkeit kann nun von den einzelnen Kompartimenten im Modell abhängen, beispielsweise von DNS, RNS oder der Gesamtmasse an Proteinen. Biologisches Wissen kann bei der richtigen Zuordnung helfen. Ist eine Zuordnung nicht möglich, können wie später dargestellt durch die automatische Modellierung verschiedene Möglichkeiten einfach erzeugt und schnell überprüft werden.

Ungeeignete Werte für die Ausbeutekoeffizienten oder unpassende Reaktionsraten können bei diesem Modelltyp dazu führen, dass „überschüssige Massen“ in ein nicht messbares Kompartiment verschoben werden, hier insbesondere in die Strukturelemente. Aufgrund der fehlenden Messwerte, verursacht dies im Rahmen einer Parameteridentifikation keinen zusätzlichen Beitrag zur Summe der Fehlerquadrate und somit kann dieser Modellfehler auch nicht minimiert werden. Bis zu einem gewissen Grad ist diese „Auslagerung“ annehmbar, da eine exakte Modellierung der biochemischen Zusammenhänge

nicht verlangt werden kann. Nach einer Modellanpassung sollten aber in den Simulationen die zeitlichen Verläufe der nicht messbaren Konzentrationen kritisch betrachtet werden. Erneut kann ein biologisches Grundverständnis hier helfen und in die Bewertung der Lösung einfließen. Unter Umständen sind andere, sinnvoller erscheinende Startwerte zu verwenden, oder andere Optimierer einzusetzen, um eine plausible Lösung zu erreichen.

Die Verwendung noch stärker strukturierter Modelle bedarf der Unterstützung durch eine erweiterte Messtechnik, die für diese Arbeit nicht zur Verfügung stand. In verschiedenen Projekten zeigte sich allerdings, dass Prozesse bereits mit mittelstrukturierten Modellen gut approximiert und in der Folge auch erfolgreich optimiert werden konnten [41, 43, 44].

Systembiologische Modelle

Systembiologische Modelle gehen noch einen großen Schritt weiter und beinhalten alle (bekannten) Stoffwechselreaktionen. Mit Hilfe dieser Modelle wird der Stofftransportweg detailliert beschrieben, aber nicht der dynamische Transport selber: Im Gegensatz zu den bisher vorgestellten Modelltypen stellen sie grundsätzlich nur die Verteilung der zellinternen Substanzen bei einer bestimmten (konstanten) Nährstoff-Situationen dar. Der messtechnische Aufwand für die notwendigen Informationen ist bedeutend. Er steigt stark an, wenn auch dynamische Nährstoff-Situationen untersucht werden sollten, wobei viele Messungen für eine dynamische Beschreibung *in vivo* nicht durchführbar sind. Somit enthalten die bisherigen systembiologischen Modelle nur vereinzelt Informationen darüber, unter welchen Bedingungen die einzelnen Reaktionen stattfinden, mit welchen Reaktionsraten diese ablaufen, oder durch welche Größen diese prinzipiell reguliert werden. Diese Informationen müssten durch Messungen ebenfalls erlangt werden.

In der aktuellen Literatur finden sich jedoch bereits einige Versuche, diese Einschränkungen zumindest teilweise aufzuheben. Dies soll durch die geschickte Verknüpfung verschiedener Informationsquellen und Modellansätzen gelingen [47, 54, 89]. Systembiologische Modelle werden hier nicht weiter betrachtet.

3.2 Messungen

Zur Modellbildung gehört auch, dass die Modellparameterwerte identifiziert werden. Dies geschieht in der Regel im Rahmen einer Parameteridentifikation, die im folgenden Ab-

schnitt beschrieben wird. Als Grundlage für diese werden zunächst Messdaten benötigt, weswegen sich dieser Abschnitt mit den Messungen befasst. Im Folgenden wird nach einigen einleitenden Sätzen die Messsituation vorgestellt, der die hier vorliegenden Arbeit zugrunde liegt.

Es lassen sich verschiedene Anforderungen an gute Messdaten formulieren. Zunächst ist die Qualität, also die Genauigkeit, der Messwerte wichtig. Speziell gilt dies, wenn Messwerte wie Konzentrationen vergleichsweise dicht an null liegen: Die Unterscheidung „null“ oder „fast null“ ist für biologische Systeme in vielen Fällen entscheidend. Die Messgenauigkeit der einzelnen Messgrößen sollte auch bestimmt werden, weil sie als Information für einige regelungstechnische Methoden wichtig ist, beispielsweise in Form der sogenannten Kovarianzmatrix des Messrauschens.

Neben der Qualität der Messdaten ist auch deren Quantität wesentlich, wenn dynamische Prozesse zu beschreiben sind. Die zeitliche Verteilung der Messungen ist ebenso wichtig: Fehlen Messungen in sehr dynamischen Zeitabschnitten einer Fermentation, gibt es keine Informationen für das Modell über das dynamische Zell- oder Stoffwechsel-Verhalten. Die Identifikation der entsprechenden Kinetikparameter ist in diesem Fall nicht möglich. Umgekehrt ergibt sich hieraus allerdings auch, dass nicht die gesamte Fermentationszeit stark beprobt werden muss, sondern nur die Zeitabschnitte der Fermentation mit hohem Informationsgehalt sorgfältig vermessen werden sollten.

Die letzte Anforderung an die Messdaten ist auch eine Forderung an die Prozessführung: Das System muss ausreichend angeregt werden, damit alle Dynamiken, die modelliert werden sollen, in Erscheinung treten und gemessen werden können. Soll das Modell beispielsweise eine Substrat-Limitation beschreiben können, so sind zur Identifikation der Parameter auch Versuche derart durchzuführen, dass eine Limitation eintritt.

Insbesondere bei einem komplexeren System ist es nicht einfach, geeignete Versuchsbedingungen zu finden, die seine gesamte Dynamik ausreizen. Auch dürften es nur die wenigsten biologischen Experimente ermöglichen, alle Informationen, die für eine Modellbildung erforderlich sind, mit einem einzigen Versuch zu erfassen. Man kann aber die Anzahl der notwendigen Versuche reduzieren, indem der Informationsgehalt der einzelnen Versuche maximiert wird. Dies ist das Ziel der *Optimalen Versuchsplanung*, die in dieser Arbeit nicht weiter behandelt wird. Die Durchführung mehrerer Versuche bleibt dennoch sinnvoll, um den Einfluss des Messrauschens reduzieren zu können und eine Reproduzierbarkeit zu überprüfen.

Ausgangspunkt der Auswahl an Messgrößen für die Modellbildung stellen Vorgängerprojekte [13] dar, in denen Messmethoden für folgende Größen etabliert wurden:

- Glucose-Konzentration (oder Konzentration alternativer C-Quelle)
- Phosphat-Konzentration (oder Konzentration alternativer PO_4-Quelle)
- Ammonium-Konzentration (oder Konzentration alternativer N-Quelle)
- DNS-Konzentration
- RNS-Konzentration
- Gesamt-Protein-Konzentration
- Biotrockenmasse
- OD (optische Dichte)

Zusätzlich werden auch die für die pH-Wert-Regelung zugeführten Volumina von Säure und Base als Messwert erfasst. Für einzelne Projekte werden weitere Messdaten erhoben, zum Beispiel für die Produkte oder Nebenprodukte (Ethanol, verschiedene Antibiotika) der jeweiligen Mikroorganismen. Auf den genannten Messgrößen basiert im Wesentlichen die Modellentwicklung, die auch zur Programmierung des Modellstrukturgenerators geführt hat.

Zur Überwachung einer laufenden Fermentation werden folgende Messgrößen zusätzlich erfasst, die aber nicht in die Modellbildung einfließen, teils weil unterlagerte Regelungen dafür sorgen, dass sie über Versuche hinweg konstant sind, teils weil davon ausgegangen wird, dass von ihnen kein limitierender Effekt ausgeht:

- pO_2 (Gelöstsauerstoff)
- pCO_2 (Gelöstkohlendioxid)
- Temperatur
- Drehzahl des Rührers
- pH-Wert
- Airflow (Begasungsrate des Fermenters)
- Sauerstoff-Konzentration (in der Abluft)
- Kohlendioxid-Konzentration (in der Abluft)

3.3 Parameteridentifikation

Da die Parameteridentifikation eine wichtige Methode bei der Modellbildung darstellt, wird sie in diesem Abschnitt näher beschrieben. Ihr grundsätzliches Ziel ist es, jene Parameterwerte zu ermitteln, mit denen die Lösung einer Modellgleichung den zeitlichen

Verlauf eines durch Messungen beobachteten realen Experiments möglichst exakt wiedergeben kann. Die Differenz $\underline{e}$ zwischen Messwertverlauf $\underline{y}^{\text{Mess}}$ und Simulationsverlauf $\underline{h}$ ergibt sich zu einem Zeitpunkt t_k durch:

$$\begin{aligned} \underline{e}_k &= \underline{y}_k^{\text{Mess}} - \underline{h}(t_k, \underline{x}_k, \underline{\theta}, \mathbf{U}) \qquad \text{oder kurz} \\ &= \underline{y}_k^{\text{Mess}} - \underline{h}_k \end{aligned} \tag{3.23}$$

Darin sind $\underline{x}$ die Zustände, $\underline{\theta}$ der Parametervektor und $\mathbf{U}$ die Stellgrößenmatrix.

Der Fehler $\underline{e}$ geht in eine Gütefunktion Φ ein, wobei sich dafür mehrere Alternativen anbieten, siehe unten. Allgemein wird durch einen mathematischen Optimierungsprozess versucht, Φ zu minimieren. Erst die Lösung des Optimierungsproblems, Gleichung 3.24, ergibt die gesuchten, optimalen Parameter $\underline{\theta}^*$:

$$\underline{\theta}^* = \arg\min_{\underline{\theta}} \Phi(\underline{\theta}) \tag{3.24}$$

$$\text{U. d. B.} \qquad \dot{\underline{x}} = f(t, \underline{x}, \mathbf{U}, \underline{\theta}), \quad \underline{x}(t=0) = \underline{x}_0$$

Im Folgenden werden verschiedene Möglichkeiten für die Funktion Φ vorgestellt.

Ein intuitiver und häufig verwendeter Ansatz minimiert die Summe der Fehlerquadrate (englisch: *least squares*, LS) zwischen Simulation und Messung, siehe Gleichung 3.25. Ein erweiterter Ansatz verwendet eine gewichtete Summe (engl.: *weighted least squares*, WLS) mit der die einzelnen Messwertreihen besser zu einander in Bezug gesetzt werden können, siehe Gleichung 3.26. Dies ist relevant, wenn sich die Größenordnungen der Messwerte der einzelnen Messgrößen oder die Anzahl der jeweiligen Messwerte deutlich voneinander unterscheiden. Mit Hilfe der Gewichtung kann auch die bewusste Entscheidung getroffen werden, dass die Modellanpassung an bestimmte Messgrößen wichtiger ist als an andere. Die Wahl der Gewichtung ist nicht einfach zu treffen, worauf noch näher eingegangen wird.

$$\Phi_{\text{LS}}(\underline{\theta}) = \sum_{k=1}^{N} \underline{e}_k{}^{\text{T}} \cdot \underline{e}_k \tag{3.25}$$

$$\Phi_{\text{WLS}}(\underline{\theta}) = \sum_{k=1}^{N} \underline{e}_k{}^{\text{T}} \cdot \mathbf{W} \cdot \underline{e}_k \tag{3.26}$$

Die Anzahl der Messzeitpunkte eines Versuchslaufs wird hier mit N bezeichnet. Sind F Experimente in die Parameteridentifikation einzubeziehen, so werden die Summen in

Gleichung 3.25 beziehungsweise 3.26 für jedes Experiment berechnet und anschließend summiert. N_F ist hier dann die Anzahl der Messdaten ${}^{(f)}\underline{y}$ aus dem Versuch f, der hochgestellte Index (f) kennzeichnet den jeweiligen Versuchslauf. Es ergibt sich:

$$\Phi_{\mathrm{WLS}} = \sum_{f=1}^{F} \sum_{k=1}^{N_F} {}^{(f)}\underline{e}_k{}^{\mathrm{T}} \cdot \mathbf{W} \cdot {}^{(f)}\underline{e}_k \tag{3.27}$$

Zur Vereinfachung der Darstellung wird im Folgenden auf die Berücksichtigung mehrerer Versuchsläufe verzichtet, das heißt $F = 1$, somit entfällt die zusätzlich notwendige Summation und auf die explizite Verwendung des hochgestellten Index (f) kann verzichtet werden.

Ein bekanntes Verfahren zur Bewertung der Differenz zwischen Messwert und Simulation ist als Maximum Likelihood Estimation (MLE) bekannt. Es berücksichtigt, dass Messwerte durch Messrauschen verfälscht werden. Gelingt es, die Messunsicherheit statistisch zu beschreiben, kann die folgende Frage gestellt werden: Wie wahrscheinlich ist es, dass das Modell die realen, verrauschten Messdaten $\underline{y}_k^{\mathrm{Mess}}$ erzeugen kann? Daraus folgt die Optimierungsaufgabe, die Modellparameter $\underline{\theta}$ so zu verändern, dass die beobachteten Messwerte den wahrscheinlichsten entsprechen.

Für lineare Systeme kann nachgewiesen werden, dass die MLE die größtmögliche Sicherheit bezüglich der identifizierten Parameter liefert: Die Kovarianz der Identifikationsfehler der Parameter wird minimal. Obwohl die Beweisführung nur für lineare Systeme gilt und somit keine Bedeutung für die hier verwendeten Systeme hat, erweist sich das Verfahren in der Praxis auch für nichtlineare Systeme als sehr leistungsfähig.

Unter der Voraussetzung, dass das Messrauschen unkorreliert und normalverteilt sowie seine Kovarianzmatrix bekannt ist, führt die mathematische Formulierung zur Form von Gleichung 3.26. Der Unterschied beider Verfahren besteht darin, dass die Gewichtungsmatrix bei der MLE nicht manuell wie bei der Weighted Least Squares (WLS) festgelegt werden muss, sondern aus einer statistischen Betrachtung der Messdaten stammt: Statt einer willkürlichen Gewichtungsmatrix wird für die MLE die Inverse der Kovarianzmatrix des Messrauschens verwendet: $\mathbf{C}_y{}^{-1}$. Für die Parameteridentifikation ergibt sich das folgende Gütefunktional:

$$\Phi_{\mathrm{MLE}}(\underline{\theta}) = \sum_{k=1}^{N_F} \underline{e}_k{}^{\mathrm{T}} \cdot \mathbf{C}_y{}^{-1} \cdot \underline{e}_k \tag{3.28}$$

Sind die Messsensoren zusätzlich unabhängig voneinander, kann folgende Vereinfachung angewandt werden: Die Elemente der Hauptdiagonalen von $\mathbf{C}_y$ entsprechen den

Varianzen der Messgenauigkeiten der einzelnen Messgrößen. Kann man diese aus technischen Beschreibungen (bei Messgeräten) entnehmen oder sie experimentell bestimmen, können diese Werte direkt verwendet werden. Die restlichen Elemente der Matrix sind in diesem einfachen Fall unabhängiger Messsensoren gleich null.

Kann man die Kovarianzmatrix der Messfehler so nicht aufstellen, muss sie mit geeigneten Verfahren vergleichsweise aufwendig aus den (verrauschten) Messdaten geschätzt werden. Eine Voraussetzung für das Schätzverfahren ist, dass die Anzahl der Messdaten mindestens so groß wie die Anzahl der Messgrößen ist. Für eine gute Schätzung sollte die Anzahl der Messdaten sehr viel größer sein: $N_F \gg \dim(\underline{e}_k)$. Die Schätzung muss in jedem Iterationsschritt der Optimierung erneuert werden, da sie abhängig von den aktuellen Parameterwerten ist. In aller Regel werden auch die Elemente außerhalb der Hauptdiagonalen dabei ungleich null. Die Schätzung erfolgt durch

$$\widetilde{\mathbf{C}}_y = \frac{1}{N_e - \dim(\widehat{\theta})} \sum_{k=1}^{N_e} \underline{e}_k \cdot \underline{e}_k{}^{\mathrm{T}}. \tag{3.29}$$

Darin ist $\dim(\widehat{\theta})$ die Anzahl der unbekannten Modellparameter. Die mitgeschätzte Kovarianzmatrix $\widetilde{\mathbf{C}}_y$ aus der MLE ist gültig für alle Messzeitpunkte. Eine Gewichtung, die beispielsweise die variierende Größenordnung der Messwerte berücksichtigt, kann auf diese Weise nicht vorgenommen werden. Eine entsprechende Variante zur Berechnung ist in [30] aufgeführt.

Für die hier betrachtete Arbeit wird eine Mischform aus WLS und MLE eingesetzt. Aus der Erkenntnis, dass Informationen bezüglich des Messrauschens in die Berechnung einfließen sollten, wird die Hauptdiagonale der $\mathbf{C}_y$-Matrix mit den entsprechenden bekannten Varianzen versehen. Anders als bei der MLE vielfach vereinfachend angenommen werden diese jedoch für jeden Zeitpunkt einzeln bestimmt, um die korrekte Wahrscheinlichkeitsdichte zu berücksichtigen: $\mathbf{C}_y{}^{(k)}$. Eine zusätzliche, diagonale Gewichtungsmatrix wird ebenfalls eingeführt. Ihre Berechnung erfolgt teilweise aufgrund von heuristischen Regeln. Zum Beispiel werden die zeitliche Messdatendichte und -anzahl einbezogen, mit dem Ziel, die Identifikationsgeschwindigkeit und -güte zu erhöhen. Die entsprechenden Algorithmen wurden in [30] beschrieben. Die Möglichkeit, direkte Vorgaben bezüglich einer Priorität von Messgrößen oder Versuchsläufen festzulegen, existiert im ABC-System ebenfalls. Die tatsächlich eingesetzte Gleichung für die Methode der erweiterten MLE (EMLE) lautet:

$$\Phi_{\mathrm{EMLE}}(\underline{\theta}) = \sum_{k=1}^{N_F} \left(\underline{e}_k{}^{\mathrm{T}} \cdot \mathbf{W}_k \circ (\mathbf{C}_y{}^{(k)})^{-1} \cdot \underline{e}_k \right) \tag{3.30}$$

Der $\circ$-Operator bezeichnet das Schur- oder elementweise Produkt[14] der Matrizen.

Unabhängig vom verwendeten Verfahren zur Berechnung von Φ gilt, dass die Güte der Parameteridentifikation unmittelbar von der Qualität (und Quantität) der Messdaten abhängt. Nachdem die Parameter eines Modells so angepasst wurden, dass das Modell die Messdaten der bereits vorhandenen Experimente beschreiben kann, ist es geeignet, die Messdaten zu interpolieren und so Zwischenwerte zu generieren. Auf diese Weise sind unter Umständen Zusammenhänge und Trends besser zu erkennen. Auch kann nun mit Hilfe des Modells berechnet werden, wie sich eine veränderte Anfangsbedingung oder ein anderes Zufütterungsprofil auswirkt. Genaugenommen handelt es sich in diesem Fall um eine Extrapolation. Ihre Zuverlässigkeit hängt unter anderem von den zuvor verwendeten Versuchen zur Parameteridentifikation ab. Um die Vorhersage-Fähigkeit zu beurteilen, gibt es verschiedene Möglichkeiten der Modellvalidierung [61], der Parameteranalyse über die Fischer-Informationsmatrix[15] [48] oder der Bootstrap-Methode [21] und die Untersuchung der Identifizierbarkeit [6, 36, 74].

3.3.1 Sequentielle Parameteridentifikation

Bei neu erstellten, komplexen Modellen (insbesondere mit nicht messbaren Zuständen) tritt oft das Problem auf, dass für die Modellparameterwerte die jeweiligen Größenordnungen nicht einmal näherungsweise bekannt sind. In dieser Situation können willkürlich gewählte Startwerte für die Parameteridentifikation so weit vom Optimum entfernt liegen, dass die Identifikation aller Parameter eine große numerische und zeitliche Herausforderung darstellt. Gradienten-basierte Optimierer können in diesem Fall sogar gänzlich versagen. Abhängig von den zur Verfügung stehenden Messdaten kann es sinnvoll sein, zunächst nur eine begrenzte Auswahl von Parametern identifizieren zu lassen oder die Bedeutung einzelner Messgrößen durch eine entsprechend starke Gewichtung zu verändern. Die Schwierigkeit hier ist, eine geeignete Auswahl oder Gewichtung zu finden.

Ein alternativer Ansatz, der sich auch automatisieren lässt, ist die *Sequentielle Parameteridentifikation* nach einer Grundidee von [42]. Eine ähnliche Methode wurde von [4]

[14] benannt nach Issai Schur (1875 – 1941, Mathematiker, Deutschland)

[15] benannt nach Sir Ronald Aylmer Fisher (1890 – 1962, Statistiker, England)

beschrieben. Bei der in dieser Arbeit implementierten Sequentiellen Parameteridentifikation werden die einzelnen Modellgleichungen in zwei oder mehr Gruppen so aufgeteilt, dass die individuellen Gruppen möglichst unabhängig voneinander sind. So kann es zum Beispiel bei den typischen Modellierungen biologischer Systeme gelingen, die biotischen Differentialgleichungen – also jene Gleichungen, die das Zellwachstum des Organismus beschreiben – von denen zu trennen, die die Erzeugung der Zwischen- oder Endprodukte beschreiben.

Die Methode der Sequentiellen Parameteridentifikation ist auf eine Weise implementiert worden, die es erlaubt, die bislang automatisch erstellten Modelldateien mit wenigen Änderungen weiter verwenden zu können. Sequentiell werden die Parameter der einzelnen Gruppen von Gleichungen identifiziert. Dazu werden die Gleichungen aus der aktuell gewählten Gruppe wie bisher durch einen Löser behandelt. Für die Übrigen werden zunächst die jeweiligen Messwerte in Zustandswerte umgerechnet und interpoliert. Dann werden die rechten Seiten der betreffenden Differentialgleichungen durch eine Ableitung der interpolierten Daten ersetzt. Effektiv entfallen damit die Parameter, die ausschließlich in den ersetzten Differentialgleichungen auftreten. Dieses Vorgehen ermöglicht es, dass nicht alle Parameter des gesamten Differentialgleichungssystems gleichzeitig bestimmt werden müssen. So kann jeweils ein Teil des Modellsystems identifiziert werden, ohne dass die übrigen Teile bereits identifiziert sein müssen.

Die Flexibilität bei dieser Erweiterung des ABC-Systems bedingt einige Einschränkungen: So ist es erforderlich, dass die zeitlichen Ableitungen der jeweiligen Zustände des Differentialgleichungssystems sich tatsächlich aus Messdaten berechnen lassen. Die entsprechenden Messgleichungen müssen sich explizit nach dem jeweiligen Zustand auflösen lassen. Es dürfen somit auch keine Abhängigkeiten von weiteren, zu bestimmenden Zuständen bestehen. Für Konzentrationen von Massenzuständen stellt diese Forderung in den meisten Fällen keine Einschränkung dar[16].

Die in der Praxis oft begrenzte Menge der zur Verfügung stehenden Messgrößen und genauen Messdaten verhindert allerdings häufig eine ausreichend gute Interpolation für die Sequentielle Parameteridentifikation. Das Automatische Probennahmesystem (AProS), siehe Kapitel 2, ermöglicht aber eine ausreichend hohe Anzahl von Messzeitpunkten. Die Sequentielle Parameteridentifikation ist daher vor allem in automatisierten Laborumgebungen vorteilhaft.

Eine zukünftig umzusetzende, neue Implementation der Sequentiellen Parameteriden-

[16] Das Volumen gilt in diesem Zusammenhang nicht als Zustand: Seine Werte können jederzeit unabhängig vom restlichen Differentialgleichungssystem berechnet werden.

tifikation könnte einige der oben genannten Einschränkungen umgehen: In einer weiteren, automatisch zu erstellenden Programmvariante eines Modells könnten in den Formalkinetiken anstelle der aus Zuständen berechneten Messwerten direkt die tatsächlich gemessenen Werte einfließen. Eine Interpolation der Messwerte kann jedoch so auch nicht vermieden werden, da tatsächliche Messwerte typischerweise nur zu sehr wenigen Löserschritt-Zeitpunkten existieren.

Durch den Ansatz der Sequentiellen Parameteridentifikation verringert sich der Aufwand des numerischen Optimierers, womit das Verfahren insgesamt zu einer Beschleunigung der Identifikation aller Parameter führen kann. Es empfiehlt sich, als letzten Schritt eine Parameteridentifikation für alle Parameter gemeinsam durchzuführen.

Auch wenn die Voraussetzungen bezüglich der Unabhängigkeit der einzelnen Gruppen nicht erfüllt werden, kann das Verfahren eingesetzt werden, um passende Startwerte für nicht messbare Zustände zu finden. So ergänzt die Methode der Sequentiellen Parameteridentifikation die gewöhnliche Parameteridentifikation.

Im ABC-System kann die Sequentielle Parameteridentifikation als Option im Rahmen einer normalen Parameteridentifikation gewählt werden. Die Zuordnung der einzelnen Gleichungen in die Gruppen muss jedoch zuvor manuell vorgenommen werden (in der `SystemInit_SI_PI`).

3.4 Versuchsabhängige Parameter

Die bisher erwähnten Optimierungsparameter $\underline{\theta}$ sind modellabhängig. Das heißt, für jeden Modellkandidaten existieren verschiedene Parametersätze. Neben diesen modellabhängigen Parametern existiert eine Gruppe weiterer Parameter, deren Einführung in der Praxis sehr hilfreich ist. Beispielsweise ist die exakte Zusammensetzung des Anfangsmediums oder die genaue Startbiomasse oft nicht bekannt. In vielen Fällen werden die entsprechenden Werte zwar gemessen, unterliegen aber dem Messrauschen. Für exponentiell wachsende Systeme kann eine falsche Startbiomasse zu großen Abweichungen führen. Im Falle von strukturierten Modellen sind die Anfangszustände von zellinternen Speichern vielfach nicht exakt bekannt und können auch nicht gemessen werden. Ein vollständiger Anfangswertvektor ist in diesen Fällen nicht bekannt, der jedoch ist die Voraussetzung zum Lösen eines Differentialgleichungssystems.

Um unbekannte oder unsichere Anfangswerte berücksichtigen zu können, werden im ABC-System X_0-Parameter verwendet. Diese Anfangswert-Parameter sind zu diesem

Zweck grundsätzlich versuchsabhängig, aber nicht (zwangsweise) abhängig von einzelnen Modellkandidaten. So kann im Rahmen einer Parameteridentifikation ein passender Wert für jedes einzelne Experiment identifiziert werden.

Die parallele Berücksichtigung von einerseits versuchs- und andererseits modellabhängigen Parametern erhöht den programmiertechnischen Aufwand. So enthält beispielsweise die Menge der Optimierungsparameter normalerweise nur jeden Modellparameter einmal (pro Modellkandidat), aber jeder X_0-Parameter muss in der gleichen Vielfachheit auftreten, wie es Experimente gibt. Für die Simulation eines einzelnen Experiments innerhalb der Parameteridentifikation muss der jeweils passende X_0-Parameterwert extrahiert und in den Anfangswertvektor $\underline{x}_0$ kopiert werden.

Bei der Verwendung von zellinternen Anfangskonzentrationen als Startwert-Parameter gilt es zu beachten, dass die Summe aller Biomassen-Anteile zusammen eins ergibt. Da dies in der Parameteridentifikation nicht automatisch berücksichtigt wird, sind für die Optimierung noch Gleichheitsnebenbedingungen hinzuzufügen. Werden die Modelle durch den in dieser Arbeit vorgestellten Modellstrukturgenerator erzeugt, werden die notwendigen Nebenbedingungen automatisch formuliert und berücksichtigt.

Um die Verwendung von Anfangswert-Parametern weiter zu motivieren, sei auf Abbildung 3.11 verwiesen. Dargestellt durch die gestrichelte Linie ist eine exponentiell wachsende Startbiomasse mit homogener Zusammensetzung. Das heißt, die gesamte Masse besteht aus Zellen, die sich mit der gleichen Geschwindigkeit vermehren können. Die Masse wurde zum Zeitpunkt $t = 0$ bestimmt und der zeitliche Verlauf ergibt sich gemäß einer unbekannten Wachstumsrate. In der Praxis ist aber die Zusammensetzung oft nicht 100 %-ig homogen: Die Anfangsmasse kann identisch sein, besteht jedoch zum Beispiel nur zur Hälfte aus noch lebenden beziehungsweise zur Teilung fähigen Zellen[17], so dass neben einem konstanten Massenanteil sich nur der Rest mit der gleichen Wachstumsrate wie im idealen Fall vermehren kann. Im Ergebnis ist ein Messwertverlauf wie durch die schwarze Linie dargestellt, zu beobachten.

Eine Überprüfung mit ausreichend vielen Messwerten würde ergeben, dass sich das Wachstum mit dem fälschlicherweise angenommenen Startwert durch keine e-Funktion approximieren lässt. Wird dies ignoriert, identifiziert eine Parameteridentifikation mit diesen Messwerten eine falsche maximale Wachstumsrate. Basiert die Identifikation auf mehreren Versuche, wird der Algorithmus einen „mittleren“ Wert für die Wachstumsrate

[17] Die Knospung von Hefe-Zellen ist ein typischer Fall für ein solches Verhalten: Die maximale Anzahl von Zellteilungen durch Knospung ist durch die Oberfläche der Zellen begrenzt.

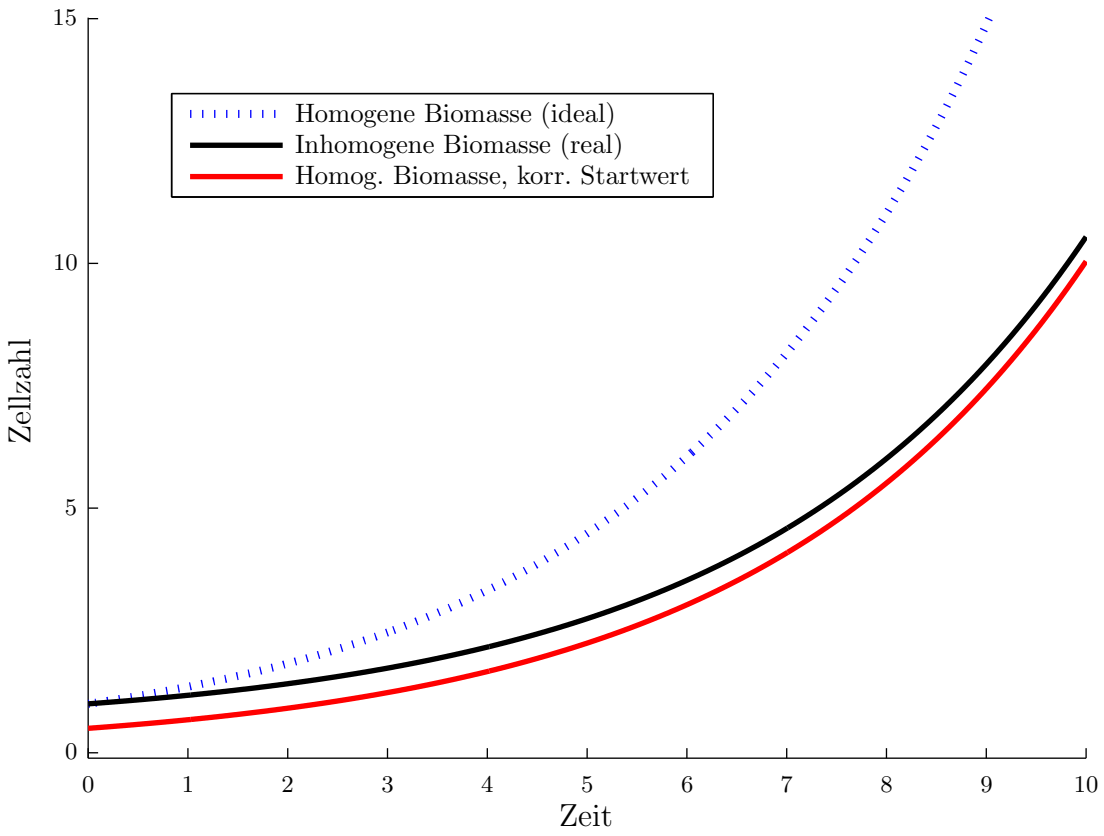


Abbildung 3.11: Verschiedene Szenarien mit unterschiedlichen Start-Biomassen

ermitteln, der im schlechtesten Fall zu keiner der bereits durchgeführten Fermentation passt. Dies ist insbesondere kritisch für die Planung neuer Versuche.

Um die Situation korrekt darzustellen, ist die Verwendung eines segregierten Modells notwendig. Aber anstatt ein solches mit großem Aufwand zu entwickeln, können die beobachteten Unterschiede auch durch eine korrigierte Startbiomasse für das Modell verringert werden. In der Abbildung zeigt die rote Linie das Wachstum mit der gleichen maximalen Wachstumsrate wie im Ausgangsfall, allerdings mit einer auf die Hälfte reduzierten Startbiomasse. Mit dieser Korrektur des Anfangswerts lassen sich die simulierten Messdaten des Beispiels durch das Modell deutlich besser beschreiben. Die Bestimmung des korrigierten Anfangswert-Parameters ist ebenfalls Aufgabe der Parameteridentifikation.

Die Ursache für ein verzögertes Wachstum kann aber auch eine Lag-Phase sein. Eine Unterscheidung ist besonders zu Beginn einer Fermentation mit wenigen Messungen oft nicht ohne weitere Informationen möglich.

3.5 Optimierung

Nichtlineare Differentialgleichungen, wie sie hier verwendet werden, lassen sich nicht analytisch lösen. Sie sind stattdessen numerisch zu behandeln. Neben der Auswahl des numerischen Lösers besitzt auch das eingesetzte Optimierungsverfahren einen großen Einfluss auf das Ergebnis einer Optimierung.

Ein allgemeines Optimierungsproblem wurde bereits im Abschnitt 3.3 über die Parameteridentifikation vorgestellt. In dieser Arbeit werden Optimierungsaufgaben als Minimierungsproblem formuliert, da die meisten Optimierungsalgorithmen ebenfalls nur nach einem Minimum suchen.

In den folgenden Abschnitten werden einige Optimierer vorgestellt, die im Rahmen dieser Arbeit eingesetzt wurden. Die Algorithmen stammen aus Matlab sowie aus den Paketen Tomlab [77] und NLopt [37]. Sie lassen sich in verschiedene Kategorien einteilen. Die Kenntnis der wichtigsten Eigenschaften der einzelnen Methoden ist vorteilhaft für eine sinnvolle Auswahl beziehungsweise effektive Anwendung. Zunächst sollen daher globale Verfahren kurz charakterisiert werden, anschließend lokale Verfahren, die weiter unterteilt werden in gradientenfreie und gradientenbasierte Verfahren. Auf eine detaillierte Beschreibung der Algorithmen wird an dieser Stelle allerdings nicht eingegangen.

3.5.1 Globale Optimierungsverfahren

Globale Verfahren suchen das globale Optimum eines Funktionals. Eine übliche Voraussetzung für numerische Verfahren ist, dass das jeweilige Optimierungsproblem beschränkter Natur ist: Für die einzelnen Parameter sind minimale und maximale Grenzen vorzugeben. In der Praxis können diese Grenzen in der Regel leicht gefunden werden. Für viele Parameter steht zum Beispiel das Vorzeichen fest, wodurch sich bereits die Null als eine Begrenzung ergibt. Durch eine später erläuterte Analyse der Formalkinetiken ergeben sich weitere Beschränkungen für numerisch sinnvolle Bereiche der Parameter[18].

[18] Für alle im ABC-System eingesetzten Modelle werden für jeden Parameter Grenzen festgelegt, unabhängig von der Wahl des Optimierungsverfahrens.

Außerdem ist eine konservative Abschätzung für die Grenzen ausreichend, um globale Verfahren einsetzen zu können. Der Rechenaufwand lässt sich indes mit kleineren Intervallen meist weiter reduzieren.

Das numerische Vorgehen der meisten globalen Optimierer ähnelt den brute-force-Methoden aus der Kryptologie, bei denen mit hohem Zeitaufwand alle denkbaren Kombinationen ausprobiert werden. Bei Modellen mit vielen Parametern ist insbesondere der Zeitaufwand für globale Optimierungsverfahren ähnlich[19], weswegen andere Verfahren häufig besser geeignet sind. Sie können aber bei kleinen oder Teilproblemen eingesetzt werden, und sind auch in der Lage, gültige (engl.: *feasible*) Parametersätze zu finden: Im Gegensatz zu gradientenbasierten Verfahren sind die Zwischenergebnisse der einzelnen Iterationsschritte bei globalen Verfahren voneinander unabhängig und führen nicht zum Abbruch des Verfahrens, wenn das Differentialgleichungssystem für einen Anfangswertvektor nicht lösbar ist[20].

Während der Entstehung dieser Arbeit wurden von den globalen Verfahren nur solche aus dem NLopt-Paket eingesetzt. Probeweise Anwendung fanden verschiedene Varianten des DIRECT-Verfahrens (*DIviding RECTangle*) [26, 38] und das CRS2-Verfahren (*Controlled Random Search 2*) [64].

Prinzipiell unterteilen DIRECT-Verfahren den Parameterraum in Hyperrechtecke und untersuchen deren Mittelpunkte, das heißt, ein Gütefunktional wird jeweils mit den entsprechenden Parameterwerten ausgewertet. Sind alle Hyperrechtecke ausgewertet, werden diese sukzessive weiter in kleinere Hyperrechtecke unterteilt. Wie oben angedeutet, wächst der Rechenaufwand exponentiell mit der Dimension (beziehungsweise der Parameteranzahl): Die Anzahl der zu untersuchenden Hyperrechtecke steigt schnell an. Außerdem ist zu beachten, dass der durchschnittliche Abstand zwischen benachbarten Gitterpunkten zusätzlich mit der Dimension wächst. Für eine vergleichbare äquidistante Abtastung müssen bei höherdimensionalen Problemen die Kantenlängen verringert werden. Der Aufwand zur Lösung des Optimierungsproblems steigt daher sogar schneller an als „nur“ mit der Potenz der Dimension. Die erwähnte Unabhängigkeit der einzelnen Iterationen eröffnet aber die Möglichkeit, den gesamten Parameterraum zu unterteilen und die entstehenden Teilprobleme auf verschiedene Rechner zu verteilen.

[19] Allerdings sind viele kryptologischen Probleme nur diskreter Natur, das heißt, es gibt eine große, aber endliche Menge von Möglichkeiten, während das Auffinden eines optimalen reellen Parameterwerts unendlich viele Möglichkeiten bietet.

[20] Einige (gradientenbasierte) Verfahren verfügen aber über gesonderte Algorithmen, um immerhin einen gültigen Anfangswertvektor suchen zu können, falls dies erforderlich ist.

Die „L“-Variante von DIRECT bevorzugt die lokale Optimierung, so dass zunächst jene Hyperrechtecke weiter unterteilt werden, die bessere Ergebnisse erzielen konnten. Die Variante arbeitet weiterhin global.

In der Praxis führten beide Varianten bei den in dieser Arbeit untersuchten Modellen innerhalb einer akzeptablen Rechenzeit (max. drei Tage) zu keinem annähernd optimalen Ergebnis. Sie wurden daher meist nur für die Suche nach gültigen Anfangswerten genutzt, aber auch in diesem Fall erwiesen sich andere Verfahren als besser geeignet.

Das CRS2-Verfahren ähnelt prinzipiell dem Downhill-Simplex-Verfahren[21] von Nelder[22] und Mead[23] [62], ist aber im Gegensatz zu letzterem ein Schwarmverfahren[24]. Es werden verstärkt zufällige Einflüsse verwendet und auch deutlich komplexere Heuristiken für die Festlegung der jeweils neuen Punkte definiert. Das CRS2-Verfahren zählt zu den globalen Algorithmen.

Für spezielle Anwendungsfälle wie die Parameteridentifikation gibt es auch globale Verfahren wie in [4] beschrieben, die sehr schnell sind. Sie basieren auf der Differentiation von Messwerten und einer geeignet zu wählenden Interpolation. Dadurch kann die Aufgabe, ein Differentialgleichungssystem numerisch zu lösen, umgangen werden; sie wird durch eine analytisch lösbare ersetzt. Für dieses Verfahren bestehen allerdings einige Voraussetzungen und Bedingungen, die den möglichen praktischen Einsatz relativieren: Für alle Zustände müssen Messdaten existieren. Diese wiederum müssen in ausreichender Anzahl vorliegen und sollten möglichst wenig verrauscht sein, da dies ansonsten zu einer schlechten Approximation durch die Interpolation führt. Je schlechter die Approximation ist, desto weniger stimmt die Lösung mit der des ursprünglichen Problems überein. In der Laborpraxis kann insbesondere die Forderung nach der Messbarkeit aller Zustände nur in wenigen Fällen erfüllt werden. Im Rahmen der vorliegenden Arbeit wurde dieses spezielle Verfahren daher nicht eingesetzt.

[21] Nicht zu verwechseln mit dem Simplex-Algorithmus zum Lösen linearer Gleichungssysteme.

[22] John Ashworth Nelder (1924 – 2010, Statistiker, England)

[23] Roger Mead (*1938, Statistiker, England)

[24] Schwarmverfahren basieren auf der gleichzeitigen Optimumssuche von (vielen) verschiedenen Ausgangspunkten aus. Auch das Simplex-Verfahren mit seinen wenigen Punkten könnte hierzu gezählt werden, jedoch sind bei diesem die Lage der einzelnen Punkte zueinander sehr stark reglementiert.

3.5.2 Lokale Optimierungsverfahren

Lokale Optimierungsverfahren verwenden meist den Gradienten der Gütefunktion oder eine Approximation desselben, um ein lokales Minimum zu finden. Sie sind prinzipbedingt beschränkt auf eine Lösung, die sich je nach Verfahren, mehr oder weniger im „Umfeld“ des Startwertes befindet. Schwarmverfahren oder Methoden mit großen stochastischen Einflüssen verwenden nur eine sehr grobe Approximation des echten Gradienten. Wie im Falle des *CRS2* kann dies sogar den Übergang von einem lokalen zu einem globalen Verfahren bedeuten[25]. Zwei häufig verwendete Verfahren werden kurz skizziert, das (reine) Gradientenverfahren und das Newton-Verfahren[26], nachdem zunächst auf geeignet gewählte Startwerte eingegangen wird.

Startwerte

Die meisten Optimierungsverfahren benötigen, mit Ausnahme weniger, meist globaler Verfahren[27], Startwerte. Gradientenbasierte Verfahren finden in der Regel sehr präzise und effizient das nächstliegende Optimum. Insbesondere bei komplexen Gütegebirgen ist dies aber nur zufällig das gesuchte, globale Optimum. Die Wahl passender Startwerte ist für das Optimierungsergebnis bei der Verwendung von Gradienten entscheidend, denn in vielen Fällen besitzen beispielsweise die Gütefunktionen der Parameteridentifikation sehr viele lokale Minima. Ohne Kenntnis geeigneter Startwerte kann daher die Anwendung gradientenfreier Verfahren, siehe unten, günstiger sein.

Gradientenbasierte Verfahren

Verwendet ein Verfahren abgesehen von einem skalaren Schrittweitenfaktor ausschließlich den aktuellen numerisch oder analytisch ermittelten Gradienten $\mathrm{grad}(\Phi)$ der Gütefunktion Φ, so spricht man vom Gradienten-Verfahren oder der Methode des steilsten Abstiegs.

---

[25] Die meisten „Hybrid“-Optimierer können jedoch nicht garantieren, globale Optima zu finden.

[26] Sir Isaac Newton (1643 – 1726, Naturforscher und Philosoph, England)

[27] Genaugenommen benötigen auch globale Verfahren Startwerte, sie werden allerdings durch die Angabe der zwingend notwendigen Intervallgrenzen häufig durch das Verfahren selbst ermittelt. Ähnliches gilt für Schwarmverfahren, bei denen die ganze Gruppen von Startwerten durch zufällige Prozesse generiert werden.

Mit

$$\mathrm{grad}(\Phi)_k = \left(\begin{array}{c} \frac{\partial \Phi}{\partial \theta_1} \\ \vdots \\ \frac{\partial \Phi}{\partial \theta_n} \end{array} \right) \Bigg|_k \tag{3.31}$$

ergibt sich für die Optimierungsparameter $\underline{\theta}$ ausgehend von einem Startpunkt $\underline{\theta}^{k=0}$ für den nächsten Iterationsschritt $k+1$ der nächste Punkt in Richtung des negativen Gradienten:

$$\underline{\theta}_{k+1} = \underline{\theta}_k - \lambda_{k+1} \cdot \mathrm{grad}(\Phi)_k. \tag{3.32}$$

λ_{k+1} stellt einen skalaren Faktor zur Kontrolle der Schrittweite dar. Abhängig vom verwendeten Algorithmus kann er in jedem Iterationsschritt neu festgelegt werden. Er dient dazu, den Einfluss des Gradienten, insbesondere bei häufigen Richtungswechseln, zu dämpfen. Der Parametervektor sollte durch eine passende Normierung einheitenlos sein. Eine Normierung ist auch deshalb empfehlenswert, um alle Parameter in der Optimierung gleich zu gewichten.

Während der negative Gradient zwar die Richtung des steilsten Abstiegs wiedergibt, muss dies nicht der direkte Weg zum Minimum sein. Das Newton-Verfahren verwendet daher nicht nur den Gradienten, sondern zusätzlich die lokale Krümmung des Gütegebirges. Bei Problemen, bei denen die Optimierungsparameter direkt quadratisch in die Gütefunktion eingehen, ist dieses Vorgehen optimal, da mit einem Iterationsschritt das Minimum erreicht werden kann.

Bei den vorliegenden Problemen gehen die Parameter allerdings in komplizierterer, nicht expliziter Form in die Funktion ein. Wird die Gütefunktion nun mittels einer Taylor-Reihenentwickung[28] durch eine quadratische Funktion approximiert, kann daraus das Minimum als Näherungslösung für das ursprüngliche Optimierungsproblem bestimmt werden, wobei die Approximation nur lokal gültig ist und iterativ wiederholt werden muss. Daher nennt sich dieses Verfahren auch Sequential Quadratic Programming (SQP) [72].

Im Ergebnis wird bei diesem Verfahren anstelle des obigen Skalars λ_{k+1} die inverse Hesse-Matrix[29] $\mathbf{H}$ von $\Phi(\underline{\theta})$ eingesetzt[30]. Der Term $\mathbf{H}^{-1} \cdot \mathrm{grad}\,\Phi_k$ wird als konjugierter Gradient bezeichnet.

[28] Brook Taylor (1685 – 1731, Mathematiker, England)

[29] Otto Hesse (1811 – 1874, Mathematiker, Deutschland)

[30] Zusätzlich muss $\mathbf{H}$ positiv definit sein, damit der angegebene Iterationsschritt durchgeführt wird.

Da nur das approximierte Problem wiederholt gelöst wird, lässt sich die Gesamteffizienz des Newton-Verfahrens noch verbessern. Gängig ist hierfür beispielsweise ein sogenannter Line-Search als Zwischenschritt: Durch den konjugierten Gradienten wird die Suchrichtung festgelegt, wodurch sich die Aufgabe auf ein eindimensionales Optimierungsproblem reduziert. Die optimale Schrittweite kann darin schnell durch einen Line-Search-Algorithmus ermittelt werden.

Meist werden die partiellen Ableitung numerisch mittels kleiner Schritte δ durch einen Differenzenquotienten approximiert, zum Beispiel im eindimensionalen Fall mit

$$\frac{\partial \Phi}{\partial \theta_1} \approx \frac{\Phi(\theta_1 + \delta) - \Phi(\theta_1)}{\delta}. \tag{3.33}$$

Die Genauigkeit des derart numerisch bestimmten Gradienten hängt von der Größe von δ ab. Allerdings begrenzt die Rechengenauigkeit von Computern das beliebige Verkleinern von δ .

Der rechnerische Aufwand ist für Gradienten-Verfahren bei vielen Optimierungsparametern hoch: Die Bestimmung des Gradienten kostet in jeder Iteration zusätzlich zur nominellen Simulation eine Simulation pro Optimierungsparameter. Der Aufwand zur Bestimmung der Hesse-Matrix beim Newton-Verfahren ist entsprechend höher, weswegen bei SQP-Verfahren direkt die inverse Hesse-Matrix approximiert wird.

Abgesehen von den Optimierungsverfahren, die Matlab zur Verfügung stellt, werden in dieser Arbeit weitere Optimierer verwendet. Die im Folgenden genannten stammen aus einer Erweiterung für Matlab; dem „Optimization Environment“-Paket von Tomlab. Nach gewonnenen Erfahrungen sind dessen Optimierer etwas leistungsstärker[31]; sie finden ein „besseres Optimum“ in kürzerer Zeit. Nach den Angaben des Herstellers [78] sollen die Algorithmen tatsächlich vielseitiger, robuster und schneller als die in Matlab sein.

Verwendet wurden in dieser Arbeit dabei *SNOPT7*, *NPsol* und *conSolve*. Diesen drei Optimierungsalgorithmen ist gemein, dass sie, wie auch der *fmincon*-Befehl von Matlab, auf dem oben erwähnten SQP-Algorithmus mit einem Line-Search-Schritt beruhen.

Wird ein SQP-Algorithmus mit guten Anfangswerten gestartet, konvergiert das Verfahren in der Regel nach einigen Anfangsiterationen recht schnell. Die merklichen Verzögerungen am Anfang werden unter anderem durch die Berechnung der Ableitung und Approximation der Krümmung des Gütefunktionals verursacht. Derartige Verzögerun-

[31] Diese Erfahrungen beruhen auf am Fachgebiet durchgeführten Vergleichstest mit der damals aktuellen Matlab-Version 2007b

gen treten auch im Verlauf der Optimierung auf, besonders dann, wenn die Startwerte schlecht gewählt waren. So zeigt sich, wie rechenaufwendig die Hilfsfunktionen dieser indirekten Verfahren sind. Die Wahl der Startwerte ist daher wichtig. Im Abschnitt 4.3 wird ein Verfahren vorgestellt, mit dessen Hilfe gute Startwerte ermittelt werden können.

Die Unterschiede zwischen den drei Algorithmen liegen in den konkreten Implementierungen, die nicht veröffentlicht sind. Vom Hersteller wird *conSolve* als Optimierer für allgemeine, nichtlineare Probleme vorgeschlagen, während *SNOPT7* speziell für dünnbesetzte und *NPsol* für dichtbesetzte Matrizen optimiert wurde. Damit ist die Auswahl zumindest theoretisch problemabhängig. In der Praxis zeigte sich allerdings, dass ein Wechsel zwischen diesen drei Verfahren (während der Optimierung) mitunter die besten Ergebnisse brachte.

Gradientenfreie Verfahren

Um den numerischen Aufwand zur Bestimmung eines Gradienten zu umgehen, gibt es viele Verfahren, die ohne eine echte Gradienteninformation arbeiten. Dabei werden häufig heuristische Regeln eingesetzt, die einen Gradienten auf einfache Art und Weise nachbilden, wodurch sie aber weniger effizient arbeiten, als es mit „echten“ Gradienteninformationen möglich wäre. Zusätzlich werden oft auch zufällige Anteile bei der Suche nach dem Optimum eingesetzt. Letzteres ermöglicht es den Verfahren, nicht nur das nächstliegende lokale Optimum zu finden, sondern auch im weiteren Umkreis zu suchen. Da eine aufwendige numerische Bestimmung des Gradienten (unter Umständen auch der zweiten Ableitung) entfällt, ist eine einzelne Optimierungsiteration schnell berechnet. Daher eignen sich diese Verfahren besonders für sogenannte Schwarm-Algorithmen, bei denen viele Optimierungen parallel verlaufen, und Informationen zwischen den einzelnen Optimierungsverläufen ausgetauscht werden.

Das Nelder-Mead-Verfahren beispielsweise, auch bekannt als Downhill-Simplex, vergleicht weit entfernte Gütefunktionalsauswertungen miteinander und bestimmt daraus nach festgelegten Regeln die neue Suchrichtung und Schrittweite. Schwarmverfahren verwenden hingegen viele Punktauswertungen und setzen für die Suche nach dem Optimum den Massenmittelpunkt des gesamten Schwarms ein.

Für die in dieser Arbeit untersuchten Probleme sind verschiedene lokale Optimierer aus dem Nonlinear Optimization (NLopt)-Paket [37] zum Einsatz gekommen. Diese werden im Folgenden kurz erläutert.

- BOBYQA [63]
  Ein Verfahren, welches Gradienten-Informationen nur indirekt im Rahmen von Trust-Region-Methoden verwendet. Es wird eine quadratische Approximation des tatsächlichen Gütefunktionals verwendet[32], die jedoch immer wieder überprüft wird, indem die Voraussagen der Approximation mit den in der nächsten Iteration tatsächlich berechneten Werten der Gütefunktion verglichen werden. Daher eignet es sich besonders für quadratische Problemstellungen.

  Der Algorithmus war in der Praxis bei den bearbeiteten Problemen sehr schnell in der Lage, gute Parameterwerte zu finden, auch wenn die Startwerte sehr schlecht gewählt waren. Allerdings wird er bei der Annäherung an das lokale Minimum langsamer, was typisch für gradientenfreie Verfahren ist.

- Nelder-Mead
  Dieses wohlbekannte, klassische Verfahren basiert auf Vergleichen von Punktauswertungen des Gütegebirges. Seine grundsätzlichen Vorteile liegen in der Einfachheit seiner Implementierung, seiner Robustheit (in den meisten Fällen stören Unstetigkeiten des Gütefunktionals nicht), und in der einfachen grafischen Darstellung seiner Funktionsweise. Es gehört für hochdimensionale Probleme aber zu den eher langsamen Verfahren. Beispielsweise kann sich die Schrittweite schon frühzeitig sehr schnell verringern, obwohl das Verfahren erst nur für einen einzigen Parameter die Nähe eines Optimums erreicht hat.

- Subplex [70]
  Der Subplex-Algorithmus unterteilt das Optimierungsproblem in Unterprobleme, die ihrerseits mit dem Nelder-Mead-Verfahren optimiert werden. Insbesondere bei hoch-dimensionalen Problemstellungen wird dieser Algorithmus damit schneller und in bestimmten Fällen auch robuster sein. Gleichzeitig behält er aber die vorteilhaften Eigenschaften des Nelder-Mead-Verfahrens bezüglich nicht-kontinuierlicher Gütefunktionen. Bei den testweisen Einsätzen im Rahmen der durchgeführten Parameteridentifikationen wurde allerdings kein nennenswerter Vorteil des Subplex-Verfahrens bemerkt. Da andere Methoden fast immer schneller waren, wurde dies nicht ausführlich untersucht.

[32] Für jeden Optimierungsparameter wird eine eigene quadratische Interpolationsfunktion verwendet.

Welcher Optimierertyp und welcher genaue Algorithmus schneller zu einem Ergebnis führt, ist vom jeweiligen Problem abhängig. Für die gradientenbasierten Verfahren gilt, dass bei hochdimensionalen Problemen die Berechnung des Gradienten und besonders der Approximation der inversen Hesse-Matrix aufwendig ist. In der gleichen Zeit kann ein gradientenfreies Verfahren viele Optimierungsschritte vorangekommen sein. Bei der Annäherung an ein Minimum werden im direkten Vergleich jedoch die gebräuchlichsten Verfahren mit Gradientenbestimmung schneller, da sie aufgrund der erwähnten quadratischen Formulierung des Problems mit Hilfe der zweiten Ableitung das Optimum analytisch approximieren. Die Schrittweite (und -richtung) kann dann so optimiert werden, dass nur noch wenige Iterationen notwendig sind. Da im allgemeinen Fall zunächst nichts über die Güte der Startwerte bekannt ist, wird eine ungünstige Optimiererauswahl oft erst während der Optimierung erkannt.

In der Praxis wurden gute Erfahrungen mit dem wechselnden Einsatz verschiedener Verfahren gemacht. Zuerst werden gradientenfreie Verfahren zur schnellen Bestimmung von geeigneten Startparameterwerten eingesetzt. Nebenbei wird gleichzeitig eine schnelle Voroptimierung erzielt. Darauf folgt der Start eines gradientenbasierten Verfahrens, das ausgehend von den bereits gefundenen Werten in kurzer Zeit das nächstliegende Optimum bestimmt. Wenn genügend Zeit zur Verfügung steht, können erneut wechselweise gradientenbasierte und -freie Verfahren benutzt werden. So können unter Umständen noch bessere lokale Minima gefunden werden.

4 Die automatisierte Modellbildung

Any sufficiently advanced technology is indistinguishable from magic.

(Arthur C. Clarke, 1987)

In den vorherigen Kapiteln wurde dargestellt, welche Vorteile sich durch die Verwendung mathematischer Modelle ergeben, wie sie prinzipiell aufgebaut werden können, und wie sie beispielsweise im Rahmen einer Parameteridentifikation eingesetzt werden. Ebenso wurde das bestehende ABC-System erläutert.

Dieses Kapitel konzentriert sich auf die Modellerzeugung und deren Automatisierung innerhalb des Rahmens des ABC-Systems. Zunächst wird der klassische Weg der Modellierung im Detail beschrieben. Dabei wird auf ein einzelnes zu erzeugendes Modell eingegangen. Anschließend werden die Methodiken des *Modellstrukturgenerators* vorgestellt, mit denen die Automatisierung der Modellierung umgesetzt wurde.

4.1 Der „RapidOptimization“-Ansatz

Der Vorgang der Modellkonstruktion umfasst grundsätzlich die Auswahl der zu modellierenden Stoffe und der Reaktionen, die zum Auf- und Abbau dieser Stoffe im Bilanzraum (dem Fermenter) führen. Im Falle eines unstrukturierten Modells sind dies beispielsweise die Substrate, ein Produkt und die Biomasse (1). Des Weiteren müssen während der Modellbildung die Reaktionen zwischen diesen Stoffen festgelegt werden (2), die Einflussfaktoren auf die Reaktionsgeschwindigkeiten bestimmt werden (3) und die formale Beschreibung der dynamischen Einflüsse mit den sogenannten Formalkinetiken ausgewählt werden (4). Im Anschluss daran müssen für die Modellparameter passende Startwerte gefunden werden (5) und als letztes sind alle vorangegangen Ergebnisse in Form eines Computerprogramms niederzuschreiben (6). Im Folgenden werden diese sechs Schritte näher erläutert, dabei wird insbesondere auf die in dieser Arbeit angestrebte Automatisierung eingegangen:

1. Die Auswahl der zu modellierenden Stoffe ergibt sich aus der Aufgabenstellung und den im Prozess eingesetzten Stoffen. Die Entscheidung kann nicht prinzipiell automatisiert werden, aber es gibt Verfahren zur automatisierten Messdatenanalyse, die bei der Auswahl helfen können, zum Beispiel in [32].

2. Die Anzahl und Definition der Reaktionen sind wichtige Modell-Entscheidungen. Sie werden im Prinzip willkürlich getroffen, wenn nicht bereits brauchbare, ähnliche Modelle vorhanden sind. Die tatsächlichen Reaktionen sind, soweit überhaupt bekannt, für die Modellierung auf dem hier eingesetzten Abstraktionsniveau meist nicht verwendbar. Dennoch können sie Hinweise für eine sinnvolle Auswahl geben.

   Auch dieser Schritt wird unten nicht durch den Modellstrukturgenerator automatisiert, allerdings können durch die Automatisierung viele Varianten von Reaktionsnetzwerken überprüft werden, so dass eine Festlegung auf ein einziges Reaktionsnetzwerk nicht notwendig sein wird.

3. Die Geschwindigkeiten vieler Reaktionen werden durch die Konzentrationen ihrer Edukte und Katalysatoren reguliert: Nur wenn alle Edukte und Katalysatoren vorhanden sind, läuft die jeweilige Reaktion ab. Je nach Reaktion sind daher eventuell mehrere Einflussgrößen zu berücksichtigen. Um das Modell möglichst einfach zu halten, ist es mitunter möglich, eine bestimmende Größe (*bottleneck*) zu erkennen und nur diese im Modell zu verwenden. Aber nicht immer ist dies zielführend, da je nach Situation verschiedene Stoffe für die Reaktionsgeschwindigkeit entscheidend sein können.

   Das Festlegen der regulierenden Größen erfolgt am besten mit Hilfe von biologischen Vorwissen: Vergleichsweise einfache Zusammenhänge sind bei Reaktionen wie dem Wachstum zu erwarten. Die Wachstumsrate ist primär von den Substratkonzentrationen abhängig. Darüber hinaus sind inhibierende Einflüsse möglich, die beispielsweise das Wachstum bei einer zu hohen Substratkonzentration einschränken. Nur wenn die entsprechenden Bedingungen in den jeweiligen Prozessen auftreten können, ist es sinnvoll, die Inhibierung in das Modell aufzunehmen.

   Insbesondere für Reaktionen zur Bildung von Sekundärmetabolit-Produkten kann die Festlegung der Einflussgrößen deutlich schwieriger sein, da die Reaktionsrate oft nicht von den Eduktkonzentrationen abhängig ist. Viele Reaktionen laufen katalytisch ab, das heißt, ein ansonsten nicht an der Reaktion beteiligter Stoff

bestimmt die Reaktionsgeschwindigkeit[1]. Vorwissen ist hier nützlich, aber häufig fehlen für abstrakte Modelle die Information über die Regulationsmechanismen.

Auch für diesen Schritt gilt, dass es aktuell keine Verfahren gibt, die die Auswahl automatisieren können. Ein Festlegen auf eine einzige Konfiguration ist jedoch nicht notwendig, wie schon im zweiten Schritt dargelegt wurde.

4. Das nächste Element der Modellerzeugung ist die Bestimmung der mathematischen Gleichungen für die Formalkinetiken. Sie definieren die Art des Einflusses der regulierenden Größen auf die Reaktionsgeschwindigkeit. Oft ergibt sich der Typ des Einflusses, beispielsweise limitierend oder inhibierend, aus der Reaktion selbst.

   Die Entscheidung über die genaue Formalkinetik erfolgte bisher für abstrakte Modelle in der Regel ebenfalls erfahrungsgetrieben mit der Auswahl eines typischen Ansatzes pro Reaktion. Nur durch aufwendiges Ausprobieren hätten verschiedene Kombinationen alternativer Ansätze miteinander verglichen werden können, was erst nach Abschluss des Modellbildungsverfahrens möglich wäre. Es ist zu vermuten, dass in vielen Modellierungsprojekten aus Zeitgründen nur sehr wenige Modellkombinationen ausgetestet werden.

   An diesem Punkt der Modellierung setzt die starke Unterstützung durch die Automatisierung an. Mit Hilfe des Modellstrukturgenerators legt der Benutzer – oder auch ein weiteres Programm – nur den Typ der Kinetik fest, der an die jeweilige Stelle des Modells einzubauen ist, zum Beispiel inhibierend oder limitierend, und alle Kombinationen der möglichen Varianten werden eigenständig in Form der Modellkandidaten erzeugt. Dabei ist das Programm bei der Eingabe des Typs sehr flexibel: Selbst die Festlegung auf einen einzigen Typ ist nicht notwendig, und umgekehrt kann die Auswahl auf bestimmte Formalkinetiken eingeschränkt werden, wenn dies gewünscht wird.

5. Der theoretische Teil der Modellbildung wird abgeschlossen, indem allen Modellparametern ein erster Wert zugewiesen wird. Dem Modellstrukturgenerator können zu diesem Zweck Startwerte übergeben werden. Fehlen diese, können sie mit Hilfe zweier unterschiedlicher Verfahren, die in den Abschnitten 4.3 und 5.2.1 vorgestellt

[1] Darunter fallen auch scheinbar katalytische Reaktionen: Reaktionen mit Stoffen, deren Verbrauch unterhalb der Nachweisgrenze üblicher Messmethoden liegt.

werden, generiert werden. Die Optimierung und endgültige Festlegung der Parameterwerte erfolgt anschließend wie üblich im Rahmen einer Parameteridentifikation.

6. Alle vorangegangenen Entscheidungen müssen in Form mathematischer Gleichungen formuliert und als ausführbares Programm kodiert werden, damit ein „Simulationsprogramm“ entsteht. Durch die automatisierte Erstellung von Programmcode lässt sich der aufwendige und fehleranfällige Prozess des manuellen Programmierens vermeiden.

Zusammenfassend stehen einem menschlichen Modellkonstrukteur und dem Modellstrukturgenerator viele Freiheitsgrade bei der Erschaffung eines Modells zur Verfügung. Allerdings erfolgen diese Entscheidungen bei einer rein manuellen Modellbildung oft willkürlich und werden selten revidiert, da jede Änderung eines Modells erneut einen signifikanten Aufwand bei der Programmierung bedeutet. Durch eine Automatisierung der Modellprogrammierung wird die dargestellte Situation deutlich verbessert. Das Problem für den Modellkonstrukteur verschärft sich noch, da oft erst nach einer Parameteridentifikation (oder sogar noch später) entschieden werden kann, ob ein Modell seinen Zweck erfüllt. Häufig wird beispielsweise die Michaelis-Menten-Funktion verwendet, doch gibt es keinen Grund, warum in Modellen der hier diskutierten abstrakten Form diese Kinetik besser geeignet sein sollte als andere mathematische Funktionen. Tatsächlich beschreibt die Michaelis-Menten-Funktion eine enzymkontrollierte Einzelreaktion. Hinter den Reaktionen abstrakter Modelle verbergen sich jedoch Hunderte von Einzelreaktionen. In ihrer Summe kann sich leicht eine gänzlich andere Form für eine zusammenfassende Kinetik ergeben, auch Funktionen mit mehr als einem Minimum und Maximum sind denkbar. Für die abstrakten Modelle enthält die im Zusammenhang mit dem Modellstrukturgenerator entwickelte Kinetikbibliothek (Abschnitt 4.4) daher auch viele „komplexe“ Funktionen.

Mit dem im Rahmen dieser Arbeit entwickelten Modellstrukturgenerator werden drei hauptsächliche Ziele erreicht:

1. Erstellung vieler verschiedener Modellvarianten vermeidet willkürliche Auswahl
2. Wegfall von notwendigen Programmierkenntnissen durch automatische Programmierung
3. Automatische Programmierung ermöglicht eine automatische Weiterverarbeitung der Modelle

Neue Erkenntnisse, die im Laufe der praktischen Arbeit mit einem Stamm von Mikroorganismen üblicherweise gewonnen werden, können einen großen Einfluss auf die

mathematischen Modelle haben. Dies kann dazu führen, dass der Prozess der Modellbildung wiederholt erneut begonnen werden muss. Insbesondere ist dies bei Mikroorganismen der Fall, bei denen noch sehr wenig über die Anzuchtbedingungen bekannt ist. Aber auch später können neue Informationen genutzt werden, um ein Modell zu verbessern. Durch die Automatisierung der Modellbildung sinkt die „Hemmschwelle“, Modelle je nach Kenntnisstand zu generieren. Somit können zu jedem Zeitpunkt Modelle bereitstehen, die verwendet werden können, um beispielsweise informative Experimente zu planen oder eine Prozessoptimierung durchzuführen. Insbesondere die alternierende Modell- und Prozessoptimierung kann durch den Einsatz und Vergleich vieler Modellkandidaten beschleunigt werden, siehe [82, 83].

In den nächsten Abschnitten werden die Elemente, die ein Modell im ABC-System ausmachen, aufgeführt und die Mechanismen des Modellstrukturgenerators zu deren Erzeugung erläutert. Die wichtigsten davon sind:

- Zustandsgrößen
- Reaktionsgleichungen
- P-Faktoren
- Reaktionskinetiken
- Messgleichungen
- Stellgrößen
- Parameter
- Probennahmen

Aus diesen Bestandteilen lässt sich ein allgemeines, vollständiges Modell, in der Art wie es in Kapitel 3 vorgestellt wurde, zusammenfügen. Der Modellstrukturgenerator stellt die einzelnen Teile automatisch zusammen und erzeugt die notwendigen Modell-Dateien für das ABC-System. Die Programmierung wurde in Matlab (2007b) durchgeführt.

4.2 Der Modellstrukturgenerator

Wie bereits für die Bilanzgleichungen aus Abschnitt 3.1 gezeigt wurde, können die einzelnen Gleichungen nach einem standardisierten Schema aufgestellt werden. Im Folgenden wird gezeigt, wie dies auch – ausgehend von Reaktionsgleichungen – für das Aufstellen des gesamten Differentialgleichungssystems gilt. Dieser Formalismus dient als Basis für das automatische Erzeugen und letztlich auch zur automatischen Programmierung von

Modellen. Neben den Reaktionen sind weitere Informationen zum Erzeugen eines fertigen Modellordners im ABC-System notwendig. Auch diese lassen sich vereinheitlichen und kompakt formulieren. Die verschiedenen Informationen, zusammen in dieser Arbeit als *Modellstrukturdefinition* bezeichnet, werden dazu in einem abstrakten Variablentyp in Matlab gespeichert; einer Strukturvariablen. Im Folgenden wird die Variable selbst mit `StrDef` bezeichnet. So erfolgt der Aufruf des Modellstrukturgenerators in Matlab einfach mit: `Modellstrukturgenerator(StrDef)`

Der Aufbau von `StrDef` ist zur Übersicht in Tabelle 4.1 dargestellt. Der jeweilige Datentyp der einzelnen Felder ist in Klammern neben dem Feldnamen angegeben. So bezeichnet *string* eine Zeichenkette, *struct* ist ein Strukturvariable, in welchem verschiedene Variablentypen zusammengefasst werden, *double* steht für einen skalaren Wert (mit doppelter Genauigkeit) und mit *bool* sind boolesche[2] Variablen gemeint (`wahr/falsch`). Keine eigenen Datentypen sind *matrix* und *vector*, mit ihrer Hilfe wird die Struktur der Daten verdeutlicht. Die eigentlichen Datentypen der betreffenden Feld-Einträge sind in der jeweiligen Beschreibung aufgeführt, siehe folgende Abschnitte.

Durch einfache Erweiterungen der Einträge können sogar Gruppen von Modellfamilien (in jeweils einem eigenen Modellordner) erzeugt werden. Dies bietet sich insbesondere für nur geringfügige Struktur-Variationen an. Aus `double`- oder `bool`-Elementen werden dann Vektoren, aus Vektoren Matrizen und so weiter. Die Definition der Zustands- und Stellgrößen bleibt konstant für alle derart erzeugten Modellfamilien.

Weil der Inhalt der `StrDef`-Variablen nicht nur von Hand, sondern inzwischen auch von Programmen des ABC-Systems automatisch erstellt werden kann, stellen die erfassten Daten nur die gemeinsamen Basisinformationen dar. Fehlende Informationen werden vom Benutzer während der Laufzeit des Modellstrukturgenerators über grafische Benutzeroberflächen (Graphical User Interfaces (GUIs)) abgefragt. Dazu gehören zum Beispiel Entscheidungen bezüglich der Anwendung von Korrekturfluiden (für die Kontrolle des pH-Werts) oder Antischaummittel, P-Faktoren oder X_0-Parameter.

In Ergänzung zur Abbildung 2.2 des zweiten Kapitels auf Seite 18 wird in Abbildung 4.1 schematisch gezeigt, wie die möglichen Arbeitsabläufe aussehen, um einen Modellordner zu erstellen: Ausgehend von Experimentaldaten, die in der MySQL-Datenbank gespeichert werden, führen aktuell drei Wege zu einer Modellstrukturdefinition (MSD). Der Modellstrukturgenerator erstellt daraus den Modellordner mit den grundlegenden

[2] benannt nach George Boole (1815-1864, Mathematiker und Philosoph, England)

`StrDef. (struct)`	`Modellstrukturdefinition`
`|- .Name (string)`	enthält den Namen des Modells
`|- .Info (string)`	enthält allgemeine Modellinformationen
`|- .Zustand (struct)`	Informationen über Zustände
`| |- .Name (string)`	Name des Zustands
`| |- .Einheit (string)`	Physikalische Einheit dieses Zustands
`| |- .Typ (string)`	Zustandstyp
`| |- .Feed (bool)`	wird gefeedet (Ja/Nein)
`| |- .gemessen (bool)`	wird gemessen (Ja/Nein)
`| |- .Zerfall (string)`	Formel für Zerfallsreaktion
`| |- .x0 (double)`	Anfangswert
`|- .rName (vector(string))`	Namen der einzelnen Reaktionen
`|- .K (matrix)`	Ausbeutekoeffizienten der Reaktionen
`|- .R (matrix)`	Formalkinetiken der Reaktionen
`|- .P (matrix)`	P-Faktoren der Reaktionen
`|- .PMod (vector(double))`	Modellkandidatenwahrscheinlichkeit
`|- .Massendefekt (vector(bool))`	Massenerhaltung der Reaktionen (Ja/Nein)
`|- .pH (bool)`	pH-Wert als Stellgröße aufnehmen (Ja/Nein)
`|- .Lagphase (bool)`	Lag-Phase vorhanden (Ja/Nein)
`|- .Erhaltung (bool)`	Zellerhaltungsverbrauch (Ja/Nein)
`|- .Diauxie (bool)`	Stoffwechsel-Phänomen (Ja/Nein)
`|- .Startwerte (struct)`	Informationen über Startwerte
`| |- .muMax (double)`	Erwartete, maximale Wachstumsrate
`| |- .muMaxMin (double)`	Intervallgrenze für Wachstumsrate
`| |- .muMaxMax (double)`	Intervallgrenze für Wachstumsrate
`| |- .KMin (matrix)`	Intervallgrenze für Ausbeutekoeffizienten
`| |- .KMax (matrix)`	Intervallgrenze für Ausbeutekoeffizienten
`| |- .Erhaltung (double)`	Erwarteter Verbrauch für Zellerhaltung
`| |- .Lagphase (vector(double))`	Erwartete Dauer der Lag-Phase
`| |- .LagphaseMin (vector(double))`	Intervallgrenze für Lag-Phase
`| |- .LagphaseMax (vector(double))`	Intervallgrenze für Lag-Phase
`|- .Messdaten (struct)`	Kopie von Messdaten

Tabelle 4.1: Aufbau der Struktur-Variablen mit der Modellstrukturdefinition

Informationen. Ein automatischer Aufruf von `ABC_Build` erzeugt zum Abschluss die kompilierten Daten (nicht dargestellt) im `auto`-Ordner.

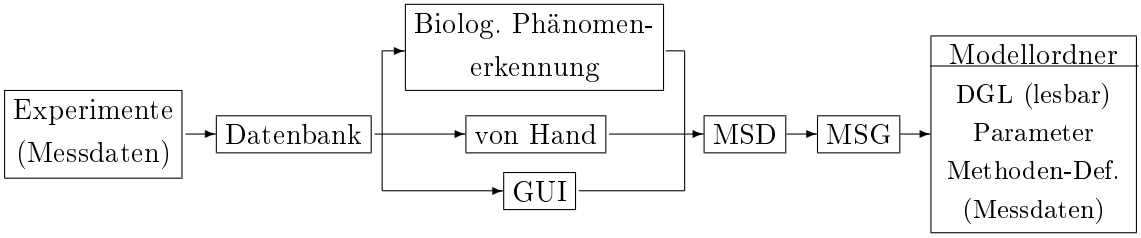


Abbildung 4.1: Erzeugungswege für Modellordner

Im Anschluss werden die einzelnen Felder der Strukturvariablen beschrieben, die zur automatischen Modellerzeugung benutzt werden. Darauf folgen einige Abschnitte, die die automatische Programmierung weiter erläutern.

4.2.1 Modellbezeichnung und -information

Der Name des Modells wird in `.Name` abgelegt. Bei der automatischen Erzeugung der Modelldateien wird dieser Name für den Modellordner innerhalb des ABC-Systems verwendet. Sind in der Variablen `StrDef` Informationen für mehrere Modellfamilien hinterlegt, so werden die Namen durch eine zusätzliche Nummerierung automatisch variiert und dem Benutzer vorgeschlagen. Zusätzliche Hinweise zu dem konkreten Modell wie beispielsweise die Herkunft der Modellstruktur können in `.Info` abgelegt werden. Abgesehen von der Dokumentation der Modellerzeugung können solche Informationen bei einer späteren Suche in der Modellbibliothek nützlich sein.

4.2.2 Zustandsgrößen

Die für alle Modellkandidaten gemeinsam definierten Zustände werden in einer Unterstruktur (`.Zustand.`) zusammengefasst, daher ist dieses Feld seinerseits auch ein `Struct`, dessen Länge der Anzahl der verwendeten Zustandsgrößen entspricht. Für jeden Zustand werden darin die folgenden Informationen abgelegt:

Das Unterfeld `.Name` legt den Namen der jeweiligen Zustandsgröße fest, in `.Einheit` wird die physikalische Einheit abgelegt. Sie wird einerseits für die Beschriftung bei grafischen Ausgaben verwendet und andererseits benutzen einige automatische Erkennungs-

routinen, nicht nur im Modellstrukturgenerator, diese Angaben als zusätzliche Informationsquelle.

Der `.Typ` eines Zustands, siehe Abschnitt 2.2, beinhaltet wesentliche Informationen darüber, welcher Art der jeweilige Zustand ist. Bei der automatischen Modellerzeugung werden diese Informationen ausgewertet, um die Differentialgleichungen passend aus den einzelnen Termen zusammenzusetzen. Die derzeit implementierten Typen sind in den Tabellen 2.1 und 2.2 ab Seite 25 aufgeführt.

Für jede Zustandsgröße wird mittels der booleschen Variable `.Feed` festgelegt, ob für das Modell die betreffende Substanz prinzipiell als Feed definiert werden soll. Dies betrifft in der Regel die Substrate. Die jeweiligen Differentialgleichungen werden in diesem Fall um einen Term für die externe Zufuhr dieses Substrats i ergänzt:

$$\dot{x}_i = (...) + u_{\text{feed,i}} \cdot c_{\text{feed,i}} \tag{4.1}$$

Hier ist $c_{\text{feed,i}}$ die Stoffkonzentration des Feeds und $u_{\text{feed,i}}$ die Feedrate. Die Einheiten werden in `.Einheit` in Form von Zeichenketten festgelegt und sind typischerweise $\mathrm{g\,l^{-1}}$ und $\mathrm{l\,h^{-1}}$. Das Produkt der beiden Größen ergibt einen Massenstrom. Sowohl die Feedkonzentration als auch die Feedrate werden automatisch in die Liste der Stellgrößen aufgenommen.

Die Software des Modellstrukturgenerators wurde speziell für Fermentationen mit chemisch-definierten Minimalmedien ausgelegt. Als Konsequenz wurden für die in dieser Arbeit betrachtete Problemklasse drei Feedings für jeweils eine C-, N- und P-Quelle als Modellbausteine fertig programmiert. Sie werden prinzipiell durch die Informationen in `.Feed` für jeden einzelnen Zustand aktiviert. Andere Situationen, mit mehr oder weniger als drei Feeds, werden erkannt. Die automatische Verarbeitung im Modell ist für einige Fälle bereits vollständig implementiert, eine Erweiterung auf alle Fälle ist ohne Probleme möglich.

Die Stellgrößen eines Modells werden in der Datei `SystemInit` definiert. Dort werden der Name, die Einheit und der Typ der jeweiligen Stellgröße festgelegt. Hinzu kommen Informationen über maximale und minimale Werte sowie die `.uOrder`, die die Art der Interpolation für die Stellgrößenverläufe einer Modellfamilie angibt, siehe Abschnitt 2.5. Aus den Informationen in der Modellstrukturdefinition erstellt der Modellstrukturgenerator die benötigten Einträge automatisch.

Da sich durch eine Substratzufuhr das Flüssigvolumen im Fermenter verändert, wird die Gleichung für das Volumen entsprechend automatisch ergänzt:

$$\dot{V} = (...) + u_{\text{feed,i}} \tag{4.2}$$

Der Inhalt der booleschen Variablen `.gemessen` gibt Auskunft darüber, ob die jeweilige Substanz automatisch auch als Messgröße ins Modell aufgenommen werden soll. Weitere Details zu den Messgrößen werden unten in einem eigenen Abschnitt aufgeführt.

Durch einen Eintrag in `.Zerfall` wird ähnlich wie durch `.Feed` ein zusätzlicher Term in die Differentialgleichung des betreffenden Zustands eingebaut, allerdings mit einem negativen Vorzeichen. Dieser Baustein ist hilfreich, wenn ein Zerfall oder Abbau des betreffenden Zustands nicht selbst als Reaktion modelliert werden soll[3]. Anstelle einer zusätzlichen Reaktion kann die Differentialgleichung des instabilen Endprodukts mit einem Zerfallsterm versehen werden, um das Modell zu vereinfachen und seine Identifizierbarkeit zu erhöhen. Durch den fertigen Baustein wird ein Modellparameter für die maximale Zerfallsgeschwindigkeit hinzugefügt und diese mit der Masse der Substanz x_i selbst multipliziert:

$$\dot{x}_i = (...) - \mu_{\text{i z,max}} \cdot x_i \tag{4.3}$$

Da die Zerfallsreaktion nicht nur eine Reaktion nullter Ordnung sein kann, sondern oft eine Abhängigkeit von verschiedenen Größen, wie beispielsweise Konzentration oder Temperatur, aufweist, ist der Modellstrukturgenerator soweit vorbereitet, dass in `.Zerfall` auch eine Zeichenkette mit einem mathematischen Ausdruck z in Matlab-Syntax übergeben werden kann. Dieser Ausdruck soll später auch Formalkinetiken aus der Kinetikbibliothek enthalten dürfen. In die betreffende Differentialgleichung wird dann später der Ausdruck z automatisch als zusätzlicher Multiplikator im Zerfallsterm eingebaut. Hilfreich ist es auch, wenn eine Bibliothek typischer Zerfallsreaktionen als auswählbare Mustervorlagen nach dem Vorbild der Kinetikbibliothek integriert würde.

Zuletzt gibt `.x0` einen Anfangswert für den jeweiligen Zustand an. Die Werte aller Zustände ergeben zusammen den Anfangswertvektor zum Lösen des Differentialgleichungssystems. Der Benutzer kann realistische Werte bei der Modellerzeugung hinterlegen, damit das Modell anschließend sofort für einen schnellen Test lauffähig ist. Eine spätere Änderung ist manuell jederzeit möglich: Die Eintragung erfolgt in der `SystemInit`, siehe Abschnitt 2.1.

[3] Das trifft auch auf Produkte zu, die z. B. schnell verdampfen und den Bilanzraum verlassen.

Am Ende der Liste mit allen betrachteten Stoffen müssen zusätzlich ein Zustand vom xTyp `KAX` (Kompartiment: Aktive Biomasse) und ein Zustand vom xTyp `X` (Biogesamtmasse) definiert werden. So können trotz der verschiedenen Modelltypen dieselben Algorithmen zur Modellerstellung verwendet werden. In Abschnitt 4.2.7 wird dies genauer erläutert. Wenn `StrDef` von Hand erstellt wird, sind die zwei Zustände manuell hinzuzufügen, die vorhandenen Programme legen sie automatisch an.

Die Anfangswerte für die einzelnen durchgeführten Experimente werden innerhalb des ABC-Systems in der Datenbank und in der Messdatenstruktur (siehe Seite 26) zu jedem Versuch im Messdatenordner gespeichert.

4.2.3 Reaktionen

Im Feld `.rName` werden die Namen für die einzelnen Reaktionen festgelegt. Obwohl eine einfache Durchnummerierung möglich ist, empfiehlt sich eine hilfreichere Namensgebung, wie zum Beispiel „r_{Wachstum}“. Der Modellstrukturgenerator versieht den Software-Code des Modells automatisch mit Kommentarzeilen, damit er für den Benutzer besser lesbar ist. Unter anderem dafür werden die hier gewählten Bezeichnungen für die Reaktionen verwendet.

Die Anzahl der Elemente in `.rName` muss der Anzahl der zu modellierenden Reaktionen entsprechen. Bei der Modellerzeugung durch den Modellstrukturgenerator wird dies automatisch überprüft.

4.2.4 Reaktionsgleichungen (Koeffizientenmatrix)

Wie bereits erwähnt, werden im Rahmen dieser Arbeit Modelle durch Massenbilanzgleichungen definiert. Für Fed-Batch-Ansätze gilt:

$$\dot{x} = \text{Zudosierung} - \text{Probennahme} + \text{Reaktion} \tag{4.4}$$

Nicht erläutert wurde bisher die Frage, woher die Informationen für die Reaktionen kommen. Ein für viele Benutzer nahe liegender Weg beginnt bei chemischen Reaktionsgleichungen. Dieser Abschnitt wird zeigen, wie derartige Gleichungen zu Modellbausteinen führen.

Dazu wird folgendes Beispiel betrachtet: Ein Stoff A reagiere mit einem Stoff B mit der Reaktionsgeschwindigkeit r zu Stoff C. Eine allgemein übliche Schreibweise dafür lautet:

$$A + B \xrightarrow{r} C$$

Berücksichtigt man die stöchiometrischen Reaktionskoeffizienten n, wird daraus:

$$n_A \cdot A + n_B \cdot B \xrightarrow{r} n_C \cdot C$$

Beispiel:

$$2\mathrm{H}_2 + \mathrm{O}_2 \xrightarrow{r} 2\mathrm{H}_2\mathrm{O} \qquad \text{(Mengeneinheiten)}$$

Für die Aufstellung von Massenbilanzen wird in Reaktionsgleichungen anstelle der Mengeneinheit [mol] eine Masseneinheit, beispielsweise [g], verwendet. Die stöchiometrischen Koeffizienten werden durch sogenannte Ausbeute- oder pseudo-stöchiometrische[4] Koeffizienten Y ersetzt. Die zu Edukten gehörenden Ausbeutekoeffizienten besitzen negative Werte.

Als Beispiel für die nächsten Abschnitte soll die vereinfachte biochemische Reaktion r_1 von Glucose („Glc“) und Ammonium („Am“) zu Biomasse („X“) dienen:

$$\mathrm{Y_{Glc}}\cdot\mathrm{Glc} + \mathrm{Y_{Am}}\cdot\mathrm{Am} \xrightarrow{r_1} \mathrm{Y_X}\cdot\mathrm{X} \qquad \text{(Masseneinheiten)} \tag{4.5}$$

Die Reaktion führt gemäß den Erläuterungen ohne Zudosierung und Probennahme zu den folgenden Bilanzgleichungen für das Zustandsraummodell:

$$\begin{aligned} \dot{m}_{\mathrm{Glc}}(t) &= Y_{Glc} \cdot r_1 \\ \dot{m}_{\mathrm{Am}}(t) &= Y_{Am} \cdot r_1 \\ \dot{m}_{\mathrm{X}}(t) &= Y_X \cdot r_1 \end{aligned} \tag{4.6}$$

Um die Reaktionen in einer digitalen Form für den Modellstrukturgenerator zu erfassen, wird eine Koeffizienten-Matrix (K-Matrix) verwendet, in der die Zeilen die einzelnen Modellzustände repräsentieren. Informationen über eine Reaktion werden in jeweils einer Spalte dieser Matrix gespeichert[5], siehe Abbildung 4.2.

Negative Vorzeichen eines Elements in der K-Matrix kennzeichnen für den Modellstrukturgenerator Edukte, der Buchstabe `k` bedeutet, dass im Modell ein zu identifizierender Parameter einzusetzen ist. Zahlenwerte stehen für konstante Werte, die nicht

[4] Der Begriff „pseudo-stöchiometrisch“ verdeutlicht u.a., dass es sich nicht um chemische, sondern biochemische Reaktionsgleichungen handelt, in denen Massen und nicht Mol verwendet werden.

[5] Die Darstellung in Matrizenform weicht von dem in Abschnitt 3.1.1 vorgestellten Formalismus für $\mathbf{M_R}$ insofern ab, als dass nun über die Anzahl der Reaktionen indiziert wird und nicht über die Anzahl der Produkte. Der hauptsächliche Unterschied besteht darin, dass hier mehrere Reaktionen mit den gleichen Edukten und Produkten einzeln referenziert werden. Dies gilt analog auch für die R- und P-Matrizen in den folgenden Abschnitten.

als Parameter in das Modell aufgenommen werden. Alle Reaktionen werden vom Modellstrukturgenerator so normiert, dass ein Ausbeutekoeffizient immer zu `1` wird, wie unten gezeigt wird. Damit entfällt ein Y-Parameter im Gleichungssystem 4.6. Eine Null in der Zeile eines Zustands gibt an, dass dieser Zustand nicht an der jeweiligen Reaktion beteiligt ist.

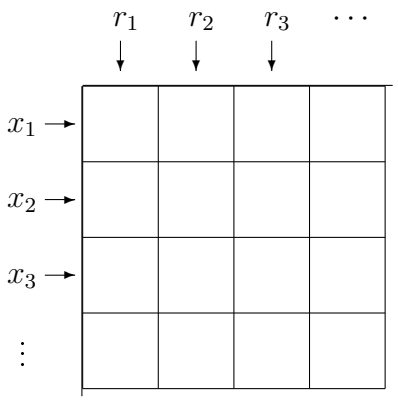


Abbildung 4.2: K-Matrix: In der Position der Elemente sind der Zustand und die Reaktion kodiert.

Das obige Beispiel wird wie folgt kodiert, mit m_{Glc}, m_{Am} und m_X als den ersten drei Modellzuständen:

$$\begin{bmatrix} -\mathrm{k} \\ -\mathrm{k} \\ \mathrm{k} \\ \vdots \end{bmatrix} \tag{4.7}$$

Tritt in einer Spalte außer Zahlenwerten nur ein einziges `(±)k` auf, und gilt die Massenerhaltung für diese Reaktion, siehe Abschnitt 4.2.9, wird für den betreffenden Parameter ein Anfangswert aus den anderen Elementen der Spalte für die Parameteridentifikation berechnet. Es wird zudem eine Vorzeichenprüfung durchgeführt und gegebenenfalls eine Warnung ausgegeben, wenn die Vorzeichen nicht zueinander passen: Hat der Benutzer beispielsweise ein `k` eingetragen, also durch das Vorzeichen ein Produkt deklariert, sollte sich aus der Berechnung kein negativer Wert für `k` (Edukt) ergeben. Die Software ist so programmiert, dass die Verwendung von mehr als einem `k`-Zeichen in einer Spalte automatisch zur Aktivierung der Massendefekt-Option für die betreffende Reaktion führt, es sei denn es handelt sich um eine interne Speichergröße.

Bei einer automatischen Messdatenanalyse (nicht Teil des Modellstrukturgenerators) ist es möglich, dass aufgrund des Messrauschens viele nur scheinbare Reaktionen erkannt werden. Die meisten falsch erkannten Reaktionen beziehungsweise fälschlicherweise erkannten Komponenten einer Reaktion sind durch sehr kleine Ausbeutekoeffizienten Y zu erkennen. Daher werden betragsmäßig sehr kleine Zahlenwerte in der Matrix wie Nullen behandelt. Als Grenzwert für die Y-Werte wird standardmäßig `sqrt(eps)`[6] in Matlab verwendet. Der genaue Zahlenwert hängt von der Rechengenauigkeit der verwendeten Hardware-Architektur und Software ab. Für typische PCs liegt der Wert unter 10^{-7}.

Die Namen der Parameter werden automatisch erzeugt und beinhalten, zu welchen Zuständen und zu welcher Reaktion der jeweilige Parameter gehört. Somit kann später, beispielsweise bei einer Parameteranalyse aber auch bei eventuellen Warn- oder Fehlermeldungen, schnell vom Namen auf die entsprechende Stelle des Differentialgleichungssystems geschlossen werden. Beispielsweise erhält der Y-Parameter der ersten Gleichung in 4.6 den vollständigen Namen $Y_{Glc\_X\_r_1}$.

Der Modellstrukturgenerator generiert aus den Informationen, die in der K-Matrix enthalten sind, das Grundgerüst für das Differentialgleichungssystem, welches schrittweise durch weitere Informationen in der `StrDef` ergänzt wird. Das oben aufgeführte Beispiel (Gleichungssystem 4.6) führt zu den folgenden, noch unvollständigen Gleichungen für die Zustände m_{Glc}, m_{Am} und m_X. Für eine bessere Verständlichkeit des Programmcodes werden bei Edukten die Minus-Zeichen für die Ausbeutekoeffizienten explizit ausgeschrieben. Die Y-Parameterwerte sind daher immer positiv.

$$\begin{aligned} \dot{m}_{Glc} &= -Y_{Glc\_X\_r_1} \cdot \mu_{\max,r_1} \cdots \\ \dot{m}_{Am} &= -Y_{Am\_X\_r_1} \cdot \mu_{\max,r_1} \cdots \\ \dot{m}_X &= +Y_{X\_GlcAm} \cdot \mu_{\max,r_1} \cdots \end{aligned} \tag{4.8}$$

Ebenso wie die maximalen Reaktionsraten $\mu_{\max}$, sind die Werte der Ausbeutekoeffizienten Y während der Modellbildung häufig noch unbekannt. Sie werden erst im Rahmen einer Parameteridentifikation bestimmt.

Um die Identifikation zu vereinfachen, kann die Anzahl der unbekannten Parameter Y wie folgt reduziert werden. In jeder Reaktion ist eine Normierung auf einen Ausbeutekoeffizienten möglich. Beispielsweise kann Gleichung 4.5 durch Y_X dividiert werden,

[6] Damit ist die Quadratwurzel der maschinenabhängigen Konstanten `eps` gemeint.

wodurch sich mit

$$\tilde{Y}_{Glc} = Y_{Glc}/Y_X \qquad \text{und} \tag{4.9}$$

$$\tilde{Y}_{Am} = Y_{Am}/Y_X \tag{4.10}$$

die folgende Vereinfachung ergibt:

$$\tilde{\mathrm{Y}}_{\mathrm{Glc}}\cdot\mathrm{Glc} + \tilde{\mathrm{Y}}_{\mathrm{Am}}\cdot\mathrm{Am} \xrightarrow{r_1} X \tag{4.11}$$

Wenn in einer Reaktion die Massen erhalten werden, also im Beispiel $\tilde{Y}_{Glc} + \tilde{Y}_{Am} = 1$ gilt, kann ein weiterer Parameter eingespart werden. Beispielsweise kann der Parameter $\tilde{Y}_{Am}$ durch $1 - \tilde{Y}_{Glc}$ ersetzt werden. Die Reaktionsgleichung 4.11 vereinfacht sich somit zu

$$\tilde{\mathrm{Y}}_{\mathrm{Glc}}\cdot\mathrm{Glc} + (1-\tilde{\mathrm{Y}}_{\mathrm{Glc}})\cdot\mathrm{Am} \xrightarrow{r_1} X. \tag{4.12}$$

Die Parameterreduktion durch Normierung wird automatisch vom Modellstrukturgenerator durchgeführt. Für die zusätzliche Parameterersetzung im Falle einer Massenerhaltung wurden viele Typen von Reaktionen vordefiniert, so dass auch sie automatisch vorgenommen werden kann. Die vordefinierten Reaktionstypen werden im Folgenden aufgelistet. Die Einträge werden kurz anhand eines Beispiels erläutert:

A $\longrightarrow$ **B**: $\mathrm{A} \longrightarrow \mathrm{B}$, ($Y_A$ wurde normiert auf 1, daher gilt auch auch $\tilde{Y}_B = 1$)

Dieser Eintrag benennt zuerst den Typ der Reaktion, hier ist die Reaktion vom Typ „Edukt (A) reagiert zum Produkt (B)“. Diese Reaktion wird im Programmcode des Modells schematisch durch $\mathrm{A} \longrightarrow \mathrm{B}$ implementiert. Als Erklärung folgt, dass durch die Normierung von Y_A auf 1 wegen der Massenerhaltung auch $\tilde{Y}_B = 1$ gilt. In diesem Fall sind keine weitere Parameter notwendig. Entsprechend vereinfachen sich die Terme für die Differentialgleichungen.

Da in einem neuen Modell nur die $\tilde{Y}$-Parameter interessieren (und keine Y-Parameter mehr auftreten), werden diese der Einfachheit halber im Folgenden (und auch im Modellcode) weiterhin mit Y bezeichnet.

A $\longrightarrow$ **B**: $\mathrm{A} \longrightarrow \mathrm{B}$
Y_A wurde normiert auf 1, daher gilt auch auch $Y_B = 1$

A + **B** $\longrightarrow$ **C**: $\mathrm{Y_A}\cdot\mathrm{A} + (1-\mathrm{Y_A})\cdot\mathrm{B} \longrightarrow \mathrm{C}$
Y_C wurde normiert auf 1, $0 < Y_A < 1$

A $\longrightarrow$ **B** + **C**: $\mathrm{A} \longrightarrow \mathrm{Y_B} \cdot \mathrm{B} + (1 - \mathrm{Y_B}) \cdot \mathrm{C}$
Y_A wurde normiert auf 1, $0 < Y_B < 1$

A + **B** $\longrightarrow$ **C** + **D**: $\mathrm{A} + \mathrm{Y_B} \cdot \mathrm{B} \longrightarrow \mathrm{Y_C} \cdot (1 + \mathrm{Y_B}) \cdot \mathrm{C} + (1 - \mathrm{Y_C})(1 + \mathrm{Y_B}) \cdot \mathrm{D}$
Y_A wurde normiert auf 1, $Y_B > 0$ und $0 < Y_C < 1$

A + **B** + **C** $\longrightarrow$ **D**: $\mathrm{A} + \mathrm{Y_B} \cdot \mathrm{B} + \mathrm{Y_C} \cdot \mathrm{C} \longrightarrow (1 + \mathrm{Y_B} + \mathrm{Y_C}) \cdot \mathrm{D}$
Y_A wurde normiert auf 1, $Y_B > 0$ und $Y_C > 0$

A $\longrightarrow$ **B** + **C** + **D**: $(1 + \mathrm{Y_C} + \mathrm{Y_D}) \cdot \mathrm{A} \longrightarrow \mathrm{B} + \mathrm{Y_C} \cdot \mathrm{C} + \mathrm{Y_D} \cdot \mathrm{D}$
Y_B wurde normiert auf 1, $Y_C > 0$ und $Y_D > 0$

A + **B** + **C** $\longrightarrow$ **D** + **E**:
$\mathrm{A} + \mathrm{Y_B} \cdot \mathrm{B} + \mathrm{Y_C} \cdot \mathrm{C} \longrightarrow \mathrm{Y_D} \cdot (1 + \mathrm{Y_B} + \mathrm{Y_C}) \cdot \mathrm{D} + (1 - \mathrm{Y_D}) \cdot (1 + \mathrm{Y_B} + \mathrm{Y_C}) \cdot \mathrm{E}$
Y_A wurde normiert auf 1, $Y_B > 0$ und $Y_C > 0$ und $0 < Y_D < 1$

A + **B** $\longrightarrow$ **C** + **D** + **E**:
$\mathrm{Y_A} \cdot \mathrm{A} + \mathrm{Y_B} \cdot \mathrm{B} \longrightarrow \mathrm{C} + \mathrm{Y_D} \cdot (\mathrm{Y_A} + \mathrm{Y_B}) \cdot \mathrm{D} + (1 - \mathrm{Y_D}) \cdot (\mathrm{Y_A} + \mathrm{Y_B}) \cdot \mathrm{E}$
Y_C wurde auf 1 normiert, $Y_A > 0$ und $Y_B > 0$ und $0 < Y_D < 1$

A + **B** + **C** $\longrightarrow$ **D** + **E** + **F**:
$\mathrm{A} + \mathrm{Y_B} \cdot \mathrm{B} + \mathrm{Y_C} \cdot \mathrm{C} \longrightarrow$
$\mathrm{Y_D} \cdot (1 + \mathrm{Y_B} + \mathrm{Y_C}) \cdot \mathrm{D} + \mathrm{Y_E} \cdot (1 + \mathrm{Y_B} + \mathrm{Y_C}) \cdot \mathrm{E} + (1 - (\mathrm{Y_D} + \mathrm{Y_E})) \cdot (1 + \mathrm{Y_B} + \mathrm{Y_C}) \cdot \mathrm{F}$
Y_A wurde normiert auf 1, $Y_B > 0$ und $Y_C > 0$ und $0 < Y_D + Y_E < 1$

A + **B** + **C** + **D** $\longrightarrow$ **E**:
$\mathrm{A} + \mathrm{Y_B} \cdot \mathrm{B} + \mathrm{Y_C} \cdot \mathrm{C} + \mathrm{Y_D} \cdot \mathrm{D} \longrightarrow (1 + \mathrm{Y_B} + \mathrm{Y_C} + \mathrm{Y_D}) \cdot \mathrm{E}$
Y_A wurde normiert auf 1, $Y_B > 0$ und $Y_C > 0$ und $Y_D > 0$

Statistisch gesehen sinkt die Wahrscheinlichkeit einer erfolgreichen Reaktion mit der Anzahl von Edukt-Molekülen, da alle Moleküle gleichzeitig an einem gemeinsamen Ort zusammentreffen müssen. Reaktionen mit vielen Edukten sind daher prinzipiell unrealistisch oder laufen zumindest nur mit einer geringen Geschwindigkeit ab. Allerdings handelt es sich hier um abstrakte Reaktionen, mit deren Hilfe ein unbekanntes Reaktionsnetzwerk abgebildet werden soll. Das Wahrscheinlichkeitsargument findet daher keine Anwendung. Dennoch war es bei den vom Autor betrachteten Beispielen nicht notwendig, Reaktionen mit mehr als insgesamt fünf Reaktionspartnern zu modellieren. Eine Erweiterung der Liste ist jedoch jederzeit möglich.

Dem Benutzer wird die Möglichkeit gegeben, für jede einzelne Reaktion festzulegen, ob die Massenerhaltung gilt, wie in Abschnitt 4.2.9 beschrieben. In der Praxis ist die

Wahlfreiheit jedoch häufig eingeschränkt. So wird zum Beispiel für die Gleichung 4.11, die ein Zellwachstum beschreibt, im Allgemeinen keine Massenerhaltung gelten, wenn das bei der aeroben Atmung entstehende Kohlendioxid nicht quantitativ gemessen wird. Das Aufstellen einer elementaren Bilanz ist daher in diesem Beispiel nicht möglich. Wenn die Massenerhaltung laut `StrDef` für eine Reaktion nicht gilt, wird nur die Normierung automatisch durchgeführt.

Zukünftig sollte die oben beschriebene automatischen Verarbeitung für den Fall der Massenerhaltung innerhalb des Modellstrukturgenerators vervollständigt werden. Einerseits müssen hierfür unterschiedliche Aufrufe für die verschiedenen Optimierer implementiert werden, andererseits muss auch beachtet werden, dass nicht alle Optimierer Nebenbedingungen berücksichtigen, wie sie beispielsweise für die vorletzte Gleichung (A+B+C→D+E+F) benötigt werden: $0 < Y_D + Y_E < 1$.

Neben den oben aufgeführten Reaktionen gibt es noch zwei Sonderfälle; Reaktionen ohne Edukt beziehungsweise ohne Produkt. Diese beiden Reaktionen können zur Vereinfachung eines Modells benutzt werden. Der erste Fall kann beispielsweise eintreten, wenn ein Produkt in einer so geringen Menge auftritt, dass sich die Produktion messtechnisch nicht durch eine Edukt-Massenverringerung nachweisen lässt. Die zweite Reaktion kann verwendet werden, wenn ein Produkt in inerte Stoffe zerfällt, die nicht im Modell enthalten sind.

A $\longrightarrow$ **(−)**: A $\longrightarrow$ (−) und

(−) $\longrightarrow$ **A**: (−) $\longrightarrow$ A

In beiden Fällen wird analog zu anderen Reaktionen automatisch das Produkt aus Ausbeutekoeffizient und maximaler Reaktionsrate ($Y \cdot \mu_{\max}$) in das Modell aufgenommen. Für beide Reaktionen gilt, dass die zwei Parameter im Differentialgleichungssystem nur in genau diesem Produkt auftreten. Daher wird aus Gründen der Identifizierbarkeit ein Parameter auf einen konstanten Wert festgesetzt. Die Automatik setzt zu diesem Zweck den Ausbeutekoeffizienten Y auf eins. Die Anpassung der Reaktionsgeschwindigkeit erfolgt somit über den $\mu_{\max}$-Parameter.

4.2.5 Reaktionskinetiken (Regulationsmatrix)

Im Modell wird die Regulation der einzelnen Reaktionen durch Formalkinetiken festgelegt. Welche Kinetikanteile in den einzelnen Reaktionen eingesetzt werden sollen, wird

durch Einträge in der Regulationsmatrix (R-Matrix) definiert, die nach dem selben Schema wie die K-Matrix aufgebaut ist und dieselbe Größe besitzt: Die Spaltenposition eines Kinetikeintrags innerhalb der Matrix legt die Reaktion fest, zu der die festgelegte Kinetik gehört, und die Zeile bestimmt, welcher Zustand als regulierende Größe darin auftritt. Bis auf konstruierte Ausnahmefälle, die in der bisherigen Praxis nicht auftraten, sind diese Angaben ausreichend, um das Differentialgleichungssystem eindeutig aufzubauen.

Als regulierende Größen werden meist die Konzentrationen der Massenzustände verwendet. Der Modellstrukturgenerator erzeugt automatisch die notwendigen Gleichungen. Abschnitt 4.2.13 erläutert dies detaillierter.

Die Eingabemöglichkeiten sind sehr flexibel ausgelegt, wie unten genauer erläutert wird. Das folgende Beispiel zeigt eine Spalte der Matrix für die Reaktion r_1, in der die Reaktion sowohl durch den ersten als auch durch den zweiten Zustand, also Glucose und Ammonium, reguliert wird. `MiMe` ist eine Abkürzung und Codewort im ABC-System für eine bestimmte Formalkinetik; die Michaelis-Menten-Kinetik (MiMe). An einer Reaktion unbeteiligte Zustände werden durch leere Elemente[7] symbolisiert (in der Matrixspalte 4.13 dargestellt durch einen Strich).

$$\begin{bmatrix} \text{MiMe} \\ \text{MiMe} \\ - \\ \vdots \end{bmatrix} \tag{4.13}$$

Das Code-Beispiel aus dem vorherigen Abschnitt auf Seite 94 erweitert sich durch die Festlegung für die Reaktion r_1 in der Spaltenmatrix 4.13 auf:

$$\begin{aligned} \dot{m}_{Glc} &= -Y_{Glc\_X\_r_1} \cdot \mu_{\max,r_1} \cdot \text{MiMe}(c_{Glc}, \text{MiMe}_{\text{Glc\_r1}}) \cdot \text{MiMe}(c_{Am}, \text{MiMe}_{\text{Am\_r1}}) \cdots \\ \dot{m}_{Am} &= -Y_{Am\_X\_r_1} \cdot \mu_{\max,r1} \cdot \text{MiMe}(c_{Glc}, \text{MiMe}_{\text{Glc\_r1}}) \cdot \text{MiMe}(c_{Am}, \text{MiMe}_{\text{Am\_r1}}) \cdots \\ \dot{m}_X &= +\quad 1 \quad \cdot \mu_{\max,r1} \cdot \text{MiMe}(c_{Glc}, \text{MiMe}_{\text{Glc\_r1}}) \cdot \text{MiMe}(c_{Am}, \text{MiMe}_{\text{Am\_r1}}) \cdots \end{aligned} \tag{4.14}$$

Im ABC-System ist zum Beispiel $\texttt{MiMe}(\texttt{c}_{\texttt{Am}}, \texttt{MiMe}_{\texttt{Am\_r1}})$ der Aufruf der MiMe-Kinetik mit der Ammonium-Konzentration c_{Am} als Einflussgröße, sowie dem Parameter $\text{MiMe}_{\text{Am\_r1}}$. Auch dieser Name wird automatisch erzeugt und ermöglicht die schnelle und vor allem eindeutige Zuordnung innerhalb des Modells.

Als Eingabemöglichkeiten für die R-Matrix stehen für die einzelnen Elemente die folgenden Optionen zur Auswahl:

[7] Genaugenommen wird das spezielle Matlab-Element `NaN` (Not a Number) verwendet.

- Wie bereits im Beispiel dargestellt, kann ein Feld der Matrix mit dem Namen einer Kinetik gefüllt werden. Eine Liste häufig verwendeter, implementierter Kinetikfunktionen wird in Abschnitt 4.4 gegeben. Die dortige Liste kann jederzeit um weitere Kinetiken erweitert werden.

- Auch mehrere, durch Leerzeichen oder Kommata getrennte Namen von Formalkinetiken dürfen eingetragen werden, zum Beispiel `MiMe,Haldane`. Der Modellstrukturgenerator erstellt dann automatisch verschiedene Modellkandidaten, die sich an der betreffenden Position im Differentialgleichungssystem durch die angegebenen Kinetiken voneinander unterscheiden.

- Durch den Eintrag einer Zahl anstelle eines Kinetiknamens erzeugt der Modellstrukturgenerator ebenfalls mehrere Modellkandidaten. Durch eine Zahl wird eine Gruppe von Kinetiken spezifiziert, die für die einzelnen Kandidaten verwendet werden. Eine `1` referenziert beispielsweise auf alle einparametrigen, limitierenden Kinetiken.

- Eine Kombination aus den drei vorherigen Eingabemöglichkeiten ist ebenfalls zulässig. So werden beispielsweise durch die Angabe `1,Haldane` Modellkandidaten erzeugt, die entweder Formalkinetiken vom Typ `1` oder die `Haldane`-Kinetik an der betreffenden Stelle im Differentialgleichungssystem enthalten.

Um die Größe einer Modellfamilie zu begrenzen, empfiehlt es sich, die Auswahl der möglichen Formalkinetiken der Situation angepasst möglichst klein zu halten. Ist für eine Reaktion nicht bekannt, ob beispielsweise eine limitierende oder inhibierende Kinetik besser geeignet ist, sollte von jedem Kinetiktyp nur eine Kinetik überprüft werden. Wenn der Kinetiktyp bekannt ist, können verschiedene Kinetiken des entsprechenden Typs ausgetestet werden. Abschnitt 5.1 erläutert genauer die Probleme, die durch große Modellfamilien entstehen, und somit den Hintergrund für diese Empfehlung.

Drei Kinetiken erfüllen einen besonderen Zweck für die Modellbildung. Mit ihrer Hilfe können Modellkandidaten erzeugt werden, die sich in ihrer effektiven mathematischen Struktur voneinander unterscheiden. So können unterschiedliche Strukturvarianten innerhalb einer Modellfamilie erzeugt werden, was ansonsten mit der aktuellen Version des Modellstrukturgenerators nicht vorgesehen ist. Durch den Vergleich der verschiedenen Varianten – bezüglich ihrer Anpassung an Messdaten – kann die bestgeeignete Struktur bestimmt werden. Die Auswertung bezüglich der Strukturunterschiede kann natürlich

erst nach einer erfolgten Parameteridentifikation aller Modellkandidaten erfolgen. Die drei Sonderkinetiken sind:

- `One`(c_A): Diese Kinetik hat einen Übergabeparameter, dessen Wert jedoch ignoriert wird, und gibt immer den Wert eins aus. Mit ihrer Hilfe kann im Vergleich mit anderen Modellkandidaten untersucht werden, ob die Einflussgröße, hier c_A, einen signifikanten Einfluss hat oder aus dem Modell entfernt werden kann. Letzteres ist der Fall, wenn der Kandidat mit der `One`-Kinetik die beste Anpassung ermöglicht.
- `Zero`(c_A): Im Gegensatz zu `One` gibt diese Kinetik immer den Wert null zurück. Sie dient dazu herauszufinden, wiederum im Vergleich mit anderen Modellkandidaten, ob die betreffende Reaktion überhaupt stattfindet. Erreicht der Modellkandidat mit der `Zero`-Kinetik die beste Anpassung, ist die Modellierung der betreffenden Reaktion offenbar nicht notwendig.
- `Identity`(c_A): Diese spezielle Formalkinetik gibt den Wert der Einflussgröße c_A direkt zurück. Auf diese Weise können zum Beispiel chemische Reaktionen erster Ordnung modelliert werden.

4.2.6 P-Faktoren (P-Matrix)

Die Geschwindigkeit einer Reaktion ist proportional zum P-Faktor, einer Zustandsgröße, wie in Abschnitt 3.1.1 erläutert wurde. Die Festlegung, welche Zustandsgröße vom Modellstrukturgenerator für die einzelnen Reaktionen verwendet werden soll, erfolgt durch entsprechende Einträge in der P-Matrix. Auch diese Matrix ist nach demselben Schema wie die K- oder R-Matrix aufgebaut.

Das Beispiel aus dem letzten Abschnitt, Gleichungssystem 4.14, fortführend, wird nun festgelegt, dass der P-Faktor der Reaktion r_1 die Biomasse m_X, der dritten Zustand des Modells, ist. In der P-Matrix wird diese Information wie folgt in der ersten Spalte gespeichert:

$$\begin{bmatrix} 0 \\ 0 \\ 1 \\ \vdots \end{bmatrix} \tag{4.15}$$

Die Eintragung einer `1` bedeutet, dass der Modellstrukturgenerator die Reaktion im Modell mit dem Zustand multipliziert, der mit der jeweiligen Zeile korrespondiert. Da im Beispiel der dritte Zustand m_X ist, erfolgt die Eintrag in der dritten Zeile.

Die Differentialgleichungen aus dem vorangegangenen Abschnitt werden gemäß der Information in der P-Matrix wie folgt erweitert:

$$\begin{aligned}
\dot{m}_{Glc} &= -Y_{Glc_X_r_1} \cdot \mu_{\max,r_1} \cdot \mathrm{MiMe}(c_{Glc}, \mathrm{MiMe}_{\mathrm{Glc_r1}}) \cdot \mathrm{MiMe}(c_{Am}, \mathrm{MiMe}_{\mathrm{Am_r1}}) \cdot \quad m_X \\
\dot{m}_{Am} &= -Y_{Am_X_r_1} \cdot \mu_{\max,r1} \cdot \mathrm{MiMe}(c_{Glc}, \mathrm{MiMe}_{\mathrm{Glc_r1}}) \cdot \mathrm{MiMe}(c_{Am}, \mathrm{MiMe}_{\mathrm{Am_r1}}) \cdot \quad m_X \\
\dot{m}_X &= \underbrace{+ \quad 1 \quad \cdot \mathrm{MiMe}(c_{Glc}, \mathrm{MiMe}_{\mathrm{Glc_r1}}) \cdot \mu_{\max,r1}}_{\text{K-Matrix}} \cdot \underbrace{\mathrm{MiMe}(c_{Am}, \mathrm{MiMe}_{\mathrm{Am_r1}})}_{\text{R-Matrix}} \cdot \underbrace{m_X}_{\text{P-Matrix}}
\end{aligned}$$

Info aus: (4.16)

Die möglichen Elemente der P-Matrix sind `0` oder `1`. Enthalten alle Elemente einer Spalte nur `0` wird die Reaktion nicht erweitert[8]. Da dieser Fall in der bisherigen Praxis jedoch nicht auftrat, wird vom Modellstrukturgenerator eine Warnung ausgeben. Umgekehrt dürfen aber mehrere Elemente einer Spalte die `1` enthalten. Der Modellstrukturgenerator interpretiert dies als Alternativen und erzeugt Modellkandidaten mit den verschiedenen Zustandsgrößen als P-Faktor.

Die Differentialgleichungen für die drei Zustände des Beispiels (Gleichung 4.16) sind nun vollständig, nachdem die Informationen aus den drei Matrizen (K-, R- und P-Matrix) extrahiert wurden. In der Regel gibt es weitere Reaktionen, die in einem Modell definiert werden, und diese werden der Reihe nach abgearbeitet. Daraus ergeben sich dann weitere Terme, die analog automatisch erstellt und zu den bisherigen Gleichungen addiert werden. Für obiges Beispiel ergibt sich folgendes Code-Fragment der automatisch erstellten Datei `Symb_f`.

Symb_f (Ausschnitt)

```
% Abkürzungen
m_Glc        =  x(1);
m_Am         =  x(2);
m_X          =  x(3);
V            =  x(4);

c_Glc        =  m_Glc / V;
c_Am         =  m_Am / V;

Glc_Feed     =  u(1) * u(2);
Am_Feed      =  u(3) * u(4);
```

[8] Der Modellstrukturgenerator fügt ersatzweise als Größe kein Zustand, sondern eine Eins als Faktor in die Gleichung ein.

```
% Reaktion: rX  (Glc + Am --> X)
rX =  MuMax.rX ...
     * MiMe([c_Glc],[MiMe_.k.Glc.rX]) ...
     * MiMe([c_Am],[MiMe_.k.Am.rX]);

% DGL-Gleichungen
% m_Glc
xdot(1) = - rX * Y_Glc_X * m_X + Glc_Feed;

% m_Am
xdot(2) = - rX * Y_Am_X * m_X + Am_Feed;

% m_X
xdot(3) =   rX * m_X;

% Volumenänderung durch Feeds, Korrekturfluide und Abdampf
xdot(4) = u(1) + u(3) + u(7) + u(8) + u(9) - u(10);
```

4.2.7 Modellergänzungen

Der zu erstellende Modelltyp, siehe Abschnitt 3.1.4, wird nicht eigens in der Modellstrukturdefinition festgelegt, sondern ergibt sich über die Arten oder Typen der verwendeten Zustände. Um dies zu verstehen, muss zunächst eine Modifikation erläutert werden, die es erlaubt, auch Zusammenhänge zwischen Größen herzustellen, die nicht Teil des eigentlichen Gleichungssystems sind. In strukturierten Modellen kann es beispielsweise gewünscht sein, dass der P-Faktor einer Reaktion die Gesamtbiomasse ist, die aber als Zustand in einem strukturierten Modell nicht enthalten ist. Daher geht der Modellstrukturgenerator davon aus, dass die Liste der Zustände in der Modellstrukturdefinition unabhängig von den übrigen Modellzuständen am Ende um die Zustände „aktive Biomasse“ und „Gesamtbiomasse“ erweitert ist, und kann im entsprechenden Fall darauf zurückgreifen. Entsprechend ist auch die Zeilenanzahl der K-, R- und P-Matrizen um zwei größer. Für diese zwei Hilfszustände werden keine Differentialgleichungen erstellt, sie können aber in den Gleichungen der übrigen Zustände auftreten.

Die einzelnen Modelltypen, ihre Erkennung und die Verwendung der zusätzlichen Zustände wird im Folgenden erläutert. Dabei wird unterschieden zwischen dem exakten xTyp-Vergleich, das heißt, die als xTyp definierte Zeichenkette muss identisch zum Suchmuster sein (Reihenfolge, Groß-/Kleinschreibung), und der einfachen Überprüfung eines Bestandteils innerhalb der xTyp-Zeichenkette: Die Suche nach `K` ist daher in diesem Fall erfolgreich bei den xTypen `K`, `KAX` oder `XKA`.

- Verwendet ein Modell keine Zustände, die das Innere einer Zelle beschreiben, kann also kein xTyp gefunden werden, der das Zeichen `K` (Kompartiment) enthält, handelt es sich um ein „unstrukturiertes Modell“. Ein Zustand, der die Biomasse beschreibt, also exakt vom xTyp `X` ist[9], muss somit bereits vorhanden sein, wohingegen eine „aktive Biomasse“ in dieser Art von Modell nicht existiert.

  Die beiden zusätzlichen Zustände besitzen deshalb keine weitere Bedeutung für diesen Modelltyp. Einträge in den zwei letzten Zeilen der drei Matrizen werden ignoriert.

- Existiert ein Zustand für die aktive Biomasse (xTyp `KAX`), aber keine Gesamtbiomasse (xTyp exakt `X`), so handelt es sich um ein Modell vom Typ „einfaches Modell mit Speicher“. Zur Berechnung der Gesamtbiomasse wird die Summe aller Zustände, die in ihrem xTyp ein `K` enthalten (also auch `KAX`), gebildet. Die zusätzlichen, redundanten Einträge für aktive Biomasse in der jeweils vorletzten Zeile der drei Matrizen werden ignoriert, stattdessen werden nur die ersten Zeileneinträge für die aktive Biomasse berücksichtigt.

- Existieren weder Zustände mit dem exakten xTyp `X` noch `KAX`, wird von einem komplexeren Modell, einem „Kompartiment-Modell“ ausgegangen. Die beiden zusätzlichen Zustände werden als mögliche P-Faktoren und regulierend eingreifende Größen benötigt. Die Gesamtbiomasse errechnet sich aus der Summe aller Zustände, die im xTyp ein `K` enthalten, während die aktive Biomasse für die Summation nur die Zustände mit dem xTyp-Anteil `KA` berücksichtigt.

4.2.8 Modellwahrscheinlichkeit

Für die einzelnen Modellkandidaten können unterschiedliche Wahrscheinlichkeiten vergeben werden, die Einfluss auf die Bearbeitungsreihenfolge bei der Parameteridentifikation haben. Derartige Informationen stammen beispielsweise aus der biologischen Phänomenerkennung [32]. Zukünftige Methoden können diese Wahrscheinlichkeiten jedoch auch anders verwenden. Ein Beispiel dafür könnten automatisierte Verfahren zur Modellfehleranalyse sein.

Wird in dem Feld `.PMod` kein Wert angegeben, so wird die Wahrscheinlichkeit gleichverteilt über die gesamte Modellfamilie.

[9] „Exakt“ meint hier, dass kein anderer Codebuchstabe enthalten ist.

4.2.9 Massendefekt

In diesem Feld wird ein Vektor mit booleschen Werten gespeichert. Jedes Element charakterisiert hierbei eine Reaktion. Es legt fest, ob in der entsprechenden Reaktionsgleichung ein Massendefekt angenommen werden soll oder nicht. Diese Auswahl hat Einfluss auf die Gestaltung der entsprechenden Differentialgleichung, insbesondere auf die Anzahl der Modellparameter darin, wie in Abschnitt 4.2.4 dargelegt wurde.

Für interne Speichergrößen, durch den xTyp `KI` als solche gekennzeichnet, wird unabhängig von der Benutzervorgabe immer davon ausgegangen, dass kein Massendefekt vorliegt. Diese Annahme hilft, um besonders im Zusammenhang mit nicht messbaren Größen sinnvolle Ergebnisse zu erzielen.

4.2.10 pH-Wert

In den hier betrachteten experimentiellen Untersuchungen wurde meist versucht, den pH-Wert nicht nur während einer Fermentation durch eine Regelung konstant zu halten, sondern wegen der erwünschten Vergleichbarkeit von verschiedenen Fermentationen diesen Wert durch einheitliche Versuchsplanung auch beizubehalten. Durch die Einführung dieser „Stellgröße“ kann jedoch auch der Einfluss verschiedener pH-Werte experimentell – und im Modell – untersucht werden.

Der pH-Wert wirkt anders als die übrigen Stellgrößen auf die zu erzeugenden Modellgleichungen ein: Statt einen zusätzlichen additiven Term zu erzeugen, siehe Gleichung 4.1 in Abschnitt 4.2.2, wird er mit einer speziellen Formalkinetik für den pH-Wert zu den jeweiligen Reaktionstermen multipliziert. Außerdem muss bei der Erstellung der Differentialgleichung für das Volumen berücksichtigt werden, dass die pH-Wert-Stellgröße das Volumen nicht verändert: Die Volumenänderung muss weiterhin über die Korrekturfluidmengen Säure und Base erfasst werden.

Durch einen Vektor von booleschen Werten WAHR/FALSCH im Feld `.pH` kann für jede einzelne Reaktion festgelegt werden, ob die spezielle pH-Kinetik zu der betreffenden Reaktion automatisch hinzugefügt werden sollen. Gegenwärtig gibt es zwei Formalkinetiken zur Auswahl, die sich in ihrer Parameteranzahl unterscheiden. Die erste repräsentiert einen gaussschen Kurvenverlauf, die zweite Kinetik stellt eine Erweiterung dazu dar, siehe auch Abbildung 4.3:

$$\text{Erw. Gauss}: \quad f_q = 1 - \left(1 - \exp\left(-\frac{1}{2} \cdot \left(\frac{pH - pH_0}{\sigma}\right)^2\right)\right)^{Exp} \tag{4.17}$$

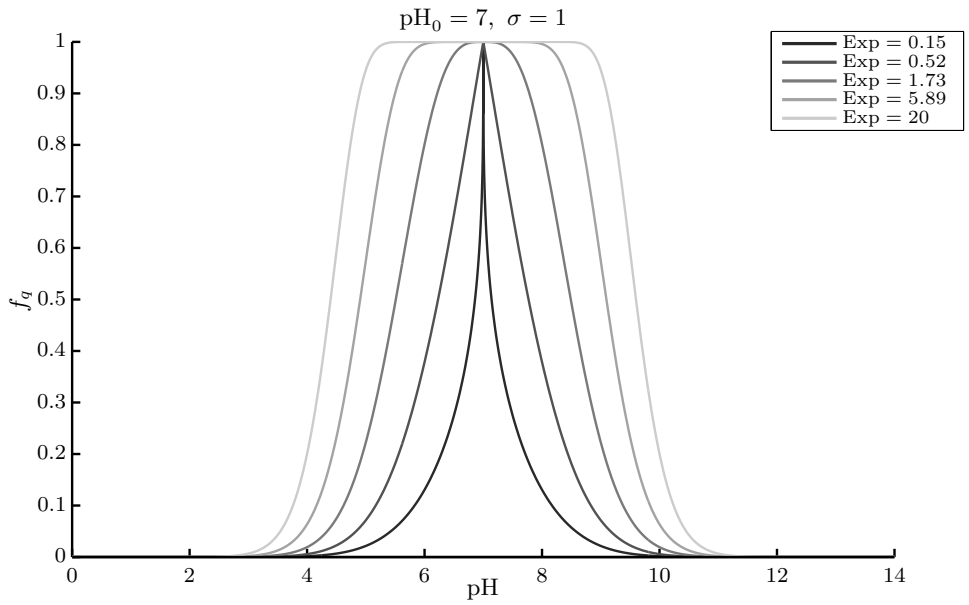


Abbildung 4.3: Darstellung der modifizierten Gauss-Kurve als pH-Kinetik

4.2.11 Spezielle Modellbausteine

Abgesehen von den Reaktionen, die der Benutzer für jedes Modell neu festlegen muss, gibt es verschiedene Eigenschaften im Verhalten von Mikroorganismen, die in den jeweiligen Modellen immer auf die gleiche Weise kodiert werden können. Für einige sind daher fertige Bausteine für die automatische Modellbildung verfügbar. Sie lassen sich über entsprechende Schalter in der Modellstrukturdefinition aktivieren. Diese biologischen Eigenschaften werden zusammen mit ihrer Implementierung im Modellstrukturgenerator im Folgenden erläutert.

Lag-Phase

Eine Lag-Phase beschreibt die zeitliche Verzögerung des Wachstums zu Beginn einer Fermentation. Meist ist der Effekt eine Folge der internen Anpassung des Zell-Metabolismus auf die neue Umgebung im Fermenter nach der Inokulation. Ist ein Lag-Effekt beobachtbar, kann ein spezieller Modellbaustein dafür aktiviert werden. Der hier vorgestellte Modellbaustein nutzt Formalkinetiken, die Lag-Phasen nur zu Beginn einer Fermentation beschreiben.

Enthält das Feld `StrDef.Lagphase` den booleschen Wert WAHR, wird bei unstrukturierten Modellen die rechte Seite der Differentialgleichung für den Zustand „Biomasse“

(xTyp: `X`) mit einer speziellen Formalkinetik multipliziert, die den Lag-Effekt beschreibt. Bei einem strukturierten Modell wird zunächst nach einem einzelnen Kompartiment „aktive Biomasse“ (xTyp: `KA`) gesucht, um hier die Gleichung zu modifizieren. Falls mehrere solcher Kompartimente existieren, wie es bei „strukturierten Modellen mittlerer Größe“ der Fall sein kann, wird die Gleichung für das dort vorhandene DNS-Kompartiment (xTyp: `KAD`) verändert. Alle Reaktionen, die in Verbindung mit dem aufgefundenen Zustand stehen, sind vom Lag-Effekt betroffen, wie in der folgenden Gleichung dargestellt wird:

$$\dot{x}_{KA} = \mathrm{Lag}(t, [\mathrm{Lag}_a, \mathrm{Lag}_T]) \cdot \sum_{i=1}^{\#\mathrm{Reak.}} r_i \cdot V \tag{4.18}$$

Der Wertebereich der „Lag“-Formalkinetik liegt zwischen null und eins. Die Kinetik-Funktion erwartet zwei Argumente: die aktuelle Zeit und einen Parametervektor, bestehend aus zwei Parametern, was durch durch die Matlab-Syntax [Lag_a, Lag_T] verdeutlicht wird:

$$\mathrm{Lag}(t, [\mathrm{Lag}_a, \mathrm{Lag}_T]) = \frac{1 - \mathrm{Lag}_a}{1 + \exp(-10 \cdot (t - (\mathrm{Lag}_T - \log(8)/10)))} + \mathrm{Lag}_a \tag{4.19}$$

Der erste Parameter der Lag-Kinetik gibt an, welchen Wert die Kinetikfunktion zum Zeitpunkt $t = 0$ zurückgibt. Durch $\mathrm{Lag}_a = 0.2$ wird beispielsweise festgelegt, dass alle betroffenen Reaktionen durch den Lag-Effekt anfangs in ihrer Geschwindigkeit nur mit 20 % der maximalen Geschwindigkeit ablaufen. Der verlangsamende Effekt reduziert sich mit fortschreitender Zeit. Der Parameter Lag_T definiert die Zeitspanne (in Stunden[10]), nach der etwa 90 % der maximalen Reaktionsraten erreicht werden[11]. Die Lag-Kinetik ahmt ein Schaltverhalten nach, siehe Abbildung 4.4 (links). Die untere zulässige Grenze für Lag_T wurde in der aktuellen Implementierung auf 0.5 h festgelegt: Kürzere Zeitspannen hätten im Rahmen der durchgeführten Versuche nur einen sehr geringen Einfluss auf den Versuchsverlauf und würden gerade zu Beginn der Fermentation viele Messungen erfordern, um den Effekt überhaupt beobachten zu können.

Startwerte für die Parameter-Optimierung können in `StrDef.STARTWERTE.Lagphase` übergeben werden. Normalerweise wird ein Vektor mit zwei Elementen erwartet. Diese

[10] beziehungsweise in der in `SystemInit` definierten Zeiteinheit

[11] Der Wert hängt von Lag_a ab, für $\mathrm{Lag}_a = 0.1$ sind es exakt 90 %.

werden den beiden Modellparametern als Startwerte zugeordnet. Ihre Reihenfolge entspricht jener in der Lag-Kinetik, also [Lag_a, Lag_T]. Um die Kompatibilität zu früheren Programmversionen zu gewährleisten, ist es auch möglich, statt eines Vektors nur einen Skalar zu übergeben. Dieser wird als Anfangswert für den Zeit-Parameter, Lag_T, interpretiert. Der erste Parameter, Lag_a, wird in diesem Fall automatisch auf den Anfangswert 0.1 gesetzt.

Gegenwärtig gibt es eine alternative „Lag2"-Kinetik für das Phänomen der Lag-Phase in der Kinetikbibliothek. Sie erlaubt einen fließenderen Übergang, siehe Abbildung 4.4 (rechts), allerdings ist der Startwert für die Kinetik nicht wählbar. Der erste Parameter Lag_a ist ein Maß für die Steigung. Der zweite Parameter Lag_T legt die Zeitspanne bis zum Erreichen des halben Endwertes, das heißt f_q=0.5, fest.

$$\mathrm{Lag2}(t, [\mathrm{Lag}_a, \mathrm{Lag}_T]) = \frac{1}{1 + \exp(-\mathrm{Lag}_a \cdot (t - \mathrm{Lag}_T))} \tag{4.20}$$

Die Unterteilung in verschiedene Kinetiken ist sinnvoll, um die Anzahl der notwendigen Parameter gering zu halten. Für jede der beiden Kinetiken werden nur zwei Parameter eingeführt. Die Abbildung 4.4 zeigt einige exemplarische Verläufe beider Kinetik-Varianten.

Als zusätzliche Angaben können in der `StrDef.Startwerte` minimale und maximale Werte für Lag_T übergeben werden (`.LagMin` und `.LagMax`). Sie werden in der `Parameter_Fcn`, siehe Seite 23, gespeichert und so bei einer Parameteridentifikation berücksichtigt. Die Werte stammen üblicherweise aus direkten Beobachtungen oder einer Messdatenanalyse wie in [32].

Der Modellstrukturgenerator nutzt die Angaben bezüglich einer Lag-Phase für alle Modellkandidaten einer Modellfamilie. Nur wenn die `StrDef` gleichzeitig mehrere Modellfamilien beschreibt, können für die einzelnen Modellfamilien unterschiedliche Einstellungen vorgenommen werden. Eine zukünftige Erweiterung des Modellstrukturgenerators kann jedoch eine versuchsabhängige Implementierung der Parameter der Lag-Phase beinhalten.

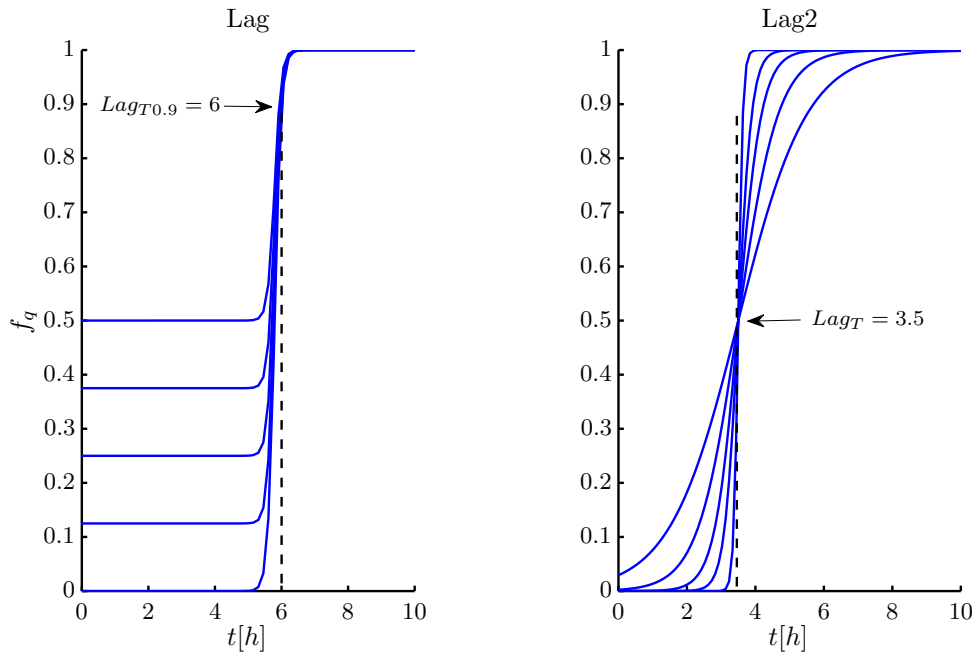


Abbildung 4.4: Verschiedene Ausprägungen beider Lag-Phasen-Formalkinetiken

Lebenserhaltung

Auch wenn Mikroorganismen nichts produzieren und sich auch nicht vermehren, benötigen sie für den lebenserhaltenden Stoffwechsel eine Kohlenstoff-Quelle[12]. Der Verbrauch ist proportional zur Biomasse (m_X).

Da die lebenserhaltende Stoffwechselreaktion im Normalfall immer abläuft, kann durch den Modellstrukturgenerator ein vorbereiteter Baustein in das Modell integriert werden. Dafür wird automatisch nach einer C-Quelle (xTyp: `SC`) gesucht, um die Differentialgleichung für einen entsprechenden Zustand m_C um einen zusätzlichen Reaktionsterm zu erweitern. Durch diesen wird das Substrat m_C verbraucht, aber kein Produkt erzeugt:

$$\dot{m}_C = \cdots - Y_{Erh} \cdot m_X \tag{4.21}$$

Allerdings wird darin noch nicht die Einschränkung berücksichtigt, dass das Substrat nur verbraucht werden kann, wenn es auch vorhanden ist. Deshalb wird die Gleichung 4.21 um einen weiteren Term ergänzt, eine Michaelis-Menten-Kinetik:

$$\dot{m}_C = \cdots - Y_{Erh} \cdot \frac{c_C}{0.01 + c_C} \cdot m_X \tag{4.22}$$

[12] Wie bisher werden hier nur die flüssigen Zucker- und Mineralsalz-Substrate betrachtet.

Der üblicherweise verwendete Parameter k_M wird durch einen konstanten, kleinen Wert (0.01) ersetzt, so dass die Kinetik sehr steil ist und wie ein Schalter wirkt, aber noch keine numerischen Probleme bei der Differentiation hervorruft, wie es ein echter Schalter mit unstetigem Verlauf täte. Auf die Einführung eines Parameters anstelle der Konstante wurde vorerst verzichtet, da zu seiner Identifikation Messungen in einem derart niedrigen Konzentrationsbereich notwendig wären, die in sehr vielen Anwendungen nicht vorgenommen werden können.

Ob der Erhaltungsstoffwechsel einen prozess-signifikanten Einfluss auf den Verbrauch des Substrats besitzt, und er damit in das Modell aufgenommen werden muss, hängt einerseits vom Prozess und andererseits vom Mikroorganismus ab. Der Modellstrukturgenerator lässt dem Benutzer die Wahlfreiheit durch das Feld `.Erhaltung` in der Strukturdefinition. Es enthält nur ein boolesches Element pro Modellfamilie. Ist der Wert WAHR, wird der erläuterte Modellbaustein automatisch in das Modell einfügt.

Diauxie-Phänomen

Organismen benötigen viele verschiedene Substanzen. Phosphate, Stickstoff- und Kohlenstoffverbindungen gehören dabei zu den nicht-gasförmigen Stoffen, die in größeren Mengen benötigt werden. In den hier betrachteten Fed-Batch-Ansätzen mit chemisch-definierten Minimalmedien wird standardmäßig jeweils nur eine dieser Verbindungen als Feeding verwendet. Im Laufe der Evolution haben sich allerdings viele Stoffwechselwege entwickelt, um verschiedene chemische Verbindungen dieser drei Grundsubstanzen aufnehmen zu können. Stehen einem Mikroorganismus beispielsweise zwei verschiedene Stickstoff-Verbindungen zur Verfügung, können häufig beide verstoffwechselt werden, allerdings meist nicht gleichzeitig. Dabei tritt ein Verzögerungseffekt auf, der im folgenden erläutert wird.

Aus der Entwicklungsgeschichte eines Organismus, aber auch aus energetischen Gründen, gibt es meist eine Bevorzugung bestimmter Verbindungen. So werden viele Organismen beispielsweise erst Glucose verstoffwechseln, bevor längerkettige Zucker aufgenommen werden. Dieses Phänomen ist auch als Katabolitrepression bekannt [8]: Vereinfacht kann man davon ausgehen, dass zunächst die Enzyme vorhanden sind, die zur Glucose-Aufnahme notwendig sind. Erst, wenn kaum noch Glucose vorhanden ist, werden weitere Enzyme gebildet, die Oligo- und Polysaccharide aufspalten können. Diese Umstellung benötigt eine messbare Zeitspanne, in der die Zellen praktisch keine Nährstoffe außer für den Erhaltungsstoffwechsel zur Verfügung haben. Die resultierende temporäre Verlang-

samung (oder sogar Aussetzung) des Biomassenwachstums während einer Fermentation wird als Diauxie bezeichnet.

Zur Beschreibung von Fermentationen, in denen Mikroorganismen mehrere alternative chemische Verbindungen verstoffwechseln können, sind Modellergänzungen notwendig. Obwohl für die im Rahmen dieser Arbeit betrachteten Prozesse keine Diauxie-Phänomen zu modellieren war, ist ein Diauxie-Baustein fertig implementiert, der durch den booleschen Wert WAHR in `StrDef.Diauxie` aktiviert werden kann. Dafür wird in den Zeichenketten der xTypen der Modellzustände nach Ziffern gesucht, die primäre, sekundäre, tertiäre, etc. Substratquellen definieren (siehe Abschnitt 2.2, „# der Substratquelle“ in Tabelle 2.1). Es gibt mehrere Möglichkeiten, den Übergang von einer primären zu einer sekundären Quelle zu modellieren, im Folgenden wird die implementierte Grundvariante vorgestellt.

Für das Wachstum der Biomasse X auf zwei alternativen Substraten S_1 (Primärquelle) und S_2 (Sekundärquelle) werden zwei zusätzliche Reaktionen definiert, r_1 und r_2:

$$S_1 \xrightarrow{r_1} X \tag{4.23}$$

$$S_2 \xrightarrow{r_2} X \tag{4.24}$$

Die beiden Reaktionen sollen durch die Konzentration des jeweiligen Edukts (c_{S_1} bzw. c_{S_2}) und durch ein virtuelles Enzym[13] reguliert werden. Das Enzym kann in einer aktivierten (EnzA) und in einer inaktivierten (Enz) Form vorliegen, beide Formen werden in das Modell als Zustand eingeführt. Die Zustände sind miteinander verkoppelt, so dass Folgendes gilt:

$$\begin{aligned} \mathrm{Enz} + \mathrm{EnzA} &= 1 \\ \mathrm{Enz} &\in [0,1] \\ \mathrm{EnzA} &\in [0,1] \end{aligned} \tag{4.25}$$

Das inaktive Enzym reguliert die Reaktion r_1 und in der aktivierten Form die Reaktion r_2. Beide Einflüsse wirken jeweils limitierend, das heißt, je mehr die aktivierte Form des Enzyms vorliegt, desto schneller läuft die Reaktion r_2 ab und umgekehrt.

$$r_1 = r_1(c_{S_1}, \mathrm{Enz}) \tag{4.26}$$

$$r_2 = r_2(c_{S_2}, \mathrm{EnzA}) \tag{4.27}$$

[13] Die Benennung *virtuelles Enzym* leitet sich von ähnlich ablaufenden enzymatischen Funktionen ab, im Modell handelt es sich tatsächlich nur um Rechengrößen.

Für die Umwandlung des Enzyms von der einen in die andere Form werden zwei neue Reaktionen, r_3 und r_4, eingeführt:

$$\mathrm{Enz} \underset{r_4}{\overset{r_3}{\rightleftharpoons}} \mathrm{EnzA} \tag{4.28}$$

Um die Bedingungen 4.25 zu erfüllen, sind die Geschwindigkeiten der beiden Umformungsreaktionen jeweils von ihrer „Eduktkonzentration“ und von der Konzentration des Primärsubstrats c_{S_1} abhängig:

$$r_3 = r_3(\mathrm{Enz}, c_{S_1}) \tag{4.29}$$

$$r_4 = r_4(\mathrm{EnzA}, c_{S_1}) \tag{4.30}$$

Die Erweiterung des Differentialgleichungssystems sieht wie folgt aus:

$$\dot{m}_X = \cdots + (r_1 + r_2) \cdot V \tag{4.31}$$

$$\dot{m}_{S_1} = \cdots - Y_1 \cdot r_1 \cdot V \tag{4.32}$$

$$\dot{m}_{S_2} = \cdots - Y_2 \cdot r_2 \cdot V \tag{4.33}$$

$$\dot{\mathrm{Enz}} = -r_3 + r_4 \tag{4.34}$$

$$\dot{\mathrm{EnzA}} = r_3 - r_4 \tag{4.35}$$

Für die beiden neuen Differentialgleichungen werden, je nach Situation, beispielsweise die folgenden Anfangswerte festgelegt:

$$\mathrm{Enz}(t = 0) = 1 \tag{4.36}$$

$$\mathrm{EnzA}(t = 0) = 0 \tag{4.37}$$

Abschließend sei erwähnt, dass Medien mit mehrfachen Substratquellen eine zusätzliche Herausforderung für die Modellidentifikation darstellen. Dies gilt erstens, weil sich die Anzahl der zu identifizierenden Parameter erhöht. Zweitens müssen ausreichend viele und qualitativ gute, insbesondere bei geringen Konzentrationen genaue Messdaten vorliegen, damit die Veränderung des Stoffwechselwegs gut beobachtet werden kann.

4.2.12 Messdaten

Wenn Modellstrukturen durch ein Programm zur Messdatenanalyse vorgeschlagen werden, können die dafür verwendeten Messdaten ebenfalls in die automatisch erzeugte Strukturdefinition integriert werden. Es ist auch möglich, Messdaten manuell hinzuzufügen. Da der Modellstrukturgenerator die `StrDef` als Datei in einem Modellordner

abspeichert, hat dies den Vorteil, dass somit einerseits Messdaten archiviert werden und andererseits die Modellerzeugung später nachvollzogen werden kann. Bei einer Weiterentwicklung des Modellstrukturgenerators sollten vorhandene Messdaten automatisch in den Messdaten-Ordner des neu zu erstellenden Modellordners kopiert werden.

Die folgenden drei Abschnitte enthalten zusätzliche Informationen zur automatischen Modellerzeugung. Die dafür notwendigen Informationen sind jedoch nicht eigens in der Modellstrukturdefinition gespeichert.

4.2.13 Messgleichungen

Messgleichungen geben den Zusammenhang zwischen Zustandsgrößen $\underline{x}$ und Messgrößen $\underline{y}$ wieder. Standardmäßig wird angenommen, dass jeder Zustand prinzipiell messbar ist, daher werden normalerweise alle Messgleichungen automatisch erzeugt. Somit können auch zellinterne Zustände im ABC-System visualisiert werden, was unter anderem für eine biologisch fundierte Beurteilung eines Modells wichtig ist, insbesondere wenn keine echten Messdaten zur Verfügung stehen.

Es gibt jedoch Verfahren, bei denen sich falsche Angaben über die Messbarkeit von Zustandsgrößen störend auswirken, da bei ihnen die Messgleichungen analytisch ausgewertet werden. Zum Beispiel ist dies der Fall bei der Überprüfung der Beobachtbarkeit eines System oder bei einer Methode zur Bestimmung der strukturellen Identifizierbarkeit [74]. Bei letzterer können die erhaltenen Ergebnisse fehlerhaft sein, wenn eine Messgröße in Wirklichkeit nicht messbar ist. Daher kann über die boolesche Feldvariable `.Zustand.gemessen` für jeden Zustand festgelegt werden, ob eine Messgleichung erzeugt werden soll.

Für die automatische Erzeugung der Messgleichungen wertet der Modellstrukturgenerator die physikalische Einheit und den xTyp der einzelnen Zustände aus, da drei Fälle zu unterscheiden sind, wobei die Automatik für weitere Fälle leicht erweiterbar ist:

1. Für massebehaftete, extrazelluläre Zustände, werden Gleichungen wie zum Beispiel $c_{Am} = m_{Am}/V$ erstellt.

2. Bei massebehafteten, zellinternen Zuständen ist für die Regulation in der Regel die zellinterne Konzentration relevant, daher werden Gleichungen nach dem Schema $g_{\mathrm{DNS}} = m_{\mathrm{DNS}}/V_x$ erstellt. Hierin ist m_{DNS} das Zellkompartiment mit der DNS-Masse und V_x das Gesamtzellvolumen. Letzteres wird – unter der Annahme,

dass die Zelldichte gleich eins ist – durch die Zellmasse ersetzt beziehungsweise automatisch aus der Summe der einzelnen Kompartimentzuständen berechnet. Zusätzlich werden für den Vergleich mit typischen Messdaten auch die Gleichungen für die c-Konzentrationen erzeugt.

3. Für massefreie Zustände wie beispielsweise dem pH-Wert wird angenommen, dass sie direkt messbar sind. Die Messgleichung lautet für diesen Fall daher $y_{pH} = x_{pH}$.

Der Modellstrukturgenerator erstellt die Datei `Symb_h`[14], in der die Messgleichungen enthalten sind. Wird in den obigen Beispiel die Glucose-Konzentration bestimmt, resultiert daraus das folgende Code-Fragment der `Symb_h`-Datei. Die Datei enthält automatisch einige hilfreiche Namensersetzungen, um die Lesbarkeit zu erhöhen:

Symb_h (Ausschnitt)

```
m_Glc       =  x(1);
...
V           =  x(end);

c_Glc      =  m_Glc / V;
...

...
y(1) = c_Glc;
```

Schließlich können manuell nach der automatischen Erstellung der Modelldateien mit wenig Aufwand noch weitere „virtuelle Messgrößen“ hinzugefügt werden. Das Verhalten einzelner Teile einer Differentialgleichung, zum Beispiel von Formalkinetiken, kann auf diese Weise visualisiert werden. Dies stellt ein wichtiges Hilfsmittel während der Modellentwicklung dar, da so das dynamische Verhalten einer Kinetik im Prozessverlauf analysiert werden kann. Für die virtuellen Messgrößen gelten jedoch ebenfalls die obigen Anmerkungen bezüglich der falschen Angabe über ihre Messbarkeit.

4.2.14 Parameter

Abgesehen von den fest vorgegebenen Parametern, wie zum Beispiel den Ausbeutekoeffizienten in der K-Matrix, erzeugt der Modellstrukturgenerator die meisten Parameter

[14] beziehungsweise `Symb_h_1`, wenn mehr als ein Modellkandidat erstellt wird

selbst. Für folgende Parameter gibt es die Möglichkeit, in der `StrDef` Startwerte oder gültige Wertebereiche anzugeben: Ausbeutekoeffizienten Y, maximale Reaktionsraten $\mu_{\max}$, Parameter einer eventuell vorhandenen Lag-Kinetik oder X_0-Parameter. Für die übrigen Kinetikparameter werden gültige Wertebereiche und auch Anfangswerte direkt aus der Kinetikbibliothek entnommen.

Generell müssen für alle Parameter, die in einem Modell definiert werden, Anfangswerte festgelegt werden. Es wäre wünschenswert, dass die Zahlenwerte bereits so gewählt werden, dass mit ihnen das Modell vorhandene Messdaten gut beschreiben kann. In vielen Fällen ist dies nicht möglich, so dass erst durch eine Parameteridentifikation entsprechende Parameterwerte gefunden werden. Aber auch dies wird umso einfacher, je besser die gewählten Startwerte sind[15].

Durch die Angabe von Grenzen kann der Suchraum während der Parameteroptimierung eingeschränkt und damit die Identifikation beschleunigt werden. Tatsächlich benötigen viele Optimierungsverfahren Grenzen für die Parameter. Im ABC-System wird daher verlangt, dass zu jedem Parameter prinzipiell ein minimaler und maximaler Grenzwert definiert wird, damit später ein beliebiges der bereit gestellten Optimierungsverfahren eingesetzt werden kann.

Alle dazu gesammelten Informationen werden in der Datei vom Typ `Parameter_Fcn` gespeichert, siehe Anhang A.1.1. Für jeden Parameter wird darin unter anderem festgelegt,

- ob es sich um einen variablen Parameter in der Parameteridentifikation handelt,
- welchen Beschränkungen er unterliegt,
- ob es ein X_0-Parameter ist,
- mit welcher Farbe er in GUIs markiert werden soll und
- welcher Anfangswert er hat.

Neue Parameterwerte, die durch eine Parameteridentifikation ermittelt werden, können auf Wunsch automatisch an das Ende der Datei `Parameter_fcn` geschrieben werden. Matlab verwendet nur die jeweils letzten Einträge. Auf diese Weise werden ältere Ergebnisse archiviert. Der Benutzer kann so zu Vergleichszwecken den Entwicklungsverlauf der Parameterwerte nach jeweils neuen Experimenten oder auch unter verschiedenen Bedingungen für die Parameteridentifikation einsehen. Da Matlab-Skript-Dateien in einem

[15] Der Abschnitt 4.3 zeigt ein Verfahren mit dessen Hilfe sinnvolle Startwerte bestimmt werden können.

normalen Texteditor bearbeitet werden können, ist es auch möglich, auf einen älteren Stand zu wechseln, falls das aktuelle Resultat weniger zufriedenstellend ist. Dieser Fall kann beispielsweise durch die Wahl ungünstigerer Startwerte eintreten.

Für die Anfangswerte von X_0-Parametern für Kompartimentzustände können optional auch prozentuale Angaben gemacht werden. Diese Werte beziehen sich auf die Gesamtbiomasse. Daher muss in `StrDef.Messdaten` in diesem Fall auch die Angabe einer Startbiomasse enthalten sein, damit die Prozentwerte intern automatisch in Massen umgerechnet werden können. Bei Nutzung dieser Option enthält das Variablenfeld in `StrDef` anstelle einer Zahl eine Zeichenkette, die aus einer Zahl und einem Prozentzeichen besteht. Aktuell wird diese Eingabemöglichkeit von einem grafischen Tool zur Modellbildung genutzt.

4.2.15 Probennahmen

Es gibt vier Probennahmetypen, die im ABC-System auftreten: Voll- oder Überstandsprobe, jeweils manuell und automatisch. In der Datei `SystemInit` werden neben dem Typ auch weitere Informationen abgelegt: Name und Einheit.

Die Informationen werden einerseits für die korrekte Berechnung des Volumens benötigt, und andererseits für die diskreten Zustandsänderungen, siehe Abschnitt 2.4.

4.3 Anfangswerte und Grenzen von Parametern

In den vorangegangenen Abschnitten wurde erläutert, wie Parameter automatisch generiert werden und wie ein Modell bezüglich der Parameter konfiguriert wird. Dieser Abschnitt konzentriert sich darauf, wie ihre Startwerte für das Modell ermittelt und Gültigkeitsbereiche für eine Parameteridentifikation festgelegt werden.

4.3.1 Übernahme aus anderen Modellen und Experimenten

Wenn die Versuchsbedingungen vergleichbar sind, können Wertebereiche oder sogar konkrete Werte der Literatur entnommen oder in eigenen Vorversuchen näherungsweise bestimmt werden. Typische Parameterwerte, die vor einer Modellbildung zur Verfügung stehen können, sind zum Beispiel maximale Wachstumsraten, Ausbeutekoeffizienten oder die Dauer einer Lag-Phase. Aber selbst die Übernahme der Lag-Phasendauer ist ohne Einschränkungen nur möglich, wenn sämtliche Fermentationsvorbereitungen (Vorkultu-

ren, Lagerung, etc.) vergleichbar mit den geplanten Experimenten sind. In der Praxis ist es oft nicht möglich, alle Details zu beachten. Für die übrigen genannten Parameter muss zusätzlich auch das mathematische Umfeld vergleichbar sein, wie im Folgenden gezeigt wird: Beispielsweise ist der Wert einer maximalen Wachstumsrate $\mu_{\max}$ von den verwendeten Formalkinetiken abhängig. Das Situation wird anhand der beiden folgenden, einfachen Modellvarianten für die spezifische Wachstumsrate μ einer Reaktion kurz aufgezeigt:

$$\mu_{\text{alt}} = \mu_{\text{alt,max}} \cdot \text{MiMe}(c_j) \qquad \text{altes Modell} \tag{4.38}$$

$$\mu_{\text{neu}} = \mu_{\text{neu,max}} \cdot \text{Haldane}(c_j) \qquad \text{neues Modell} \tag{4.39}$$

Für dieses veranschaulichende Beispiel wird angenommen, dass das alte Modell bereits identifiziert wurde und für das neue Modell noch ein geeigneter Startwert für $\mu_{\text{neu,max}}$ gesucht wird. Für die Parameter der Haldane-Kinetik wurden ebenfalls bereits Werte ermittelt, und zwar hier so, dass der Wertebereich der Kinetik zwischen null und 0.5 liegt. Da im Falle der MiMe-Kinetik der Wertebereich unabhängig von ihrem Parameter zwischen null und eins liegt, muss der Wert für $\mu_{\text{neu,max}}$ daher in erster Näherung $\mu_{\text{neu,max}} = 2 \cdot \mu_{\text{alt,max}}$ sein.

Zu beachten ist, dass sich auf diese Weise nur eine grobe Anpassung durchführen lässt: Durch die im Experiment aufgetretenen Konzentrationen c_j kann es möglich sein, dass nur ein kleiner Bereich des theoretischen Wertebereichs relevant ist[16]. Entscheidend ist der maximale Wert, der in dem jeweiligen Versuch erreicht wurde. Die oben angegebene Anpassung ist unter Umständen weniger erfolgreich und kann verbessert werden durch:

$$\mu_{\text{neu,max}} = \mu_{\text{alt,max}} \cdot \frac{\max(\mu_{\text{alt}})}{\max(\mu_{\text{neu}})} \tag{4.40}$$

Darin sind $\max(\mu_{\text{alt}})$ und $\max(\mu_{\text{neu}})$ die maximalen, tatsächlich erreichten Werte der Kinetik(gruppe). Allerdings ist es in der Praxis nicht immer möglich, zuverlässige Werte für die Wachstumsrate durch Messungen zu erlangen.

Im Zusammenhang mit Multimodellen tritt das Problem der Anfangswerte für Parameter häufig auf. Abschnitt 5.2.1 wird sich nochmals mit diesem Thema beschäftigen und stellt ein Verfahren zur Startwertberechnung auch für andere Parameterwerte vor.

[16] Der beste Weg, Informationen für eine gute Anpassung zu erzielen, besteht darin, in Experimenten den gesamten prozessrelevanten Konzentrationsbereich zu beproben.

4.3.2 Kinetikparametergrenzen

Es gibt häufig Situationen, in denen weder eigene noch fremde Quellen genutzt werden können, um sinnvolle Startwerte für die Modellparameter zu finden. Dies betrifft insbesondere die Kinetikparameter, die oft den Großteil der Modellparameter ausmachen. Selbst die Größenordnung der Werte ist nur selten bekannt, so dass beliebige Werte ausgewählt werden könnten. Wie bereits erwähnt, ist aber das Ergebnis einer Parameteridentifikation und ihre Laufzeit sehr von den Anfangswerten abhängig. In diesem Abschnitt wird gezeigt, wie die Situation für die Kinetikparameter verbessert werden kann.

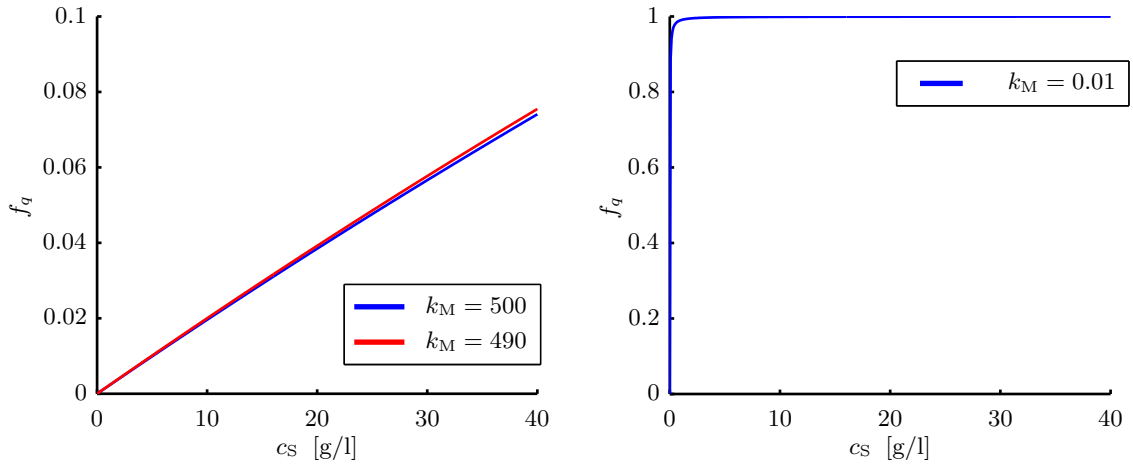
Auch wenn keine guten Anfangswerte für Kinetikparameter bekannt sind, kann das Optimierungsproblem der Parameteridentifikation vereinfacht werden, indem der Parametersuchraum eingegrenzt wird. Ohne Kenntnis eines Modells kann hierzu die folgende Überlegung angestellt werden. Es sollten nur jene Parameterwerte zulässig sein, die bei kleinen Änderung ihres Werts nicht zu numerischen Problemen bei der Berechnung des Gradienten des Funktionswerts (der Kinetik) führen. Hierbei gibt es zwei Fälle zu unterscheiden, die anschließend anhand von Beispielen näher erläutert werden:

- Ein zu kleiner Gradient wird durch numerisches Rauschen schnell beeinflusst und senkt die Effizienz des Optimierungsverfahrens. Außerdem führt er unter Umständen zu einem vorzeitigen Abbruch der Optimierung.

- Ein zu großer Gradient führt zu vielen kleinen Schritten des numerischen Integrators. Dies verlängert die benötigte Rechendauer und birgt die Gefahr von numerischen Ungenauigkeiten.

In Abbildung 4.5(a) sind zwei Michaelis-Menten-Kinetiken mit ähnlichen, im Vergleich zum Konzentrationsbereich von c_S sehr großen Parameterwerten dargestellt, wodurch sich praktisch ununterscheidbare und nahezu lineare Verläufe ergeben. Es handelt sich um ein zur besseren Veranschaulichung stark übertriebenes Beispiel, denn wenn in Experimenten die Substratkonzentration immer im Bereich $c_{Glc} \leq 40\ \mathrm{g\,l^{-1}}$ liegt, ergibt es aus Gründen der Identifizierbarkeit wenig Sinn, eine MiMe-Kinetik zu verwenden, deren k_M-Wert zehnfach höher als die maximal beobachtete Konzentration ist. Bei einer numerischen Gradientenbestimmung mittels Differenzenquotient wird der Parameterwert typischerweise nur um sehr kleine Werte verändert. Auch der Funktionswert ändert sich dementsprechend wenig. Die Differenz unterschreitet eventuell sogar die Größenordnung

des numerischen Rauschens ($\approx 10^{-8}$), wodurch sich die Genauigkeit des berechneten Gradienten deutlich verschlechtert.



(a) Zwei Verläufe mit $k_M = 490$ bzw. $500\,\mathrm{g\,l^{-1}}$

(b) $k_M = 0.01\,\mathrm{g\,l^{-1}}$

Abbildung 4.5: Verschiedene Ausprägungen der MiMe-Kinetik $f_q = \frac{c_S}{c_S + k_M}$

Abbildung 4.5(b) zeigt das andere Extrem: Der Funktionswert erreicht schon bei sehr geringen Konzentrationen (verglichen mit dem – hier angenommenen – maximalen Wert von $40\,\mathrm{g\,l^{-1}}$) seinen Endwert. Bereits winzige Änderung in diesem Konzentrationsbereich führen zu signifikanten Wertänderungen. Dies zwingt die Schrittweitensteuerung des numerischen Integrators zu sehr kleinen Schritten, und die wiederum können zu erhöhten numerischen Fehlern führen. Auch in diesem Fall ist keine Identifizierbarkeit des Kinetikparameters gegeben.

Die Beschränkung des Gradientenwerts kann durch die Festlegung von maximalen und minimalen Grenzwerten für die zu bestimmenden Parameter gewährleistet werden. Die zulässigen Wertebereiche für alle Parameter der in der Kinetikbibliothek enthaltenen Kinetiken wurden nach intensiven Studien in den Programmen der einzelnen Kinetiken festgelegt.

Eine automatische, dynamische Anpassung der Parametergrenzen an die jeweilige Situation ist in zukünftigen Arbeiten angedacht, denn die Festlegung der Grenzbereiche ist stark von dem Konzentrationsbereich abhängig, der in dem Prozess erreicht wird: Beispielsweise sind die Grenzwerte für den Parameter k_M in der MiMe-Kinetik für Glucose mit typischen Konzentrationen von 0 bis $40\,\mathrm{g\,l^{-1}}$ anders zu wählen als für zellinterne

Konzentrationen, die zwischen 0 und $1\,\mathrm{g\,g^{-1}}$ liegen. Die in der Kinetikbibliothek hinterlegten Grenzwerte gelten daher aktuell für beide Situationen. Für den Parameter der MiMe-Kinetik gilt beispielsweise[17]: $10^{-4} \leq k_\mathrm{M} \leq 50$

Generell ist zu prüfen, ob in den Experimentaldaten Messwerte aus dem Konzentrationsbereich enthalten sind, der maßgebend für den ermittelten Parameterwert ist. Für den dargestellten Verlauf in Abbildung 4.5(b) sind beispielsweise Messungen aus dem Bereich 0 bis $1\,\mathrm{g\,l^{-1}}$ wichtig. Fehlen entsprechende Messdaten, kann das Erreichen der Grenzwerte zu hilfreichen Schlussfolgerungen für die Modellbildung führen, wie am folgenden Beispiel der MiMe-Kinetik dargestellt wird:

1. Wenn bei der Optimierung die maximale Grenze für den Kinetikparameter k_M erreicht wird, kann es möglich sein, dass eine Vergrößerung des Wertes für k_M zu einer noch bessere Anpassung an die Messdaten führt. Dadurch sänke der Funktionswert f_q allerdings weiter und somit muss eine Anhebung des μ_max-Parameters erfolgen. Es kann nun geprüft werden, ob eine bessere Anpassung erreichbar ist, wenn die fragliche Reaktion ganz aus dem Modell entfernt wird. Außerdem ist in Abbildung 4.5(a) zu erkennen, dass der Funktionsverlauf im dargestellten Bereich praktisch linear ist. Auch dies kann ein Hinweis auf einen strukturellen Modellfehler sein.

2. Wird für k_M in einer Parameteridentifikation die untere Grenze erreicht, kann auch dies zu einer Überprüfung der Modellstruktur genutzt werden, denn die Kinetik ist praktisch gesehen immer zu 100 % aktiv. Die Reaktion ist für die Beschreibung der Messdaten offenbar erforderlich, es scheint aber keine Abhängigkeit von der betrachteten Konzentration notwendig zu sein. Entweder läuft die entsprechende Reaktion tatsächlich immer ab, in diesem Fall könnte das Modell vereinfacht werden, oder vielleicht ist eine andere Konzentration als regulierende Größe besser geeignet.

Die Schlussfolgerungen sind von der jeweiligen Kinetik abhängig. Eine generelle Strukturuntersuchung kann anhand eines Modellkandidaten mit der MiMe-Kinetik bereits durchgeführt werden. Für eine automatische Analyse sind die Schlussfolgerungen für die verschiedenen Kinetiken noch zu überprüfen. Eine zukünftige Integration dieser Funktion, zusammen mit einem automatischen Vorschlag für eine Modellstrukturverbesserung, ist sinnvoll.

[17] Die physikalische Einheit des Parameters ist abhängig von der Einheit der Konzentration.

4.4 Kinetikbibliothek

Wie in Abschnitt 4.1 bereits dargelegt, ist es sinnvoll, viele verschiedene Formalkinetiken zu prüfen, um ein Modell an Messdaten anpassen zu können. Um nicht dem Anwender die Kenntnis über geeignete Formalkinetiken abzuverlangen, sind aktuell rund 50 verschiedene Kinetiken in der Kinetikbibliothek des ABC-Systems abgelegt, siehe Tabelle 4.2 . Dabei handelt es sich jeweils um eigenständige Matlab-Programme, die nach einem einheitlichen Aufruf- und Rückgabewert-Muster programmiert und in einem speziellen Ordner im ABC-System zusammengefasst sind. Die Programme können jeweils vier verschiedene Funktionen erfüllen. Die Art des Programmaufrufs entscheidet über die konkrete Funktionsausführung, Details werden in Anhang A.3 beschrieben:

1. Wird ein Kinetik-Programm, beispielsweise die Haldane-Kinetik

   $f_q = c/(k_\mathrm{M} + c + \frac{c^2}{k_\mathrm{M}+k_\mathrm{I}})$,

   in Matlab einfach mit `Haldane('')` aufgerufen, erzeugt dies einen grafischen Überblick über typische Funktionsverläufe anhand einer Reihe von Standardparameterwerten, die in der Kinetik selbst gespeichert sind. Ein Beispiel dafür zeigt Abbildung 4.6.

2. Dem Programm können beim Aufruf auch konkrete Werte für eine bestimmte Konzentration c und den Parameterwerten $(k_\mathrm{M}, k_\mathrm{I})$ übergeben werden. Durch zum Beispiel `Haldane(5,[1 10])` wird der Wert für f_q direkt ausgerechnet, mit $c = 5, k_\mathrm{M} = 1$ und $k_\mathrm{I} = 10$ (jeweils in $\mathrm{g\,l^{-1}}$). Diese Variante wird bei der Simulation von Modellen üblicherweise verwendet. Neben der Rückgabe des Zahlenwerts für f_q ist optional eine grafische Ausgabe möglich. In der Grafik werden dann die f_q-Verläufe bis zum angegebenen Konzentrationswert (hier $5\,\mathrm{g\,l^{-1}}$) dargestellt. Dies ist für eine Modellanalyse sowohl während der Modell- als auch der Prozessentwicklung hilfreich.

3. Für das automatische Kompilieren eines Modells werden auch Hilfsdateien mit analytisch bestimmten Ableitungen benötigt. Das symbolische Rechnen ist eine spezielle Fähigkeit von Matlab, die hierfür genutzt wird. Zu diesem Zweck muss die Formel für die Kinetik in einer symbolischen Form vorliegen. Die Rückgabe dieser Aufrufvariante ist daher eine ableitbare Funktionsgleichung der Kinetik.

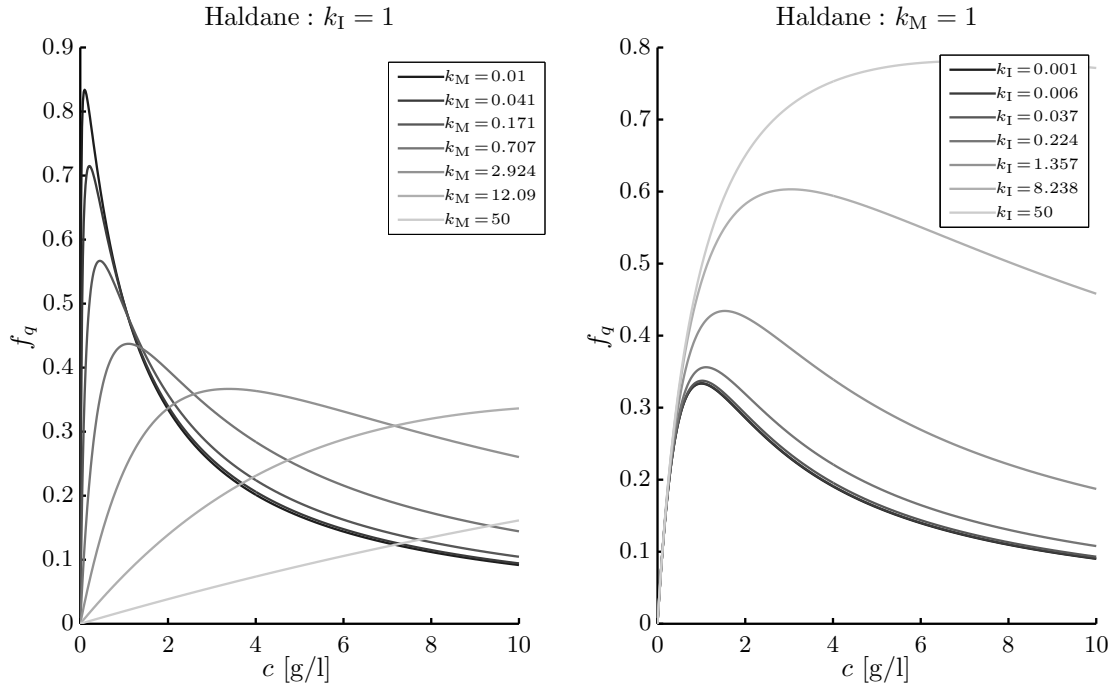


Abbildung 4.6: Darstellung verschiedener Formen der Haldane-Kinetik

4. Durch eine spezielle Aufrufvariante von Matlab können Details über die Kinetik abgerufen werden. Informationen wie den Kinetiktyp, Parameteranzahl und -namen, typische Anfangswerte und zulässige Grenzwerte werden vom Modellstrukturgenerator bei der automatischen Modellerzeugung verwendet. Weitere Informationen wie zum Beispiel über die Art der typischerweise verwendeten Einflussgröße und wie diese auf die Reaktionsgeschwindigkeit einwirkt, sind bereits gespeichert, werden aber erst zukünftig für die automatische Modellierung genutzt. Detaillierte Informationen hierzu gibt das in Anhang A.3 aufgeführte vollständige Programmlisting der Michaelis-Menten-Kinetik.

Der hauptsächliche Nutzen der Kinetikbibliothek besteht darin, dass durch die Standardisierung der Kinetik-Programme die automatische Modellbildung ermöglicht wird. Zu diesem Zweck werden die Formalkinetiken zunächst nach Typen eingeteilt, die ähnliche Verläufe kategorisieren. Beispielsweise vereint der Typ `1` die limitierenden, einfachen Formalkinetiken (mit einem Parameter), Typ `-1` die inhibierenden, einfachen Kinetiken und Typ `2` die limitierenden Kinetiken mit zwei Parametern. Da viele der komplexeren Kinetiken kein eindeutig limitierendes oder inhibierendes Verhalten zeigen, gilt die Ver-

einfachung, dass die mathematischen Funktionen umso komplexer werden, je größer die absolute Typnummer ist. Jedoch brechen viele Sonderfälle mit diesem Schema. Einige Kinetiken werden beispielsweise ausschließlich zur Beschreibung der Reaktionsgeschwindigkeit in Abhängigkeit von Temperatur oder pH-Wert anstatt einer Konzentration eingesetzt. Eine Ergänzung der Kinetikbibliothek erfolgt laufend und ist jederzeit möglich.

Für die automatisierte, aber auch manuelle Abfrage der vorhandenen Kinetik-Programme wird das Programm `Kinetikbibliothek` verwendet. Der Aufruf ohne Parameter listet alle Formalkinetiken auf, die zur Verfügung stehen, zusammen mit dem jeweiligen Typ. Ein Aufruf mit einer Zahl, beispielsweise `Kinetikbibliothek(1)` listet alle Formalkinetiken des Typs `1` auf. Diese Aufrufvariante wird speziell von der automatischen Modellbildung genutzt: Entsprechend der Einträge in der R-Matrix liefert die Kinetikbibliothek eine Liste von Kinetiknamen, die dem jeweiligen Typ angehören. Der Modellstrukturgenerator bildet anschließend alle Kinetik-Kombinationen und erzeugt die einzelnen Modellkandidaten mit den jeweils dazu gehörenden Kinetikparametern und Anfangswerten. Die Parameternamen und Anfangswerte werden automatisch durch die vierte Aufrufvariante der einzelnen Kinetiken (siehe oben) ermittelt.

Der Modellstrukturgenerator und die Kinetikbibliothek kennen auch spezielle Typen: Typ `0` vereint beispielsweise die Menge der Kinetiken vom Typ `1` und `-1`. Der Typ `1a` vereint die Menge des Typs `1` und der „One“-Kinetik. Weitere Sonderfälle sind einfach zu ergänzen.

Kinetikname	**Typ**	**Formel** $f_q =$
Typ 1 und Typ -1	0	
Typ 1 und One	1a	
Luong[18]	-100	$\max(0, 1 - (c/k_i)^{c_{\max}})$
InhibMoser	-4	$1 - (c^\lambda/(k + c^\lambda))$
Aiba	-1	$\exp(-k \cdot c)$
Jerusaliwski	-1	$k/(k + c)$
Ming	1	$c^2/(c^2 + k)$
MiMe	1	$c/(c + k_{\mathrm{M}})$

[18] Anmerkung: Die dargestellte Luong-Kinetik ist durch die Maximum-Funktion nicht symbolisch verarbeitbar. Die Funktion ist jedoch ersetzbar durch:

$$\max(a, b) = \frac{a + b + \sqrt{(a - b)^2}}{2} \tag{4.41}$$

Tessier	1	$1 - \exp(-\frac{c}{k})$
Ratkowsky	2	$k_1 \cdot (T - T_{\min}) \cdot (1 - \exp(k_2 \cdot (T - T_{\max})))^2$
Haldane	2	$c/(k_{\mathrm{M}} + c + \frac{c^2}{k_{\mathrm{M}} + k_{\mathrm{I}}})$
Weibull	2	$k \cdot \lambda \cdot (\lambda \cdot c)^{k-1} \cdot \exp(-(\lambda \cdot c)^k)$
Edwards	2	$\exp(-c/k_i) - \exp(-c/k_s)$
Sokol_Howell1	2	$c/(k + c^2)$
Moser	2	$c^\lambda/(k + c^\lambda)$
TOptimum	2	$k_1 \cdot \exp(k_2 \cdot T - k_3 \cdot T^{(k_4 \cdot T)})$
MiMeJeru	2	$c/(c + k_m) \cdot k_j/(k_j + c)$
Rossner	2	$(1 + (k_i/k_m) \cdot (k_i - 1)^{(1-k_i)/k_i}) \cdot c/(1 + c + (c/k_m)^{k_i})$
Rossner1	2	$(1 + (2/k_m)) \cdot c/(1 + c + (c/k_m)^2)$
Jost	2	$c^2/((k_1 + c) \cdot (k_1 + k_2 + c))$
AibaSchalter	3	$\exp(-k \cdot c) \cdot c^4/(c^4 + 10^{-5})$
MiMeJeruSchalter	3	$\frac{1}{2} \cdot (c/(c + k_m) + k_j/(k_j + c)) \cdot c/(c + 10^{-3})$
Identity	11	c
Erhaltung	12	$c/(c + 0.01)$
Zero	13	0
pHGauss	40	$\exp(-0.5 \cdot (((pH - pH_0)/\sigma)^2))$
pHGaussExp	40	$1 - \left(1 - \exp(-\frac{1}{2} \cdot (\frac{pH - pH0}{\sigma})^2)\right)^k$
KonzSchalter	50	$(1 - k)/(1 + \exp(-5 \cdot (c - (c_{0.5} - \log(8)/10000)))^2) + k$
Lag	50	$(1 - k)/(1 + \exp(-10 \cdot (t - (T_{0.9} - \log(8)/10)))) + k$
Lag2	50	$1/(1 + \exp(-k \cdot (t - T)))$
One	99	1

Tabelle 4.2: Auszug aus der Kinetikbibliothek

5 Multimodell-Anwendungen

Few things are harder to put up with than a good example.

(Mark Twain (1835-1910))

In diesem Kapitel wird zunächst kurz erklärt, wie der Modellstrukturgenerator praktisch eingesetzt wird. Dabei geht es weniger um das Erzeugen der entsprechenden Dateien, als vielmehr um die Darstellung des typischen Arbeitsablaufs. Es wird zunächst dargestellt, wie die Größe der Modellfamilie durch Kombinatorik sehr schnell wachsen kann, weswegen der Modellstrukturgenerator trotz der Automatisierung gezielt eingesetzt werden sollte. Ein Vorschlag für einen ressourcenschonenden Arbeitsablauf wird vorgestellt. Die letzten drei Abschnitte zeigen den Einsatz der typischen Verfahren Parameteridentifikation und Trajektorienplanung sowie ein modelldiskriminierendes Planungsverfahren jeweils speziell für Multimodelle.

5.1 Zeitproblem / Kombinatorik

Im Kapitel 4 wurde mit der Modellstrukturdefinition eine kompakte Beschreibung zur Kodierung von Modellen vorgestellt, mit der durch einfache Erweiterungen und mit Hilfe des Modellstrukturgenerators anstelle eines einzelnen Modells ganze Modellfamilien für das ABC-System erzeugt werden können. Die Größe einer Modellfamilie kann sehr schnell mit einer steigenden Anzahl von Variationsmöglichkeiten wachsen, wie nachstehendes Rechenbeispiel demonstriert:

Die folgenden Varianten in einer Modellstruktur seien vorgegeben: An vier Stellen innerhalb eines beliebigen Differentialgleichungssystems werden anstelle einer festen Formalkinetik die Mitglieder des Kinetiktyps `1` eingesetzt. Der Kinetiktyp `2` wird an drei weiteren Stellen verwendet. Für den ersten Typ stehen vier Formalkinetiken zur Auswahl, für den zweiten fünf. Zusätzlich werden in fünf Reaktionen noch je zwei Alternativen für

die P-Faktoren definiert. Insgesamt ergibt sich bereits für dieses scheinbar überschaubare Beispiel eine Modellfamiliengröße von

$$4^4 \cdot 3^5 \cdot 5^2 = 1.555.200 \tag{5.1}$$

Modellkandidaten. Die Anzahl der Variationsmöglichkeiten im Beispiel ist keineswegs unrealistisch hoch. In strukturierten Modellen mittlerer Größe, beispielsweise mit 12 Modellzuständen und 10 Reaktionen, die jeweils von einer oder mehreren Größen reguliert werden, existieren – konservativ geschätzt – mindestens 15 Stellen mit Kinetiken im Differentialgleichungssystem, die unabhängig voneinander variiert werden können. Das ABC-System ist ursprünglich tatsächlich nicht für Multimodelle entwickelt worden, aber selbst unabhängig davon, stößt eine automatisierte Verarbeitung hier an ihre Grenzen, wie im Folgenden gezeigt wird.

1. Bei den verwendeten Windows-Betriebssystemen (WinXP und Win7, 64bit) kann beobachtet werden, dass die Performanz der Dateisystemverwaltung signifikant sinkt, wenn die Anzahl der Dateien in einem Ordner sehr groß wird. Pro Kandidat werden je nach Konfiguration 40 – 50 Dateien erzeugt, so dass schnell mehrere tausend Dateien in einem Modellordner abgelegt werden. Deutlich wird die Performanz-Einbuße beispielsweise beim Kompilieren der einzelnen Modellkandidaten. Während die ersten innerhalb von jeweils 20 – 30 Sekunden auf einem modernen PC erzeugt werden, benötigt der hundertste Modellkandidat bereits etwa die doppelte Zeit. Die angegebene Zeitdauer hängt natürlich sehr vom Betriebssystem, dem verwendeten Compiler, dem Speichermedium (Festplatte) und vom Modell ab, aber die Verlangsamung ist in verschiedenen Konfigurationen zu beobachten. Um diesen Performanz-Verlust durch das Betriebssystem zu begrenzen, werden die Programmdateien für die Kandidaten im `auto`-Ordner automatisch in Blöcke von jeweils 50 Kandidaten eingeteilt, die in jeweils eigenen Unterordnern gespeichert werden[1]. Die Anzahl der Dateien darin ist somit auf etwa 2500 begrenzt.

   Durch die gesammelten Erfahrungen wurde der Modellstrukturgenerator so programmiert, dass er automatisch eine vom Benutzer zu bestätigende Warnung ausgibt, wenn die Anzahl der zu erstellenden Modellkandidaten größer als 1.000 ist. Unabhängig vom verwendeten Compiler für die C-Programme sind für die Erstellung von 1.000 Modellen etliche Stunden einzuplanen. Überschreitet die Anzahl

[1] Die Verwaltung der Ordner übernimmt das ABC-System. Ein Benutzereingriff ist nicht notwendig

25.000, so bricht das Programm planmäßig mit einer Fehlermeldung ab und empfiehlt, mehrere kleinere Modellfamilien zu definieren. Diese harte Grenze wurde gewählt, da Matlab nach vielen Stunden des Modellerstellens mehrfach wegen fehlendem Arbeitsplatzspeicher abstürzte.

Als Strategie zur Modellentwicklung hat sich folgendes prinzipielles Vorgehen bewährt:

a) Auswahl geeigneter Modelltypen (unstrukturiert,...)
b) Festlegen einer Modellstruktur für jeden Modelltyp (Reaktionsschema)
c) Erzeugen einer kleinen Modellfamilie für jede Struktur (MiMe, Haldane, ...)
d) Parameteridentifikation für jeden Modellkandidaten
e) Vergleich der Einzelergebnisse
f) Auswahl der am besten geeigneten Struktur (limitierend, ...)
g) Erzeugen einer größeren Modellfamilie für diese Struktur (MiMe, Ming, ...)
h) Parameteridentifikation der gesamten Modellfamilie
i) Auswahl der besten Modellkandidaten
j) Planung eines neuen Versuchs

Durch diese Abfolge kann der Performanz-Verlust begrenzt werden. Außerdem können schnell Informationen über passende oder unpassende Modellstrukturen gewonnen werden.

2. Während das Erzeugen einer Modellfamilie ein einmaliger Vorgang ist, und daher die Zeiten zur Erstellung einer mittelgroßen Familie als noch akzeptabel eingestuft werden können, ist der zeitliche Aufwand für die Identifikation aller Parameter erheblich höher: Für die Größe der oben genannte Modellfamilie, siehe Gleichung 5.1, ergibt sich eine extrapolierte Rechenzeit in der Größenordnung von Jahren, selbst unter der optimistischen Annahme, dass eine einzelne Parameteridentifikation im Durchschnitt nur wenige Minuten dauert.

Modellfamilien sollten daher von vornherein so klein gehalten werden, dass erstens die beschriebene Performanz-Einbuße erträglich und zweitens die realistisch geschätzte Gesamtzeit für die Parameteridentifikationen sich in einem sinnvollen zeitlichen Rahmen bewegt. Andernfalls können die Modellfamilien nur stichprobenartig untersucht werden. Modellfamilien mit einer Größe bis etwa 1.000 Kandidaten haben sich in der Praxis als gut beherrschbar herausgestellt.

5.2 Multimodell-Parameteridentifikation

Auch bei kleineren Modellfamilien besteht ein Interesse daran, den Zeitaufwand für die Identifikation aller Modellkandidaten zu reduzieren. Zwei Ansätze hierzu werden im Folgenden erläutert. Der folgende Abschnitt stellt zunächst ein Verfahren vor, mit dem die Rechenzeit für eine einzelne Parameteridentifikation reduziert wird. Um in einer Modellfamilie darüber hinaus möglichst schnell die guten Modellkandidaten zu finden, wurde eine heuristische Suche entwickelt, siehe Abschnitt 5.2.2.

5.2.1 Kinetikabgleich

Durch die automatische Erzeugung von Modellkandidaten ergeben sich innerhalb einer Modellfamilie viele recht ähnliche Varianten. Wie bereits dargelegt, müssen ihre jeweiligen Modellparameter identifiziert werden, damit die Modelle optimal zu den vorhandenen Messdaten passen. Der Zeitaufwand für jede Parameteridentifikation ist hoch, insbesondere wenn keine guten Startwerte bekannt sind. Abschnitt 4.3 stellte erste Ansätze vor, mit deren Hilfe der Zeitaufwand reduziert werden kann. Die Suche nach geeigneten Startwerten kann aber speziell durch die Ähnlichkeit zwischen den einzelnen Modellkandidaten noch weiter beschleunigt und verbessert werden, wie im Folgenden gezeigt wird.

Voraussetzung ist ein erster, bereits identifizierter Modellkandidat, dessen Anpassung an die Messdaten von akzeptabler Güte ist. Die Startwerte für einen zweiten noch zu identifizierenden Modellkandidaten können schnell durch folgende Methode bestimmt werden, wenn die beiden Kandidaten sich aufgrund einer ausgetauschten Formalkinetik möglichst nur in einem einzigen Parameter unterscheiden.

Alle Werte von Parametern, die dem bereits identifizierten und dem neuen, noch zu identifizierenden Modellkandidaten gemeinsam sind, werden vom neuen Kandidaten übernommen. Je besser die Anpassung des alten Modells war, desto besser sind insgesamt die Startbedingungen für die nachfolgende Identifikation des neuen Modells. Der oder die übrigen Parameter gehören zu eben jenem Formalkinetik-Unterschied zwischen den beiden Kandidaten. In einer einfachen, algebraischen Optimierung[2] wird die Summe der Fehlerquadrate zwischen den Verläufen der beiden entsprechenden Formalkinetiken minimiert. Auf diese Weise können die Parameter der neuen Formalkinetik so ermittelt werden, dass die Verläufe der beiden Formalkinetiken bestmöglich übereinstimmen. Die Methode, wie auch das Programm, wird `Kinetikabgleich` genannt.

[2] Da keine Simulation eines Modells notwendig ist, kann der Zeitaufwand vernachlässigt werden.

Für die Optimierung wird der gesamte Konzentrationsbereich von null bis zu einem anzugebenden Konzentrationsmaximum betrachtet. Dabei werden 150 äquidistante Stützstellen in diesem Bereich berücksichtigt. Anstelle eines maximalen Werts für die Konzentration können auch Messdaten übergeben werden. Die Optimierung betrachtet in diesem Fall automatisch nur die entsprechenden Konzentrationspunkte auf den Verläufen. Dies verbessert insbesondere bei Kinetiken, deren qualitative Verläufe sehr unterschiedlich ausfallen können, die Anpassung in den praktisch relevanten Konzentrationsbereichen.

Da die Wertebereiche der verschiedenen Kinetiken nicht identisch sein müssen, siehe Abschnitte 4.3.1 und 4.3.2, wird zusätzlich ein Ausgleichsfaktor bestimmt. Der ermittelte Wert wird der Einfachheit halber zum $\mu_{\max}$-Parameter für die betreffende Reaktion des zweiten Modells multipliziert, obwohl er genaugenommen kein Bestandteil der Kinetik ist.

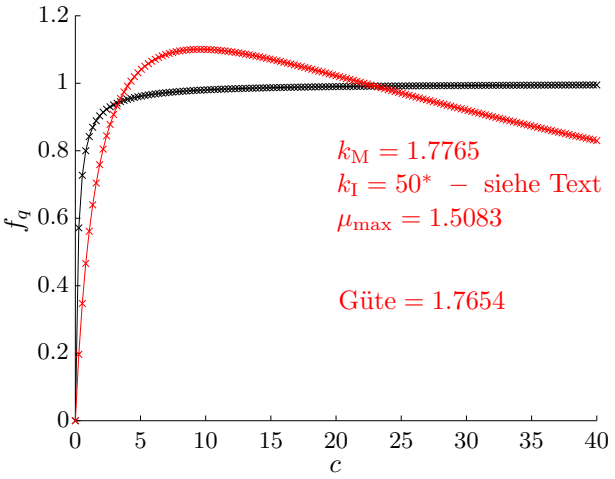
Zusammen ergibt sich ein voroptimierter Vektor von Anfangswerten für den zweiten Modellkandidaten. Die benötigte Rechenzeit für die optimale Anpassung der Parameterwerte wird in der Regel mit diesem deutlich reduziert.

Die Abbildungen 5.1(a) und (b) zeigen exemplarisch das Endergebnis eines Kinetikabgleichs von einer Michaelis-Menten-Kinetik mit dem Parameterwert $k_{\mathrm{M}} = 0.2$ zu einer Haldane-Kinetik, wobei eine maximale Konzentration von $40\,\mathrm{g\,l^{-1}}$ angenommen wird. Die grafische Ausgabe wird von der Funktion automatisch erstellt. Der Aufruf im ABC-System lautet: `Kinetikabgleich('MiMe',40,0.2,'Haldane')`

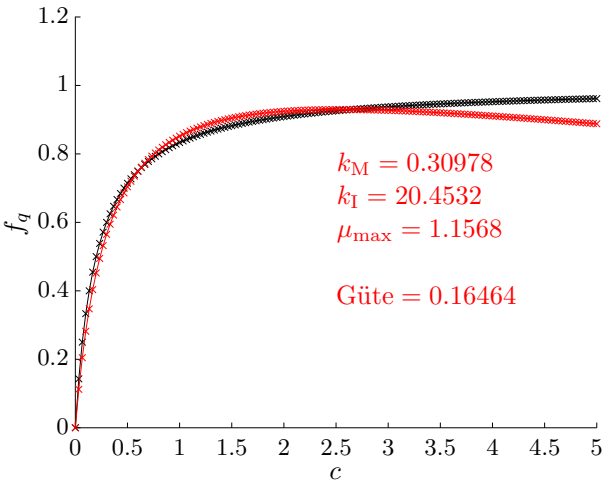
Die erste Abbildung zeigt einen Kinetikabgleich, bei dem keine vorgegebenen Messwerte für die Konzentration c verwendet wurden. Für die Berechnung der zweiten Abbildung wurde angenommen, dass nur Messwerte aus dem Konzentrationsbereich 0 – $5\,\mathrm{g\,l^{-1}}$ vorliegen (in diesem synthetischen Beispiel äquidistant und zu Vergleichszwecken ebenfalls 150 Einzelwerte). Die verschiedenen Messwertverläufe sind durch rote bzw. schwarze Kreuze dargestellt. Die Optimierung erzielt erwartungsgemäß unterschiedliche Ergebnisse.

Der Stern beim Wert k_{I} im oberen Bild deutet an, dass die maximale Grenze für diesen Parameter erreicht wurde. Ohne diese Begrenzung würde der Unterschied in den Parameterwerten zwischen beiden Fällen daher noch deutlicher ausfallen. Aus den Werten geht auch hervor, dass eine Anpassung des $\mu_{\max}$-Parameters allein durch den Austausch einer Formalkinetik notwendig ist, um vergleichbare Ergebnisse zu erzielen.

Es kann aber auch die Situation eintreten, dass ein Kinetikabgleich nicht zu einem besseren Startvektor führt. Diese kann beispielsweise eintreten, wenn bereits eine manuell gestartete Parameteridentifikation für den nächsten Modellkandidaten erfolgt ist



(a) Ohne Messdaten



(b) Mit Messdaten

Abbildung 5.1: Konvertieren einer „MiMe“- in eine „Haldane“-Kinetik mit angepassten Parameter der Haldane-Kinetik. MiMe - schwarz, Haldane - rot

oder ein vorheriger Lauf der Multimodell-Parameteridentifikation abgebrochen wurde. Bevor der neue Parametervektor übernommen wird, werden daher die Gütewerte vor und nach dem Kinetikabgleich miteinander verglichen. Es kann auch vorkommen, dass das Ursprungsmodell keine gute Anpassung erzielt hat. Daher wird zusätzlich auch ein Kinetikabgleich mit dem bisher besten Modellkandidaten durchgeführt und der daraus folgende Gütewert in den Vergleich mit einbezogen.

5.2.2 Heuristische Suche im Kandidatenbaum

In der Praxis ist es durchaus sinnvoll, ein nächstes Experiment planen und durchführen zu können, ehe die Identifikation eines großen Modellbaums abgeschlossen ist, da man beispielsweise an langfristige Terminpläne des Labors gebunden ist. Dies führt daher zu der Fragestellung: Wenn bis zu einem bestimmten Zeitpunkt nur ein Teil der Modellfamilie fertig identifiziert sein wird, wie kann man möglichst schnell viele der guten Kandidaten identifizieren?

Die Parameteridentifikation einer gesamten Modellfamilie kann einerseits beschleunigt werden, indem die Identifikationen der einzelnen Kandidaten so durchgeführt werden, dass aufeinanderfolgende Modelle möglichst wenige unterschiedliche Formalkinetiken aufweisen. Unter dieser Bedingung kann der im letzten Abschnitt beschriebene Kinetikabgleich effektiv eingesetzt werden. Zu diesem Zweck werden die Modellkandidaten in einer baum-ähnlichen Datenstruktur angeordnet. Diese Vorsortierung führt in Verbindung mit dem Kinetikabgleich bereits zu großen Vorteilen hinsichtlich des Rechenaufwands.

Andererseits ist es sinnvoll, mit der Parameteridentifikation der vielversprechendsten Modellkandidaten zu beginnen, und so zu jedem Zeitpunkt das (bis dahin) beste Modell für die Planung oder Regelung des nächsten Versuchs verwenden zu können. Die restlichen Identifikationen laufen parallel zum Experiment weiter, für den Fall, dass ein noch besseres Modell entdeckt wird, welches dann für einen nächsten Versuch benutzt werden kann. Zur Umsetzung dieser zweiten Idee ist das Baum-Schema ebenfalls hilfreich, da darin eine heuristische Suche nach den vielversprechendsten Kandidaten durchgeführt werden kann.

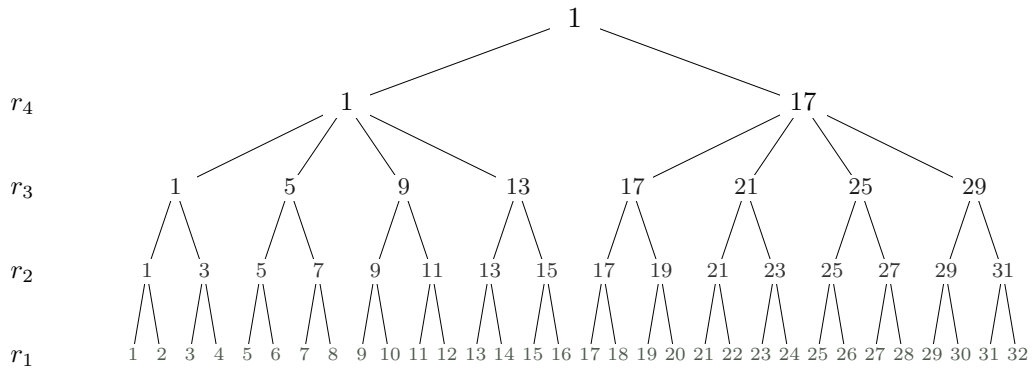


Abbildung 5.2: Datenmodell: Baumstruktur einer Modellfamilie mit vier Variationsstellen: r_1 bis r_4 (siehe Text)

Vor der Beschreibung des Suchalgorithmus wird das Baum-Schema und einige seiner Eigenschaften erläutert: Alle Kandidaten einer Modellfamilie besitzen die gleiche mathematische Struktur ihrer Differentialgleichungssysteme. Die Unterschiede zwischen ihnen stellen „Variationsstellen“ in der Struktur dar, die in einem Modellkandidaten durch eine bestimmte Formalkinetik konkretisiert wird. Jede horizontale Ebene in dem Baum-Schema symbolisiert eine solche Stelle. Die unterschiedlichen Möglichkeiten werden durch die Anzahl der „Kinder“ eines „Elters“ abgebildet. Abbildung 5.2 zeigt exemplarisch den Aufbau eines solchen Baums. Die Modellfamilie, die zu der dargestellten Baumstruktur führt, verfügt über vier Variationsstellen (in den Reaktionen r_1 bis r_4), und darin jeweils zwei (r_1, r_2, r_4) oder vier (r_3) möglichen Formalkinetiken, also über insgesamt $2^3 \cdot 4^1 = 32$ Kandidaten.

Alle erstellten Bäume besitzen die Eigenschaft, dass jede Verbindung zwischen zwei Modellkandidaten nur einem einzigen Modellunterschied (Formalkinetik) entspricht. Im Baum verbundene Modellkandidaten sind also paarweise optimal geeignet für einen Kinetikabgleich.

Die implementierte heuristische Suche nach den besten Kandidaten funktioniert nach den folgenden Regeln[3]. Sie beginnt mit dem Modellkandidat #1, dem ersten Elter:

[3] Im Ausblick, Kapitel 6, werden Vorschläge zu einer Verbesserung dieser Regeln genannt.

1. Identifiziere die Parameter des aktuellen Modellkandidaten (=Elter).
2. Identifiziere die Parameter aller Kinder dieses Elters.
3. Wähle anhand der Güte das beste Kind als das neue Elter aus.
4. Wiederhole Schritte 2 – 3, solange Kinder existieren.
5. Wähle anhand der Güte aus den übrigen Modellkandidaten das neue Elter aus.
6. Beginne erneut bei Schritt 2 mit diesem Kandidaten.
7. Der Modellbaum ist vollständig identifiziert.

Die heuristische Suche basiert auf der Annahme, dass ein Modellkandidat um so schlechtere Gütewerte erhält, je mehr Formalkinetiken er besitzt, die von denen des besten (aber unbekannten) Kandidaten abweichen. Durch den iterativen, paarweisen Vergleich, wird die Anzahl der „falschen“ Kinetiken möglichst schnell reduziert.

Der Baum enthält viele Kandidaten mehrfach, wie auch der Abbildung 5.2 zu entnehmen ist. Dies ist notwendig, um die oben genannten Eigenschaften bezüglich der minimalen Änderungen zwischen benachbarten Kandidaten zu erhalten. Parameteridentifikationen werden dabei aber nicht unnötig wiederholt.

Um sich den Stand der laufenden Identifikationen anzeigen zu lassen, oder um die Verteilung der Gütewerte im Modellbaum zu analysieren, sind im ABC-System verschiedene Funktionen verfügbar. Die interaktive grafische Ausgabe einer dieser Funktionen, siehe Abbildung 5.2, bietet ein Kontextmenü für jeden einzelnen Modellkandidaten. Die verfügbaren Funktionen sind beispielsweise:

- Anzeige von Gütewert sowie Formalkinetik an der entsprechenden Variationsstelle
- Aufruf der verschiedenen Modelldateien im Editor
- Start einzelner Parameteridentifikation durch eine grafische Auswahl
- Darstellung der Verläufe der identifizierten Kinetiken

Letzteres ist eine hilfreiche Funktion, um potentielle Schwächen in einer Modellstruktur oder Kinetikauswahl aufzudecken.

5.3 Multimodell-Trajektorienplanung

Es gibt verschiedene Arten von Prozessführungsstrategien. Eine davon ist die Steuerung eines Prozesses ohne Rücksicht auf Störungen oder Berücksichtigung verfügbarer Messwerte. Das heißt, der jeweilige Prozess wird nur nach einem vorher festgelegten Plan gesteuert. Im Falle der hier behandelten Fermentationen bedeutet dies, dass das Zufütterungsprofil $\mathbf{U}$ vor Versuchsbeginn festgelegt und ohne Veränderungen ausgeführt

wird. Mit Hilfe eines Modells kann das Zufütterungsprofil so optimiert werden, dass in der Simulation ein definiertes Ziel, beispielsweise die maximale Menge eines Produkts m_{Produkt}, erreicht wird. Das Optimierungsproblem zur Minimierung der Güte Φ für ein solchermaßen vereinfachtes Beispiel lautet[4]:

$$(\mathbf{U}^*, \underline{x}_0^*) = \arg\min_{\mathbf{U}, \underline{x}_0} \Phi(\mathbf{U}, \underline{x}_0) \tag{5.2}$$

$$\Phi(\mathbf{U}, \underline{x}_0) = -m_{\text{Produkt}}(t = t_{\text{end}}) \tag{5.3}$$

$$\text{U. d. B.} \qquad \dot{\underline{x}} = f(t, \underline{x}(t), \mathbf{U}, \underline{\theta}), \quad \underline{x}(t = 0) = \underline{x}_0$$

$$\underline{u}_{\min} \leq \underline{u} \leq \underline{u}_{\max}$$

Um die Werte für $m_{\text{Produkt}}(t = t_{\text{end}})$ zu bestimmen, wird das Differentialgleichungssystem in jeder Iteration des Optimierungsprozesses neu mit einem veränderten $\mathbf{U}$ gelöst. Das Zufütterungsprofil $\mathbf{U}^*$ bestimmt dann zusammen mit den Anfangswerten $\underline{x}_0^*$ den „Weg“ der Differentialgleichungslösung, der das gesetzte Ziel erreicht. Deswegen nennt man das Planungsverfahren auch Trajektorienplanung (gr.: „Bahnkurve“).

Das Vorliegen einer kompletten Modellfamilie nach der oben beschriebenen automatischen Modellbildung führt zu der Frage: Mit welchem Modellkandidaten soll eine Prozessplanung durchgeführt werden? Zur Beantwortung ist eine Maßzahl für die Modellgüte der einzelnen Modelle notwendig. Klassischerweise wird hier die Anpassungsgüte an vorhandene Messdaten, also die erreichte Güte bei der Parameteridentifikation, gewählt. Allerdings zeigt sich häufig, dass es nicht nur einen eindeutig besten Modellkandidaten gibt, sondern mehrere, vergleichbar gute. Es gibt andere Methoden, wie zum Beispiel die Akaike-Modellwahrscheinlichkeit, die neben der Anpassungsgüte auch die Anzahl der Parameter eines Modells berücksichtigt [16]. Allerdings reduziert auch diese Methode die Anzahl der Modellkandidaten nicht zwangsläufig auf eins.

Die Rangfolge der einzelnen Modellkandidaten wird außerdem durch das Messrauschen beeinflusst. Die Auswahl des Modellkandidaten mit der besten Güte führt daher nicht garantiert zum besten Modell[5]. Welches Modell für die Planung verwendet werden sollte, hängt allerdings auch davon ab, wie gut die extrapolierenden Fähigkeiten der einzelnen Modelle sind. Jenes Modell zu finden, das den geringsten Fehler bei einer Vorhersage macht, kann nicht anhand der Anpassungsgüte entschieden werden. Dies gilt insbesondere, wenn mehrere, vergleichbar gute Kandidaten gefunden wurden. Mit Hilfe eines realen Anwendungsfalls soll dies verdeutlicht werden.

[4] Im industriellen Umfeld würden z. B. Energie- und Rohstoffkosten mitberücksichtigt werden.

[5] Das „beste“ Modell in diesem Zusammenhang gibt die wahren, unverrauschten Messdaten wieder.

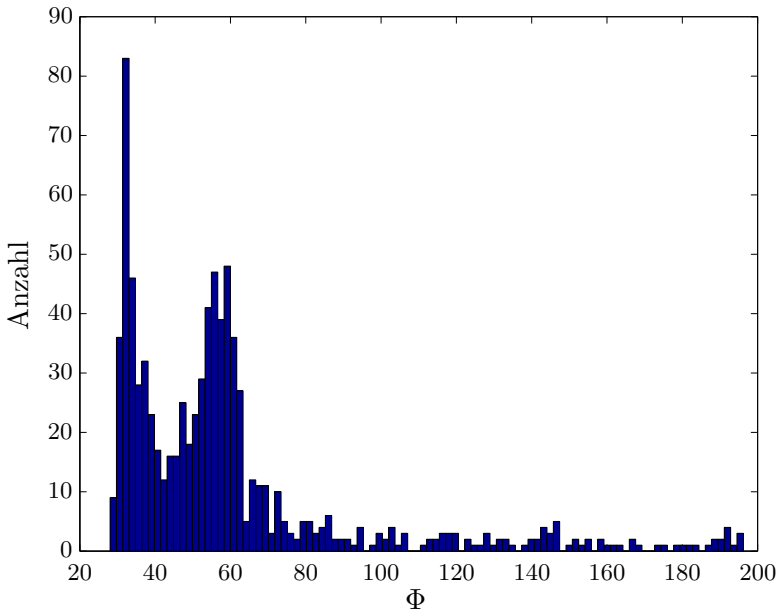


Abbildung 5.3: Histogramm: Verteilung der erreichten Gütewerte

Für die Anpassung an drei verschiedene Kultivierungen von *Paenibacillus polymyxa* wurden verschiedene Modelltypen mit unterschiedlichen Modellstrukturen untersucht. Die im Folgenden gezeigten Ergebnisse basieren auf einer Modellfamilie mit 1152 Modellkandidaten vom Typ „einfaches Modell mit Speicher“ (siehe Abschnitt 3.1.4). Details zu den Fermentationen und den erzeugten Modellen finden sich in Anhang A.1. Innerhalb der Modellfamilie wurden die Formalkinetiken für die Wachstumsreaktion, zwei verschiedene Reaktionen für den Aufbau sowie eine Abbau-Reaktion eines internen Phosphatspeichers variiert. Abbildung 5.3 zeigt die Gütewert-Verteilung nach der Parameteridentifikation aller Kandidaten dieser Modellfamilie, bei denen eine Güte $\Phi < 200$ erreicht wurde (rund 840 Modellkandidaten).

Für die weitere Betrachtung werden nur die zehn besten Modellkandidaten ausgewählt. Sie approximieren die Messdaten gleichermaßen gut; ihre Gütewerte sind nahezu identisch. Dargestellt in den Abbildungen 5.4 und 5.5 sind die Anpassungen aller zehn Modellkandidaten an zwei der drei für die Parameteridentifikation verwendeten Experimente: „PPdef10“ und „PPdef12“. Abbildung 5.4 zeigt die geringsten Abweichungen zwischen Modellsimulationen und den echten Messwerten. Größere Unterschiede sind in Abbildung 5.5 zu erkennen. Das dritte Experiment („PPdef11“) ist sehr ähnlich zu

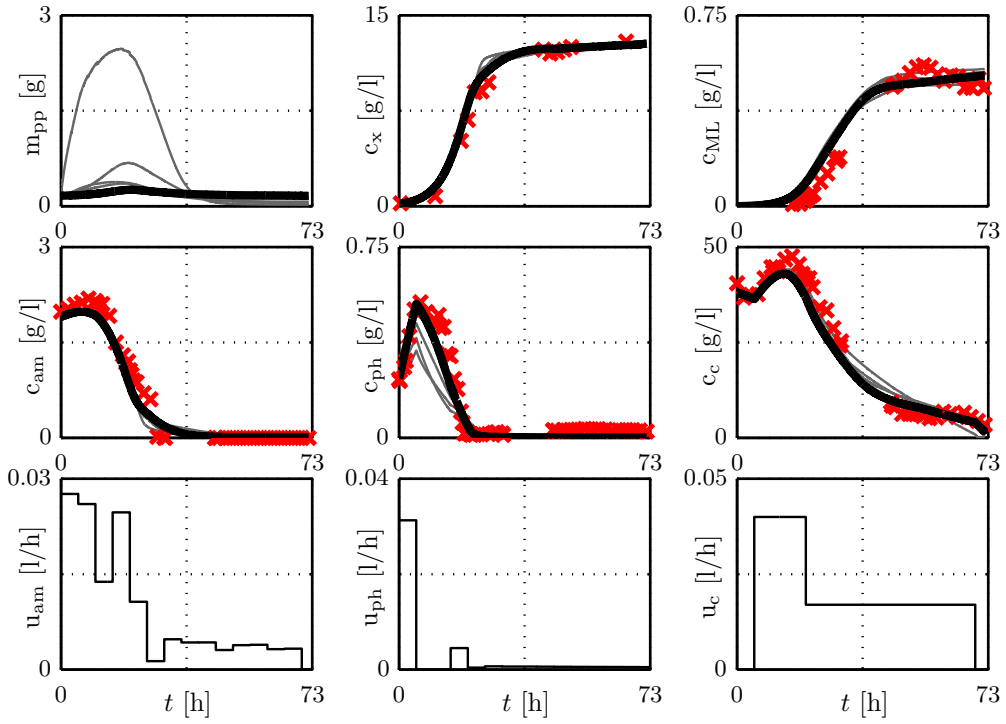


Abbildung 5.4: Anpassung an „PPdef10“ durch die besten zehn Modelle. In schwarz dargestellt ist das beste Modell (übrige: grau), Messdaten sind als rote Kreuze abgebildet. Erste Zeile: Masse des P-Speichers m_{PP}, Konzentration der Biotrockenmasse c_X und Produktkonzentration c_{ML} ; Zweite Zeile: Substratkonzentrationen: c_{am} Ammonium, c_{ph} Phosphat, c_c Glucose; Dritte Zeile: Zudosierraten der drei Substrate

„PPdef10“ durchgeführt worden, daher wird hier auf die bildhafte Darstellung verzichtet.

Insgesamt kann eine befriedigende und sehr ähnliche Anpassung durch alle zehn Modellkandidaten festgestellt werden. In schwarz dargestellt ist das Modell mit der besten Güte. Zu erkennen ist insbesondere in Abbildung 5.5, dass dieses Modell nicht bei jeder einzelnen Messgröße die beste Anpassung zeigt. Im Folgenden werden die einzelnen Modellkandidaten in der Reihenfolge ihrer Anpassungsgüte benannt, das heißt, „Modell #1“ zeigt die beste Anpassung, und so weiter.

Dass sich die bislang sehr ähnlichen, strukturell sogar identischen Modellkandidaten unter veränderten Versuchsbedingungen deutlich unterscheiden können, zeigt eine

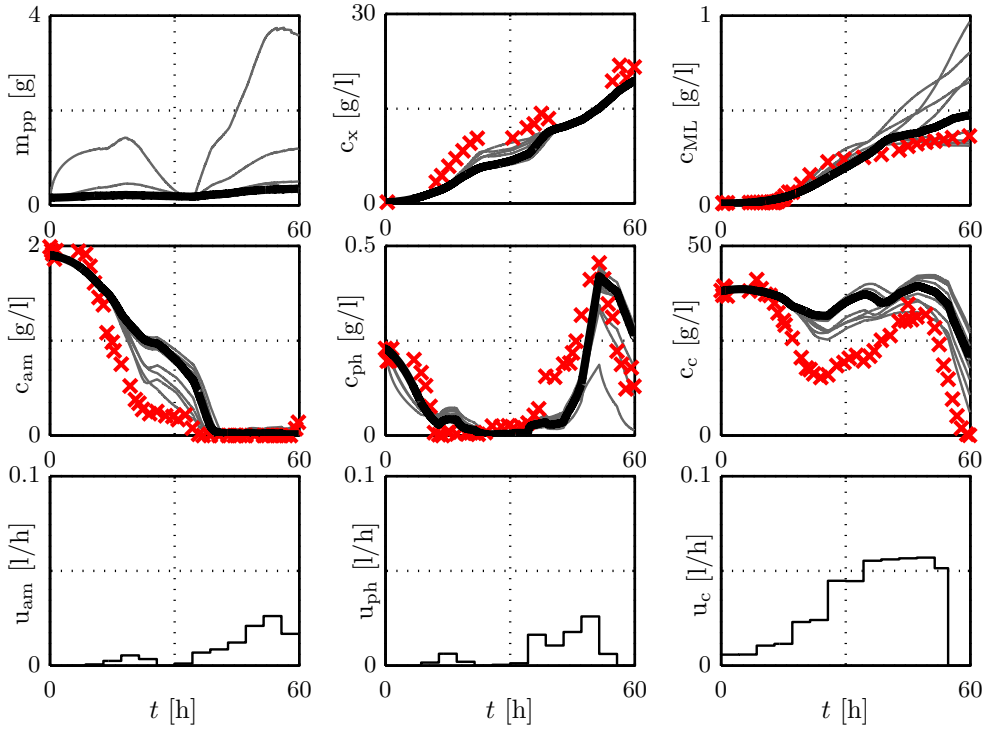


Abbildung 5.5: Anpassung an „PPdef12“ durch die besten zehn Modelle, siehe Abb. 5.4

durchgeführte Trajektorienplanung. Als Grundlage für die Optimierung wurde Modell #1 gewählt. Das Ziel der Prozessplanung war eine Maximierung der Produktmenge. Als Ergebnis der Optimierung wurde ein optimales Zufütterungsprofil $\mathbf{U}^*_{\#1}$ ermittelt.

Wie oben erwähnt, muss Modell #1 nicht zwangsläufig die beste Wahl für ein Planungsverfahren sein. Eventuell kann das nächstbeste Modell (Modell #2) das wahre, unverrauschte Verhalten der Mikroorganismen besser approximieren. Zum Vergleich wird eine Simulation von Modell #2 mit dem Zufütterungsprofil $\mathbf{U}^*_{\#1}$ in Abbildung 5.6 gezeigt.

Die Auswirkungen der mathematischen Unterschiede zwischen den beiden Modellkandidaten sind unter den gewählten Prozessbedingungen sehr deutlich. Sogar die vorgegebenen Grenzwerte für die Planung werden bei verschiedenen Messgrößen (Phosphat und Glucose) durch die Simulation von „Modell #2“ verletzt.

Vor der Durchführung des geplanten Versuchs kann nicht entschieden werden, welche Simulation eher das Verhalten der Mikroorganismen wiedergibt. Daher kann auch nicht

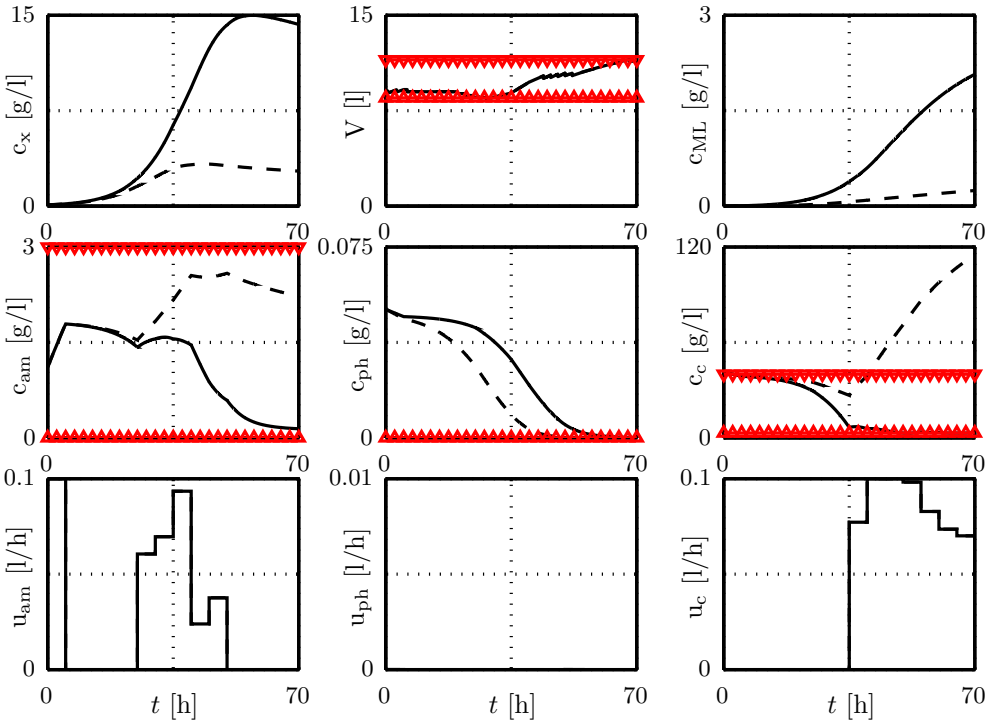


Abbildung 5.6: Trajektorienplanung auf Basis von Modell #1 (durchgezogene Linie) zur Produktmaximierung (rote Markierungen zeigen Grenzwerte für die Optimierung) und Vergleichssimulation mit dem Modell #2 (Strichlinie); untere Reihe: Zufütterungsprofil $\mathbf{U}^*_{\#1}$

davon ausgegangen werden, dass $\mathbf{U}^*_{\#1}$ tatsächlich dem gesuchten $\mathbf{U}^*$ entspricht.

Anstatt sich bei der Planung des nächsten Versuchs für einen einzigen Modellkandidaten zu entscheiden, wird hier vorgeschlagen, mehrere Modellkandidaten gleichzeitig bei der Optimierung zu berücksichtigen. Dieses Verfahren trägt daher den *Multimodell-Trajektorienplanung*.

Kurz skizziert schaltet das ABC-System während der Optimierung zwischen den verschiedenen Modellkandidaten automatisch hin und her und simuliert ihren jeweiligen Prozessverlauf. Der wesentliche Unterschied zur normalen (Einzel-Modell-)Trajektorienplanung besteht im Gütefunktional, in dessen Berechnung anstelle einer einzigen Simulation nun die verschiedenen Simulationen der einzelnen Modellkandidaten einfließen. Untersucht wurden unterschiedliche Ansätze zur Verrechnung der Simulationsergebnisse,

zum Beispiel die Maximierung des Mittelwerts der durch die einzelnen Modellkandidaten vorausgesagten Produktmenge, oder anstelle des Mittelwerts auch der Median oder das Minimum. Welcher Ansatz am besten geeignet ist, hängt von weiteren Prozesszielen, wie Sicherheit oder Robustheit, ab. Der Minimum-Ansatz ist zum Beispiel eher für Worst-case-Planungen geeignet. Zusätzlich können die einzelnen Kandidaten im Gütefunktional gewichtet werden, um Modellwahrscheinlichkeiten oder die Unsicherheit der Planung zu berücksichtigen.

Eine ähnliche, tatsächlich im Experiment realisierte Trajektorienplanung, basierend auf neun Modellen des gleichen Typs wie oben, ist in Abbildung 5.7 dargestellt. Die zeitlich hohe Dichte der Messungen für Substrate und Produkt (c_{ML}) wurde mit Hilfe des Automatischen Probennahmesystems (Kapitel 2) erreicht. Das Optimierungsziel für den dargestellten Versuch war eine Maximierung des ungewichteten Mittelwerts der Produkt-Endwerte c_{ML}.

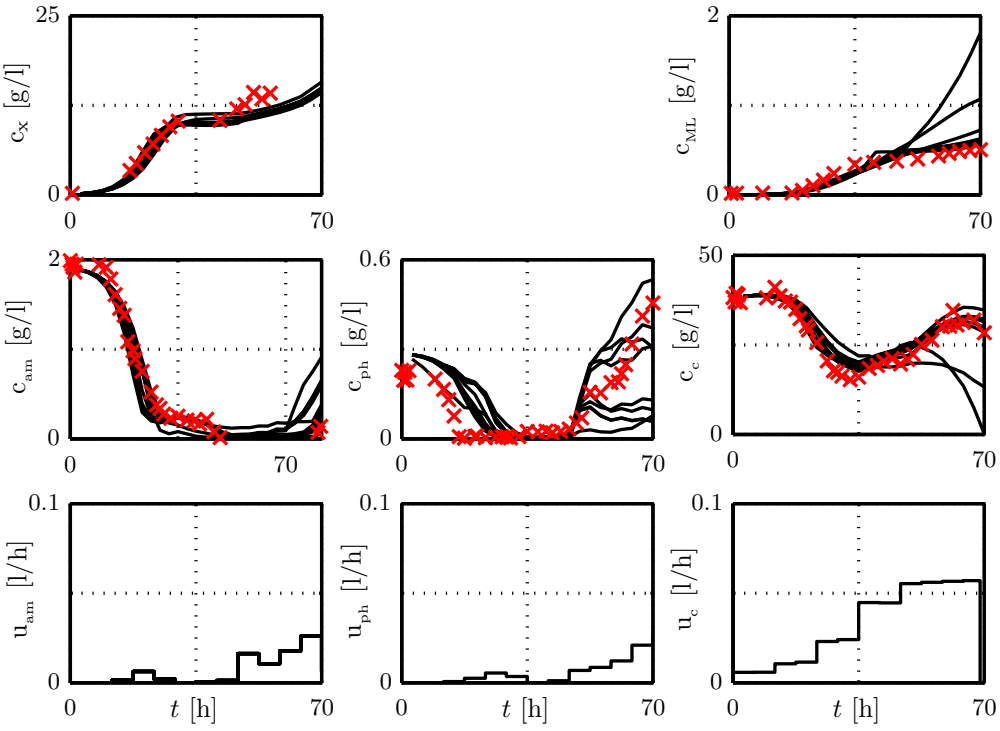


Abbildung 5.7: Planung (Striche) und Umsetzung (rote Kreuze) einer Multimodell-Trajektorienplanung

Die Abbildung zeigt erneut die nun großen Unterschiede zwischen den Simulationen der einzelnen Modellkandidaten, aber auch, dass der tatsächliche Verlauf der verschiedenen Messgrößen weitgehend gut beschrieben wird. Berücksichtigt man die Tatsache, dass es sich bei den für das Beispiel verwendeten Modellkandidaten lediglich um einfache Modelle mit einem zellinternen Speicher handelt, die zudem nur auf der Basis von drei Experimenten ermittelt wurden, zeigen die Simulationen durch die automatisch erstellten Modellkandidaten bereits eine tendenziell zufriedenstellende Übereinstimmung mit den realen Messwerten. Einige der Modellkandidaten sind jedoch weniger gut geeignet, den neuen Versuch angemessen zu beschreiben. Von weiteren Multimodell-Trajektorienplanungen können diese Kandidaten ausgeschlossen werden.

Sind prinzipielle Unterschiede zwischen Messdaten einerseits und den Simulationen einer großen Gruppe von Modellkandidaten andererseits zu beobachten, können hieraus unter Umständen Informationen gewonnen werden, um die Modelle strukturell zu verbessern: Wenn eine hinreichend große Anzahl von Formalkinetiken eingesetzt wurde, können ungeeignete Formalkinetiken als primäre Ursache für die bestehenden Modellfehler ausgeschlossen werden. Stattdessen ist der Fehler in der Modellstruktur zu suchen. Hier kann zukünftig versucht werden, ein automatisiertes Verfahren „Lernen-aus-Fehlern“ zu entwickeln, welches an dieser Stelle ansetzt und Vorschläge für eine Verbesserung der Modellstruktur unterbreitet.

Die Entscheidung für einen oder mehrere Modellkandidaten ist um so schwieriger, je weniger experimentelle Daten zur Bewertung vorliegen. Stehen viele Experimente für eine Parameteridentifikation zur Verfügung, mittelt sich der beeinflussende Effekt durch das Messrauschen heraus. Ebenso kann die Extrapolationsgüte der Modelle mit einer steigenden Anzahl unterschiedlicher Versuche als Datenbasis für die Parameteridentifikation verbessert werden. Die Gruppe zu berücksichtigender Modellkandidaten wird daher mit der Durchführung weiterer Experimente tendenziell kleiner. Die vorliegende Arbeit setzt jedoch bei der ersten Modellentwicklung an. Nur in wenigen Fällen werden in dieser Situation bereits umfangreiche Messreihen zur Verfügung stehen. Besonders in Abbildung 5.7 zeigt sich auch, dass ein einzelner Modellkandidat die Realität im Detail nicht so gut approximiert, wie die Gemeinschaft mehrerer Modellkandidaten es vermag.

5.4 Modelldiskriminierende Versuchsplanung

Im letzten Abschnitt wurde erläutert, wie ähnlich die Verläufe verschiedener Modellkandidaten sein können, und daher eine Planung nicht nur auf einem Modell beruhen sollte. Dies trifft besonders auf die frühe Modellentwicklungsphase zu. Später kann es sinnvoll oder erwünscht sein, die Anzahl der Modellkandidaten zu reduzieren. Typischerweise führt die Durchführung verschiedener Experimente mit einer anschließenden Parameteridentifikation bereits dazu, dass die Menge passender Modellkandidaten abnimmt. Ist dies nicht der Fall, kann das Gütefunktional für eine Versuchsplanung so formuliert werden, dass durch den Versuch die Unterschiede zwischen den einzelnen Kandidaten maximal herausgearbeitet werden. Nach der Realisierung des geplanten Versuchs kann durch den Vergleich zwischen Simulationen und den erhobenen Messdaten die Anzahl der zu berücksichtigenden Kandidaten reduziert werden. Dementsprechend nennt sich dieses Verfahren *Modelldiskriminierende Versuchsplanung*.

Zuerst werden die Modellkandidaten, die Messgrößen und die Messzeitpunkte ausgewählt, zu denen die einzelnen Modellsimulationen zum Zwecke der Diskriminierung miteinander verglichen werden sollen. Die ausgewählte Anzahl der Modelle wird im Folgenden mit N_M, die der Messgröße mit N_Y und die der Messzeitpunkte mit N_t bezeichnet. Der Vergleich wird durch ein Gütefunktional quantifiziert. Im Folgenden werden zwei verschiedene Gütefunktionale vorgestellt, die im Rahmen dieser Arbeit verwendet wurden. Im ersten, quadratischen Ansatz wird wie bei Fehlerquadraten die paarweise Differenz der einzelnen Simulationsergebnisse gebildet, quadriert und aufaddiert, siehe Gleichung 5.4. Da die maximale Differenz gesucht wird und Optimierer üblicherweise nach einem Minimum suchen, wird die Summe mit -1 multipliziert:

$$\Phi_{\text{quad}}(\underline{x}(\mathbf{U}), \underline{x}_0) = -\sum_{i=1}^{N_M-1} \sum_{j=i+1}^{N_M} \sum_{n=1}^{N_Y} \sum_{t_k=1}^{N_t} \left({}^{i}y_n(t_k)^{\text{sim}} - {}^{j}y_n(t_k)^{\text{sim}}\right)^2 \tag{5.4}$$

Hier stehen ${}^{i}y$ und ${}^{j}y$ für die Simulationen der Modellkandidaten i und j.

Werden zur Vereinfachung die Anfangsbedingungen $\underline{x}_0$ konstant gehalten und nur das Zufütterungsprofil $\mathbf{U}$ variiert, ergibt sich als Ergebnis der Optimierung ein optimales Zufütterungsprofil $\mathbf{U}^*$, welches Versuchsbedingungen erzeugt, die die größten Unterschiede zwischen den ausgewählten Modellkandidaten voraussagt.

Als Erweiterung können die einzelnen Messgrößen und/oder Zeitpunkte zusätzlich gewichtet werden. Dies ist in [22] detailliert beschrieben. Wird der Ansatz allerdings bei

mehr als zwei Modellkandidaten verwendet, kann der Fall eintreten, dass nur der Verlauf eines Modells weit von den übrigen entfernt ist, beziehungsweise einige wenige Maximalabstände bevorzugt werden, aber keine gleichmäßige Verteilung erzielt wird. Im Verlauf der Arbeit wurden daher Varianten des quadratischen Gütefunktionals untersucht, die beispielsweise die einzelnen Fehlerquadrate zusätzlich normieren, allerdings ohne eindeutig positive Ergebnisse, weswegen hier nur die Grundvariante vorgestellt wurde.

Signifikante Verbesserungen bringt ein zweiter Ansatz, der die Modellwahrscheinlichkeiten der einzelnen Modellkandidaten sowie das Messrauschen berücksichtigt[6].

Für die folgenden Ausführungen werden alle ausgewählten Messdaten zu den N_t Zeitpunkten in einem Vektor $\underline{\mathrm{Y}}$ zusammengefasst:

$$\underline{\mathrm{Y}} = \begin{pmatrix} \underline{y}^{(t_1)} \\ \underline{y}^{(t_2)} \\ \vdots \\ \underline{y}^{(N_t)} \end{pmatrix} \tag{5.5}$$

Unter der Annahme, dass das Modell M^* die Realität korrekt beschreibt, führt ein Versuch idealerweise zu den Messdaten $\underline{\mathrm{Y}}^*$. In der Praxis sind Messungen jedoch verrauscht. Nimmt man ein normalverteiltes Rauschen an, lässt sich dieses durch seine Kovarianzmatrix $\mathbf{C}_y$ beschreiben. Wegen der Erweiterung des Messgrößenvektors von $\underline{y}$ auf $\underline{\mathrm{Y}}$, muss auch die verwendete Kovarianzmatrix analog erweitert werden: $\mathbf{C}_{\underline{\mathrm{Y}}}$. Damit ergibt sich für die tatsächlich zu messenden Werte $\underline{\mathrm{Y}}$, unter Verwendung von $Q = \dim(\underline{\mathrm{Y}})$ und $|\mathbf{C}_{\underline{\mathrm{Y}}}| = \det(\mathbf{C}_{\underline{\mathrm{Y}}})$, die folgende Funktion für die Wahrscheinlichkeitsdichte:

$$f^*(\underline{\mathrm{Y}}) = \frac{1}{\sqrt{(2\pi)^Q \cdot |\mathbf{C}_{\underline{\mathrm{Y}}}|}} \cdot \exp\left(-\frac{1}{2}(\underline{\mathrm{Y}} - \underline{\mathrm{Y}}^*)^{\mathrm{T}} \cdot \mathbf{C}_{\underline{\mathrm{Y}}}^{-1} \cdot (\underline{\mathrm{Y}} - \underline{\mathrm{Y}}^*)\right) \tag{5.6}$$

Die Wahrscheinlichkeitsdichte kann auch für alle N_M Modelle bestimmt werden, die die Messdaten $\underline{\mathrm{Y}}$ beschreiben: $f_1(\underline{\mathrm{Y}}), f_2(\underline{\mathrm{Y}}), \ldots, f_{N_M}(\underline{\mathrm{Y}})$. Mit Hilfe dieser Werte kann eine Modelldiskriminierung **nach** einem erfolgten Versuch durchgeführt werden. Dies ist kein Planungsverfahren und wird hier deshalb nicht weiter betrachtet.

Bei der Modelldiskriminierenden Versuchsplanung sollen **vor** der tatsächlichen Durchführung des geplanten Versuchs die Vorhersagen durch die verschiedenen Modellkandidaten diskriminiert werden. Dazu wird durch die Festlegung eines Stellgrößenprofils $\mathbf{U}$ mit Hilfe des Modells M_i ein Messdatenvektor $\underline{\mathrm{Y}}_i$ berechnet. Für die geplanten und somit

[6] Die Idee zu diesem Ansatz stammt von M. Kawohl (persönliche Mitteilung).

unverrauschten Werte ergibt sich für den Wert der Wahrscheinlichkeitsdichte f_i an der Stelle $\underline{\mathrm{Y}}_i$:

$$f_i(\underline{\mathrm{Y}} = \underline{\mathrm{Y}}_i) = \frac{1}{\sqrt{(2\pi)^Q \cdot |\mathbf{C}_{\underline{\mathrm{Y}}}|}} \cdot \underbrace{\exp\left(-\frac{1}{2}(\underline{\mathrm{Y}}_i - \underline{\mathrm{Y}}_i)^{\mathrm{T}} \cdot \mathbf{C}_{\underline{\mathrm{Y}}}^{-1} \cdot (\underline{\mathrm{Y}}_i - \underline{\mathrm{Y}}_i)\right)}_{=1} \tag{5.7}$$

Setzt man die simulierten Werte $\underline{\mathrm{Y}}_i$ in eine der anderen Dichtefunktionen f_j ein, folgt daraus:

$$f_j(\underline{\mathrm{Y}}_i) = \frac{1}{\sqrt{(2\pi)^Q \cdot |\mathbf{C}_{\underline{\mathrm{Y}}}|}} \cdot \underbrace{\exp\left(-\frac{1}{2}(\underline{\mathrm{Y}}_i - \underline{\mathrm{Y}}_j)^{\mathrm{T}} \cdot \mathbf{C}_{\underline{\mathrm{Y}}}^{-1} \cdot (\underline{\mathrm{Y}}_i - \underline{\mathrm{Y}}_j)\right)}_{\leq 1} \tag{5.8}$$

„Fremde“ Modelle M_j $(i \neq j)$ beschreiben die Simulationsdaten $\underline{\mathrm{Y}}_i$ meist mit einer geringeren Wahrscheinlichkeitsdichte als das zur Berechnung verwendete Modell. Das Ziel der Modelldiskriminierenden Versuchsplanung ist es, einen Versuch durch Ermittlung eines geeigneten Stellgrößenprofils $\mathbf{U}^*$ zu finden, so dass gemäß Gleichung 5.8 die Messdaten $\underline{\mathrm{Y}}_i$ von Modell M_i durch die anderen Modellen M_j, $i \neq j$, nur mit einer geringen Wahrscheinlichkeitsdichte beschrieben werden, im Idealfall sogar $f_j(\underline{\mathrm{Y}}_i) \approx 0$ für $i \neq j$. Außerdem folgt aus Gleichung 5.7, dass gleichzeitig $f_j(\underline{\mathrm{Y}}_i)$ für $i = j$ maximal sein soll. Diese Anforderungen lassen sich über den Satz von Bayes[7] miteinander verknüpfen. In diesem Satz

$$\mathrm{P}(\mathrm{M}_i|\underline{\mathrm{Y}}) = \frac{\mathrm{P}(\underline{\mathrm{Y}}|\mathrm{M}_i) \cdot \mathrm{P}(\mathrm{M}_i)}{\sum_{j=1}^{N_M} \mathrm{P}(\underline{\mathrm{Y}}|\mathrm{M}_j) \cdot \mathrm{P}(\mathrm{M}_j)} \tag{5.9}$$

ist $\mathrm{P}(\mathrm{M}_i)$ die A-priori-Modellwahrscheinlichkeit dafür, dass das Modell M_i zutreffend ist. Ohne Vorwissen wird jedem Modell zunächst die gleiche Wahrscheinlichkeit zugeordnet: $\mathrm{P}(\mathrm{M}_i) = 1/N_M$. Als A-posteriori-Wahrscheinlichkeit wird $\mathrm{P}(\mathrm{M}_i|\underline{\mathrm{Y}})$ bezeichnet, sie drückt die Wahrscheinlichkeit aus, dass das Modell M_i unter Berücksichtigung der vorhandenen (oder für die Modelldiskriminierende Versuchsplanung: der geplanten) Messdaten $\underline{\mathrm{Y}}$ korrekt ist. Umgekehrt stellt $\mathrm{P}(\underline{\mathrm{Y}}|\mathrm{M}_i)$ die Wahrscheinlichkeit dafür dar, dass durch das Modell M_i die Messdaten $\underline{\mathrm{Y}}$ erzeugt werden. Letzteres wird durch die Wahrscheinlichkeitsdichtefunktion Gleichung 5.6 ausgedrückt, während das Ziel der Modelldiskriminierende Versuchsplanung der Maximierung der A-posteriori-Wahrscheinlichkeit entspricht. In Gleichung 5.9 wird daher $\mathrm{P}(\underline{\mathrm{Y}}|\mathrm{M}_i)$ durch $f_i(\underline{\mathrm{Y}}_i)$ und $\mathrm{P}(\underline{\mathrm{Y}}|\mathrm{M}_j)$ durch $f_j(\underline{\mathrm{Y}}_i)$ ersetzt.

[7] Thomas Bayes (um 1700 – 1761, Mathematiker, England)

Für das gesuchte Stellgrößenprofil $\mathbf{U}^*$, das zu den geplanten Messvektoren $\underline{\mathrm{Y}}_j$ mit der Eigenschaft führt, dass N_M Modelle optimal voneinander diskriminiert sind, gilt somit:

$$\mathrm{P}(\mathrm{M}_i|\underline{\mathrm{Y}}_i) = \frac{f_i(\underline{\mathrm{Y}}_i) \cdot \mathrm{P}(\mathrm{M}_i)}{f_i(\underline{\mathrm{Y}}_i) \cdot \mathrm{P}(\mathrm{M}_i) + \underbrace{\sum_{j=1, i\neq j}^{N_M} f_j(\underline{\mathrm{Y}}_i) \cdot \mathrm{P}(\mathrm{M}_i)}_{\approx 0}} \qquad \approx 1 \tag{5.10}$$

Die Suche nach einem ideal modelldiskriminierenden Versuch entspricht daher der Suche nach dem Maximum von Gleichung 5.10, wobei für M_i alle Modellkandidaten von 1 bis N_M einzusetzen ist. Das Optimierungsproblem lautet daher, unter Beachtung der Tatsache, dass Optimierer nach einem Minimum suchen:

$$\mathbf{U}^* = \arg\min_{\mathbf{U}} \Phi(\underline{x}(\mathbf{U})) \tag{5.11}$$

$$\Phi(\underline{x}(\mathbf{U})) = \frac{1}{N_M} \sum_{i=1}^{N_M} (1 - \mathrm{P}(\mathrm{M}_i|\underline{\mathrm{Y}}_i)) \tag{5.12}$$

Der Vorfaktor $1/N_M$ hat keinen Einfluss auf die Optimierung, er normiert aber das Ergebnis so, dass am Gütewert der Erfolgsgrad der Modelldiskriminierung schnell abgelesen werden kann.

Der Vorteil gegenüber dem quadratischen Gütefunktional besteht darin, dass nicht versucht wird, die Differenz zwischen den einzelnen Modell-Trajektorien so weit wie möglich zu maximieren, sondern nur so weit wie notwendig, um ausreichend distinkte Gaußkurven zu erhalten. Eine weitere Vergrößerung des Abstands der simulierten Messwerte hätte nur einen geringen Einfluss auf den Gütewert, da die überlappende Fläche der verschiedenen Gauss-Kurven und somit $f_j(\underline{\mathrm{Y}}_i)$ für $i \neq j$ immer geringer wird. Abbildung 5.8 veranschaulicht dies anhand eines Beispiels. Dargestellt ist eine Messgrößen-Trajektorie zweier unterschiedlicher Modelle. Bei Betrachtung der nominellen Trajektorien (farbige Linien) sind deutliche Unterschiede zwischen beiden Verläufen zu erkennen. Wird jedoch das Messrauschen mit berücksichtigt, verändert sich die Situation: Durch die Unsicherheiten ergibt sich eine Wahrscheinlichkeitsdichte für den in der Praxis messbaren Verlauf der jeweiligen Trajektorie. In der Abbildung ist die Dichte für den Endwert der Trajektorien durch senkrecht stehende Gaußkurven der entsprechenden Farbe dargestellt. Ein Überlappungsbereich beider Kurven ist zu erkennen, ergibt eine Messung einen Konzentrationswert in diesem Bereich, kann eine sichere Zuordnung zu einem der beiden Modelle nicht vorgenommen werden.

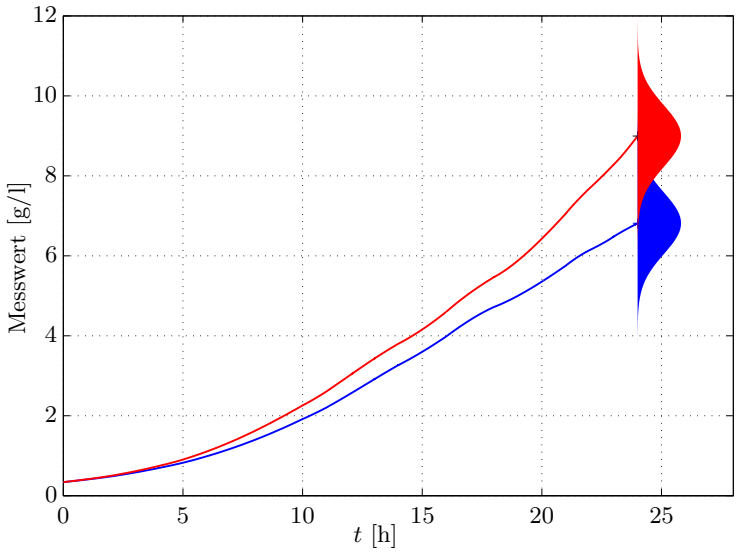


Abbildung 5.8: Auswirkungen von Modellunsicherheiten

Die beiden vorgestellten Gütefunktionale wurden mit verschiedenen Modellen und Startwerten überprüft. Durch das Bayes-Gütefunktional in der Modelldiskriminierenden Versuchsplanung werden Trajektorien ermittelt, die die einzelnen Modellsimulationen besser diskriminieren. Es ist daher zu bevorzugen. Allerdings waren zunächst viele Resultate nicht zufriedenstellend.

Die in der Praxis für Optimierungen wichtigen Gütegebirge sind in der Regel nur an wenigen, vom Optimierer bestimmten, diskreten Punkten bekannt. Beispielhaft soll deshalb nun ein synthetisch erzeugtes Gütegebirge dabei helfen, die teilweise schlechten Optimierungsresultate nachzuvollziehen und idealerweise auch vermeiden zu können. In einer modellfreien Simulationsstudie wurden dazu die Produktendmengen von vier „Modellen“ als Variablen des Bayes-Gütefunktionals, Gleichung 5.12 verwendet. Es wurden also keine Modelle simuliert, sondern lediglich die Endmengen variiert. Für die ersten beiden „Modellkandidaten“ wurden die Endmengen m_1 und m_2 in kleinen Schritten von 0 bis 30 in das Gütefunktional eingesetzt. Die Endmengen der Modelle 3 und 4 wurden konstant gesetzt ($m_3 = 10$ und $m_4 = 15$), um das entstehende Gütegebirge grafisch veranschaulichen zu können. Somit sind m_1 und m_2 die einzigen Veränderlichen und spannen die Grundfläche des Gütegebirges auf (x- und y-Achse), siehe Abbildung 5.9. Für die Kovarianzwerte wurden $C_y(m_1) = C_y(m_2) = 1$ und $C_y(m_3) = C_y(m_4) = 0.25$ verwendet. Wie nochmals betont sei, zeigt die Abbildung eine rein theoretische Betrachtung: Beliebige Produktendwerte sind in der Regel nicht erreichbar. Die tatsächlich möglichen

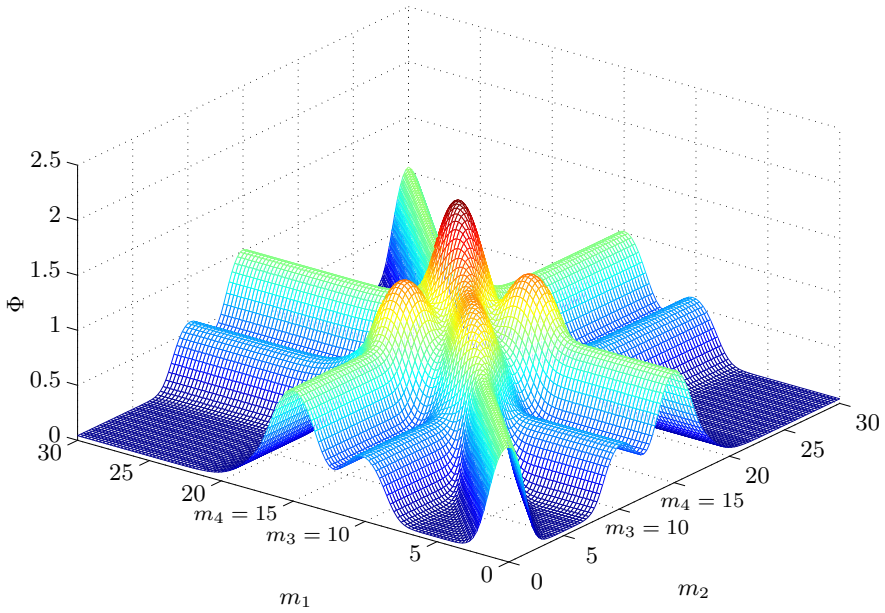


Abbildung 5.9: Simuliertes Gütegebirge eines Vier-Modell-Problems, m_1 und m_2 sind die zu diskriminierenden Vorhersagewerte der „Modelle“ 1 und 2 - siehe Text

Zustandsverläufe echter Modelle schränken den Lösungsraum auf einen kleinen (unter Umständen verzweigten und mehrteiligen) Bereich des dargestellten Gütegebirges ein.

Die Abbildung zeigt, dass sich die Güte erhöht (verschlechtert), wenn die variablen Endwerte m_1 und m_2 sich einem der beiden konstanten Werte m_3 oder m_4 nähern: In diesen Fällen führen die ähnlichen Zahlenwerte zu einer schlechten Diskriminierung. Aus dem gleichen Grund ist die Güte ebenfalls hoch, wenn sich die variablen Werte der Hauptdiagonalen nähern, denn dort ist $m_1 = m_2$.

Diese einfache Simulationsstudie zeigt eine Vielzahl von lokalen Minima, die aufgrund der Problemstellung automatisch entstehen. Selbst bei der Verwendung von nur zwei Modellen gäbe es aufgrund der beiden Lösungsmöglichkeiten ($m_1 > m_2$ und $m_1 < m_2$) bereits zwei Minima. Je mehr Modelle in die Gütefunktion eingehen, desto „zerfurchter“ wird das entstehende Gütegebirge. Diese Zerfurchung, die auch bei den anderen Gütefunktionalen zur Modelldiskriminierung auftritt, stellt für die ursprünglich eingesetzten, gradientenbasierten Optimierer ein ernstes Problem dar. Das Ergebnis dieser Simulationsstudie ist daher, dass die Ursache für die schlechten Diskriminierungsresultate in einer ungünstigen Wahl von Startwerten für die Optimierung lag. Für die Modelldis-

kriminierende Versuchsplanung empfiehlt es sich daher, keine (rein) gradientenbasierten Optimierer einzusetzen.

Der Informationsgewinn einer Modelldiskriminierenden Versuchsplanung steht in Konkurrenz zu dem aus anderen Verfahren. Insbesondere zu Beginn der Modellentwicklung ist jedoch eine Trajektorienplanung zur Produktmaximierung keine sinnvolle Wahl, da in dieser Phase die Modellierung der Produktbildung oft noch rudimentär ist. Die Modelldiskriminierende Versuchsplanung ist vor allem nützlich, wenn eine Anzahl von potentiellen Modellkandidaten schnell reduziert werden soll. Der Vorteil einer Modelldiskriminierenden Versuchsplanung wird so wie beschrieben erst nach der Realisierung des geplanten Versuchs sichtbar. Da die Differenz zwischen den Verläufen der einzelnen Modellkandidaten maximiert wird, zeigt sich aber schon das mögliche „Spektrum“ der verwendeten Modellstruktur. Auch daraus lassen sich unter Umständen Schlüsse auf Modelldefizite ziehen. Eventuell lassen sich für große Modellfamilien bereits Simulationsstudien durchführen, mit dem Ziel, die Familie in Gruppen von Modellkandidaten einzuteilen, die ein ähnliches Verhalten aufweisen. Dies könnte genutzt werden, um Repräsentanten jeder Gruppe auszuwählen und weitere Untersuchungen und Parameteridentifikationen zunächst nur mit diesen durchzuführen.

6 Zusammenfassung und Ausblick

The best way to predict the future is to invent it.

(Alan Kay)

In dieser Arbeit wurde ein Überblick über die entwickelten Methoden und Programme gegeben. Grundlage für die Programmierung des Modellstrukturgenerators sind dabei umfangreiche informationstheoretischen Datenstrukturen. Hierzu zählen einerseits Strukturvariablen wie die Modellstrukturdefinition, für die zunächst nach eingehender Analyse ermittelt wurde, welche Informationen für eine automatische Erzeugung notwendig und sinnvoll sind. Parallel hierzu wurde auch das kompakte Formulierungssystem für Modelle entwickelt (P-, R- und K-Matrix). Weitere, hier nicht erläuterte Strukturvariablen wurden ebenfalls entworfen, um Funktionen wie zur Darstellung eines Modellbaums mit den notwendigen Informationen zu versorgen. Alle Informationen werden erzeugt und in Dateien gesichert. Das Design der entwickelten Strukturen und der Inhalt der einzelnen Dateien wurde zudem immer unter den Gesichtspunkten Lesbarkeit, Flexibilität, Erweiterbarkeit und Übersichtlichkeit beurteilt. Da dies oftmals widersprüchliche Anforderungen sind, waren individuelle Abwägungen zu treffen. Alle neu entwickelten Programme sind kompatibel zu bereits existierenden Programmen wie zum Beispiel der Parameteridentifikation, die weiterhin auch manuell oder für nicht automatisch erstellte Modelle aufgerufen werden können.

Mit Hilfe der entwickelten informationstechnischen Strukturen konnte gezeigt werden, dass der implementierte Modellstrukturgenerator eine erfolgreiche Umsetzung des Konzepts der automatischen Modellerzeugung darstellt. Seine wichtigsten Merkmale und Vorzüge werden im Folgenden zusammengefasst.

1. Die in Kapitel 4 erläuterte, standardisierte Modellstrukturdefinition ist ein Schlüssel zur Automatisierung. Der Inhalt der Definitionsvariablen kann auf verschiedene Weise festgelegt werden. Solange das Format der Definitionsvariablen eingehalten

wird, kann der Modellstrukturgenerator beliebige Herkunftsquellen bearbeiten. Eine Erweiterung oder Schnittstelle zu anderen Quellen oder Programmen ist daher einfach umzusetzen.

2. Da Computer-Modelle nur genutzt werden können, wenn sie in Form von Funktionen oder ausführbaren Programmen vorliegen, übernimmt der Modellstrukturgenerator die Umsetzung von der Modellstrukturdefinition in ausführbare Matlab- und C-Programme. Durch die automatische Programmierung werden manuelle Fehler vermieden und Standards bei den erzeugten Programmen, zum Beispiel durch einheitliche Variablenbezeichnungen oder Kommentarzeilen, erreicht. Des Weiteren reduziert der Modellstrukturgenerator den Zeitaufwand für die Programmierung eines Modells so stark, dass er im Rahmen der Modellentwicklung nicht mehr auffällt, was einen enormen Fortschritt darstellt.

3. Da in der Kinetik-Definition (R-Matrix) und bei den P-Faktoren (P-Matrix) die Verwendung von Platzhaltern beziehungsweise Alternativen möglich ist, können ohne manuellen Aufwand viele verschiedene Modellvariationen für biotechnologische Prozesse automatisch erzeugt werden. Der Anwender kann somit ohne weiteres Zutun Varianten von Formalkinetiken und – in begrenztem Umfang – auch von Modellstrukturen erstellen lassen und miteinander vergleichen, um die beste Anpassung an Messdaten zu finden. Mit geringem Aufwand kann die Modellstrukturdefinition auch modifiziert werden, so dass auch größere strukturelle Modellunterschiede erzeugt werden können. Manuell wären derart umfangreiche Untersuchungen nicht oder nur schwer durchführbar, wodurch neue Methoden und Entwicklungen möglich werden.

4. Durch die allgemeine Beschreibung von Modellen ist es möglich, den Modellstrukturgenerator auch unter anderen Laborbedingungen einzusetzen. So ist beispielsweise die Anzahl der Feeds und Korrekturfluide nicht fest vorgegeben.

Im Kapitel 5 wurden Probleme, die erst beim Einsatz von größeren Modellfamilien entstehen, zusammen mit Lösungsansätzen vorgestellt: Die Multimodell-Trajektorienplanung umgeht eine prinzipiell unsichere Modellauswahl, indem mehrere Modelle gleichzeitig verwendet werden. Für eine Reduzierung der Kandidatenanzahl ist die modelldiskriminierende Versuchsplanung geeignet.

Insgesamt ist es mit dem Modellstrukturgenerator nun möglich, Methoden für Modelle zu entwickeln, ohne auf die Programmierung derselben angewiesen zu sein. Ein

gutes Beispiel ist die automatische biologische Phänomenerkennung [32], die ohne den Modellstrukturgenerator nicht entwickelt hätte werden können[1].

Der Modellstrukturgenerator ist noch kein fertiges Produkt, seine Entwicklung kann weiter fortgeführt und seine Einsatzbereiche erweitert werden. In den einzelnen Abschnitten dieser Arbeit stehen hierzu bereits einige Anmerkungen. Einige Details zu diesen und weitere Ideen werden im Folgenden aufgeführt.

Zusätzliche Formalkinetiken können einfach zur Kinetikbibliothek hinzugefügt werden, so dass die Funktionalität schnell erweitert werden kann. Die automatische Programmierung ist aktuell noch nur in der Lage, Modelle nach dem erläuterten Schema zu erstellen. Der Modellstrukturgenerator erstellt beispielsweise für die einzelnen Reaktionen selbsttätig die notwendigen Parameter. Wenn, vielleicht aufgrund von biologischem Vorwissen, zwei Parameter aber den gleichen Wert haben sollen oder in einem anderen, bestimmten Verhältnis zu einander stehen sollen, kann dies gegenwärtig nicht berücksichtigt werden. Die Modelldateien können natürlich von Hand modifiziert werden, allerdings wird dies nur bei einer beschränkten Anzahl von Modellkandidaten praktikabel sein. Eine Erweiterung, die derartige, manuell vorgegebene Nebenbedingungen automatisch in Modelle einbaut (beispielsweise in die `SystemInit`), würde die Flexibilität des Modellstrukturgenerators deutlich vergrößern. Bestehende Modellbausteine, wie für die Lag-Phase, können auch noch erweitert oder neue Bausteine hinzugefügt werden, um den biologischen Hintergrund der Modelle zu unterstützen. Ein Beispiel dafür könnte der Energie-Stoffwechsel sein, zu dessen Modellierung entsprechende Messungen notwendig sind. Auch könnten Modelle automatisch um ein Atmungsmodell(baustein) ergänzt werden.

Im Arbeitsumfeld, in dem diese Dissertation eingebunden ist, werden, abgesehen von der Begasung des Fermenters, fast ausschließlich Versuche mit flüssigen, chemisch-definierten Substraten und Kulturmedien durchgeführt. Dies führt einerseits zu einer verbesserten Reproduzierbarkeit und auch zu einfacheren Modellen. Beides wiederum erhöht die Identifizierbarkeit. Aus diesem Grund ist die Software des Modellstrukturgenerators bisher so konzipiert, ebenfalls nur Modelle mit flüssigen, chemisch-definierten Substraten und Kulturmedien zu erstellen. Dies hat zur Folge, dass andere Situationen noch nicht in der Standardisierung des Differentialgleichungssystems enthalten sind. Durch eine zusätzliche Betrachtung von beispielsweise Gas-Daten innerhalb des Modellstrukturgenerators wird das Programm vielseitiger anwendbar.

[1] Persönliche Mitteilung von S. Herold, 2015

Eine zukünftige Verbesserung stellt die automatische Parameterbegrenzung dar. Wie in Abschnitt 4.3 dargestellt, wurden für die Kinetiken bereits numerisch sinnvolle Wertebereiche definiert. Für eine konkrete Anwendung einer Kinetik auf bestimmte Messgrößen oder Konzentrationsbereiche können sich engere Wertebereiche als praktikabler erweisen. Die entsprechenden Auswirkungen auf die Kinetiken sollten systematisch untersucht werden, und für eine Automatisierung ist eine dynamische Beschreibung der sinnvollen Parametergrenzen in den jeweiligen Kinetiken ideal. Dabei ist darauf zu achten, dass bei einer Parameteridentifikation zwar die minimalen und maximalen Werte der einzelnen Messgrößen leicht zu ermitteln sind, aber bei einer Trajektorienplanung werden häufig neue Wertebereiche erreicht. Die Grenzen müssen also auf die denkbaren und zulässigen Werte angepasst werden, die der Benutzer vorgeben muss. Darüber hinaus kann auch das Erreichen der Grenzen für Kinetikparameter zu Modellverbesserungen genutzt werden, wie am Beispiel der Michaelis-Menten-Kinetik gezeigt wurde. Das Aufstellen entsprechender automatischer Verfahren zur Modellverbesserung, auch für andere Kinetiken, ist eine sinnvolle Erweiterung.

In Abschnitt 5.2 wurde die heuristische Suche nach den vielversprechendsten Modellkandidaten in einem Baum-Schema vorgestellt. Einfache Erweiterungen des Programms können die Auswahl des ersten Modellkandidaten ermöglichen und auch die Abarbeitungsreihenfolge der Variationsstellen frei wählbar machen. Insbesondere Letzteres könnte verwendet werden, um die heuristische Suche zu optimieren. Algorithmen könnten beispielsweise ermitteln, welche Variationsstelle die stärksten Unterschiede zwischen den Modellkandidaten hervorruft und so schneller oder effektiver die besten Kandidaten durch die Suche ermitteln.

Die Einsatzmöglichkeiten des Modellstrukturgenerator können auch zu vollkommen neuen Verfahren führen. Ein Stichwort hierzu ist das „Lernen aus Fehlern“. Schon bei einem einzelnen Modell kann die Frage gestellt werden, wie die Abweichungen zwischen Modellapproximation und den echten Messdaten zu Vorschlägen zur Verbesserung des Modells führen kann. Diese Verfahren müssen noch entwickelt und automatisiert werden, bieten aber im Hinblick auf die automatische Modellerzeugung und für Modellfamilien ein noch größeres Potential, wie kurz auf Seite 139 erwähnt wurde. Ein Teil eines solchen Verfahrens könnte auch eine Analyse der identifizierten Kinetikparameter enthalten, wie es auf auf Seite 119 angedeutet wurde: Aus dem Erreichen von zuvor festgelegten Grenzwerten können zumindest für einige Formalkinetiken Schlüsse für eine Modellverbesserung gezogen werden. Auch identifizierte Anfangswertparameter (Abschnitt 3.4) können Informationen beinhalten: Wenn trotz identischer Vorkulturführung sehr unterschiedli-

che Werte für einen Anfangswert für die verschiedenen Experimente ermittelt werden, kann – nach Ausschluss anderer Ursachen – zum Beispiel gefolgert werden, dass ein unberücksichtigter Einflussfaktor die Fermentationen beeinflusst.

Ein anderer Ansatz für das „Lernen aus Fehlern“ kann sich aus der Sequentiellen Parameteridentifikation ergeben, siehe Kapitel 3.3.1: Wenn, wie für die Sequentielle Parameteridentifikation vorausgesetzt, die Menge und Qualität der Messdaten ausreichend ist, um eine zuverlässige Interpolation zu erhalten, können unter Umständen die einzelnen Teile eines Differentialgleichungssystems mit Hilfe der Sequentiellen Parameteridentifikation sogar auf Modellstrukturfehler untersucht werden. Lassen sich beispielsweise alle Reaktionen eines Modells bis auf die Produktsynthese wie beschrieben durch Messdateninterpolationen ersetzen, und die Parameteridentifikation kann dennoch keine Parameterkombination ermitteln, um die Produktsynthese korrekt zu beschreiben, muss der Modellfehler mit hoher Wahrscheinlichkeit im Bereich der Produktsynthese-Reaktion zu finden sein.

Zukünftige Anwendungen für Multimodelle sind zum Beispiel auch in der Prozessregelung vorstellbar: Für eine nichtlineare modellprädiktive Regelung (NMPC) kann es eine lohnenswerte Strategie sein, mehrere Modellkandidaten zu verwenden, die jeweils verschiedene Phasen der Experimente optimal beschreiben. Auch für eine Prozessüberwachung können verschiedene Modelle gleichzeitig zum Einsatz kommen, die neben dem planmäßigen Prozessverlauf auch bestimmte Störungen durch Fehler beschreiben. So kann unter Umständen nicht nur ein Fehler schneller erkannt, sondern parallel dazu gleich diagnostiziert werden. Diese Anwendungen sind nicht alle neu, aber durch die automatische Programmierfähigkeit und die standardisierte Modellstrukturdefinition des Modellstrukturgenerators kann der notwendige Aufwand erheblich reduziert werden.

Die Methodik des Modellstrukturgenerators ist nicht auf die Erzeugung von Modellen für biotechnologische Prozesse beschränkt. Wenn eine Anpassung der Modellstrukturdefinition möglich ist, kann die Software prinzipiell auch für andere Prozesse eingesetzt werden, um Modelle automatisch zu programmieren. Anwendungsfälle, bei denen innerhalb einer Modellstruktur Funktionsteile beziehungsweise mathematische Terme ausgetauscht werden sollen, können darüber hinaus zusätzlich Nutzen aus der automatischen Erzeugung der Modellvarianten ziehen.

A Anhang

Das, wobei unsere Berechnungen versagen, nennen wir Zufall.

(Albert Einstein)

A.1 Beispielmodell

A.1.1 Mikroorganismus: *Paenibacillus polymyxa*

Die realen Experimente, die in Kapitel 5 beschrieben sind, stellen Fermentationen mit einem Wildtyp des Stammes *Paenibacillus polymyxa* dar. Dieser wurde vom Fachgebiet Mess- und Regelungstechnik als erster Beispiel-Organismus für die automatische Modellierung ausgewählt. Er ist interessant, da er ein neuartiges Antibiotikum, Macrolactin[1](ML), produziert [57]. Ein Prozess zur optimierten Macrolactin-Produktion war noch nicht bekannt. Die Messung des Macrolactin-Gehalts in Überstandsproben wurde mit Hilfe einer HPLC (High Performance Liquid Chromotography) durchgeführt.

Für die Modellierung der Fermentationen wurden Modellstrukturen verschiedener Komplexität getestet, die den jeweiligen Kenntnisstand widerspiegelten. Erste Modellfamilien mit unstrukturierten Modellkandidaten modellierten zunächst die Substratabhängigkeit des Wachstums mit verschiedenen Formalkinetiken. Durch die Kombination der einzelnen Varianten wurden mit Hilfe des Modellstrukturgenerators über 2000 Modellkandidaten untersucht. Den Verbrauch der Substrate wie auch das Zellwachstum konnten viele Modellkandidaten trotz der unterschiedlichen mathematischen Beschreibung gut abbilden.

[1] Die Macrolactine sind genaugenommen eine ganze Gruppe von Verbindungen mit antimikrobiellen Eigenschaften, die Gegenstand aktueller Forschung sind. Für den prinzipiellen Einsatz der automatischen Modellierung war der genaue von *P. polymyxa* produzierte Typ unwesentlich und war am Anfang der Forschungsarbeit auch nicht bekannt. Später stellte sich heraus, dass es sich um das Macrolactin-D handelte. Die Unterscheidung der Macrolactin-Typen war mit den Mitteln des Fachgebiets nicht möglich, aber wie erwähnt auch nicht notwendig.

Die Synthese des Produkts, das einem Sekundärmetabolismus entstammt, konnte mit derart einfachen Modellen allerdings nicht zufriedenstellend abgebildet werden. Nach einer eingehenden Analyse der Messdaten wurden im nächsten Schritt Speicher für Glucose und/oder Phosphat als zusätzliche Zustände in die Modellstruktur eingefügt. Die automatisch erzeugten rund 1000 Modellkandidaten zeigten eine verbesserte Beschreibung der Wachstumsphase, wenn auch nur eine geringfügige. Die Macrolactin-Synthese konnte dagegen nicht verbessert werden: Der Beginn der Synthese, zum Beispiel, hing offenbar nicht von den verwendeten Modellzuständen ab.

Zuletzt wurde daher die Modellstruktur um Kompartimente erweitert, um den Start der Macrolactin-Produktion besser abbilden zu können. Als zusätzliche Zustände wurden DNS, RNS und die Gesamtproteinmasse (Pr) hinzugenommen, sowie die Zustände Nukleotide (Nu), Aminosäuren (As) und zelluläre Strukturelemente (U). Für die drei zuletzt genannten Zustände waren keine Messdaten verfügbar. Schnell wurde erkannt, dass durch die Erweiterungen auf einen Phosphat-Speicher verzichtet werden konnte. Ein Glucose-Speicher musste jedoch aufgrund von Beobachtungen weiterhin berücksichtigt werden. Einen solchen, in Form eines Exopolysaccharids (EPS), bestätigten Projektpartner später.

Um ein gut passendes Modell zu finden, wurden mit dem Modellstrukturgenerator mehrere, leicht unterschiedliche Modellstrukturen untersucht. Das beste Modell gehörte schließlich zu einer Gruppe von 144 Modellkandidaten, mit jeweils 13 Zuständen und zwölf definierten Reaktionen. Sie sind in Tabelle A.1 zusammen mit den Bezeichnungen für die jeweiligen Reaktionsgeschwindigkeiten aufgeführt.

Einige von den Reaktionen insbesondere für die nicht gemessenen Modellzustände, entstammen allgemeinen biologischen Verständnis, wie zum Beispiel die Synthese von

$C + Am \xrightarrow{r_{As}} As$	(a)	$Nu \xrightarrow{r_{DNS}} DNS$	(g)
$C + Ph + As \xrightarrow{r_{Nu}} Nu$	(b)	$Pr \xrightarrow{r_{zPr}} As + U$	(h)
$C \xrightarrow{r_{U}} U$	(c)	$RNS \xrightarrow{r_{zRNS}} Nu + U$	(i)
$C \xrightarrow{r_{ML}} ML$	(d)	$DNS \xrightarrow{r_{zDNS}} U$	(j)
$As \xrightarrow{r_{Pr}} Pr$	(e)	$C \xrightarrow{r_{EPS}} EPS$	(k)
$Nu \xrightarrow{r_{RNS}} RNS$	(f)	$EPS \xrightarrow{r_{zEPS}} C$	(l)

Tabelle A.1: Das für den *P. polymyxa* verwendete Reaktionschema. Abkürzungen sind im Text erläutert

Makromolekülen (DNS, RNS, Pr) aus Vorstufen (Nu, Aa). Die Reaktionen h), i), j) und l) stellen entsprechende Zerfallsreaktionen dar. Die übrigen sind aus einer Analyse der Experimente und den vorherigen Modellen abgeleitet worden. Das Ergebnis der Modellierung ist in Gleichung A.6 zusammengefasst.

Für eine detailliertere Beschreibung der biologischen Hintergründe des Modells sei auf [81] verwiesen. Hier sollen nur der Einsatz der automatischen Modellierung und ihre Ergebnisse vorgestellt werden.

Innerhalb der zuletzt ausgewählten Modellfamilie wurde

- der Zerfall von RNS abhängig von Nukleotiden durch zwei Alternativen von inhibierenden,
- der Zerfall von Proteinen abhängig von Ammonium durch zwei Alternativen von inhibierenden,
- die Produktion von Macrolactin abhängig von Aminosäuren durch drei Alternativen von limitierenden,
- die Produktion von Macrolactin abhängig von Glucose durch drei Alternativen von limitierenden und
- die Produktion von Macrolactin abhängig von Nukleotiden durch zwei Alternativen von hybriden (wechselweise limitierend und inhibierend wirkenden)

Formalkinetiken beschrieben. Außerdem wurden zwei Alternativen für die P-Faktoren für die Reaktionsgeschwindigkeiten der Macrolactin-Produktion und der Zellerhaltung untersucht.

Das Volumen wurde durch die folgende Gleichung bestimmt:

$$V(t) = u_{\mathrm{Am}}(t) + u_{\mathrm{Ph}}(t) + u_{\mathrm{C}}(t) \tag{A.1}$$

Im Modell galten zunächst alle formalen Kinetiken für die dynamischen Reaktisengeschwindigkeiten (μ) als unbekannt, ebenso die pseudo-stöchiometrischen Koeffizienten (Y). Für letztere existierten aus Erhaltungsgründen dennoch einige algebraische Beziehungen:

$$Y_1 = 1 + Y_{\mathrm{PhNu}} + Y_{\mathrm{CNu}} \tag{A.2}$$

$$Y_2 = Y_{\mathrm{AmAs}} - 1 \tag{A.3}$$

$$Y_3 = 1 - Y_{\mathrm{AsPr}} \tag{A.4}$$

$$Y_4 = 1 + Y_{\mathrm{NuRNS}} \tag{A.5}$$

$$
\begin{aligned}
\frac{d}{dt}\begin{pmatrix} m_{\mathrm{DNS}}(t) \\ m_{\mathrm{RNS}}(t) \\ m_{\mathrm{Pr}}(t) \\ m_{\mathrm{As}}(t) \\ m_{\mathrm{Nu}}(t) \\ m_{\mathrm{U}}(t) \\ m_{\mathrm{Am}}(t) \\ m_{\mathrm{Ph}}(t) \\ m_{\mathrm{C}}(t) \\ m_{\mathrm{EPS}}(t) \\ m_{\mathrm{ML}}(t) \end{pmatrix} &= \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ c_{\mathrm{Am,f}} \cdot u_{\mathrm{Am}}(t) \\ c_{\mathrm{Ph,f}} \cdot u_{\mathrm{Ph}}(t) \\ c_{\mathrm{C,f}} \cdot u_{\mathrm{C}}(t) \\ 0 \\ 0 \end{pmatrix} - V(t) \cdot \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ \mu_{\mathrm{M}}(t) \\ 0 \\ 0 \end{pmatrix} \\
&+ V(t) \cdot \begin{pmatrix} 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 \\ 0 & 0 & 0 & 0 & -1 & Y_{\mathrm{AsPr}} \\ -1 & 0 & -1 & Y_{\mathrm{NuRNS}} & 0 & 0 \\ 0 & 1 & 0 & Y_4 & 0 & Y_3 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \cdot \begin{pmatrix} r_{\mathrm{DNS}}(t) \\ r_{\mathrm{zDNS}}(t) \\ r_{\mathrm{RNS}}(t) \\ r_{\mathrm{zRNS}}(t) \\ r_{\mathrm{Pr}}(t) \\ r_{\mathrm{zPr}}(t) \end{pmatrix} \\
&+ V(t) \cdot \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & Y_1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ -Y_{\mathrm{AmAs}} & 0 & 0 & 0 & 0 & 0 \\ 0 & -Y_{\mathrm{PhNu}} & 0 & 0 & 0 & 0 \\ Y_2 & -Y_{\mathrm{CNu}} & -1 & -1 & 1 & -1 \\ 0 & 0 & 0 & 1 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} r_{\mathrm{As}}(t) \\ r_{\mathrm{Nu}}(t) \\ r_{\mathrm{U}}(t) \\ r_{\mathrm{EPS}}(t) \\ r_{\mathrm{zEPS}}(t) \\ r_{\mathrm{ML}}(t) \end{pmatrix}
\end{aligned} \tag{A.6}
$$

A.1.2 Beurteilung der Modellierung

Das reine Wachstum konnte mit einer Vielzahl von den oben angedeuteten Modellkandidaten gut beschrieben werden. Die automatische Modellierung half hier mit der Auf-

findung der besten Kombination von Formalkinetiken. Obwohl nicht nur eine einzelne herausragende Kombination gefunden wurde, kann festgestellt werden, dass in diesem Beispiel nur etwa ein Prozent der untersuchten Kandidaten zur Spitzengruppe gehörte. Entsprechend gering wäre die Chance gewesen, per Hand zufällig einen der Spitzenkandidaten zu programmieren. Andererseits konnte auch mit dem Kompartimentmodell der beobachtete Start der Macrolactin-Synthese nicht korrekt beschrieben werden, obwohl die Anpassung für mehrere, verschiedene Experimente quantitativ überzeugend war. Die Validierungsexperimente zeigten größere Abweichungen, was sich auch in der Realisierung des Versuchs mit der Multimodell-Trajektorienplanung offenbarte, siehe Abbildung 5.7. Die Grundlage für ein zukünftig noch zu verbesserndes Modell stellen die Ergebnisse jedoch dar. Die teilweise nur geringen Güteunterschiede zwischen den einzelnen Modellkandidaten scheinen den erhöhten Aufwand bei der Verwendung von Multimodellen nicht überall zu rechtfertigen. Allerdings sollte nicht die Güte der Anpassung bei Identifikationsexperimenten für diese Beurteilung herangezogen werden, sondern die Güte der Voraussage der Validierungsexperimente.

Bei diesem Fallbeispiel existierten jedoch mehrere Gründe, warum die Anpassung beziehungsweise Validierung speziell der Macrolactin-Synthese problematisch war: Als erstes fiel auf, dass bei der Wiederholung einzelner Messwertbestimmungen die Messwerte nicht reproduzierbar waren. Daraufhin wurde die Methode zur Messung der Macrolactin-Konzentration im Laufe der Zeit verändert. Es wurden zwar Anstrengungen unternommen, die Vergleichbarkeit zu gewährleisten, aber bei späterer Analyse kamen Zweifel an deren Erfolg auf. Es zeigte sich schließlich, dass Macrolactin offenbar nicht stabil war: Wiederholte Analysen der gleichen Proben ergaben zum Teil deutlich geringere Konzentrationswerte. Da die Macrolactine prinzipiell stabile Verbindungen sind[2], kann nur vermutet werden, dass es Wechselwirkungen mit den restlichen Bestandteilen der Probe gab. Eine optimierte Probenaufbereitung hätte die Messergebnisse und damit vermutlich auch die Ergebnisse der Modellierung verbessern können.

Als zweiter Grund ist zu nennen, dass *P. polymyxa* ein Sporenbildner ist. Zwar war es das Ziel durch die Prozessführung die Sporenbildung zu vermeiden, allerdings konnte die Ursache für den Beginn der Sporenbildung nicht eindeutig erkannt werden beziehungsweise getrennt werden vom Beginn der Produktbildung: In unterschiedlichen Mengen wurden Sporen in allen Fermentationen gefunden, in denen auch Macrolactin produ-

[2] Persönliche Mitteilung von G. Molinari, ZEIM, Helmholtz Zentrum für Infektionsforschung, Braunschweig (09.01.2014)

ziert wurde. Einfache Versuche, den Vorgang der Sporulation in das Wachstumsmodell aufzunehmen scheiterten. Die Kopplung verschiedener Modellreaktionen an die DNS-Konzentration brachten keine Besserung. Ein möglicher Grund dafür ist, dass die zur Verfügung stehenden Messwerte der DNS unerwartet starke Schwankungen aufwiesen, wofür keine Ursache ermittelt werden konnte. Die Modelle waren nicht in der Lage, diese Schwankungen zu simulieren. Eine (unüberprüfte) Vermutung ist, dass die Schwankungen im Zusammenhang mit der Sporenbildung hervorgerufen wurden: In der Phase der Sporenbildung wird die DNS zunächst verdoppelt, bevor die Zellwand der Sporen so stabil wird, dass sich ihr Inneres einer Messung entzieht. In der Folge davon könnten die Messwerte vorübergehend zu hoch ausfallen, bevor sie dann wieder fallen.

A.1.3 Verwendete Modellstrukturdefinition

Im Folgenden sind die wichtigsten Elemente der Modellstrukturdefinition aufgeführt, die zum Erstellen von Modellkandidaten für den Stamm *P. polymyxa* mit Hilfe des Modellstrukturgenerators notwendig sind.

$$\texttt{.rName} = \text{„}r_{\mathrm{DNS}}, r_{\mathrm{zDNS}}, r_{\mathrm{RNS}}, r_{\mathrm{zRNS}}, r_{\mathrm{Pr}}, r_{\mathrm{zPr}}, r_{\mathrm{As}}, r_{\mathrm{Nu}}, r_{\mathrm{U}}, r_{\mathrm{EPS}}, r_{\mathrm{zEPS}}, r_{\mathrm{ML}}\text{“} \tag{A.7}$$

$$\texttt{.K} = \begin{bmatrix} 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & k & 1 & -1 & 0 & 0 & 0 & 0 \\ -1 & 0 & -1 & k & 0 & 0 & 0 & k & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & k & 0 & k & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -k & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -k & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & k & -k & -1 & -1 & 1 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{A.8}$$

$$
\texttt{.R} = \begin{bmatrix}
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & -1 & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
1 & \cdot & \cdot & \cdot & 1 & \cdot & \cdot & 1 & 1 & \cdot & \cdot & 2 \\
1 & -1 & 1 & 1 & \cdot & \cdot & 1 & \cdot & 1 & \cdot & \cdot & -1 \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & -1 & 1 & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 2 & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 1 & \cdot & \cdot & 2 & -1 & 2 \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot
\end{bmatrix} \tag{A.9}
$$

$$
\texttt{.P} = \begin{bmatrix}
1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix} \tag{A.10}
$$

A.1.4 Automatisch erzeugte Modelldateien

Im Folgenden werden die Matlab-Programmlistings der vom Modellstrukturgenerator automatisch erstellten Dateien für das im Abschnitt A.1.1 aufgeführte Modell wiedergegeben. Dabei handelt es sich um die `SystemInit` mit den allgemeinem Modelldefinitionen und -informationen, `Symb_f` mit den Zustandsgleichungen, `Symb_h` mit den Messgleichungen, `Parameter_Fcn` mit den Parameterdefinitionen und -werten, `Rauschen` mit den Angaben zur Genauigkeit der Messgrößen und um `x2x_` für die Definition der diskreten Zustandsänderungen.

Die hier aufgelisteten Programme sind weitgehend im Originalzustand, das heißt, es sind keine zusätzlichen Kommentare eingefügt worden. Allerdings sind zu lange Zeilen gekürzt oder umgebrochen worden.

SystemInit

```
function SystemInit

% P. polymyxa
% 12 Zustände; 8 Messgrößen
% Dieser Modellordner wurde erstellt von: Norman Violet

global SYSTEM;

% Name für die Modellklasse
SYSTEM.ModellName   = 'P.polymyxa';
SYSTEM.Info         = 'Beste Anpassung für P.p., veröffentlicht CAB2010';

SYSTEM.ModellAnzahl = 72;
ABC_Waiting('start','Lade SystemInit');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Definitionen für Modellklasse
%
%%%%% Einstellungen für Erweiterungen %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
SYSTEM.SETTINGS.ReportSave      = 1;  % Datei mit Reports schreiben: 0-Nein 1-Ja
SYSTEM.SETTINGS.Symbolisch      = 1;  % symb_f symbolisch ausgewertbar: 0-Nein, 1-Ja
SYSTEM.SETTINGS.Erweiterungen   = { 'Systemidentifikation.Parameteridentifikation';
                                   % 'Prozessfuehrung.Trajektorienplanung';
                                   % 'Prozessfuehrung.Sigmapunkt Trajektorienplanung'
                                  };
SYSTEM.SETTINGS.JACOBIANS       = {'dfdx','dfdp', 'dfdu', 'dx_dx','dx0dp'};
SYSTEM.SETTINGS.DATABASE.use_DB = true;

% Zeiteinheit
SYSTEM.tEinheit     = 'h';
% Simulationsgeschwindigkeit
SYSTEM.TMP.Throttle = 4;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% MODELL-Beschreibungen:
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Wahrscheinlichkeiten für die Modellkandidaten
SYSTEM.MM.Wahrscheinlichkeit = [ ...
1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  ...
1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  ...
1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1 ];
```

```
%% Schleife über alle Modelle
for mod_id = 1 : 72,

if mod(mod_id,50) == 1, % alle 50 Modelle Zeile einfügen
   ABC_Report(['Lade SystemInit ...  Modelle ' num2str(mod_id) '...'
       num2str(min(SYSTEM.ModellAnzahl,mod_id+49))]);
     ABC_Waiting;
end

ABC_Select_Modell(mod_id,true);

% Anzahl der Zustandsgrößen
SYSTEM.n = 12;
% Anzahl der Messgrößen
SYSTEM.m = 8;
% Anzahl der Stellgrößen
SYSTEM.p = 10;
% Anzahl der diskreten Zustandsänderungsarten
SYSTEM.q = 4;

%% Bezeichungen der Zustandsgrößen
SYSTEM.ZDGL.xName(1,1)     = {'m_{DNS}'};   SYSTEM.ZDGL.xEinheit(1,1) = {'g'};
SYSTEM.ZDGL.xLabelPos(1,1) = {'lo'};
SYSTEM.ZDGL.xTyp(1,1)      = {'KD'};

SYSTEM.ZDGL.xName(2,1)     = {'m_{RNS}'};   SYSTEM.ZDGL.xEinheit(2,1) = {'g'};
SYSTEM.ZDGL.xLabelPos(2,1) = {'lo'};
SYSTEM.ZDGL.xTyp(2,1)      = {'KI'};

SYSTEM.ZDGL.xName(3,1)     = {'m_{Pr}'};    SYSTEM.ZDGL.xEinheit(3,1) = {'g'};
SYSTEM.ZDGL.xLabelPos(3,1) = {'lo'};
SYSTEM.ZDGL.xTyp(3,1)      = {'KI'};

SYSTEM.ZDGL.xName(4,1)     = {'m_{As}'};    SYSTEM.ZDGL.xEinheit(4,1) = {'g'};
SYSTEM.ZDGL.xLabelPos(4,1) = {'lo'};
SYSTEM.ZDGL.xTyp(4,1)      = {'KINC'};

SYSTEM.ZDGL.xName(5,1)     = {'m_{Nu}'};    SYSTEM.ZDGL.xEinheit(5,1) = {'g'};
SYSTEM.ZDGL.xLabelPos(5,1) = {'lo'};
SYSTEM.ZDGL.xTyp(5,1)      = {'KIPC'};

SYSTEM.ZDGL.xName(6,1)     = {'m_{U}'};     SYSTEM.ZDGL.xEinheit(6,1) = {'g'};
SYSTEM.ZDGL.xLabelPos(6,1) = {'lo'};
SYSTEM.ZDGL.xTyp(6,1)     = {'K'};

SYSTEM.ZDGL.xName(7,1)     = {'m_{Am}'};    SYSTEM.ZDGL.xEinheit(7,1) = {'g'};
SYSTEM.ZDGL.xLabelPos(7,1) = {'lo'};
SYSTEM.ZDGL.xTyp(7,1)      = {'SN'};
```

```
SYSTEM.ZDGL.xName(8,1)     = {'m_{Ph}'};    SYSTEM.ZDGL.xEinheit(8,1) = {'g'};
SYSTEM.ZDGL.xLabelPos(8,1) = {'lo'};
SYSTEM.ZDGL.xTyp(8,1)      = {'SP'};

SYSTEM.ZDGL.xName(9,1)     = {'m_{C}'};     SYSTEM.ZDGL.xEinheit(9,1) = {'g'};
SYSTEM.ZDGL.xLabelPos(9,1) = {'lo'};
SYSTEM.ZDGL.xTyp(9,1)      = {'SC'};

SYSTEM.ZDGL.xName(10,1)     = {'m_{EPS}'};  SYSTEM.ZDGL.xEinheit(10,1) = {'g'};
SYSTEM.ZDGL.xLabelPos(10,1) = {'lo'};
SYSTEM.ZDGL.xTyp(10,1)      = {'IC'};

SYSTEM.ZDGL.xName(11,1)     = {'m_{ML}'};   SYSTEM.ZDGL.xEinheit(11,1) = {'g'};
SYSTEM.ZDGL.xLabelPos(11,1) = {'lo'};
SYSTEM.ZDGL.xTyp(11,1)      = {'Z'};

SYSTEM.ZDGL.xName(12,1)     = {'V'};        SYSTEM.ZDGL.xEinheit(12,1) = {'l'};
SYSTEM.ZDGL.xLabelPos(12,1) = {'lo'};
SYSTEM.ZDGL.xTyp(12,1)      = {'V'};

% Zustandsbeschränkungen
SYSTEM.ZDGL.xmin = [eps;eps;eps;eps;eps;eps;eps;eps;eps;eps;eps;eps];
SYSTEM.ZDGL.xmax = [Inf;Inf;Inf;Inf;Inf;Inf;Inf;Inf;Inf;Inf;Inf;Inf];

% Ungleichungsbedingungen für den Systemzustand
% in der Form xA*x<=xb
% SYSTEM.ZDGL.xA = []
% SYSTEM.ZDGL.xb = []

%% Def. der Anfangsbed.
SYSTEM.ZDGL.t0 = 0;
SYSTEM.ZDGL.x0 = [0;0;0;0;0;0;0;0;0;0;0;10.01];

SYSTEM.ZDGL.x0_berechnen    = {''; ''; ''; ''; ''; ''; ''; ''; ''; ''; ''; ''};

SYSTEM.ZDGL.P0 = diag([0.001 0.001 0.001 0.001 0.001 0.001 0.001 0.001 ...
                       0.001 0.001 0.001 0.001]);

%% Bezeichnungen der kontinuierlichen Stellgrößen
SYSTEM.ZDGL.uName(1,1)     = {'u_{am}'};      SYSTEM.ZDGL.uEinheit(1,1) = {'l/h'};
SYSTEM.ZDGL.uLabelPos(1,1) = {'lo'};          SYSTEM.ZDGL.uTyp(1,1)     = {''};

SYSTEM.ZDGL.uName(2,1)     = {'c_{feed,am}'}; SYSTEM.ZDGL.uEinheit(2,1) = {'g/l'};
SYSTEM.ZDGL.uLabelPos(2,1) = {'lo'};          SYSTEM.ZDGL.uTyp(2,1)     = {''};

SYSTEM.ZDGL.uName(3,1)     = {'u_{ph}'};      SYSTEM.ZDGL.uEinheit(3,1) = {'l/h'};
SYSTEM.ZDGL.uLabelPos(3,1) = {'lo'};          SYSTEM.ZDGL.uTyp(3,1)     = {''};
```

```
SYSTEM.ZDGL.uName(4,1)     = {'c_{feed,ph}'}; SYSTEM.ZDGL.uEinheit(4,1) = {'g/l'};
SYSTEM.ZDGL.uLabelPos(4,1) = {'lo'};          SYSTEM.ZDGL.uTyp(4,1)     = {''};

SYSTEM.ZDGL.uName(5,1)     = {'u_{c}'};       SYSTEM.ZDGL.uEinheit(5,1) = {'l/h'};
SYSTEM.ZDGL.uLabelPos(5,1) = {'lo'};          SYSTEM.ZDGL.uTyp(5,1)     = {''};

SYSTEM.ZDGL.uName(6,1)     = {'c_{feed,c}'};  SYSTEM.ZDGL.uEinheit(6,1) = {'g/l'};
SYSTEM.ZDGL.uLabelPos(6,1) = {'lo'};          SYSTEM.ZDGL.uTyp(6,1)     = {''};

SYSTEM.ZDGL.uName(7,1)     = {'BASE'};        SYSTEM.ZDGL.uEinheit(7,1) = {'l/h'};
SYSTEM.ZDGL.uLabelPos(7,1) = {'lo'};          SYSTEM.ZDGL.uTyp(7,1)     = {''};

SYSTEM.ZDGL.uName(8,1)     = {'ACID'};        SYSTEM.ZDGL.uEinheit(8,1) = {'l/h'};
SYSTEM.ZDGL.uLabelPos(8,1) = {'lo'};          SYSTEM.ZDGL.uTyp(8,1)     = {''};

SYSTEM.ZDGL.uName(9,1)     = {'AFOAM'};       SYSTEM.ZDGL.uEinheit(9,1) = {'l/h'};
SYSTEM.ZDGL.uLabelPos(9,1) = {'lo'};          SYSTEM.ZDGL.uTyp(9,1)     = {''};

SYSTEM.ZDGL.uName(10,1)    = {'Abdampf'};     SYSTEM.ZDGL.uEinheit(10,1)= {'l/h'};
SYSTEM.ZDGL.uLabelPos(10,1)= {'lo'};          SYSTEM.ZDGL.uTyp(10,1)    = {''};

SYSTEM.ZDGL.umin           = zeros(size(SYSTEM.ZDGL.uName));
SYSTEM.ZDGL.umax           = ones(size(SYSTEM.ZDGL.uName)) * inf;
SYSTEM.ZDGL.uOrder         = 0;

%% Bezeichnungen der Messgrößen
SYSTEM.MGL.yName(1,1)     = {'c_{X}'};  SYSTEM.MGL.yEinheit(1,1) = {'g/l'};
SYSTEM.MGL.yLabelPos(1,1) = {'lo'};     SYSTEM.MGL.yTyp(1,1)     = {'X'};

SYSTEM.MGL.yName(2,1)     = {'c_{DNS}'};SYSTEM.MGL.yEinheit(2,1) = {'g/l'};
SYSTEM.MGL.yLabelPos(2,1) = {'lo'};     SYSTEM.MGL.yTyp(2,1)     = {'KD'};

SYSTEM.MGL.yName(3,1)     = {'c_{RNS};  SYSTEM.MGL.yEinheit(3,1) = {'g/l'};
SYSTEM.MGL.yLabelPos(3,1) = {'lo'};     SYSTEM.MGL.yTyp(3,1)     = {'KI'};

SYSTEM.MGL.yName(4,1)     = {'c_{Pr}'}; SYSTEM.MGL.yEinheit(4,1) = {'g/l'};
SYSTEM.MGL.yLabelPos(4,1) = {'lo'};     SYSTEM.MGL.yTyp(4,1)     = {'KI'};

SYSTEM.MGL.yName(5,1)     = {'c_{Am}'}; SYSTEM.MGL.yEinheit(5,1) = {'g/l'};
SYSTEM.MGL.yLabelPos(5,1) = {'lo'};     SYSTEM.MGL.yTyp(5,1)     = {'SN'};

SYSTEM.MGL.yName(6,1)     = {'c_{Ph}'}; SYSTEM.MGL.yEinheit(6,1) = {'g/l'};
SYSTEM.MGL.yLabelPos(6,1) = {'lo'};     SYSTEM.MGL.yTyp(6,1)     = {'SP'};

SYSTEM.MGL.yName(7,1)     = {'c_{C}'};  SYSTEM.MGL.yEinheit(7,1) = {'g/l'};
SYSTEM.MGL.yLabelPos(7,1) = {'lo'};     SYSTEM.MGL.yTyp(7,1)     = {'SC'};
```

```
SYSTEM.MGL.yName(8,1)     = {'c_{ML}'}; SYSTEM.MGL.yEinheit(8,1) = {'g/l'};
SYSTEM.MGL.yLabelPos(8,1) = {'lo'};     SYSTEM.MGL.yTyp(8,1)     = {'Z'};

% Messgrößenbeschränkungen für Simulation
SYSTEM.MGL.ymin = zeros(SYSTEM.m,1);
SYSTEM.MGL.ymax = inf(SYSTEM.m,1);

SYSTEM.MGL.MessdatenFile = 'Messdaten_file';
SYSTEM.MGL.Cy            = @Rauschen;

%% Bezeichnungen der zeitdiskreten Zustandsänderungsarten
SYSTEM.ZDGL.zName(1,1)    = {'Manuelle Vollprobe'};
SYSTEM.ZDGL.zEinheit(1,1) = {'l'};

SYSTEM.ZDGL.zName(2,1)    = {'Manuelle Ueberstandsprobe'};
SYSTEM.ZDGL.zEinheit(2,1) = {'l'};

SYSTEM.ZDGL.zName(3,1)    = {'Autosampler'};
SYSTEM.ZDGL.zEinheit(3,1) = {'l'};

SYSTEM.ZDGL.zName(4,1)    = {'FIAVolumen'};
SYSTEM.ZDGL.zEinheit(4,1) = {'l'};

%% Modellfunktionen und symbolische Auswertung:
SYSTEM.ZDGL.ParameterFile = ['Parameter_FCN_' num2str(mod_id)];
SYSTEM.ZDGL.fSymb         = ['symb_f_' num2str(mod_id)];
SYSTEM.MGL.hSymb          = 'symb_h_1';
SYSTEM.ZDGL.zFile         = 'x2x__1';

SYSTEM.ZDGL.Substitutions = ''; % Z.B. {'x(1)+x(2)+x(3)+x(4)+x(5)+x(6)'
SYSTEM.ZDGL.usesimple     = false;

%% Lösungsalgorithmus
SYSTEM.ZDGL.SOLVER.Name         = 'ode15s';
SYSTEM.ZDGL.SOLVER.OPTIONS      = odeset('Rel',1e-006,'Abs',1e-008, ...
                                         'InitialStep', 0.001);
SYSTEM.ZDGL.SOLVER.OPTIONS.hmin = 1e-010;
SYSTEM.ZDGL.SOLVER.COPTIONS     = struct('Solver','ode15s','RelTol',1e-006, ...
                                         'AbsTol', 1e-008,'MinimalStep',1e-010, ...
                                           'InitialStep',0.001,'use_F',0);
SYSTEM.ZDGL.SOLVER.useSparse    = 0;

% Numerische Genauigkeit
SYSTEM.ZDGL.Genauigkeit = 20;   % Genauigkeit der Konstanten im Modell
SYSTEM.MGL.Genauigkeit  = 20;   % Genauigkeit der Konstanten in der Messgleichung

%% Private Definitionen
% SYSTEM.PRIVATE.<...> = ....
```

```
SYSTEM.SoundDir = fullfile(SYSTEM.ABCDir{1},'sound');

%% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

end % Schleife über alle Modelle

ABC_Waitbar('close');

end
```

Symb_f

```
function xdot = f(t,x,u,Parameter)

MiMe_                   =  Parameter.MiMe;
AibaSchalter_           =  Parameter.AibaSchalter;
MiMeJeruSchalter_       =  Parameter.MiMeJeruSchalter;
Jeru_Quorum_            =  Parameter.Jeru_Quorum;
Moser_                  =  Parameter.Moser;
InhibMoser_             =  Parameter.InhibMoser;
Y                       =  Parameter.Y;
MuMax                   =  Parameter.MuMax;
Aiba_                   =  Parameter.Aiba;
Haldane_                =  Parameter.Haldane;
AibaSchalter_           =  Parameter.AibaSchalter;

% --- Automatisch erzeugte Größen ----
m_DNA          =  x(1);
m_RNA          =  x(2);
m_Pr           =  x(3);
m_As           =  x(4);
m_Nu           =  x(5);
m_U            =  x(6);
m_am           =  x(7);
m_ph           =  x(8);
m_c            =  x(9);
m_EPS          =  x(10);
m_ML           =  x(11);
V              =  x(12);
Vx             =  m_DNA + m_RNA + m_Pr + m_As + m_Nu + m_U;
c_x            =  Vx / V;
g_DNA          =  m_DNA / Vx;
g_RNA          =  m_RNA / Vx;
g_Pr           =  m_Pr / Vx;
g_As           =  m_As / Vx;
g_Nu           =  m_Nu / Vx;
g_U            =  m_U / Vx;
```

```
c_am          =  m_am / V;
c_ph          =  m_ph / V;
c_c           =  m_c / V;

am_Feed       =  u(1) * u(2);
ph_Feed       =  u(3) * u(4);
c_Feed        =  u(5) * u(6);

% rDNA  (Nu --> DNA)
rDNA =  MuMax.rDNA ...
     * MiMe([g_As],[MiMe_.k.As.rDNA]) ...
     * MiMe([g_Nu],[MiMe_.k.Nu.rDNA]);

% rzDNA  (DNA --> U)
rzDNA =  MuMax.rzDNA ...
      * AibaSchalter([g_Nu],[AibaSchalter_.k.Nu.rzDNA]);

% rRNA  (Nu --> RNA)
rRNA =  MuMax.rRNA ...
     * MiMe([g_Nu],[MiMe_.k.Nu.rRNA]);

% rzRNA  (RNA --> Nu + U)
rzRNA =  MuMax.rzRNA ...
      * Aiba(g_Nu,[Aiba_.k.Nu.rzRNA]);

% rPr  (As --> Pr)
rPr =  MuMax.rPr ...
    * MiMe([g_As],[MiMe_.k.As.rPr]);

% rzPr  (Pr --> As + U)
rzPr =  MuMax.rzPr ...
     * Aiba(c_am,[Aiba_.k.am.rzPr]);

% rAs  (am + c --> As)
rAs =  MuMax.rAs ...
    * MiMe([g_Nu],[MiMe_.k.Nu.rAs]) ...
    * MiMe([c_am],[MiMe_.k.am.rAs]) ...
    * MiMe([c_c],[MiMe_.k.c.rAs]);

% rNu  (As + ph + c --> Nu)
rNu =  MuMax.rNu ...
    * MiMe([g_As],[MiMe_.k.As.rNu]) ...
    * JeruSchalter([c_ph],[MiMeJeruSchalter_.km.ph.rNu,MiMeJeruSchalter_.kj.ph.rNu]);

% rU  (c --> U)
rU =  MuMax.rU ...
   * MiMe([g_As],[MiMe_.k.As.rU]) ...
   * MiMe([g_Nu],[MiMe_.k.Nu.rU]);
```

```
% rEPS  (c --> EPS)
rEPS =  MuMax.rEPS ...
     * Jeru_Quorum([c_DNA],[Jeru_Quorum_.k.DNA.rEPS]) ...
     * Moser([c_c],[Moser_.k.c.rEPS, Moser_.lambda.c.rEPS]);

% rzEPS  (EPS --> c)
rzEPS =  MuMax.rzEPS ...
      * InhibMoser([c_c],[InhibMoser_.k.c.rzEPS, InhibMoser_.lambda.c.rzEPS]);

% rZ  (c --> ML)
rZ =  MuMax.rZ ...
   * Haldane(g_As,[Haldane_.k_m.As.rZ, Haldane_.k_i.As.rZ]) ...
   * AibaSchalter(g_Nu,[AibaSchalter_.k.Nu.rZ]) ...
   * Haldane(c_c,[Haldane_.k_m.c.rZ, Haldane_.k_i.c.rZ]);

%% DGL-Gleichungen
 % m_DNA
xdot(1) = rDNA * m_DNA - rzDNA * m_DNA;

% m_RNA
xdot(2) = rRNA * m_DNA - rzRNA * m_RNA;

% m_Pr
xdot(3) = rPr * m_RNA - rzPr * m_Pr;

% m_As
xdot(4) = - rPr * m_RNA + Y.As.Pr * rzPr * m_Pr + rAs * m_Pr - rNu * m_Pr;

% m_Nu
xdot(5) = - rDNA * m_DNA - rRNA * m_DNA + Y.Nu.RNA * rzRNA * m_RNA + 2 * ...
          (1 + Y.ph.Nu + Y.c.Nu) * rNu * m_Pr;

% m_U
xdot(6) = rzDNA * m_DNA + (1 - Y.Nu.RNA) * rzRNA * m_RNA + (1 - Y.As.Pr) * ...
          rzPr * m_Pr + rU * m_Pr;

% m_am
xdot(7) = - Y.am.As * rAs * m_Pr + am_Feed;

% m_ph
xdot(8) = - Y.ph.Nu * rNu * m_Pr + ph_Feed;

% m_c
xdot(9) = - (1 - Y.am.As) * rAs * m_Pr - Y.c.Nu * rNu * m_Pr - rU * m_Pr - ...
          rEPS * m_Pr + rzEPS * m_EPS - rZ * m_DNA - Y.Erhaltung * ...
          Erhaltung(c_c,[]) * V_x + c_Feed;
```

```
% m_EPS
xdot(10) = rEPS * m_Pr - rzEPS * m_EPS;

% m_ML
xdot(11) = rZ * m_DNA;

% Volumen
xdot(12) = u(1) + u(3) + u(5) + u(7) + u(8) + u(9) - u(10);

%{
 Info: ;
% Modellstrukturgenerator: Minimalmedien: 0 (bool);
%}
```

Symb_h

```
function y = h(t,x,u,Parameter);

X0        = Parameter.X0;
Y         = Parameter.Y;
MuMax     = Parameter.MuMax;
MiMe_     = Parameter.MiMe;
Aiba_     = Parameter.Aiba;
pHGauss_  = Parameter.pHGauss;

m_X       =  x(1);
m_Am      =  x(2);
m_Ph      =  x(3);
m_Glc     =  x(4);
m_Eth     =  x(5);
m_EPG     =  x(6);
m_CO2     =  x(7);
O2        =  x(8);
pO2       =  x(9);
V         =  x(end);

% --- Automatisch erzeugte Größen ----
c_Am          =  m_Am / V;
c_Ph          =  m_Ph / V;
c_Glc         =  m_Glc / V;
c_Eth         =  m_Eth / V;
c_EPG         =  m_EPG / V;
c_X           =  m_X / V;

% Virtuelle Messgröße (von Hand hinzugefügt)
r_X   =  MuMax.r_X * MiMe([c_Glc],[MiMe_.k.Glc.r_X]);
```

```
y(1)  = c_Am;
y(2)  = c_Ph;
y(3)  = c_Glc;
y(4)  = c_Eth;
y(5)  = c_EPG;
y(6)  = c_X;
y(7)  = m_CO2/(1.26 * V);
y(8)  = O2;
y(9)  = pO2;
y(10) = r_X;
```

Parameter_Fcn

```
function [Parameter,ParNames,x0ParNames,ParColors] = Parameter_FCN_1
MiMe.k.As.rDNA                          =      0.10000000;
MiMe.k.Nu.rDNA                          =      0.10000000;
MuMax.rDNA                              =      0.50000000;
AibaSchalter.k.Nu.rzDNA                 =      3.70000000;
MuMax.rzDNA                             =      0.50000000;
MiMe.k.Nu.rRNA                          =      0.10000000;
MuMax.rRNA                              =      0.50000000;
MuMax.rzRNA                             =      0.50000000;
MiMe.k.As.rPr                           =      0.10000000;
MuMax.rPr                               =      0.50000000;
MuMax.rzPr                              =      0.50000000;
Y.As.Pr                                 =      0.76400000;
MiMe.k.Nu.rAs                           =      0.10000000;
MiMe.k.am.rAs                           =      0.10000000;
MiMe.k.c.rAs                            =      0.10000000;
MuMax.rAs                               =      0.50000000;
MiMe.k.As.rNu                           =      0.10000000;
MiMeJeruSchalter.km.ph.rNu              =      0.25000000;
MiMeJeruSchalter.kj.ph.rNu              =      0.50000000;
MuMax.rNu                               =      0.50000000;
Y.Nu.RNA                                =      0.84800000;
MiMe.k.As.rU                            =      0.10000000;
MiMe.k.Nu.rU                            =      0.10000000;
MuMax.rU                                =      0.50000000;
Y.am.As                                 =      0.70800000;
Y.ph.Nu                                 =      0.97633136;
Y.c.Nu                                  =      0.97041420;
Jeru.k.DNA.rEPS                         =      0.01000000;
Moser.k.c.rEPS                          =      0.50000000;
Moser.lambda.c.rEPS                     =      3.00000000;
MuMax.rEPS                              =      0.50000000;
InhibMoser.k.c.rzEPS                    =      0.50000000;
InhibMoser.lambda.c.rzEPS               =      3.00000000;
```

```
MuMax.rzEPS                          =      0.50000000;
MuMax.rZ                             =      0.50000000;
Y.Erhaltung                          =      1.00000000;
Aiba.k.Nu.rzRNA                      =      1.00000000;
Aiba.k.am.rzPr                       =      1.00000000;
Haldane.k_m.As.rZ                    =      1.00000000;
Haldane.k_i.As.rZ                    =      1.00000000;
AibaSchalter.k.Nu.rZ                 =      1.00000000;
Haldane.k_m.c.rZ                     =      1.00000000;
Haldane.k_i.c.rZ                     =      1.00000000;
Y.Erhaltung             = 1e-6;

%% Definition der freien Parameter
ParNames = { ...
'MiMe.k.As.rDNA'
'MiMe.k.Nu.rDNA'
'MuMax.rDNA'
'AibaSchalter.k.Nu.rzDNA'
'MuMax.rzDNA'
'MiMe.k.Nu.rRNA'
'MuMax.rRNA'
'MuMax.rzRNA'
'MiMe.k.As.rPr'
'MuMax.rPr'
'MuMax.rzPr'
'Y.As.Pr'
'MiMe.k.Nu.rAs'
'MiMe.k.am.rAs'
'MiMe.k.c.rAs'
'MuMax.rAs'
'MiMe.k.As.rNu'
'MiMeJeruSchalter.km.ph.rNu'
'MiMeJeruSchalter.kj.ph.rNu'
'MuMax.rNu'
'Y.Nu.RNA'
'MiMe.k.As.rU'
'MiMe.k.Nu.rU'
'MuMax.rU'
'Y.am.As'
'Y.ph.Nu'
'Y.c.Nu'
'Jeru.k.DNA.rEPS'
'Moser.k.c.rEPS'
'Moser.lambda.c.rEPS'
'MuMax.rEPS'
'InhibMoser.k.c.rzEPS'
'InhibMoser.lambda.c.rzEPS'
'MuMax.rzEPS'
```

```
'MuMax.rZ'
'Y.Erhaltung'
'Aiba.k.Nu.rzRNA'
'Aiba.k.am.rzPr'
'Haldane.k_m.As.rZ'
'Haldane.k_i.As.rZ'
'AibaSchalter.k.Nu.rZ'
'Haldane.k_m.c.rZ'
'Haldane.k_i.c.rZ'};

%% Einhängen
Parameter.MiMe                    = MiMe;
Parameter.AibaSchalter            = AibaSchalter;
Parameter.MiMeJeruSchalter        = MiMeJeruSchalter;
Parameter.Moser                   = Moser;
Parameter.InhibMoser              = InhibMoser;
Parameter.Y                       = Y;
Parameter.MuMax                   = MuMax;
Parameter.Aiba                    = Aiba;
Parameter.Haldane                 = Haldane;
Parameter.AibaSchalter            = AibaSchalter;

%% CONS
Parameter.CONS.min = [ ...
    0.01000000          ,  ... % MiMe.k.As.rDNA
    0.01000000          ,  ... % MiMe.k.Nu.rDNA
    0.00000000          ,  ... % MuMax.rDNA
    0.01000000          ,  ... % AibaSchalter.k.Nu.rzDNA
    0.00000000          ,  ... % MuMax.rzDNA
    0.01000000          ,  ... % MiMe.k.Nu.rRNA
    0.00000000          ,  ... % MuMax.rRNA
    0.00000000          ,  ... % MuMax.rzRNA
    0.01000000          ,  ... % MiMe.k.As.rPr
    0.00000000          ,  ... % MuMax.rPr
    0.00000000          ,  ... % MuMax.rzPr
    0.00000000          ,  ... % Y.As.Pr
    0.01000000          ,  ... % MiMe.k.Nu.rAs
    0.01000000          ,  ... % MiMe.k.am.rAs
    0.01000000          ,  ... % MiMe.k.c.rAs
    0.00000000          ,  ... % MuMax.rAs
    0.01000000          ,  ... % MiMe.k.As.rNu
    0.00100000          ,  ... % MiMeJeruSchalter.km.ph.rNu
    0.00100000          ,  ... % MiMeJeruSchalter.kj.ph.rNu
    0.00000000          ,  ... % MuMax.rNu
    0.00000000          ,  ... % Y.Nu.RNA
    0.01000000          ,  ... % MiMe.k.As.rU
    0.01000000          ,  ... % MiMe.k.Nu.rU
    0.00000000          ,  ... % MuMax.rU
```

```
    0.00000000          ,  ... % Y.am.As
    0.00000000          ,  ... % Y.ph.Nu
    0.00000000          ,  ... % Y.c.Nu
    0.00100000          ,  ... % Jeru.k.DNA.rEPS
    0.00100000          ,  ... % Moser.k.c.rEPS
    1.00000000          ,  ... % Moser.lambda.c.rEPS
    0.00000000          ,  ... % MuMax.rEPS
    0.00100000          ,  ... % InhibMoser.k.c.rzEPS
    1.00000000          ,  ... % InhibMoser.lambda.c.rzEPS
    0.00000000          ,  ... % MuMax.rzEPS
    0.00000000          ,  ... % MuMax.rZ
    0.00000000          ,  ... % Y.Erhaltung
    0.01000000          ,  ... % Aiba_.k.Nu.rzRNA
    0.01000000          ,  ... % Aiba_.k.am.rzPr
    0.01000000          ,  ... % Haldane_.k_m.As.rZ
    0.01000000          ,  ... % Haldane_.k_i.As.rZ
    0.01000000          ,  ... % AibaSchalter_.k.Nu.rZ
    0.01000000          ,  ... % Haldane_.k_m.c.rZ
    0.01000000          ,  ... % Haldane_.k_i.c.rZ
];

Parameter.CONS.max = [ ...
   50.00000000          ,  ... % MiMe.k.As.rDNA
   50.00000000          ,  ... % MiMe.k.Nu.rDNA
   20.00000000          ,  ... % MuMax.rDNA
  100.00000000          ,  ... % AibaSchalter.k.Nu.rzDNA
   20.00000000          ,  ... % MuMax.rzDNA
   50.00000000          ,  ... % MiMe.k.Nu.rRNA
   20.00000000          ,  ... % MuMax.rRNA
   20.00000000          ,  ... % MuMax.rzRNA
   50.00000000          ,  ... % MiMe.k.As.rPr
   20.00000000          ,  ... % MuMax.rPr
   20.00000000          ,  ... % MuMax.rzPr
    1.00000000          ,  ... % Y.As.Pr
   50.00000000          ,  ... % MiMe.k.Nu.rAs
   50.00000000          ,  ... % MiMe.k.am.rAs
   50.00000000          ,  ... % MiMe.k.c.rAs
   20.00000000          ,  ... % MuMax.rAs
   50.00000000          ,  ... % MiMe.k.As.rNu
  100.00000000          ,  ... % MiMeJeruSchalter.km.ph.rNu
  100.00000000          ,  ... % MiMeJeruSchalter.kj.ph.rNu
   20.00000000          ,  ... % MuMax.rNu
    1.00000000          ,  ... % Y.Nu.RNA
   50.00000000          ,  ... % MiMe.k.As.rU
   50.00000000          ,  ... % MiMe.k.Nu.rU
   20.00000000          ,  ... % MuMax.rU
    1.00000000          ,  ... % Y.am.As
  500.00000000          ,  ... % Y.ph.Nu
```

```
  500.00000000          ,  ... % Y.c.Nu
  100.00000000          ,  ... % Jeru.k.DNA.rEPS
  100.00000000          ,  ... % Moser.k.c.rEPS
   25.00000000          ,  ... % Moser.lambda.c.rEPS
   20.00000000          ,  ... % MuMax.rEPS
  100.00000000          ,  ... % InhibMoser.k.c.rzEPS
   25.00000000          ,  ... % InhibMoser.lambda.c.rzEPS
   20.00000000          ,  ... % MuMax.rzEPS
   20.00000000          ,  ... % MuMax.rZ
    5.00000000          ,  ... % Y.Erhaltung
  100.00000000          ,  ... % Aiba_.k.Nu.rzRNA
  100.00000000          ,  ... % Aiba_.k.am.rzPr
   50.00000000          ,  ... % Haldane_.k_m.As.rZ
   50.00000000          ,  ... % Haldane_.k_i.As.rZ
  100.00000000          ,  ... % AibaSchalter_.k.Nu.rZ
   50.00000000          ,  ... % Haldane_.k_m.c.rZ
   50.00000000          ,  ... % Haldane_.k_i.c.rZ
];

%% ParColors
ParColors = [ 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 3 3 4 5 5 6 6 ...
              7 7 7 7 8 8 8 8 4 9 9 9 ];

%% X0 ParNames
x0ParNames = {''};

%% Ergebnis der Parameteridentifikation gestartet am 18-Mar-2010 08:04:29 beendet
% am 18-Mar-2010 08:19:50
% (Gesamtdauer: 15.3568 min -- aktuell gültiges TimeOut: NaN min
% Verwendeter Optimierer: SNOPT7

% verwendete Versuche:
% PPdef11_Daten PPdef12_Daten PPdef13_Daten PPdef14_Daten PPdef15_Daten
% Modifizierte Startparameterwerte:
% MiMe.k.As.rDNA : 0.69622;
% MiMe.k.Nu.rDNA : 0.40132;
% MuMax.rDNA : 20;
% AibaSchalter.k.Nu.rzDNA : 7.8828;
% MuMax.rzDNA : 0.745;
% MiMe.k.Nu.rRNA : 0.01;
% MuMax.rRNA : 2.1401;
% MuMax.rzRNA : 0.10163;
% MiMe.k.As.rPr : 0.01;
% MuMax.rPr : 1.6157;
% MuMax.rzPr : 0.27408;
% Y.As.Pr : 0.95974;
% MiMe.k.Nu.rAs : 0.096172;
% MiMe.k.am.rAs : 0.01598;
```

```
% MiMe.k.c.rAs : 0.8908;
% MuMax.rAs : 0.55623;
% MiMe.k.As.rNu : 1.596;
% MiMeJeruSchalter.km.ph.rNu : 2.1913;
% MiMeJeruSchalter.kj.ph.rNu : 1.1429;
% MuMax.rNu : 0.048601;
% Y.Nu.RNA : 0.83654;
% MiMe.k.As.rU : 0.33735;
% MiMe.k.Nu.rU : 0.78435;
% MuMax.rU : 4.287;
% Y.am.As : 0.37469;
% Y.ph.Nu : 21.4711;
% Y.c.Nu : 24.7288;
% Jeru.k.DNA.rEPS : 3.4488;
% Moser.k.c.rEPS : 0.076842;
% Moser.lambda.c.rEPS : 3.8525;
% MuMax.rEPS : 0.16664;
% InhibMoser.k.c.rzEPS : 19.987;
% InhibMoser.lambda.c.rzEPS : 1;
% MuMax.rzEPS : 0.14629;
% MuMax.rZ : 9.2149;
% Y.Erhaltung : -5.6847e-009;
% Aiba.k.Nu.rzRNA : 0.010002;
% Aiba.k.am.rzPr : 0.01;
% Haldane.k_m.As.rZ : 0.43108;
% Haldane.k_i.As.rZ : 3.9294;
% AibaSchalter.k.Nu.rZ : 0.67703;
% Haldane.k_m.c.rZ : 14.157;
% Haldane.k_i.c.rZ : 2.3386;

% variierte Parameter:
% MiMe.k.As.rDNA MiMe.k.Nu.rDNA MuMax.rDNA AibaSchalter.k.Nu.rzDNA MuMax.rzDNA ...
% MiMe.k.Nu.rRNA MuMax.rRNA MuMax.rzRNA MiMe.k.As.rPr MuMax.rPr MuMax.rzPr ...
% Y.As.Pr MiMe.k.Nu.rAs MiMe.k.am.rAs MiMe.k.c.rAs MuMax.rAs MiMe.k.As.rNu ...
% MiMeJeruSchalter.km.ph.rNu MiMeJeruSchalter.kj.ph.rNu MuMax.rNu Y.Nu.RNA ...
% MiMe.k.As.rU MiMe.k.Nu.rU MuMax.rU Y.am.As Y.ph.Nu Y.c.Nu ...
% Jeru.k.DNA.rEPS Moser.k.c.rEPS Moser.lambda.c.rEPS MuMax.rEPS ...
% InhibMoser.k.c.rzEPS InhibMoser.lambda.c.rzEPS MuMax.rzEPS MuMax.rZ Y.Erhaltung ...
% Aiba.k.Nu.rzRNA Aiba.k.am.rzPr Haldane.k_m.As.rZ Haldane.k_i.As.rZ ...
% AibaSchalter.k.Nu.rZ Haldane.k_m.c.rZ Haldane.k_i.c.rZ
% Start Güte: 275.4351
% Erreichte Güte: 113.2472

% Parametername                                   Wert
Parameter.MiMe.k.As.rDNA                          = 0.721628882446727;
Parameter.MiMe.k.Nu.rDNA                          = 0.48798053471185;
Parameter.MuMax.rDNA                              = 20;  % hit CON
Parameter.AibaSchalter.k.Nu.rzDNA                 = 12.6704647027631;
```

```
Parameter.MuMax.rzDNA                                = 0.296945309487111;
Parameter.MiMe.k.Nu.rRNA                             = 0.0359039622656748;
Parameter.MuMax.rRNA                                 = 1.87982664272685;
Parameter.MuMax.rzRNA                                = 0.0670459577076377;
Parameter.MiMe.k.As.rPr                              = 0.0191492031726463;
Parameter.MuMax.rPr                                  = 1.0837589188428;
Parameter.MuMax.rzPr                                 = 0.217597735431258;
Parameter.Y.As.Pr                                    = 0.994824887734149;
Parameter.MiMe.k.Nu.rAs                              = 0.106817822674858;
Parameter.MiMe.k.am.rAs                              = 0.0691274185689602;
Parameter.MiMe.k.c.rAs                               = 1.02826420679482;
Parameter.MuMax.rAs                                  = 0.575726565894733;
Parameter.MiMe.k.As.rNu                              = 1.72296520891481;
Parameter.MiMeJeruSchalter.km.ph.rNu                 = 2.202558920339;
Parameter.MiMeJeruSchalter.kj.ph.rNu                 = 1.14291843582947;
Parameter.MuMax.rNu                                  = 0.0787587462816176;
Parameter.Y.Nu.RNA                                   = 0.732916097517101;
Parameter.MiMe.k.As.rU                               = 0.127875308440333;
Parameter.MiMe.k.Nu.rU                               = 0.586521573882603;
Parameter.MuMax.rU                                   = 5.92685943044948;
Parameter.Y.am.As                                    = 0.423431648499535;
Parameter.Y.ph.Nu                                    = 23.8610127891855;
Parameter.Y.c.Nu                                     = 29.7199073569475;
Parameter.Jeru.k.DNA.rEPS                            = 3.43639253150305;
Parameter.Moser.k.c.rEPS                             = 0.0799509358153129;
Parameter.Moser.lambda.c.rEPS                        = 3.96877161560734;
Parameter.MuMax.rEPS                                 = 0.145843484274625;
Parameter.InhibMoser.k.c.rzEPS                       = 19.6136874202509;
Parameter.InhibMoser.lambda.c.rzEPS                  = 1.05018490260694;
Parameter.MuMax.rzEPS                                = 0.166824688175684;
Parameter.MuMax.rZ                                   = 7.22257966225748;
Parameter.Y.Erhaltung                                = 0.00720411849581306;
Parameter.Aiba.k.Nu.rzRNA                            = 0.0123486657197447;
Parameter.Aiba.k.am.rzPr                             = 0.0270707324029118;
Parameter.Haldane.k_m.As.rZ                          = 0.58288656013766;
Parameter.Haldane.k_i.As.rZ                          = 3.92724703998016;
Parameter.AibaSchalter.k.Nu.rZ                       = 0.712781396700575;
Parameter.Haldane.k_m.c.rZ                           = 19.1782292469775;
Parameter.Haldane.k_i.c.rZ                           = 2.23789573552726;
```

Rauschen

```
function R = Rauschen(y,y_parameter)
% Für jede Messgröße wird hier die Varianz für das Messrauschen definiert.
% Typischerweise wird dies in Form eines (zu quadrierenden) Polynoms angeben, so
% dass das Messrauschen einen Absolutwert und einen Proportionalanteil beinhaltet.
% Das noch nicht quadrierte Polynom stellt damit den zu erwartenden Fehler dar;
```

```
% die Standardabweichung
% Hinweis: Rauschwerte "1" werden standardmäßig vergeben für virtuelle Messgrößen

global SYSTEM

y = y(:);

if nargin < 2 || isempty(y_parameter)
  y_parameter = zeros(size(y));
end

R = zeros(SYSTEM.m);

sigmaTS =   [0.025 0.02;
             0.020 0.03;
             0.015 0.04;
             0.010 0.09;
             0.005 0.11;
             0.002 0.3];

sigmaBTM = [0.25/12 0.05];
R(1,1)   = 1/5 * polyval(sigmaBTM,y(1))^2;

sigmaDNA = .05;
R(2,2)   = polyval(sigmaDNA,y(2))^2;

sigmaRNA = .05;
R(3,3)   = polyval(sigmaRNA,y(3))^2;

sigmaPr = .05;
R(4,4)  = polyval(sigmaPr,y(4))^2;

sigmaN  = [0.015/2 0.003];
R(5,5)  = 10 * polyval(sigmaN,y(5))^2;

sigmaP  = [0.011/0.6 0.007];
R(6,6)  = polyval(sigmaP,y(6))^2;

sigmaC  = [.5/40 .25];
R(7,7)  = polyval(sigmaC,y(7))^2;

sigmaZ  = .01;
R(8,8)  = polyval(sigmaZ,y(8))^2;

end
```

x2x_

```
function x_ = x2x_(t,x,z)

x_ = x;
V  = x(end);

% Die Konzentration der Biomasse-Zustände (inkl. Kompartimente) ändern sich bei den
% Überstandsproben  nicht, aber bei den Vollprobenahmen (1) und (4)
x_([ 1, 2, 3, 4, 5, 6]) = x([ 1, 2, 3, 4, 5, 6]) * (1-(z(1)+z(4))/V);

% Die restlichen Zustände (inkl. Volumen) ändern sich bei jeder Probenahme
x_([ 7, 8, 9,10,11,12]) = x([ 7, 8, 9,10,11,12]) * (1-(z(1)+z(2)+z(3)+z(4))/V);

end
```

A.2 Typenkodierung für Stellgrößen und Probennahmen

Stellgröße	**Typ**	**Stellgröße**	**Typ**
Konzentration	c	Korrekturfluid	K
Säure	A	Base	B
Antischaum	F	Abdampf	V
Stickstoffquelle	N	Phosphatquelle	P
Kohlenstoffquelle	C	Sonst. Feeding	Z
Substrat	S	Gas	G
# der Substratquelle	[1-9] pH-Wert	H	

Tabelle A.2: Stellgrößentypen (uTyp)

Probennahme	**Typ**	**Probennahme**	**Typ**
Vollprobe	V	Überstandsprobe	K

Tabelle A.3: Probennahmetypen (zTyp)

A.3 Programmbeispiel einer Formalkinetik

MiMe

```
function ausgabe = MiMe(varargin)
% varargout = MiMe(5,0.1,'');
% varargout = MiMe(varargin)
%
% Michaelis-Menten-Kinetik-Funktion
%
% Anzahl der Übergabeparameter entscheidet über Funktionsresultat!
% - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
% Kein Übergabeparameter ->
% Es wird ein Strukt zurückgegeben, welches Informationen über die Kinetik
% enthält. Diese Informationen benötigt z.B. der automatische Modellgenerator.
% Beispiel:
% MiMe = struct ...
%   einfluss:          Liste der Einflussgrößen
%     name: 'c_gl':    Name des Einflussgröße (Glucose-Konzentration)
%     typ: 'Substrat': Einflußgröße ist vom Typ Substrat
%     einfluss: '+':   Art der Korrelation zwischen Wachstumsrate und Einflussgröße
%   Parameter:         Liste der Parameter
%     name: 'k_m':     Name des Parameters
%     values:          Liste von Parameternwerten
%        minimum:      Minimaler Wert, den der Parameter annehmen darf (in PI)
%        start:        Startwert für die Parameteridentifikation
%        maximum:      Maximaler Wert, den der Parameter annehmen darf (in PI)
% . . . . . .
% Ein Übergabeparameter (Typ: Charakter/String z.B. 'display' or 'view') ->
%  Zur Veranschaulichung der Funktion wird eine figure geöffnet und typische
%  Verläufe gezeichnet
% . . . . . .
% Zwei Übergabeparameter (Typ: Array, Typ: Array) ->
%  MiMe(X,Parameter) gibt die genaue Wachstumsrate der Michaelis-Menten Kinetik bei
%  der angebenenen Substratkonzentration (X) und dem Parameterwert (k_m) zurück.
% . . . . . .
% Drei Übergabeparameter (Typ: Array, Typ: Array, Typ: String) ->
%  MiMe(X,Parameter) gibt grafisch den Verlauf der Wachstumsrate der Michaelis-Menten
%  Kinetik für den Parameterwert (k_m) und der Substratkonzentration (X) von Null bis
%  zum angegebenen Wert (X) zurück.

MiMe.einfluss{1}.name     = 'c_{S}';
MiMe.einfluss{1}.typ      = 'Substrat';
MiMe.einfluss{1}.einfluss = '+';
MiMe.parameter{1}.name    = 'k';
```

```
MiMe.parameter{1}.values  = struct('minimum',1e-4,'startwert',0.1,'maximum',50);
MiMe.kinetiktyp           = 1; % limitierend
MiMe.beispielParameter    = [0.1;1;10];

% Kinetik-Funktion --------------------
function mu = MiMe_fcn(x,p)
  mu = x ./ (x+p);
end
function_str = '$\mu = \frac{c_S}{c_S + k}$';
% -------------------------------------
plot_flag  = false;
max_points = 500;

if nargin == 0 % Kein Übergabeparameter -> Strukt ausgeben
  ausgabe = MiMe;
else % Übergabeparameter auswerten
  if nargin == 1,
    plot_flag = true;
    i = 10; % Anzahl der parametrischen Beispiele
    % Ausprägung der Parameter:
    M = MiMe.parameter;
    p = logspace(log10(M{1}.values.minimum),log10(M{1}.values.maximum),i);
    x = linspace(0,10,max_points); % Verteilt Einflussgröße auf typischen Bereich
  elseif nargin == 2,
    X= varargin{1}(:);
    Parameter = varargin{2}(:);
  elseif nargin == 3,
    plot_flag = true;
    X         = linspace(0,varargin{1}(end),max_points);
    Parameter = varargin{2}(:);
  end

  if nargin > 1
    ausgabe = MiMe_fcn(X,Parameter);
  end

  if plot_flag
    f = figure;
    set(f,'Name','MiMe');
    hold on;
    legend_str = [];
    if nargin == 1,
      for j=1:i, % Schleife über die Parameterwert-Liste
        color = j/i * [0.9 0.9 0.9]; % Farbdefinition pro Parameterkurve
        plot(x,MiMe_fcn(x,p(j)),'Color',color);
        % Erzeugt Eintrag für die Legende:
```

```
        legend_str{end+1} = [MiMe.parameter{1}.name ' = ' num2str(p(j))];
      end
    else
      plot(X,ausgabe);
      legend_str = {[MiMe.parameter{1}.name ' = ' num2str(Parameter)]};
    end
    ylabel('\mu');
    xlabel(MiMe.einfluss{1}.name)
    legend(legend_str,'FontSize',8);
    title('MiMe');
    text(8,0.15,function_str,'Interpreter','latex','FontSize',14);
  end
end

clear MiMe;

end % function
```

Literaturverzeichnis

[1] Extensible Markup Language (XML). http://www.w3.org/XML/

[2] Systems Biology Markup Language (SBML). http://sbml.org/

[3] SBML Software Matrix. http://sbml.org/SBML_Software_Guide/SBML_Software_Matrix. Version: Januar 2016

[4] André, B. ; Marquardt, W.: Incremental and simultaneous identification of reaction kinetics: methods and comparison. In: Chemical Engineering Science 59 (2004), Nr. 13, S. 2673–2684

[5] Antoniotti, M. ; Policriti, A. ; Ugel, N. ; Mishra, B.: Model Building and Model Checking for Biological Processes. In: Cell Biochemistry and Biophysics 38 (2003), Nr. 3, S. 271–286

[6] Audoly, S. ; Bellu, G.: Global Identifiability on Nonlinear Models of Biological Systems. In: IEEE Transactions of Biomedical Engineering 48 (2001), S. 55–65

[7] Bastin, G. (Hrsg.) ; Dochain, D. (Hrsg.): On-line Estimation and Adaptive Control of Bioreactors. Elsevier, 1990

[8] Berg, J. M. ; Tymoczko, J. L. ; Stryer, L.: Biochemistry. 5. Freeman, 2002 http://www.ncbi.nlm.nih.gov/books/NBK22512/#A4446

[9] Bettenhausen, K. D. ; Marenbach, P. ; Freyer, S. ; Rettenmaier, H. ; Nieken, U.: Self-Organizing Structured Modelling of a Biotechnological Fed-Batch Fermentation by Means of Genetic Programming. In: International Conference on Genetic Algorithms in Engineering Systems: Innovations and Applications Bd. 414, 1995, S. 481–486

[10] BMBF: Die deutsche Biotechnologie-Branche / biotechnologie.de (BMBF). 2013. – Forschungsbericht

[11] Bogaerts, Ph. ; Wouwer., A. V.: Systematic Generation of Identifiable Macroscopic Reaction Schemes. In: 8th International Conference on Computer Applications in Biotechnology (CAB8), 2001, S. 13–18

[12] Bonvin, D. ; Rippin, D. W. T.: Target Factor Analysis for the Identification of Stoichiometric Models. In: Chemical Engineering Science 45 (1990), Nr. 12, S. 3417–3426

[13] Büdenbender, C.: Modellentwicklung und Trajektorienplanung für Fed-Batch Fermentationen mit komplexen Nährmedien, Technische Universität Berlin, Dissertation, 2004.

[14] "Bundesministerium für Bildung und Forschung (BMBF)": Weiße Biotechnologie - Chancen für eine bio-basierte Wirtschaft. 2012

[15] "Bundesministerium für Bildung und Forschung (BMBF)": Weiße Biotechnologie - Chancen für eine bio-basierte Wirtschaft. 2015

[16] Burnham, K. P. ; Anderson, D. R.: Model selection and multimodel inference. 2nd. Springer, 2002

[17] Calzone, L. ; Chabrier-Rivier, N. ; Fages, F. ; Soliman, S.: Machine learning biochemical networks from temporal logic properties. In: Transactions on Computational Systems Biology VI 4220 (2006), S. 68–94

[18] Demain, A. L.: Pharmaceutically active secondary metabolites of microorganisms. In: Applied Microbiology and Biotechnology 52(4) (1999), S. 455–463

[19] Demain, A. L.: Microbial biotechnology. In: Trends in Biotechnology 18 (2000), S. 26–31.

[20] Droste, P. ; Nöh, K. ; Wiechert, W.: Omix - A Visualization Tool for Metabolic Networks with Highest Usability and Customizability in Focus. In: Chemie Ingenieur Technik 85 (2013), Nr. 6, S. 849–862

[21] Efron, B. ; Tibshirani, R. J.: An Introduction to the Bootstrap. Chapman & Hall/CRC, 1993

[22] Espie, D. ; Macchietto, S.: The Optimal Design of Dynamic Experiments. In: AIChE Journal Bd. 35, 1989, S. 223–229

[23] Fages, F. ; Soliman, S.: Abstract interpretation and types for systems biology. In: Theoretical Computer Science 403 (2008), Nr. 1, S. 52–70

[24] Funahashi, A. ; Matsuoka ; Jouraku, A. ; Morohashi, M. ; Kikuchi, N. ; Kitano, H.: CellDesigner 3.5: A Versatile Modeling Tool for Biochemical Networks. In: Proceedings of the IEEE Bd. 96, 2008, S. 1254–1265

[25] Funahashi, A. ; Tanimura, N. ; Morohashi, M. ; Kitano, H.: CellDesigner: a process diagram editor for gene-regulatory and biochemical networks. In: BIOSILICO (2003), Nr. 1, S. 159–162

[26] Gablonsky, J. M. ; Kelley, C. T.: A locally-biased form of the DIRECT algorithm. In: Journal of Global Optimization 21(1) (2001), S. 27–37

[27] Ginkel, M. ; Kremling, A. ; Nutsch, T. ; Rehner, R. ; Gilles, E. D.: Modular modeling of cellular systems with ProMoT/Diva. In: Bioinformatics 9 (2003), Nr. 9, S. 1169–1176

[28] Gutenkunst, R. N. ; Atlas, J. C. ; Casey, F. P. ; Daniels, B. C. ; Kuczenski, R. S. ; Waterfall, J. J. ; Myers, C. R. ; Sethna, J. P.: SloppyCell. `http://sloppycell.sourceforge.net`. Version: 2007

[29] Haunschild, M. D. ; Freisleben, B. ; Takors, R. ; Wiechert, W.: Investigating the dynamic behaviour of biochemical networks using model families. In: Bioinformatics 21 (2005), Nr. 8, S. 1617–1625

[30] Heine, T.: Modellgestützte Überwachung und Führung von Fed-Batch-Prozessen zur Antibiotikaproduktion, Technische Universität Berlin, Dissertation, 2004

[31] Heller, K. J. ; Antranikian (Hrsg.): Angewandte Mikrobiologie. 2006, S. 511–527

[32] Herold, S.: Automatic Generation of Process Models for Fed-Batch Fermentations Based on the Detection of Biological Phenomena, Technische Universität Berlin, Dissertation, 2015

[33] Herold, S. ; King, R.: Automatic identification of structured process models based on biological phenomena detected in (fed-)batch experiments. In: Biosystems Engineering 37 (2013), Nr. 7, S. 1289–1304

[34] Herwig, Christoph ; Posch, Andreas: Herausforderungen und Trends für zukünftige Bioprozesse. In: pharmind 75 (2013), Nr. 10, S. 1688–1694

[35] Hoops, S. ; Sahle, S. ; Gauges, R. ; Lee, C. ; Pahle, J. ; N.Simus ; Singhal, M. ; Xu, L. ; Mendes, P. ; Kummer, U.: COPASI - a COmplex PAthway SImulator. In: Bioinformatics 22 (2006), Nr. 24, S. 3067–3074

[36] Isermann, R.: Identifikation dynamischer Systeme, Band I. Berlin : Springer Verlag, 1992

[37] Johnson, Steven G.: The NLopt nonlinear-optimization package. `http://ab-initio.mit.edu/nlopt`. Version: September 2012

[38] Jones, D. R. ; Perrtunen, C. D. ; Stuckmann, B. E.: Lipschitzian optimization without the Lipschitz constant. In: Journal of Optimization Theory and Applications 79 (1993), S. 157

[39] Junker, B. H. ; Wang, H. Y.: Bioprocess Monitoring and Computer Control: Key Roots of the Current PAT Initiative. In: Biotechnology and Bioengineering (2006), S. 226–261

[40] Kawohl, M.: Robuste optimale Prozessplanung und -führung nichtlinearer Systeme unter Verwendung der Unscented-Transformation, Technische Universität Berlin, Dissertation, 2014

[41] Kawohl, M. ; Heine, T. ; King, R.: Model Based Estimation and Optimal Control of Fed-Batch Fermentation Processes for the Production of Antibiotics. In: Journal of Chemical Engineering and Processing 46 (2007), Nr. 11, S. 1223–1241

[42] King, R.: Mathematische Modelle myzelförmig wachsender Mikroorganismen. Düsseldorf : VDI Fortschritt-Berichte, 1992

[43] King, R.: A Structured Mathematical Model for a Class of Organisms: 1. Development of a Model for *Streptomyces tendae* and Application of Model-Based Control. In: Journal of Biotechnology 52 (1997), S. 219–234.

[44] King, R. ; Büdenbender, C.: A structured mathematical model for a class of organisms. 2. Application of the model to other strains. In: Journal of Biotechnology 52 (1997), S. 235–244.

[45] Leifheit, J. ; Heine, T. ; Kawohl, M. ; King, R.: Rechnergestützte halbautomatische Modellierung biotechnologischer Prozesse. In: at Automatisierungstechnik 55 (2007), Nr. 5, S. 211–218

[46] Leifheit, J. ; King, R.: (Semi-)automatic modeling of biological reaction systems with TAM-B. In: 9th International Conference on Computer Applications in Biotechnology (CAB9) (2004)

[47] Leighty, R. W. ; Antoniewicz, M. R.: Dynamic metabolic flux analysis (DMFA): a framework for determining fluxes at metabolic non-steady state. In: Metabolic Engineering 13 (2011), Nr. 6, S. 745–755

[48] Majer, C. P.: Parameterschätzung, Versuchsplanung und Trajektorienoptimierung für verfahrenstechnische Prozesse, Universität Stuttgart, Dissertation, 1997

[49] Makanay, C.: Die, die in uns leben. `http://gutmicrobiota.univie.ac.at/fileadmin/user_upload/ag_dome_gut_microbiota/Public_Outreach/2012_Die_die_in_uns_leben.pdf`. Version: Nov 2012

[50] Mandenius, C.-F.: Recent developments in the monitoring, modeling and control of biological production systems. In: Bioprocess and Biosystems Engineering 26 (2004), Nr. 6, S. 347–351

[51] Mangold, M. ; Angeles-Palacios, O. ; Ginkel, M. ; Kremling, A. ; Waschler, R. ; Kienle, A. ; D.Gilles, E.: Computer Aided Modeling of Chemical and Biological Systems: Methods, Tools and Applications. In: Industrial & Engineering Chemistry Research 44 (2005), Nr. 8, S. 2579–2591

[52] "MathWorks": Matlab. `http://de.mathworks.com/products/matlab`

[53] McNaught, A. D. (Hrsg.) ; Wilkinson, A. (Hrsg.): IUPAC. Compendium of Chemical Terminology. Blackwell Scientific Publications, Oxford, 2006 `http://goldbook.iupac.org`

[54] Meadows, A. L. ; Karnik, R. ; Lam, H. ; Forestell, S. ; Snedecor, B.: Application of dynamic flux balance analysis to an industrial Escherichia coli fermentation. In: Metabolic Engineering (2010)

[55] Mirschel, S. ; Steinmetz, K. ; Rempel, M. ; Ginkel, M. ; Gilles, E. D.: ProMoT: Modular Modeling for Systems Biology. In: Bioinformatics 25 (2009), Nr. 5, S. 687–689

[56] Mishra, B.: A Symbolic Approach to Modeling Cellular Behavior. In: Proceedings of HiPC 2002, 2002, S. 725–732

[57] Mondol, M. A. M. ; Kim, J. H. ; Lee, H.-S. ; Lee, Y.-J. ; Shin, H. J.: Macrolactin W, a new antibacterial macrolide from marine Bacillus sp. In: Bio-organic & Medicinal Chemistry Letters 21 (2011), Nr. 12, S. 3832–3835

[58] Moraru, I. I. ; Schaff, J. C. ; Slepchenko, B. M. ; Blinov, M. L. ; Morgan, F. ; Lakshminarayana, A. ; Gao, F. ; Li, Y. ; Loew, L. M.: Virtual Cell modelling and simulation software environment. In: IET Systems Biology 2 (2008), Nr. 5, S. 352–362

[59] Myers, C. R. ; Gutenkunst, R. N. ; Sethna, J. P.: Python unleashed on systems biology. In: Computing in Science and Engineering 9 (2007), Nr. 3, S. 34–37

[60] "National Instruments": LabView. `http://www.ni.com/labview`

[61] Naylor, T. H. ; Finger, J. M.: Verification of Computer Simulation Models. In: Management Science 14 (1967), Nr. 2, S. B92–B101

[62] Nelder, J. A. ; Mead, R.: A Simplex Method for Function Minimization. In: Computer Journal 7 (1965), S. 308–313

[63] Powell, M. J. D.: The BOBYQA algorithm for bound constrained optimization without derivatives / Department of Applied Mathematics and Theoretical Physics. Cambridge England, 2009 (NA2009/06). – Forschungsbericht

[64] Price, W. L.: Global optimization by controlled random search. In: Journal of Optimization Theory and Applications 40(3) (1983), S. 333–348

[65] Pászthory, E.: Salpetergewinnung und Salpeterwirtschaft vom Mittelalter bis in die Neuzeit. In: Chemie in unserer Zeit 1 (1995), S. 8–20

[66] Reiss, J.: Schimmelpilze in der Heilkunde. In: Zeitschrift für Mykologie 60 (1994), Nr. 2, S. 349–357

[67] Roemer-Mähler, J.: In der Bekanntmachung vom 17. Oktober 2008 / Bundesministerium für Bildung und Forschung. 2008. – Forschungsbericht

[68] Ross, B. J.: The evolution of higher-level biochemical reaction models. In: Genetic Programming and Evolvable Machines 13 (2012), S. 3–31

[69] Rossner, N.: Robuste modellgestützte Prozessführung auf Basis von Gauß'schen Mischdichten am Beispiel der Bray-Liebhafsky-Reaktion und der autotrophen Kultivierung von *Ralstonia eutropha* H16, Technische Universität Berlin, Dissertation, 2014

[70] Rowan, T.: Functional Stability Analysis of Numerical Algorithms, Department of Computer Sciences, University of Texas at Austin, PhD thesis, 1990

[71] Saez-Rodriguez, J. ; Mirschel, S. ; Hemenway, R. ; Klamt, S. ; Gilles, E. D. ; M. ; Ginkel.: Visual setup of logical models of signaling and regulatory networks with ProMoT. In: BMC Bioinformatics 7 (2006), Nr. 506

[72] Schittkowski, K.: On the Convergence of a Sequential Quadratic Programming Method with an Augmented Lagrangian Line Search Function / Systems Optimization laboratory, Stanford University, Stanford, CA,. 1982. – Forschungsbericht

[73] Schumann, W.: Biotop Mensch. Wiley Online Library, 2011 `http://onlinelibrary.wiley.com/doi/10.1002/biuz.201110450/epdf`

[74] Sedoglavic, A.: A Probabilistic Algorithm to Test Local Algebraic Observability in Polynomial Time. In: Journal of Symbolic Computation 33 (2002), S. 735–755

[75] Stone, M. J. ; Williams, D. H.: On the evolution of functional secondary metabolites (natural products). In: Molecular Microbiology 6(1) (1992), S. 29–34

[76] Tabarez, M. R.: Discovery of the new antimicrobial compound 7-O-malonyl macrolactin A, Universität Braunschweig, Dissertation, 2005

[77] "TOMLAB Optimization": TOMLAB. `http://tomopt.com/`

[78] "TOMLAB Optimization": About Tomlab. http://tomopt.com/tomlab/about, September 2012

[79] Ulber, R. ; Soyez, K.: Vom Wein zum Penicillin - 5000 Jahre Biotechnologie. In: Chemie in unserer Zeit 38 (2004), S. 172–180

[80] Vining, L. C.: Secondary metabolism, inventive evolution and biochemical diversity - a review. In: Gene 115(1-2) (1992), S. 135–140

[81] Violet, N. ; Fischer, E. ; Heine, T. ; King, R.: A Software Supported Compartmental Modeling Approach for *Paenibacillus polymyxa*. In: Computer Applications in Biotechnology 11 (2010), Nr. 1, S. 377–382

[82] Violet, N. ; King, R.: Entwicklung eines Rapid-Prototyping-Ansatzes zum schnellen und effizienten Aufbau optimaler Prozessführungsstrategien für biotechnologische Produktionsprozesse. In: DECHEMA (2009). `http://www.dechema.de/index.php?id=123445`

[83] Violet, N. ; Rossner, N. ; Heine, T. ; King, R.: RapOpt - An automation tool for production-orientated run-to-run model evolution. In: Troch, I. (Hrsg.) ; Breitenecker, F. (Hrsg.): Proceedings of the 6th Vienna conference on mathematical modelling (MathMod). ARGESIM, 2009, S. 2339–2346

[84] Vollbrecht, D.: Feststoff-Fermentation - Ein historischer Überblick. In: Chemie Ingenieur Technik 10 (1997), Nr. 69, S. 1403–1408

[85] Watzdorf, R. ; Allgöwer, F. ; Helget, A. ; Marquardt, W. ; Gilles, E. D.: Dynamische Simulation verfahrenstechnischer Prozesse und Anlagen - Ein Vergleich von Werkzeugen. In: Simulationstechnik 9. ASIM-Symposium (1994), S. 83–88

[86] Weidemann, A. ; S.Sahle, S. R. ; Gauges, R. ; Gabdoulline, R. ; Surovtsova, I. ; N, N. S. ; Besson, B. ; Rojas, I. ; Wade, R. ; Kummer, U.: SYCAMORE - a SYstems biology Computational Analysis and MOdeling Research Environment. In: Bioinformatics 24 (2008), S. 1463–1464

[87] Wong, W. C. ; Dudinsky, L. A. ; Garcia, V. M. ; Ott, C. M. ; Castro, V. A.: Efficacy of various chemical disinfectants on biofilms formed in spacecraft potable water system components. In: Biofouling: The Journal of Bioadhesion and Biofilm Research 26 (2010), Nr. 5

[88] Yarbrough, G.G. ; Taylor, D.P. ; Rowlands, R.T. ; Crawford, M.S. ; Lasure, L.L.: Screening microbial metabolites for new drugs-theoretical and practical issues. In: The Journal of Antibiotics 46(4) (1993), S. 535–544

[89] Zamorano, F. ; Wouwer, A. V. ; Jungers, R. M. ; Bastin, G.: Dynamic metabolic models of CHO cell cultures through minimal sets of elementary flux modes. In: Journal of Biotechnology (2012)

ISBN 978-3-7418-0473-1