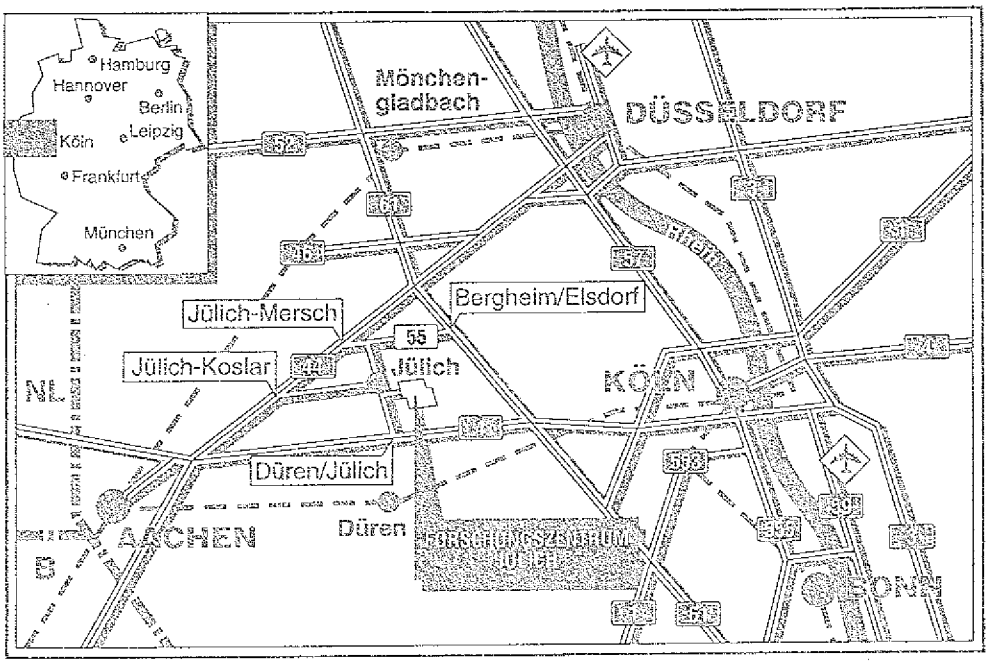
FORSCHUNGSZENTRUM JÜLICH GmbH



Institut für Biotechnologie

Interaktive Datenanalyse bei biotechnischen Prozeßdaten

Wolfgang Wiechert



Berichte des Forschungszentrums Jülich ; 2457
ISSN 0366-0885
Institut für Biotechnologie Jül-2457
D 5 (Universität Bonn)

Zu beziehen durch: Forschungszentrum Jülich GmbH · Zentralbibliothek
Postfach 1913 · D-5170 Jülich · Bundesrepublik Deutschland
Telefon: 02461/61-6102 · Telefax: 02461/61-6103 · Telex: 833556-70 kfa d

Interaktive Datenanalyse bei biotechnischen Prozeßdaten

Wolfgang Wiechert

Zusammenfassung

Konzeption

Die vorliegende Dissertation stellt ein *Konzept zur Analyse biotechnischer Prozeßdaten* vor. Es orientiert sich an den Anforderungen der Bioprozeßüberwachung und -kontrolle und vereinigt bestehende sowie neu vorgestellte Ansätze in einem einheitlichen Rahmen. Der *experimentelle Zyklus* der Verfahrensentwicklung teilt sich hierbei auf in die interaktive Arbeit am Datenanalyse-Rechner (graphische Workstation) und die Durchführung des Experiments, bei der die zuvor konfigurierten Datenanalyse-Bausteine zum Einsatz kommen können (Downloading auf einen Prozeßrechner).

Aufgrund einer *Analyse der Datenlage* und der *experimentellen Situation* in der Biotechnologie sowie diverser Simulationsbeispiele ist eine generelle Einengung des Methodenspektrums nicht begründbar. Es müssen daher

- *deterministische* (auf Differentialgleichungen basierende),
- *stochastische* (auf stochastischen Prozessen basierende),
- *statistische* (auf Modellen der Statistik basierende) und
- *logisch-heuristische* Ansätze

zu einem *Hybridkonzept* vereinigt werden. Das Konzept erlaubt die Anwendung verschiedenster Methoden und ihre Kombination durch *Vernetzung*. Die entstehenden Netzwerke von Datenanalysebausteinen können dynamisch verändert werden (*Modell- bzw. Methodenwechsel* bei Phasenübergängen). Insbesondere wird die Integration regelungstechnischer Elemente in *regelbasierte Überwachungsstrategien* ermöglicht.

Stoffwechselmodellierung

Auf dem Gebiet der *Stoffwechselmodellierung* ermöglicht die Begriffsbildung der *verallgemeinerten Reaktionsgleichungen* in Kombination mit der Annahme *abstrakter zellinterner Intermediate* eine unifizierende Theorie. Sie vereinigt unter anderem bestehende Konzepte der phänomenologischen Modellierung, des Pathway Modelling und der Wachstumsbeschreibung durch globale Reaktionsgleichungen. Für alle Ansätze ist ein *allgemeines Stoffwechsel-Modell* des Typs

$$\dot{X} = N \cdot X_1 \cdot r(X)$$

(X Zustandsvektor, speziell X_1 die Biomasse, N stöchiometrische Matrix, $r(X)$ Vektor der spezifischen Reaktionsraten) begründbar. Für die klassischen phänomenologischen Wachstumsmodelle kann ferner eine von klaren Grundannahmen ausgehende *deduktive Herleitung* gegeben werden. Diese axiomatische Vorgehensweise wird am einfachen linearen Stoffwechselmodell, am Erhaltungsstoffwechselmodell und am einem Backhefemodell mit Flaschenhalseffekt demonstriert.

Die exakte Stöchiometrie des Stoffwechsels muß für die Aufstellung des allgemeinen Modells nicht bekannt sein. Die *Bilanzierung* kann mittels einer Erhaltungsmatrix E mit

$$E \cdot \dot{X} = 0 \implies E \cdot N = 0$$

nachträglich durchgeführt werden. Dies bewirkt eine klare begriffliche Trennung zwischen den Konzepten des *theoretischen, stöchiometrischen* und *empirischen Reaktionsraums* sowie der zugehörigen statistischen Modelle.

Die auf *stochastischen Zustandsraummodellen* aufsetzenden Methoden der (nichtlinearen) Beobachtung und Filterung erscheinen in diesem Zusammenhang als Erweiterung der Bilanzierung im Falle unterbestimmter Systeme von Erhaltungsgleichungen.

Methodenspektrum

Auf dieser Basis wird für den Bereich der deterministischen, statistischen und stochastischen Modelle ein *einheitliches Methodenspektrum* begründet, das im wesentlichen auf den Techniken

- der *stationären* und *instationären Identifikation*,
- der *linearen Schätztheorie* sowie
- der *Zustandsbeobachtung* und *(nichtlinearen) Filterung*

beruht.

Eine *kritische Analyse* der klassischen *Parameteranpassungsmethode* (an Anpassungsbeispielen bei Sinus-Funktionen, exponentiellem Wachstum mit Störeinflüssen und praktischer k_La-Wert-Bestimmung) zeigt allerdings die Problematik von *strukturierten* und *Variable-Yield-Modellen* (gekennzeichnet durch zustandsabhängiges $N = N(X)$) in Hinblick auf die Datenanalyse auf. In diesem Kontext erfährt die vorgeschlagene *explorative Hypothesenbildung* aufgrund graphischer Auftragungen stationärer und instationärer Daten eine starke Aufwertung.

Als Bindeglied zwischen den Bausteinen

- der *signalbezogenen* Regelungstechnik (numerischer Ein- und Ausgang) einerseits und
- der *logisch-heuristischen* Prozeßüberwachung (logisch-kategorieller Ein- und Ausgang) andererseits
- werden *statistische* Bausteine (numerischer Eingang und logisch-kategorieller Ausgang) und *Selektoren* (logisch-kategorieller Eingang und numerischer Ausgang)

eingeführt, wodurch eine *3-Ebenen-Architektur* für die Bioprozeßüberwachung nahegelegt wird.

Simulation und Visualisierung

Zentrale Elemente des Datenanalysesystems sind *interaktive Simulation* und *dynamische Visualisierung*. Der Einsatz von Techniken zur Erzeugung *realistischer Rauschterme* (am Beispiel einer OD-Sonde) und die *Simulation stochastischer Prozesse* (am Beispiel der stochastischen Wachstumsmodelle) stellt für die Biotechnologie einen Fortschritt dar. Insbesondere ermöglichen sie einen *didaktischen Zugang* zur Kalman-Filter-Technik, die eine Berücksichtigung von Zeitverzögerungen und nicht normalverteilten Meß-Rauschtermen umfaßt.

Die extensive Verwendung von *Visualisierungstechniken* der *interaktiven* und *Explorativen Datenanalyse* (dynamische Simulation, Punktwolkendarstellungen, graphische Parametrisierung etc.) ermöglicht ein bisher nicht erreichtes kreatives Arbeiten mit dem Datenmaterial, was an verschiedenen Beispielen gezeigt werden kann.

Rechner-Implementierung

Die *Abstraktion* der verschiedenen Methoden zu Datenanalysebausteinen und die *Modularisierung* der Bausteine selbst wird so weit betrieben, daß eine unmittelbare *Rechner-Implementierung* möglich ist. Insbesondere werden die Probleme

- der *interaktiven Konfiguration*,
- *Verschaltung* und
- *Ablaufsteuerung* bei Netzwerken von kommunizierenden Bausteinen sowie
- der *Prozeßanbindung* grundsätzlich gelöst.

Eine Reihe von *Anwendungsbeispielen* beweist die Tragfähigkeit der Konzeption. Hierzu gehören

- ein *graphischer Dateneditor*,
- ein *interaktiv parametrisierbares Kalman-Filter*,
- ein *Sprungdetektor* für die Erkennung von Lag-Phasen,
- ein *Ausreißer-Detektor*,
- ein Programm zur interaktiven *Bestimmung des $k_L a$-Werts* mit der dynamischen Methode,
- ein *Klassifikator* zur *On-Line-Optimierung*,
- ein interaktiver Zugang zur Konzeption *instationärer Identifikationsexperimente* und
- ein *interaktiv konfigurierbares Erweitertes Kalman-Filter*.

Programmierkonzepte

Die bisher bei der Bioprozeßüberwachung wenig verwendeten *modernen Programmiertechniken* (speziell die *objektorientierten* und *deklarativen* Konzepte) zur konkreten Umsetzung des Systems erfahren eine ausführliche Diskussion und Bewertung. Dazu wurde eine umfangreiche Software erstellt. Sie umfaßt

- die objektorientierte Verwaltung *elementarer Datentypen* (dynamische Vektoren und Matrizen, speichergesicherte Strings, Fehlerbehandlung),
- einen speziellen *Maskengenerator* zum deklarativ formulierbaren Aufbau von Benutzerschnittstellen für modellgestützte Datenanalyse-Bausteine,
- ein *On-Line-Graphik-Paket* (dynamische Visualisierung),
- einen *Scanner* und einen frei programmierbaren *Operator-Präzedenz-Parser* für die Übersetzung von arithmetischen Ausdrücken und Deklarationssprachen sowie die *Formel-Differentiation* und
- *Numerik-Routinen* zur deterministischen, statistischen und stochastischen Simulation.

Die wichtigen technischen Einzelheiten der Programmierung sind dokumentiert. Es wird gezeigt, daß sich mit Hilfe der objektorientierten und deklarativen Programmierung viele bisher aufgetretene Probleme prinzipiell lösen lassen. Dies kann am *Prototyp* des interaktiv konfigurierbaren Kalman-Filters, das in dieser Komplexität bisher nicht realisiert werden konnte, demonstriert werden.

Inhaltsverzeichnis

0 Einleitende Bemerkungen 1
- 0.1 Zur Entstehung . . . 1
- 0.2 Zur Form . . . 4
  - 0.2.1 Homogenität . . . 4
  - 0.2.2 Neue Zusammenhänge . . . 4
  - 0.2.3 Interdisziplinarität . . . 5
  - 0.2.4 Didaktik . . . 6
  - 0.2.5 Visualisierung . . . 6
  - 0.2.6 Stoffaufbereitung . . . 6
- 0.3 Zum Aufbau . . . 6
  - 0.3.1 Anwendungsorientierter Praktiker . . . 6
  - 0.3.2 Modellierer und Datenanalytiker . . . 6
  - 0.3.3 Programmierer . . . 7
  - 0.3.4 Eigenleistung . . . 7

I Problemstellung der Dissertation 9

1 Die experimentelle Situation in der Biotechnologie 12
- 1.1 Der experimentelle Zyklus . . . 12
  - 1.1.1 Laborpraxis . . . 14
  - 1.1.2 Kooperation von Theoretiker und Praktiker . . . 14
  - 1.1.3 Industriepraxis in der Produktion . . . 15
- 1.2 Die besondere Problematik bei biotechnischen Prozeßdaten . . . 15
  - 1.2.1 Ein Beispiel für eine aufwendige Instrumentierung . . . 15
  - 1.2.2 Klassifikation von Bioprozeßdaten in Hinblick auf die Datenanalyse . . . 15
    - 1.2.2.1 Abtastung . . . 15
    - 1.2.2.2 Zeitverzögerung . . . 16
    - 1.2.2.3 Genauigkeit . . . 17
    - 1.2.2.4 Verzerrungen . . . 19
    - 1.2.2.5 Wertebereich . . . 19
    - 1.2.2.6 Bereichsschranken . . . 20
    - 1.2.2.7 Physikalische Kategorie . . . 20
    - 1.2.2.8 Beziehung zu den Systemzustandsgrößen . . . 20
    - 1.2.2.9 Automatisierbarkeit . . . 21
- 1.3 Anforderungen an ein Datenanalysesystem . . . 22
  - 1.3.1 Vereinheitlichendes theoretisches Grundkonzept . . . 22
  - 1.3.2 Graphische Umgebung . . . 22
    - 1.3.2.1 Interaktive Bedieneroberfläche . . . 22
    - 1.3.2.2 Visualisierung von Daten . . . 22
    - 1.3.2.3 Benutzerführung . . . 23
    - 1.3.2.4 Dokumentation . . . 24
  - 1.3.3 Werkzeuge . . . 24

1.3.3.1 Verarbeitung aller Datentypen 24
1.3.3.2 Zukünftige Meßtechnik 24
1.3.3.3 Moderne Analyseverfahren 24
1.3.4 Datenverwaltung 24
1.3.5 Implementierung 24
1.3.5.1 Offene Architektur 25
1.3.5.2 Automatisierbarkeit 25
1.3.5.3 Geschwindigkeit 25

2 Beispiel einer Datenanalysesitzung 26
2.1 Beispieldatensätze 26
2.1.1 Konzentrationsverläufe 26
2.1.2 Bemerkungen zur Simulation 26
2.1.2.1 Modellannahmen 26
2.1.2.2 Datenerfassung 28
2.1.2.3 Meßrauschen 28
2.1.3 Ziel der Auswertung 28
2.1.4 Weitere Datensätze 29
2.2 Datenaufbereitung mit einem graphischen Editor 29
2.2.1 Phasensetzung 30
2.2.2 Detaildarstellung 30
2.2.3 Ausreißerauslöschung 30
2.3 Konfiguration eines Kalman-Filters 30
2.3.1 Prozeßmodell 32
2.3.2 Theorie 33
2.3.3 Datenflußschema 34
2.3.4 Interaktive Konfiguration 35
2.3.5 Compiler, Differenzierer und numerische Realisierung 37
2.3.6 Durchlauf 37
2.3.7 Graphische Parametrisierung 39
2.3.8 Erweiterung des Filters 39
2.4 Ausreißer-Erkennung 40
2.4.1 Signifikanztests 40
2.4.2 Verschaltung von Datenanalysebausteinen 41
2.4.3 Durchlauf 42
2.4.4 Makro-Programmierung und Prozeßanbindung 42
2.5 Phasendetektion 44
2.5.1 Umparametrisierung des Kalman-Filters 44
2.5.2 Kumulative-Summen-Tests 44
2.5.3 Datenflußschema und Verschaltung 46
2.5.4 Durchlauf 46

3 Konkretisierung des Datenanalysekonzeptes 48
3.1 Ein Spektrum von Datenanalysemethoden 48
3.1.1 Modellgestützte Verfahren 48
3.1.1.1 Deterministische Simulation 48
3.1.1.2 Stochastische Simulation 49
3.1.1.3 Kurvenanpassung 49
3.1.1.4 Zustandsbeobachter 49
3.1.1.5 Modellgestützte Filter 49
3.1.2 Lineare Systemtheorie 49
3.1.2.1 Lineare Filter und Beobachter 50
3.1.2.2 Pole-Placement-Techniken 50
3.1.3 Regler und Steuerungen 50
3.1.3.1 Dualität zu den Beobachtern und Filtern 50

3.1.3.2 Optimale Kontrolle 50
3.1.4 Nichtparametrische Methoden 50
3.1.4.1 Ausgleichsverfahren und glättende Splines 50
3.1.4.2 Fensterschätzer 51
3.1.4.3 Frequenzanalytische Methoden 51
3.1.5 Tests und Klassifikationsverfahren 51
3.1.5.1 Ein- und Mehrfachhypothesentests 51
3.1.5.2 Klassifikatoren 51
3.1.6 Schätzverfahren 51
3.1.6.1 Schätzer für Erwartungswerte, Varianzen und Verteilungen 51
3.1.6.2 Lineare Schätzer 51
3.1.6.3 Hauptkomponentenanalyse 52
3.1.6.4 Kumulative Summen 52
3.1.7 Transformationen und Algorithmen 52
3.1.7.1 Transformationen 52
3.1.7.2 Algorithmen 52
3.1.8 Visualisierungsmethoden 52
3.2 Das Modularisierungsschema für einen Datenanalysebaustein 52
3.2.1 Theorie und Methodik 53
3.2.2 Konfiguration 54
3.2.3 Compiler 55
3.2.4 Benutzerführung 55
3.2.5 Visualisierung 56
3.2.6 Numerik 56
3.2.7 Datenfluß 56
3.3 Das Arbeiten mit einem interaktiven Datenanalysesystem 57
3.3.1 Funktionalität 57
3.3.2 Analogie zu einem Werkzeugkasten 58
3.4 Interaktive Computergraphik und Explorative Datenanalyse 58
3.4.1 Interaktive Computergraphik 59
3.4.2 Explorative Datenanalyse 59
3.4.3 Explorative und konfirmatorische Datenanalyse 60
3.4.4 Programmiertechnische Entwicklungen 60
3.5 Gesamt-Ablaufsteuerung des Systems 60
3.5.1 Eine generische Architektur für informationsverarbeitende Systeme 61
3.5.2 Theoretische Grundlagen und Methodik 61
3.5.2.1 Eine vereinheitlichende Theorie der biotechnischen Datenanalyse 61
3.5.2.2 Auswahl der Methoden 61
3.5.2.3 Homogenität des Methodenspektrums 61
3.5.3 Oberfläche 61
3.5.3.1 Kommunikation mit den Werkzeugen 61
3.5.3.2 Weitere Aufgaben 62
3.5.3.3 Vereinheitlichung und Akzeptanz 63
3.5.3.4 Realisierung 63
3.5.4 Werkzeuge 63
3.5.4.1 Wartung und Analyse 63
3.5.4.2 Transformation 63
3.5.4.3 Graphische Präsentation 64
3.5.4.4 Meta-Werkzeuge 64
3.5.5 Daten 64
3.5.5.1 Datenrepräsentation und Speicherverwaltung 64
3.5.5.2 Datenablage 64
3.5.6 Implementierung 64
3.5.6.1 Programmiertechniken 64

3.5.6.2 Datentypen . . . 65
3.5.6.3 Weitere Gebiete der Programmierung . . . 65
3.6 Formulierung des Arbeitsziels . . . 65
3.6.1 Vorhandene Arbeitsumgebung . . . 65
3.6.1.1 Hardware . . . 65
3.6.1.2 Software . . . 66
3.6.2 Arbeitsziel . . . 66

4 Einordnung in das Umfeld der Bioprozeßüberwachung 67
4.1 Ein erweiterter Modellbegriff . . . 67
4.1.1 Eine Modellhierarchie . . . 67
4.1.1.1 Deterministische Modelle . . . 67
4.1.1.2 Stochastische Modelle . . . 68
4.1.1.3 Statistische Modelle . . . 68
4.1.1.4 Logische Modelle . . . 68
4.1.1.5 Heuristische Modelle . . . 68
4.1.2 Vergleich der verschiedenen Modelltypen . . . 68
4.2 Expertensysteme und klassische Regelungstechnik . . . 69
4.2.1 Vergleich der Ansätze . . . 70
4.2.2 Kritikpunkte an der heuristischen Vorgehensweise . . . 70
4.2.2.1 Informationsdichte . . . 70
4.2.2.2 Übertragbarkeit . . . 70
4.2.2.3 Implizite Numerik . . . 70
4.2.2.4 Magische Konstanten . . . 71
4.2.3 Forderungen an heuristische Modelle . . . 71
4.3 Versuch einer Synthese . . . 71
4.3.1 Eine Hierarchie von Prozeßüberwachungskomponenten . . . 71
4.3.2 Beispiele . . . 72
4.3.3 Realisierung der Schnittstellen . . . 72
4.3.4 Frame-basierte Expertensysteme . . . 73
4.4 Von der Datenanalyse zur Prozeßüberwachung . . . 73
4.4.1 Der Entwicklungszyklus . . . 73
4.4.2 Praktische Konsequenzen für den Laboreinsatz . . . 74
4.5 Zur Problematik des On-Line-Einsatzes . . . 74
4.5.1 Besonderheiten der On-Line-Anbindung . . . 75
4.5.2 Methoden zur Anbindung von Datenanalysebausteinen an den Prozeß . . . 75
4.5.3 Vergleich der Methoden . . . 77

II Modelle für biotechnische Prozesse 79

5 Deterministische Modelle 82
5.1 Modellierung biochemischer Reaktionen . . . 83
5.1.1 Chemische Reaktionsgleichungen . . . 83
5.1.1.1 Beschreibung von Einzelreaktionen . . . 83
5.1.1.2 Verallgemeinerte Reaktionsgleichungen . . . 84
5.1.1.3 Verallgemeinerte stöchiometrische Koeffizienten . . . 85
5.1.1.4 Normierung von Koeffizienten . . . 85
5.1.1.5 Beispiel . . . 86
5.1.2 Reaktionssysteme . . . 87
5.1.2.1 Beschreibung eines Reaktionssystems . . . 87
5.1.2.2 Reaktionskinetiken . . . 87
5.1.2.3 Der theoretische Reaktionsraum . . . 90
5.1.2.4 Ein Beispiel . . . 90
5.1.3 Annahmen zur Modellvereinfachung . . . 92

5.1.3.1 Separierung der Zeitskalen bei biologischen Systemen 92
5.1.3.2 Stationaritätskonzepte 93
5.1.3.3 Ein allgemeiner Modellierungsansatz für Reaktionssysteme 93
5.1.3.4 Ein Beispiel 94
5.2 Modellierung des Wachstums von Mikroorganismen 95
5.2.1 Ein einfaches Wachstumsmodell 95
5.2.1.1 Unstrukturierte Modelle 95
5.2.1.2 Das allgemeine unstrukturierte Wachstumsmodell 97
5.2.1.3 Spezifische Wachstums-, Substrataufnahme- und Produktbildungsrate . . 97
5.2.1.4 Formalkinetik 98
5.2.2 Ein Stoffwechselmodell 98
5.2.2.1 Energie-, Aufbau- und Erhaltungsstoffwechsel 98
5.2.2.2 Spezifische Raten 99
5.2.2.3 Auswertung der Stationaritäts-Annahmen 100
5.2.2.4 Formalkinetiken für den Erhaltungsstoffwechsel 101
5.2.3 Produktbildung 102
5.2.3.1 Modellannahmen für die Produktbildung 102
5.2.3.2 Auswertung der Modellannahmen 103
5.2.3.3 Zusammenfassung 103
5.2.4 Wachstum auf zwei Substraten 104
5.2.4.1 Stoffwechselwege bei *Saccharomyces cerevisia* 104
5.2.4.2 Formalisierung 104
5.2.4.3 Der Flaschenhals-Effekt 106
5.2.4.4 Wahl der Kinetiken 107
5.2.4.5 Das Modell von Sonnleitner und Käppeli 108
5.2.5 Methodische Bemerkungen 109
5.2.5.1 Allgemeiner Ansatz 109
5.2.5.2 Deduktive und evolutive Modellierungstechniken 109
5.2.5.3 Grundannahmen 109
5.2.5.4 Ablaufschema 110
5.2.5.5 Didaktische Konsequenzen 110
5.2.5.6 Konsequenzen für die Datenanalyse 111
5.2.6 Eine allgemeine Modellstruktur 112
5.2.6.1 Formalisierung der Vorgehensweise 112
5.2.6.2 Basisdarstellung des theoretischen Reaktionsraums 113
5.2.6.3 Einige Beispiele 113
5.2.6.4 Ein Vergleich mit bestehenden Ansätzen 114
5.2.6.5 Modelle mit veränderlicher Stöchiometrie 114
5.2.6.6 Globale Wachstumsgleichungen 115
5.2.7 Strukturierte Modelle 115
5.2.7.1 Instationäre zellinterne Größen 115
5.2.7.2 Regulation einer Enzymkonzentration 116
5.2.7.3 Die Problematik strukturierter Modelle 116
5.3 Modellierung von Bioreaktoren 117
5.3.1 Das allgemeine Reaktormodell 117
5.3.1.1 Zustandsvariablen 117
5.3.1.2 Konvektion, Diffusion und Reaktion 118
5.3.1.3 Praktische Anwendbarkeit 118
5.3.2 Kompartimentierung 118
5.3.2.1 Einteilung in Kompartimente und Homogenitätsannahmen 118
5.3.2.2 Modellierung eines einzelnen Kompartiments 120
5.3.2.3 Modellierung des Gesamtsystems 120
5.3.2.4 Zusammenfassung 120
5.3.3 Beispiele 121

5.3.3.1 Betriebsweisen des Rührkessels . . . 121
5.3.3.2 Strömungsrohrreaktor . . . 122
5.3.4 Gasübergang . . . 122
5.3.4.1 Der kL-Wert . . . 122
5.3.4.2 Abhängigkeit von verschiedenen Faktoren . . . 122
5.3.4.3 Quasistationaritätsannahme . . . 123
5.4 Zustandsraummodelle . . . 123
5.4.1 System- und Meßgleichung . . . 123
5.4.1.1 Systemgleichung . . . 123
5.4.1.2 Meßgleichung . . . 124
5.4.2 Beispiel: Monod-Modell für die kontinuierliche Kultur . . . 124
5.4.2.1 Systemgleichung . . . 124
5.4.2.2 Meßgleichung . . . 124

6 Statistische Modelle **125**
6.1 Grundbegriffe . . . 125
6.1.1 Veranschaulichung von Zufallsvariablen . . . 127
6.1.2 Veranschaulichung von Verteilungen . . . 127
6.1.3 Veranschaulichung von Kovarianzmatrizen . . . 128
6.1.3.1 Berechnung der Hauptachsen . . . 130
6.1.3.2 Berechnung der Wurzel . . . 130
6.2 Verteilungen . . . 131
6.2.1 Transformation . . . 131
6.2.1.1 Lineare Transformation . . . 131
6.2.1.2 Polarkoordinatentransformation . . . 132
6.2.2 Faltung . . . 132
6.2.2.1 Unsymmetrische Verteilungen . . . 132
6.2.3 Verteilungen mit unendlicher Varianz und beschränkte Verteilungen . . . 134
6.2.3.1 Verteilungen mit unendlicher Varianz . . . 134
6.2.3.2 Beschränkte Verteilungen . . . 134
6.2.4 Mischverteilungen . . . 134
6.2.4.1 Ausreißerverteilungen . . . 135
6.2.4.2 Mehrgipflige Verteilungen . . . 135
6.3 Ein Fallbeispiel: Messung der optischen Dichte . . . 135
6.3.1 Unverrauschtes Signal . . . 136
6.3.2 Normalverteilter absoluter Meßfehler . . . 136
6.3.3 Normalverteilter relativer Meßfehler . . . 136
6.3.4 Bereichsabhängiger Meßfehler . . . 137
6.3.5 Modellierung der Blasenverteilung . . . 140
6.3.6 Bereichsbeschränkung . . . 140
6.3.7 Verwendung der empirischen Verteilungsfunktion . . . 141
6.4 Das statistische Klassifikationsmodell . . . 141
6.4.1 Aufgabenstellung der Klassifikation . . . 141
6.4.1.1 Ein Beispiel . . . 141
6.4.2 Problematik der Klassifikation . . . 141
6.4.3 Das Mischverteilungsmodell für die Klassifikation . . . 142
6.4.3.1 Einschränkung auf die Klassen . . . 142
6.4.3.2 Mischung . . . 142
6.5 Lineare Modelle und Regressionsmodelle . . . 143
6.5.1 Lineare Ansätze . . . 143
6.5.1.1 Das lineare Modell . . . 143
6.5.1.2 Lineare Regressionsmodelle . . . 143
6.5.1.3 Fehler in den Variablen . . . 144
6.5.1.4 Faktorenanalytisches Modell . . . 145

6.5.1.5 Gegenüberstellung der linearen Ansätze . . . 145
6.5.2 Nichtlineare Ansätze . . . 145
6.5.2.1 Nichtlineares Regressionsmodell . . . 145
6.5.2.2 Nichtlineare Verallgemeinerung des Faktoren-Modells . . . 147
6.6 Bilanzierung . . . 147
6.6.1 Erhaltungsgrößen . . . 147
6.6.1.1 Definition einer Erhaltungsgröße . . . 147
6.6.1.2 Beispiel . . . 148
6.6.2 Bilanzierung von Biomasse . . . 149
6.6.2.1 Die Elementarzusammensetzung von Biomasse . . . 149
6.6.2.2 Beispiel . . . 150
6.6.3 Anwendung linearer Modelle auf die Bilanzierung . . . 150
6.6.3.1 Das allgemeine Stoffwechselmodell als lineares Modell . . . 150
6.6.3.2 Die Erhaltungsgleichungen als lineares Modell . . . 152
6.6.3.3 Das allgemeine Stoffwechselmodell als Faktoren-Modell . . . 154
6.6.4 Die Beziehung zwischen theoretischem, stöchiometrischem und empirischem Reaktionsraum . . . 155
6.6.4.1 Unterraumbeziehungen . . . 155
6.6.4.2 Vergleich mit bestehenden Ansätzen . . . 156
6.6.4.3 Modelle mit variabler Stöchiometrie . . . 156
6.6.5 Vorgehensweise im offenen System . . . 157
6.7 Sprung-Modelle . . . 157
6.7.1 Modellstruktur . . . 157
6.7.2 Beispiele . . . 158
6.7.2.1 Ausfall eines Meßgerätes . . . 158
6.7.2.2 Übergang von der Lag-Phase in die exponentielle Wachstumsphase . . . 158

7 Stochastische Modelle 159
7.1 Stochastische Prozesse . . . 159
7.1.1 Informelle Definition eines stochastischen Prozesses . . . 159
7.1.2 Veranschaulichung von stochastischen Prozessen . . . 159
7.1.3 Wahrscheinlichkeits-Dichte und Momente stochastischer Prozesse . . . 161
7.2 Stochastische Differentialgleichungen . . . 161
7.2.1 Definition von stochastischen Differentialgleichungen . . . 161
7.2.1.1 Naive Definition . . . 161
7.2.1.2 Der Wiener-Prozeß . . . 161
7.2.1.3 Exakte Definition . . . 162
7.2.1.4 Der Ito-Kalkül . . . 163
7.2.2 Dichtefunktionen, Erwartungsvektoren und Kovarianzmatrizen bei stochastischen Differentialgleichungen . . . 163
7.2.2.1 Die Fokker-Planck-Gleichung . . . 163
7.2.2.2 Die Matrix-Riccati-Differentialgleichung . . . 164
7.2.2.3 Der nichtlineare Fall . . . 165
7.2.3 Eigenschaften der Matrix-Riccati-Differentialgleichung . . . 165
7.2.3.1 Linearität . . . 165
7.2.3.2 Symmetrie . . . 166
7.2.3.3 Lösung der homogenen Gleichung . . . 166
7.2.3.4 Stabilität . . . 167
7.3 Einige Beispiele . . . 167
7.3.1 Die Brownsche Bewegung . . . 167
7.3.1.1 Berechnung des Erwartungsvektors . . . 167
7.3.1.2 Berechnung der Kovarianzmatrix . . . 168
7.3.2 Linearer Prozeß 1. Ordnung . . . 169
7.3.2.1 Berechnung des Erwartungswertes . . . 169

7.3.2.2 Berechnung der Varianz . . . 169
7.3.2.3 Bemerkungen zur Korrelationszeit . . . 169
7.3.3 Linearer Prozeß 2.Ordnung . . . 171
7.3.3.1 Berechnung des Erwartungsvektors . . . 171
7.3.3.2 Berechnung der Kovarianzmatrix . . . 172
7.3.3.3 Weitere Beispiele . . . 172
7.4 Ein stochastisches Wachstumsmodell . . . 174
7.4.1 Deterministisches Modell . . . 174
7.4.2 Störung der Biomasse . . . 175
7.4.3 Störung des Wachstums . . . 175
7.4.4 Dämpfung der Wachtumsschwankungen . . . 176
7.4.5 Bedeutung der stochastischen Modellierung . . . 176
7.5 Stochastische Zustandsraummodelle . . . 177
7.5.1 Meß- und Systemrauschen . . . 177
7.5.2 Die Matrix-Riccati-Differentialgleichung bei einem stochastischen Zustandsraummodell . . . 178
7.5.3 Biotechnisches Beispiel . . . 178

8 Logische und heuristische Modelle 179
8.1 Aussagenlogische Ausdrücke . . . 179
8.2 Regelsysteme . . . 179
8.3 Algorithmische Darstellung . . . 180
8.4 Syntaktische Klassifikation . . . 181
8.5 Weitere Ansätze . . . 181

III Methoden zur modellgestützten Datenanalyse 183

9 Deterministische Methoden 186
9.1 Deterministische Simulation . . . 186
9.1.0.1 Steife Systeme . . . 186
9.1.0.2 Beseitigung von Steifheiten durch Stationaritätsannahmen . . . 187
9.1.0.3 Steifheiten bei stochastischen Systemen . . . 187
9.1.1 Schrittweitensteuerung . . . 187
9.2 Beobachter . . . 188
9.2.1 Aufbau eines Beobachters . . . 188
9.2.2 Lineare zeitinvariante Theorie . . . 189
9.2.2.1 Der Konvergenzsatz bei korrektem Modell . . . 190
9.2.2.2 Eindimensionaler Fall . . . 190
9.2.2.3 Ein zweidimensionales Beispiel . . . 190
9.2.2.4 Auswirkungen von Modellfehlern . . . 192
9.2.3 Nichtlineare Beobachter . . . 193
9.2.3.1 Linearisierung . . . 193
9.2.3.2 Verschiedene Linearisierungsansätze . . . 193
9.2.3.3 Konstruktion eines Beobachters auf der Basis der Linearisierung . . . 194
9.2.3.4 Ein Beispiel . . . 194
9.2.4 Adaptive Beobachter . . . 195
9.2.4.1 Umwandlung einzelner Parameter zu Zustandsvariablen . . . 195
9.2.4.2 Ein eindimensionales Beispiel . . . 195

10 Statistische Methoden 196
10.1 Zufallszahlengeneratoren . . . 196
10.1.1 Kongruenzverfahren zur Erzeugung gleichverteilter Zufallszahlen . . . 196
10.1.2 Erzeugung von Zufallszahlen mit beliebiger Verteilung . . . 197
10.1.2.1 Eindimensionale Transformation . . . 197
10.1.2.2 Die Box-Muller-Transformation . . . 198
10.1.2.3 Skalierung . . . 199
10.1.2.4 Rejektion . . . 199
10.1.3 Kombination von Verteilungen . . . 199
10.1.3.1 Faltung . . . 199
10.1.3.2 Beschränkte Verteilungen . . . 199
10.1.3.3 Mischverteilungen . . . 200
10.1.4 Ein Baukasten zur Erzeugung von Zufallszahlen . . . 200
10.2 Regression . . . 201
10.2.1 Gütekriterien für Schätzer . . . 201
10.2.1.1 Definition von "besten" Schätzern . . . 201
10.2.1.2 Beispiele für Minimierungskriterien . . . 201
10.2.1.3 Beispiele für spezielle Klassen von Schätzern . . . 202
10.2.2 Lineare Regression . . . 202
10.2.2.1 Lineare Regression bzgl. des Kovarianzkriteriums . . . 202
10.2.2.2 Lineare Regression bzgl. des Kleinste-Quadrate-Kriteriums . . . 203
10.2.3 Nichtlineare Regression . . . 203
10.2.3.1 Linearisierung . . . 204
10.2.3.2 Optimierverfahren . . . 204
10.2.3.3 Problematik der nichtlinearen Optimierung . . . 204
10.2.4 Parameterschätzung beim Sprung-Modell . . . 205
10.2.4.1 Diskrete Form des Modells . . . 205
10.2.4.2 KQ-Schätzung . . . 205
10.2.4.3 Konstruktion der KQ-Schätzer . . . 205
10.3 Kritische Bemerkungen zur Anwendung der Parameteranpassung . . . 206
10.3.1 "Vertikale" und "horizontale" Minimierungskriterien . . . 206
10.3.1.1 Parameteranpassung bei einer Sinus-Funktion . . . 206
10.3.1.2 Ursache der schlechten Konvergenz . . . 207
10.3.1.3 Ein lokaler Ansatz zur globalen Stabilisierung . . . 207
10.3.1.4 Verallgemeinerung der Ergebnisse . . . 209
10.3.2 Biotechnologische Relevanz . . . 211
10.3.2.1 Parameteranpassung bei Lag-Phasen und Wachstumsstörungen . . . 211
10.3.2.2 Das Phasenkonzept für die Datenanalyse und Prozeßüberwachung . . . 211
10.4 Stationäre Identifikation . . . 212
10.4.1 Der stationäre Betrieb der kontinuierlichen Kultur . . . 212
10.4.1.1 Die stationären Gleichungen . . . 212
10.4.1.2 Praktische Durchführung der Methode . . . 213
10.4.2 Beispiel: Parameterbestimmung bei Modellen mit Erhaltungsstoffwechselterm . . . 213
10.4.2.1 Messung der Linearkombinationen . . . 213
10.4.2.2 Auftragung der gemessenen Werte . . . 214
10.4.2.3 Informationsgehalt der Auftragungen . . . 214
10.4.2.4 Ein Literaturbeispiel . . . 217
10.5 Statistische Tests . . . 218
10.5.1 Likelihood-Quotienten-Test . . . 218
10.5.2 Ausreißertest . . . 219
10.5.2.1 Fehler 1. und 2. Art . . . 220
10.5.3 Sprungdetektion . . . 220
10.6 Faktorenanalyse . . . 221
10.6.1 Hauptkomponentenanalyse . . . 221

10.6.2 Nichtlineare Hauptkomponentenanalyse . . . 221
10.7 Klassifikation . . . 222
10.7.1 Das Klassifikationsproblem . . . 222
10.7.2 Einige Arten von Klassifikatoren . . . 222
10.7.2.1 Likelihood-Klassifikator . . . 222
10.7.2.2 Abstandsklassifikator . . . 223
10.7.2.3 Gebietsklassifikation . . . 223
10.7.3 Heuristische Klassifikation . . . 223
10.7.4 Lernstichproben . . . 225
10.8 Nichtparametrische Glättungsverfahren . . . 225

11 Stochastische Methoden **226**
11.1 Stochastische Simulation . . . 226
11.1.1 Das stochastische Euler-Verfahren . . . 226
11.1.1.1 Naiver Lösungsansatz . . . 226
11.1.1.2 Korrekte Formulierung des Verfahrens . . . 227
11.1.2 Verfahren höherer Ordnung . . . 229
11.1.2.1 Übertragung der Verfahren in die stochastische Kategorie . . . 229
11.1.2.2 Konvergenz der Verfahren . . . 229
11.1.2.3 Konvergenzordnung der Verfahren . . . 230
11.1.2.4 Zusammenfassung . . . 230
11.1.3 Hybride Verfahren . . . 231
11.2 Filter . . . 231
11.2.1 Lineare zeitinvariante Theorie . . . 231
11.2.1.1 Die Gleichungen des linearen Filters . . . 231
11.2.1.2 Der Konvergenzsatz . . . 232
11.2.1.3 Konstruktion eines optimalen Filters . . . 232
11.2.1.4 Eindimensionales Beispiel . . . 233
11.2.2 Nichtlineare Theorie . . . 234
11.3 Kalman-Filter . . . 236
11.3.1 Die Filter-Gleichungen . . . 236
11.3.1.1 Arbeitsweise des semi-diskreten Filters . . . 236
11.3.1.2 Herleitung der Updating-Gleichungen . . . 237
11.3.1.3 Der kontinuierliche Fall . . . 238
11.3.2 Beispiele . . . 239
11.3.2.1 Eindimensionaler Fall . . . 239
11.3.2.2 Ein zweidimensionales Beispiel . . . 239
11.3.3 Parametrisierung eines Kalman-Filters . . . 239
11.3.4 Erweiterte und adaptive Kalman-Filter . . . 242
11.3.5 Numerik des Kalman-Filters . . . 242

12 Logische und heuristische Methoden **243**
12.1 Aussagenlogische Methoden . . . 243
12.2 Beweismethoden für Regelsysteme . . . 243
12.3 Algorithmische Modelle . . . 244
12.4 Syntaktische Klassifikation . . . 244
12.5 Weitere Methoden . . . 244

13 Graphische Methoden **245**
13.1 Graphische Präsentation . . . 245
13.1.1 Visuelle Perzeption . . . 245
13.1.2 Konfiguration graphischer Bausteine . . . 246
13.2 Visualisierungstechniken . . . 246
13.2.1 Hervorhebung und Inspektion von Daten . . . 246
13.2.2 Graphische Dateneingabe und -veränderung . . . 246

13.2.2.1 Dateneingabe . . . 246
13.2.2.2 Datenveränderung . . . 247
13.2.3 Auskoppeln von Teildatensätzen . . . 247
13.2.4 Dynamische Graphik . . . 247
13.2.5 3-dimensionale Darstellungen . . . 248
13.2.6 Höhere Dimensionen . . . 248
13.3 Anforderungen an die graphischen Gestaltungsmöglichkeiten . . . 248
13.3.1 Koordinatensysteme . . . 248
13.3.2 Darstellung von Punkten und Linien . . . 248
13.3.3 On-Line-Graphik . . . 248
13.3.4 Nicht gestellte Anforderungen . . . 249
13.4 Implementierungsansätze . . . 249
13.4.1 Graphisches Basissystem . . . 249
13.4.2 Interaktive Gestaltung . . . 249
13.4.3 Konfiguration . . . 249

IV Technische Details der Programmierung 251

14 Programmierkonzepte 254
14.1 Modularisierung . . . 254
14.1.1 Überblick über die Module . . . 254
14.1.2 Code-Statistik . . . 256
14.1.3 Einfluß verschiedener Programmiertechniken auf die Module . . . 256
14.2 Objektorientierte Programmierung . . . 257
14.2.1 Grundbegriffe . . . 257
14.2.1.1 Klassen und Methoden . . . 257
14.2.1.2 Vererbung . . . 258
14.2.1.3 Objekte . . . 260
14.2.2 Das Prinzip der späten Bindung . . . 260
14.2.2.1 Prozeduraler Programmierstil . . . 261
14.2.2.2 Objektorientierter Programmierstil . . . 262
14.2.3 Einsatzgebiete und Sprachen für die Objektorientierte Programmierung . . . 262
14.2.3.1 Programmiersprachen . . . 263
14.2.3.2 Umsetzung objektorientierter Konzepte in einer klassischen prozeduralen Programmiersprache . . . 263
14.2.3.3 Weitere Anmerkungen . . . 264
14.3 Weitere Konzepte der Objektorientierten Programmierung . . . 264
14.3.1 Konstruktion, Kapselung und Destruktion . . . 264
14.3.1.1 Konstruktion . . . 265
14.3.1.2 Destruktion . . . 265
14.3.1.3 Kapselung . . . 266
14.3.2 Überladen . . . 266
14.3.2.1 Überladen von Funktionen . . . 266
14.3.2.2 Überladen von Operationen . . . 267
14.3.3 Nachrichten . . . 268
14.3.3.1 Nachrichten in SmallTalk . . . 268
14.3.3.2 Programmdokumentation und versionsunabhängige Konfiguration . . . 268
14.3.4 Offenheit objektorientierter Systeme . . . 269
14.4 Deklarative Sprachkonzepte . . . 269
14.4.1 Deklarative Programmierumgebungen . . . 269
14.4.2 Anregungen für die prozedurale Programmierung . . . 269
14.4.2.1 Beispiel für den deklarativen Programmierstil . . . 270
14.4.2.2 Imitation in einer prozeduralen Umgebung . . . 270

14.5 Programmierung paralleler Prozesse . . . 271
14.5.1 Warteschlangen . . . 272
14.5.2 Prozeßstatus . . . 272
14.5.3 Inter-Prozeß-Kommunikation . . . 272
14.6 Auswahl der Programmiersprache für diese Arbeit . . . 272

15 Datentypen 274
15.1 Strings . . . 274
15.1.1 Pascal- und C-Strings . . . 274
15.1.2 Erweiterung der C-Strings . . . 275
15.1.3 Restrisiken . . . 276
15.1.4 Operationen auf erweiterten C-Strings . . . 276
15.1.5 Objektorientierte Erweiterung . . . 276
15.2 Vektoren und Matrizen . . . 276
15.2.1 Anforderungen an dynamische Vektoren und Matrizen . . . 276
15.2.2 Realisierung als Datenstruktur . . . 277
15.2.2.1 Einträge in der Struktur . . . 277
15.2.2.2 Unterstützte Typen und Default-Werte . . . 278
15.2.3 Operationen . . . 279
15.2.3.1 Dimensionsoperationen und ihre Umsetzung . . . 279
15.2.3.2 Weitere Operationen . . . 279
15.2.4 Dimensionslisten . . . 280
15.2.4.1 Ein Beispiel . . . 280
15.2.4.2 Arbeitsweise von Dimensionslisten . . . 280
15.2.4.3 Definition von Dimensionslisten . . . 280
15.2.4.4 Constraint Propagating . . . 282
15.2.5 Vorteile und Einschränkungen . . . 282
15.3 Sequentielle Speicher . . . 283
15.3.1 Anforderungen an sequentielle Speicher . . . 283
15.3.1.1 Typ der Elemente . . . 283
15.3.1.2 Operationen . . . 284
15.3.1.3 Performanz . . . 285
15.3.2 Implementierung von sequentiellen Speichern . . . 285
15.3.2.1 Mögliche Datenstrukturen . . . 285
15.3.2.2 Umgesetzte Typen von sequentiellen Speichern . . . 285
15.4 Fehlerbehandlung . . . 286
15.4.1 Aufgaben einer Fehlerprüfung und -behandlung . . . 287
15.4.2 Realisierung der Fehlerbehandlung . . . 287
15.4.2.1 Fehlerdaten . . . 287
15.4.2.2 Fehlerbehandlungsmodi . . . 288
15.4.3 Operationen zur Fehlerbehandlung . . . 288
15.5 Objektorientierte Realisierung der Datentypen . . . 288
15.5.1 Klassenbibliotheken . . . 289
15.5.2 Kopieroperationen . . . 289

16 Benutzeroberflächen 290
16.1 Anforderungen und Lösungsansätze . . . 290
16.1.1 Anforderungen an eine Benutzeroberfläche für Datenanalyse und Prozeßüberwachung 290
16.1.2 Verschiedene Ansätze zur Beschreibung von Benutzeroberflächen . . . 291
16.1.2.1 Interface-Definitions-Sprachen . . . 291
16.1.2.2 Deklarative Formulierung . . . 291
16.1.2.3 Datenstrukturen aus Menüobjekten . . . 292
16.2 Konzeption und Arbeitsweise des Maskengenerators . . . 292
16.2.0.4 Nachrichtenbasierte Kontrolle . . . 293
16.2.0.5 Ein Beispiel für eine Maskendefinition . . . 293

16.2.1 Nachrichten . . . 294
16.2.1.1 Der Nachrichten-Stack . . . 295
16.3 Lösung der gestellten Probleme . . . 295
16.3.1 Übersichtliche Codierung . . . 295
16.3.2 Wissenschaftliche Datentypen . . . 295
16.3.3 Dynamische Strukturen . . . 296
16.3.4 Konfiguration . . . 296
16.3.4.1 Dateiformat . . . 296
16.3.4.2 Abspeichern . . . 296
16.3.4.3 Einlesen . . . 297
16.3.5 Hilfestellung . . . 297
16.3.6 Dokumentation . . . 297

17 Graphik **298**
17.1 Anforderungen an ein Graphiksystem für die Datenanalyse . . . 298
17.1.1 Erfahrungen aus der Anwendungspraxis . . . 298
17.1.2 Ein Anforderungskatalog . . . 299
17.2 Offene und portable Konzeption . . . 300
17.3 Koordinaten . . . 300
17.3.1 Koordinatenebenen . . . 300
17.3.2 Koordinatentransformationen . . . 300
17.3.3 Aspekt, Rotation und Scherung . . . 301
17.3.4 Welten . . . 302
17.4 Graphische Grundelemente . . . 302
17.4.1 Verwendung von Vektorfonts . . . 304
17.5 Zeichenstifte . . . 304
17.5.1 Virtuelle Zeichenstifte . . . 304
17.5.1.1 Beschreibungsdaten . . . 304
17.5.1.2 Operationen . . . 304
17.5.2 Welt-Zeichenstifte . . . 305
17.5.2.1 Attribute . . . 305
17.5.2.2 Operationen . . . 305
17.6 Koordinatensysteme . . . 305
17.6.1 Grundelemente eines Koordinatensystems . . . 305
17.6.2 Formulierung des Koordinatenlayouts . . . 307
17.6.3 Realisierung des Layouts . . . 308
17.6.4 Mögliche Erweiterungen . . . 308
17.7 Meta-Files und Hard-Copies . . . 309
17.7.1 Meta-Files . . . 309
17.7.1.1 Realisierung . . . 309
17.7.2 Konvertierung in andere Graphikformate . . . 309
17.7.3 Hard-Copies . . . 309
17.7.3.1 Realisierung und Format . . . 309
17.7.4 Konvertierung in andere Graphikformate . . . 309
17.8 Fallstricke . . . 309
17.8.1 Clipping . . . 310
17.8.2 Virtuelle Koordinaten . . . 311
17.8.3 Metafile-Formate . . . 311
17.8.4 Ganzzahliges Rechnen . . . 312
17.8.5 Koordinatenlayout . . . 312
17.8.6 Cursor . . . 313
17.8.7 Linienmusterzähler . . . 313

18 Compiler 315
18.1 Anforderungen . . . 315
18.2 Prinzipieller Aufbau eines Compilers . . . 316
18.3 Sprachklassen und Sprachanalyse . . . 317
18.3.1 Grammatiken . . . 317
18.3.2 Chomsky-Hierarchie . . . 318
18.3.3 Unterscheidung der kontextsensitiven Sprachen . . . 320
18.3.4 Scanner- und Parser-Generatoren . . . 321
18.4 Der Scanner . . . 321
18.4.1 Ringpuffer und Lookahead-Puffer . . . 322
18.4.2 Funktionen . . . 323
18.4.3 Zusätzliche Wahlmöglichkeiten bei der Initialisierung . . . 323
18.5 Der prädiktive Parser . . . 323
18.5.1 Eine prädiktive Grammatik . . . 324
18.5.2 Rekursiver Abstieg . . . 324
18.5.3 Mehrdeutigkeiten . . . 326
18.6 Der Operator-Präzedenz-Parser . . . 326
18.6.1 Präzedenz . . . 326
18.6.2 Assoziativität . . . 327
18.6.3 Die Operatortabelle . . . 327
18.6.4 Die Bezeichnertabelle . . . 328
18.6.5 Arbeitsweise des Operator-Präzedenz-Parsers . . . 328
18.7 Codegenerierung . . . 329
18.7.1 Umgekehrte Polnische Notation . . . 329
18.7.2 Übersetzung und Auswertung der Umgekehrten Polnischen Notation . . . 329
18.7.3 Maschinencode-Generierung für arithmetische Ausdrücke . . . 330
18.7.4 Übersetzung von Kontrollstrukturen . . . 331
18.8 Differenzieren und Vereinfachen . . . 331
18.8.1 Differenzieren . . . 331
18.8.2 Vereinfachen . . . 332

19 Numerik 334
19.1 Verfahren der linearen Algebra . . . 334
19.2 Verfahren für gewöhnliche Differentialgleichungen . . . 334
19.3 Zufallszahlengeneratoren und Verteilungen . . . 335
19.4 Statistische Verfahren . . . 335
19.5 Verfahren für stochastische Differentialgleichungen . . . 335

20 Datenflußmanagement und Datenverwaltung 337
20.1 Statistische und regelungstechnische Ansätze . . . 337
20.1.1 Das Kanalkonzept bei regelungstechnischen Systemen . . . 337
20.1.2 Das Transformationskonzept bei statistischen Systemen . . . 338
20.1.3 Eine Hybridlösung . . . 338
20.2 Anforderungen an ein biotechnologisches Datenanalysesystem . . . 341
20.2.1 Speicherplatzbedarf . . . 341
20.2.2 Ein Anforderungskatalog . . . 341
20.2.2.1 Datentypen . . . 341
20.2.2.2 Zeitausdehnung . . . 341
20.2.2.3 Zugriff . . . 342
20.2.2.4 Zeitrepräsentation . . . 342
20.2.2.5 Interpolation . . . 342
20.2.2.6 Markierung . . . 342
20.2.2.7 Vektor- und Matrizen-Repräsentation . . . 342
20.3 Ein erweitertes Kanalkonzept . . . 342
20.3.1 Erweiterte Kanäle . . . 342

20.3.2 Kanalverwaltung . . . 343
20.3.3 Verwirklichung des Phasen-Konzepts für die Datenanalyse . . . 343
20.3.4 Kommunizierende Objekte . . . 344
20.3.5 Objektorientierte Realisierung . . . 344
20.4 Datenbasis . . . 344

V Anwendungsbeispiele 347

21 Ein frei konfigurierbares Kalman-Filter 350
21.1 Benutzeroberfläche und Funktionen . . . 350
21.1.1 Datei-Eingabe und Ausgabe . . . 350
21.1.2 Modelleingabe . . . 351
21.1.3 Parameter- und Variableneingabe . . . 351
21.1.4 Bereichsgrenzen für die Rauschterme . . . 351
21.1.5 Graphik-Konfiguration . . . 351
21.1.6 Übersetzung, Fehlerprüfung und Durchlauf . . . 351
21.2 Implementierung . . . 352
21.2.1 Datenstruktur . . . 352
21.2.2 Oberfläche . . . 353
21.2.3 Operationen . . . 353
21.3 Einsatzbereiche . . . 354
21.3.1 Parameter- und Modellstudien . . . 354
21.3.2 Praktische Erfahrungen . . . 354
21.3.3 Downloading . . . 354
21.4 Vergleich mit anderen Systemen . . . 354

22 Weitere Anwendungsbeispiele 357
22.1 Instationäre Identifikation . . . 357
22.1.1 Prinzip der instationären Identifikation . . . 357
22.1.1.1 Stationäre und instationäre Identifikation mittels Parameteridentifikation 357
22.1.1.2 Instationäre Identifikation mittels Filterung . . . 358
22.1.1.3 Voraussetzungen für die Anwendbarkeit der Methode . . . 358
22.1.2 Ein Beispiel . . . 358
22.1.2.1 Batch-Kultur . . . 359
22.1.2.2 Kontinuierliche Kultur . . . 359
22.1.2.3 Verlängerte Batch-Kultur mit Pulszufütterung . . . 359
22.1.2.4 Verlängerte Batch-Kultur mit stetiger Zufütterung . . . 362
22.1.2.5 Instationär geführte kontinuierliche Kultur . . . 362
22.1.2.6 Fed-Batch-Kultur . . . 363
22.1.3 Weitere Entwicklungen . . . 363
22.1.3.1 Modularer Aufbau im Rahmen des Datenanalysesystems . . . 363
22.2 On-Line-Optimierung . . . 363
22.2.1 Anwendungsbereiche für heuristische Methoden . . . 363
22.2.2 Das Optimierungsproblem . . . 364
22.2.2.1 Heuristische Vorgehensweise eines Betreibers . . . 364
22.2.3 Klassifikationsansatz . . . 365
22.2.3.1 Merkmalsraum . . . 365
22.2.3.2 Berechnung des Merkmalsvektors . . . 366
22.2.3.3 Lernstichprobe . . . 366
22.2.3.4 Heuristischer Klassifikator . . . 366
22.2.3.5 Durchlauf . . . 368
22.2.4 Ausblick über weitere Entwicklungen . . . 368
22.2.4.1 Weitere Probleme bei der On-Line-Optimierung . . . 368
22.2.4.2 Modularer Aufbau . . . 371

22.2.4.3 Weiterentwicklung . . . 371
22.3 Dynamische Methode zur Bestimmung des kLa-Werts . . . 371
22.3.1 Durchführung des Experiments . . . 371
22.3.1.1 Grundannahmen . . . 371
22.3.1.2 Ablauf . . . 372
22.3.2 Theoretischer Kurvenverlauf . . . 372
22.3.3 Auswertung des Experiments . . . 372
22.3.3.1 Parameteranpassung . . . 372
22.3.3.2 Problematik der Parameteranpassung . . . 372
22.3.3.3 Hybrid-Methode . . . 373
22.3.4 Realisierung . . . 374
22.3.4.1 Interaktive Lösung . . . 374
22.3.4.2 On-Line-Realisierung im Rahmen des Datenanalyse-Konzepts . . . 375

VI Anhang 377

A Technische Anforderungen für eine Implementierung 379
A.0.5 Software . . . 379
A.0.5.1 Programmiersprachen . . . 379
A.0.5.2 Programmierung interaktiver und heuristischer Komponenten . . . 379
A.0.5.3 Benutzeroberfläche . . . 380
A.0.5.4 Graphik . . . 380
A.0.5.5 Numerik . . . 380
A.0.5.6 Betriebssystem . . . 381
A.0.6 Hardware . . . 381
A.0.6.1 Systemarchitektur . . . 381
A.0.6.2 Rechengeschwindigkeit . . . 381
A.0.6.3 Speicherplatz . . . 382
A.0.7 Entscheidung für eine Konfiguration . . . 382
A.0.7.1 Auswahlkriterien . . . 382
A.0.7.2 Auswahl der Hardware . . . 382
A.0.7.3 Auswahl der Software . . . 383
A.0.7.4 Mixed Language Programming . . . 383

B Notationen 384
B.1 Reaktionssysteme und Kinetik . . . 384
B.1.1 Verallgemeinerte Reaktionssysteme . . . 384
B.1.2 Enzymkinetik . . . 384
B.1.3 Kinetik-Funktionen . . . 385
B.2 Stoffwechsel-Modelle . . . 385
B.2.1 Soffwechselwege . . . 385
B.2.2 Stoffkonzentrationen bzw. -mengen . . . 386
B.2.3 Stoffwechselwege . . . 386
B.2.4 Wachstumskinetik . . . 386
B.2.5 Stöchiometrische Koeffizienten . . . 387
B.2.6 Allgemeines Stoffwechselmodell . . . 387
B.2.7 Bilanzierung . . . 387
B.2.8 Empirischer Reaktionsraum . . . 387
B.3 Spezielle Modelle . . . 388
B.3.1 Backhefe-Modell . . . 388
B.3.2 Bioreaktoren . . . 388
B.3.3 Gasübergang . . . 388
B.4 Wahrscheinlichkeitstheorie . . . 389
B.4.1 Grundbegriffe . . . 389

B.4.2 Verteilungen . . . 389
B.4.3 Hauptachsentransformation . . . 389
B.5 Statistik . . . 389
B.5.1 Schätztheorie . . . 389
B.5.2 Lineare und nichtlineare Modelle . . . 390
B.5.3 Testtheorie . . . 390
B.5.4 Sprungdetektion . . . 390
B.5.5 Klassifikation . . . 391
B.6 Deterministische und stochastische Prozesse . . . 391
B.6.1 Stochastische Prozesse . . . 391
B.6.2 Stochastische Differentialgleichungen . . . 391
B.7 Zustandsraum-Modelle, Beobachter und Filter . . . 392
B.7.1 Zustandsraum-Modelle . . . 392
B.7.2 Lineare Zustandsraum-Modelle . . . 392
B.7.3 Filter und Beobachter . . . 392

Abbildungsverzeichnis

0.1 Abhängigkeitsbaum für die theoretischen Grundlagen des Erweiterten Kalman-Filters . . 5

1.1 Varianten des experimentellen Zyklus in der Biotechnologie 13
1.2 Grobschema des Methanolstoffwechsels von *Candida boidinii* (nach [WNSW90]) 16
1.3 Meßgrößen bei einem aufwendig instrumentierten Fermenter 17
1.4 Charakterisierung biotechnischer Meßdaten 18
1.5 Graphische Visualisierung numerischer Daten 23

2.1 Diauxisches Wachstum von Backhefe auf Glucose 27
2.2 Typisches Signal einer gasanalytischen Messung (instationärer Prozeßverlauf) 29
2.3 Graphischer Dateneditor 31
2.4 Struktur eines stochastischen Zustandsraummodells 32
2.5 Datenflußschema eines Erweiterten Kalman-Filters 35
2.6 Interaktive Konfiguration eines Erweiterten Kalman-Filters 36
2.7 Durchlauf eines Erweiterten Kalman-Filters zur Schätzung der Wachstumsrate 38
2.8 Schätzung von Substratkonzentration und Ertragskoeffizient bei Vorliegen einer Zeitverzögerung (stark ausschnittvergrößert) 40
2.9 Datenflußschema eines Signifikanztests 41
2.10 Datenflußschema eines Ausreißer-Detektors 42
2.11 Durchlauf eines Aussreißer-Detektors für die Messung der Biomassekonzentration 43
2.12 Träge und übersensibel parametrisiertes Kalman-Filter zur Wachstumsratenschätzung . . 45
2.13 Datenflußschema für einen Sprung-Detektor 46
2.14 Durchlauf eines Sprung-Detektors zur Lag-Phasen-Erkennung 47

3.1 Modularisierungsschema für einen Datenanalysebaustein 54
3.2 Oberfläche eines interaktiven Datenanalysesystems 57
3.3 Generische Architektur für informationsverarbeitende Systeme 62

4.1 Bewertung von Modellierungsansätzen unter verschiedenen Kriterien 69
4.2 3-Ebenen-Architektur für die Bioprozeßüberwachung 72
4.3 Evolutionäre Strategie zur Modellbildung bei biotechnischen Prozessen 74
4.4 Ein-/Ausgangsschema eines Bioprozesses 75
4.5 Methoden zur Anbindung eines Datenanalysesystems an den Prozeß 76

5.1 Verschiedene Kinetiken 89
5.2 Theoretischer Reaktionsraum bei einer Enzymreaktion 91
5.3 Verschiedene Zeitkonstanten bei biologischen Systemen (nach [Roe83]) 92
5.4 Verschiedene Stoffwechselmodelle 96
5.5 Stoffwechselschema der Backhefe *Saccharomyces cerevisia* 105
5.6 Mögliche Arbeitsweisen des Flaschenhalses bei *Saccharomyces cerevisia* 107
5.7 Ablaufschema für die Stoffwechsel-Modellierung 111
5.8 Kompartimentierung von Bioreaktoren 119
5.9 Betriebsweisen im Rührkesselreaktor 121

6.1 Visualisierung 1-dimensionaler Wahrscheinlichkeits-Verteilungen 128
6.2 Visualisierung 2-dimensionaler Wahrscheinlichkeits-Verteilungen 129
6.3 Einige Beispiele für Verteilungen 133
6.4 Aufbau einer Trübungssonde 136
6.5 Typisches Signal einer Trübungssonde 137
6.6 Modellierung des Meßrauschens bei einer Trübungsmessung (Teil 1) 138
6.7 Modellierung des Meßrauschens bei einer Trübungsmessung (Teil 2) 139
6.8 Visualisierung eines Merkmalsraums 143
6.9 Verschiedene lineare Modellansätze 146
6.10 Dimensionsbeziehungen beim theoretischen Reaktionsraum 151
6.11 Dimensionsbeziehungen beim stöchiometrischen Reaktionsraum 154
6.12 Sprungmodell 158

7.1 Visualisierung von stochastischen Prozessen 160
7.2 Lösung einer stochastischen Differentialgleichung mit zustandsabhängigem Rauschterm. . 165
7.3 Brownsche Bewegung in 1 und 2 Dimensionen 168
7.4 Linearer Prozeß 1. Ordnung 170
7.5 Linearer Prozeß 2. Ordnung 173
7.6 Stochastisches Wachstumsmodell (Teil 1) 174
7.7 Stochastisches Wachstumsmodell (Teil 2) 175

9.1 Allgemeines Schema eines Zustandsbeobachters 189
9.2 1-dimensionaler Beobachter 191

10.1 Box-Muller-Transformation 198
10.2 Niveauliniendiagramm für die Anpassung einer Sinusfunktion 208
10.3 "Vertikale" und "horizontale" Minimierungskriterien 209
10.4 Global stabilisierte Parameteranpassung 210
10.5 Stationäre Identifikation beim Monod-Modell 215
10.6 Stationäre Identifikation beim Inhibierungs-Modell 216
10.7 Wachstumskinetik von *Candida boidinii* 217
10.8 Verschiedene Methoden der Klassifikation 224

11.1 Stochastisches Euler-Verfahren für den zweidimensionalen Brownschen Prozeß 228
11.2 Eine Vergleichsrelation zwischen Kovarianzmatrizen 233
11.3 Asymptotische Varianz eines linearen Filters 1.Ordnung 234
11.4 1-dimensionales lineares Filter 235
11.5 Arbeitsweise des semikontinuierlichen Kalman-Filters 237
11.6 Durchlauf eines 2-dimensionalen linearen Kalman-Filters 240
11.7 Durchlauf eines 2-dimensionalen linearen Kalman-Filters bei inkorrekter Modellierung des Meßrauschens. 241

14.1 Statistik der im Rahmen dieser Arbeit erstellten Programm-Module 256
14.2 Einfluß der verschiedenen Programmiertechniken auf die erstellten Programm-Module . . 257
14.3 Zeigerliste mit den Bausteinen eines Netzwerks 261
14.4 Konzepte und Strukturen für die Programmierung paralleler Prozesse 271

15.1 Unterschiede zwischen Pascal- und C-Strings 275
15.2 Realisierung von dynamischen Vektoren und Matrizen 277
15.3 Durchführung einer Dimensionsoperation (Einfügens eines Elementes) 279
15.4 Funktionen, Vektoren und Matrizen eines Erweiterten Kalman-Filters mit ihren Dimensionierungen 281
15.5 Dimensionsliste für das Kalman-Filter 282
15.6 Verschiedene Arten von sequentiellen Speichern 283
15.7 Realisierung von sequentiellen Speichern 285

15.8 Kopieroperationen unterschiedlicher Tiefe 289

16.1 Nachrichtenbasierte Kontrolle einer Benutzeroberfläche 292

17.1 Koordinatenebenen 301
17.2 Graphische Grundelemente und ihre Attribute 303
17.3 Aufbau von Koordinatensystemen ("Explosionszeichnung") 306
17.4 Ganzzahliger Überlauf bei der Koordinatentransformation 310
17.5 Interferenzmuster bei Verwendung von Linienmusterzählern 313

18.1 Ablaufphasen eines Compilers 317
18.2 Syntaxbaum zu einer kontextfreien Grammatik 319
18.3 Syntaxgraph zu einer regulären Grammatik 320
18.4 Pufferanordnung des Scanners 322

20.1 Kanalkonzept und Ablaufsteuerung in einem regelungstechnischen Datenanalysesystem . . 339
20.2 Transformationskonzept und Ablaufsteuerung in einem statistischen Datenanalysesystem . 340
20.3 Erweitertes Kanalkonzept 343
20.4 Zustands-Transitions-Diagramm 345

21.1 Einsatz eines Kalman-Filters zur Wachstumsratenschätzung bei einer Fed-Batch-Kultur von *Candida boidinii* (aus [WNSW90]) 355

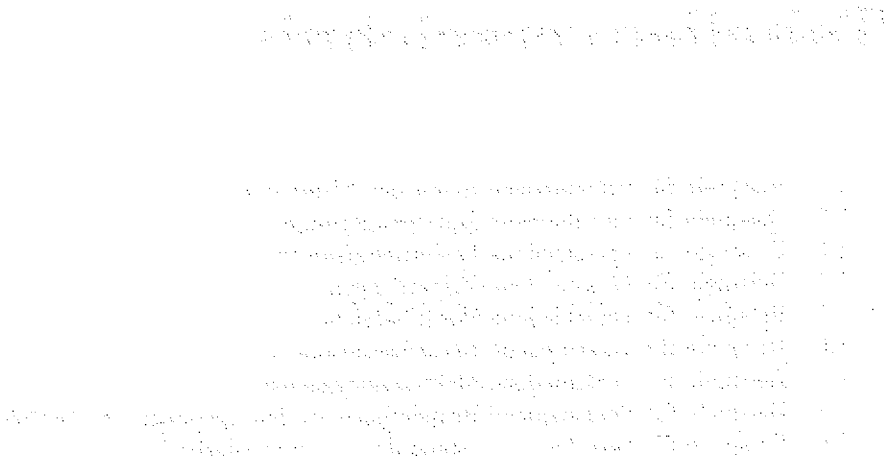
22.1 Instationäre Open-Loop-Identifikation 360
22.2 Identifikation bei einer instationär geführten Fed-Batch-Kultur 361
22.3 Identifikation bei einer instationär geführten kontinuierlichen Kultur 362
22.4 Zwei Beispiele für die heuristische Optimierung eines Abwasserprozesses durch einen Betreiber 365
22.5 Interaktive Zuordnung eines Kurvenverlaufs zu einer Klasse 367
22.6 Auftragung der Objekte zweier Klassen im Merkmalsraum 368
22.7 Heuristischer Klassifikator 369
22.8 Durchlauf des heuristischen Klassifikators 370
22.9 Idealer und realer Zeitverlauf des dynamischen kLa-Experiments 373
22.10Parameteranpassung für das kLa-Modell bei nicht berücksichtigter Zeitverzögerung 374
22.11Durchlaufbeispiel für die interaktiv gesteuerte kLa-Bestimmung 375

Tabellenverzeichnis

1.1 Beispiele für verschiedene Arten der Abtastung 16
1.2 Beispiele für verschiedene Zeitverzögerungen 17
1.3 Beispiele für verschiedene Fehlerbehaftungen 19
1.4 Beispiele für verschiedene Verzerrungen 19
1.5 Beispiele für verschiedene Wertebereiche 19
1.6 Beispiele für verschiedene Bereichsschranken 20
1.7 Beispiele für verschiedene Meßwertkategorien 21
1.8 Beispiele für verschiedene Beziehungen zu den Systemzustandsgrößen 21
1.9 Beispiele für verschiedene Grade der Automatisierbarkeit 22

15.1 Unterschiede zwischen den verschiedenen Realisierungen eines sequentiellen Speichers . . . 286

22.1 Numerische Ergebnisse für die Schätzer im kLa-Modell bei nicht berücksichtigter Zeitverzögerung 373



Kapitel 0

Einleitende Bemerkungen

Der Umfang der vorliegenden Dissertation geht deutlich über das übliche Maß hinaus. Es erscheint mir daher angebracht, einige Bemerkungen zur Entstehung des Textes vorwegzustellen.

0.1 Zur Entstehung

Die Festlegung des Themengebiets datiert auf das Jahr 1986. Es sollten Anwendungen der stochastischen Filter- und Identifikationstheorie auf biotechnische Prozesse behandelt werden. Für unsere Arbeitsgruppe stellte dieser Anwendungsbereich Neuland dar, so daß zunächst ein Fundament zu erarbeiten war. Der interdisziplinäre Charakter der Biotechnologie erforderte den Erwerb von Kenntnissen auf den Gebieten Mikrobiologie, Bioverfahrenstechnik, Meß- und Regelungstechnik und Informatik. Da ich zuvor auf dem Gebiet der Reinen Mathematik über Probleme der Algebra und Topologie diplomiert hatte, bedeutete dies für mich allerdings zusätzlich auch das Studium von angewandten mathematischen Themen wie der Theorie dynamischer Systeme, stochastischer Prozesse, Statistik und Numerik. Die für ein neues Themengebiet naturgemäß zunächst unscharfe Themenstellung ließ von vornherein einige Kurskorrekturen erwarten, was meinem Doktorvater Herrn Priv.Doz.Dr. Wolfgang Martin einige Flexibilität abverlangt hat. Er war für mich vor allem auf den Bereichen der Statistik, Datenanalyse und Software-Entwicklung ein kompetenter Gesprächspartner, wofür ich ihm an dieser Stelle besonders danken möchte.

Das Unternehmen wäre ohne die vielen persönlichen Kontakte und fachlichen Auseinandersetzungen, die oft für entscheidende Wendungen gesorgt haben, sicherlich zum Scheitern verurteilt gewesen. Der bestehende Ansatz hat erst dadurch seine Komplexität erhalten, daß ich bemüht war, keine der gemachten Erfahrungen zu vernachlässigen: Meine Überzeugung ist heute, daß kein festes Methodenspektrum — sei es das Spektrum der klassischen Kontrolltheorie, der Fuzzy Control Theory oder der Regelsysteme der Künstlichen Intelligenz — der Komplexität der Bioprozeßüberwachung gerecht werden kann. Die Aufgabe eines Systemtheoretikers auf dem Feld der Bioprozeßüberwachung und -kontrolle kann daher kaum darin bestehen, existierende Ansätze weiter zu verfeinern, um so immer kompliziertere Detaillösungen zu erstellen. Gefordert ist vielmehr ein hybrider Ansatz, der die verschiedensten Techniken miteinander verbindet und so das Gesamtproblem handhabbar macht. Einen Schritt in dieser Richtung stellt die vorliegende Arbeit dar.

Die für mich wichtigste Erfahrung war die praktische Kultivierung von Mikroorganismen im Labor. Dieser Zugang wurde mir vor allem ermöglicht durch Herrn Jörg Severin, Herrn Axel Wohfarth, Herrn Ewald Müller und Herrn Dr. Erwin Galinski vom Institut für Mikrobiologie und Biotechnologie der Universität Bonn, die mir bei meinen Versuchen mit Petri-Schale, Schüttelkolben und Laborfermenter stets mit Rat und Tat zur Seite gestanden haben. Meine heutige Sichtweise biotechnischer Systeme ist geprägt von der Skepsis des Mikrobiologen: Ein Bioreaktor ist kein rein technisches System, sondern in erster Linie ein biologisches. Der Unterschied besteht in einer weitaus geringeren Vorhersagbarkeit des biologischen Systems aufgrund der enormen Zahl von möglichen Einflußgrößen und Stoffwechselmechanismen, die zum Teil nicht einmal qualitativ bekannt sind. Eine Modellierung muß hier stark vereinfachen mit der Folge, daß die Übertragung der Ergebnisse von einer experimentellen Situation in eine andere in der Regel scheitert. Insbesondere bei auftretenden Störungen erlaubt ein Modell, welches bei normalem Prozeßver-

lauf gute Anpassungsergebnisse liefert, keine Vorhersage mehr. Ich habe mich bemüht, die Konsequenzen dieser im wesentlichen stochastischen Sichtweise des mikrobiellen Wachtums in der vorliegenden Arbeit aufzuzeigen. Sie schlägt sich im Phasenmodell des Datenanalyse-Konzepts nieder.

Mit Herrn Jörg Severin verbindet mich eine lange interdisziplinäre Zusammenarbeit, die in der Entwicklung des Fermentationsversuchs für das Bonner biotechnologische Praktikum, dem Seminar "Mathematische Grundlagen der Prozeßsimulation" am Haus der Technik in Essen und schließlich im Vorlesungsseminar "Mathematische Methoden in Mikrobiologie und Biotechnologie" viele Früchte getragen hat. Das Bemühen um die didaktische Aufbereitung des Stoffs hat auch das Datenanalyse-Konzept stark beeinflußt. Vor allem die diversen Visualisierungstechniken sind hier entwickelt und erprobt worden. Ich vertrete die Auffassung, daß die wissenschaftliche und die didaktische Aufbereitung eines Stoffs nicht voneinander zu trennen sind. Es widerspricht daher nicht den Zielen einer Dissertation wenn der vorliegende Text auch unter didaktischen Gesichtspunkten aufgebaut ist.

Bereits in die Anfangsphase der Arbeit fällt meine Begegnung mit Herrn Dr. Dirk Schürbüscher vom Institut für Biotechnologie 2 des Forschungszentrums Jülich, die einen seither zunehmend intensivierten Austausch begründet hat. Erst durch die Konfrontation mit den konkreten Fragestellungen der biotechnischen Verfahrensentwicklung war mir eine realistische Einschätzung der regelungstechnischen Anforderungen und Entwicklungen auf diesem Gebiet möglich. Zu einer möglichst praxisnahen Ausrichtung der Gesamtkonzeption hat das Gespräch mit Herrn Schürbüscher, der auch das Seminar in Essen mit getragen hat, und später auch mit Herrn Stephan Freyer, Herrn Michael Wedy und Herrn Rolf Eberhardt sehr beitragen. Der Einsatz des Kalman- Filters am *Candida boidinii*-Prozeß war ein erstes konkretes Ergebnis dieser Zusammenarbeit.

Ein Grundpfeiler der Modellierung biologischer Systeme ist die Simulation auf dem Computer. Mit der Neueröffnung der Abteilung Theoretische Biologie im Jahre 1986 an der Universität Bonn, der ich seit ihrer Gründung angehöre, fand ich ein optimales Umfeld für das Erlernen der Kunst des Modellierens. Durch den Austausch mit Herrn Prof.Dr. Wolfgang Alt erschlossen sich mir viele Techniken, die in keinem Buch schriftlich fixiert sind. Obwohl wir an verschiedenen biologischen Systemen arbeiten, hat eine Hinwendung zu einer mehr stochastischen Sichtweise fast gleichzeitig stattgefunden, was sich sehr befruchtend ausgewirkt hat. Die numerischen Verfahren zur Simulation zufälliger Vorgänge sind selbst unter Modellierern oft kaum bekannt. Mit der in dieser Arbeit präsentierten stochastischen Wachstums-Simulation und der realistischen Simulation biotechnischer Meßvorgänge ist Neuland betreten worden. Insbesondere hat sich gezeigt, daß ein allgemein verständlicher Zugang zur stochastischen Filter-Theorie über die Simulation und Visualisierung stochastischer Prozesse möglich ist. Das Kalman-Filter wird auf diese Weise eingeführt, wobei gleichzeitig deutlich wird, daß das semikontinuierliche Kalman-Filter gewissermaßen die "natürliche" Formulierung des Filters darstellt.

Entscheidend konkretisiert hat sich die Konzeption nach dem Besuch der Tagung "Computer Applications in Fermentation Technology" im September 1988 in Cambridge. Hier wurden erstmals in größerem Ausmaß Ansätze zur Bioprozeßüberwachung mit Hilfe von Expertensystemen vorgestellt und in einen Gegensatz zur klassischen, auf Differentialgleichungs-Modelle gestützten Prozeßkontrolle gesetzt. Das Resumée dieser Tagung war für mich die Einsicht, daß keine der beiden Sichtweisen auf Dauer zum Ziel führen kann: Deterministische Modelle versagen, sobald kritische Zustände und Störungen im Prozeßverlauf auftreten, wogegen die regelbasierte Kontrolle, angewendet auf mathematisch traktierbare Teilsysteme, eher einen Rückfall in die Frühzeit der Regelungstechnik darstellt. Die eindrucksvollen Vorträge von Herrn Prof. Stephanopoulos und Herrn Prof. Halme, die bereits Ansätze zur Vermittlung aufwiesen, haben mir einige Anregungen gegeben, die ich nun ausgeführt habe.

Der Gedanke einer Hybridlösung, die den simultanen Einsatz verschiedenster Techniken ermöglicht, ist schnell formuliert. Die praktische Durchführung auf dem Computer wirft dagegen große Probleme auf: Differentialgleichungs-Modelle und Regelsysteme sind einander so wesensfremd, daß eine Kommunikation zwischen entsprechenden Bausteinen zur Prozeßüberwachung nur in sehr beschränktem Umfang möglich ist. Der Schlüssel zur Lösung war — neben einer weitgehenden Abstraktion von Datenanalyse-Techniken zu Bausteinen der Prozeßüberwachung — die Einführung einer vermittelnden Ebene. Sie enthält Bausteine, die auf der einen Seite den numerischen Input modellgestützter Verfahren akzeptieren und auf der anderen Seite den logischen und kategorialen Input regelbasierter Expertensysteme. Hierzu gehören z.B. statistische Tests oder Klassifikatoren. Diese Strukturierung konnte ich auf einem Poster zur DECHEMA-Jahrestagung der Biotechnologen im Juli 1989 erstmals skizzieren.

Das Hybrid-Konzept wirft nun eine weitere Schwierigkeit auf, die für einen praktischen Einsatz nicht zu unterschätzen ist: Wie kann man ein großes und heterogenes Methodenspektrum, das insbesondere komplizierte regelungstechnische Bausteine enthält, so aufarbeiten, daß es bei einem Praktiker im Labor Akzeptanz findet? Dies ist eine Frage der Mensch-Maschine-Kommunikation, aber auch der didaktischen Aufbereitung der eingesetzten Verfahren. Den Ausschlag für die endgültige Formulierung des Themas gab schließlich das Seminar "Nichtparametrische Regression und Interaktive Datenanalyse" unter der Leitung von Prof.Dr. Peter Huber vom MIT und Dr. Werner Stuetzle von der Universität Washington im November 1988 in Dortmund. Die Computerdemonstrationen zur interaktiven und Explorativen Datenanalyse gaben einen Eindruck von der Leistungsfähigkeit extensiv eingesetzter Visualisierungstechniken. Nach dem Motto "Ein Bild sagt mehr als tausend Worte." lassen sich auch komplizierte Sachverhalte in kurzer Zeit vermitteln. Dies trifft auch auf die Erfassung des Informationsgehaltes großer Datensätze bei geeigneter visueller Präsentation zu, worauf die Explorative Datenanalyse entscheidend aufbaut. Die Einsicht, daß Datenanalyse und Bioprozeßüberwachung sehr eng miteinander verbunden sind, führte zur Schwerpunktsetzung auf Datenanalyse und Interaktivität. Die Konzeption ist jedoch dergestalt, daß zugleich mit der Analyse biotechnischer Prozeßdaten Komponenten für die Prozeßüberwachung- und kontrolle entstehen. Das Kapitel 2 verdeutlicht dies an einem Beispiel. Es entspricht im wesentlichen einem Vortrag, den ich auf der Tagung "Computereinsatz in der Biotechnologie" im Oktober 1989 in Braunschweig gehalten habe.

Die vorgestellte vereinheitlichende Konzeption bleibt so lange eine bloße Behauptung, wie nicht der konkrete Beweis für die Realisierbarkeit erbracht ist. Dabei kann es sich nicht um die Erstellung eines fertigen Systems handeln; dies wäre ein Lebenswerk! Vielmehr muß an einem Prototyp, der alle wesentlichen Aspekte beispielhaft vorführt, gezeigt werden, daß der Ansatz tragfähig ist. Solche Prototypen liegen nun in Form eines frei konfigurierbaren Kalman-Filters als Beispiel eines komplexen Datenanalysebausteins und diverser ausgearbeiteter Beispiele zur Integration verschiedener Methodiken vor.

Die Auffassung vieler Laien, daß zur Programmierung großer Systeme lediglich das Erlernen einer Programmiersprache erforderlich sei, ist grob falsch. Entscheidend ist die Programmiertechnik. Dies hat zur Entstehung einer eigenen Disziplin in der Informatik, dem Software-Engineering, geführt. Hier fand ich in Wolfgang Martin über seine Tätigkeit bei der Deutschen Software AG einen kompetenten Betreuer. Ich verdanke ihm einige entscheidende Anregungen. Die Beschäftigung mit modernen Programmiertechniken, vor allem dem objekt-orientierten und dem deklarativen Paradigma, schlägt sich im Programmierteil der Arbeit nieder. Ich habe mich bemüht, den Nutzen dieser Entwicklungen für die Bioprozeßüberwachung, wo sie bisher kaum eine große Rolle gespielt haben, herauszuarbeiten. Er liegt vor allem in der Zeitersparnis bei der Erstellung großer Systeme. Dieser ökonomische Gesichtspunkt ist ein für die moderne Forschung nicht mehr zu vernachlässigender Faktor.

Die Programmiertechnik ist die eigentliche Methodik im praktischen Teil der Arbeit, genau wie der Versuchsaufbau zur Methodik des Experimentators gehört. Den Software-Grundlagen soll daher ein ihrer Bedeutung entsprechender Raum zugewiesen werden. Interessanterweise geht kaum eine Publikation im Bereich der Bioprozeßüberwachung auf Details im Zusammenhang mit der Programmiertechnik ein. Viele Detail-Lösungen werden nur als "Folklore" der Programmierung mündlich weitergegeben. Ein Neuanfänger muß jeden Fehler selber machen, was den Einstieg erheblich erschwert. Ein extremes Beispiel stellt die Entwicklung von Formelcompilern dar: Allein an einem einzigen Institut sind mir vier verschiedene Eigenentwicklungen bekannt. Bei keiner davon wurde die Compilerbau-Literatur herangezogen! Ich habe versucht, die verwendeten Grundlagen aus der Informatik in gegebener Kürze darzustellen, in der Hoffnung, daß damit zumindest die eine oder andere "Neuerfindung des Rades" verhindert wird.

Ich freue mich, daß ich bei einem Vortrags-Aufenthalt an der Universität Halle Herrn Prof.Dr. Goldschmidt und seiner Arbeitsgruppe "in die Karten sehen" durfte. Der Austausch hat auf beide Seiten befruchtend gewirkt, so daß eine Integration des beschriebenen frei konfigurierbaren Kalman-Filters in die "Modellbank Biotechnologie" nun in Planung ist. Weiterhin danke ich Herrn Eberhardt vom Forschungszentrum Jülich und seinem Team für viele Anregungen. Ich hoffe, daß sich unsere Entwicklungen in den nächsten Jahren eng miteinander verzahnen lassen.

Fast alle wichtigen wissenschaftlichen Entwicklungen finden an mehreren Stellen zugleich statt, wie man an unzähligen Beispielen illustrieren kann. Mit dem Fertigwerden der Dissertation erscheinen zugleich Entwürfe anderer Arbeitsgruppen. Nennen möchte ich hier [CFB90] [HMN+90]. Dies zeigt, daß die biotechnische Datenanalyse ein äußerst aktuelles Gebiet ist, auf dem sich eine Weiterarbeit lohnt. Ich

freue mich, daß ich in den nächsten Jahren am Institut für Biotechnologie 2 des Forschungszentrums Jülich die Gelegenheit erhalten werde, meine Konzepte auf einer leistungsfähigen graphischen Workstation einer Realisierung näher zu führen. In diesem Kontext ist die vorliegende Dissertation nicht nur der Abschlußbericht einer wissenschaftlichen Arbeit, sondern zugleich auch Projektskizze für das weitere Forschungsvorhaben.

Es soll nicht unerwähnt bleiben, daß das nun abgeschlossene Unternehmen ohne die großzügige finanzielle und materielle Unterstützung durch das BMFT-Stipendium "Angewandte Mikrobiologie und Biotechnologie" sowie durch Herrn Prof.Dr. Wandrey aus Jülich und Herrn Prof.Dr. Alt aus Bonn nicht möglich gewesen wäre. Aus den bereitgestellten Mitteln konnte ein 386er PC mit guter graphischer Ausstattung beschafft werden, der in den letzten beiden Jahren fast jeden Tag voll ausgelastet war. Ein einfaches Rechenexempel soll illustrieren, warum die Anschaffung einer so leistungsfähigen Maschine sinnvoll war: Ein Übersetzungslauf für ein großes Programm (10000 Zeilen und mehr) mit dem Turbo-C-Compiler dauert auf diesem Gerät ca. 2 Minuten. In den letzten beiden Jahren summiert sich dies bei ca. 10 Compiler-Läufen pro Stunde auf:

×	2	(Jahre)				
×	50	(Wochen)			96000	Minuten
×	6	(Tage)	entsprechend	=	1600	Stunden
×	8	(Stunden)		=	200	Tage
×	10	(Läufe)				
×	2	(Minuten)				

Eine langsamere Maschine (z.B. ein 286er PC) benötigt hierfür um einen Faktor 2-3 länger! Ein weiterer Kommentar erübrigt sich.

Schließlich muß ich den stillen Helfern danken, die mir bei der Fertigstellung der Arbeit beigestanden haben. Frau Gabriela Weilkes und meine Eltern haben mit spitzer Feder den Text durchgesehen; Wolfgang Martin hat die inhaltlichen Korrekturen hinzugefügt. Die leidige Arbeit der Literaturaufnahme haben mir Frau Weilkes, Frau Roswitha Emmerich und Herr Guido Scholz abgenommen. Diejenigen, die mir Datenmaterial und Software-Teile überlassen haben, möchte ich hier nicht in einer Reihe auflisten. Sie sind im Text an den entsprechenden Stellen namentlich erwähnt.

0.2 Zur Form

0.2.1 Homogenität

Der vorliegende Text stellt eine Gesamtkonzeption zur interaktiven Datenanalyse bei biotechnischen Prozeßdaten vor. Voraussetzung für einen derartigen Entwurf ist die Integration eines großen Methodenspektrums in einen einheitlichen Rahmen. Als adäquate Darstellungsform erscheint mir ein Lehrbuch-Stil, bei dem Theorie und Methoden systematisch und vollständig abgehandelt werden. Um die Homogenität des Textes nicht zu beeinträchtigen, habe ich darauf verzichtet, eigene Resultate und bestehende Ergebnisse durch Zitation oder Verweis auf einen Anhang voneinander zu trennen. Ich hoffe, daß sich auf diese Weise die Einheitlichkeit der Konzeption in den Ausführungen widerspiegelt.

Ein Beispiel hierzu zeigt deutlich, welche Zersplitterung der Text erfahren hätte, wenn einige Grundlagen durch Verweis auf andere Literatur bereitgestellt worden wären: Die Abbildung 0.1 zeigt einen Abhängigkeitsbaum für die Grundlagen des Erweiterten Kalman-Filters. Von unmittelbarer Wichtigkeit für die interaktive Datenanalyse ist hiervon nur der der obere Teil (Grundstruktur und Modelle) sowie der untere Teil (Simulation, Visualisierung). Alle im mittleren Teil des Baums aufgeführten Grundlagen können in entsprechenden Lehrbüchern nachgeschlagen werden. Die Auslassung dieser Grundlagen würde aber in einem zusammenhanglosen Text resultieren, der von einem Leser mit breiter mathematischer und regelungstechnischer Vorbildung ausgeht!

0.2.2 Neue Zusammenhänge

Es hat sich erwiesen, daß der Text durch die kurzgefaßte Integration der Grundlagen nur um etwa ein Drittel länger geworden ist. Dies ist vor allem der Tatsache zuzuschreiben, daß die stochastischen und

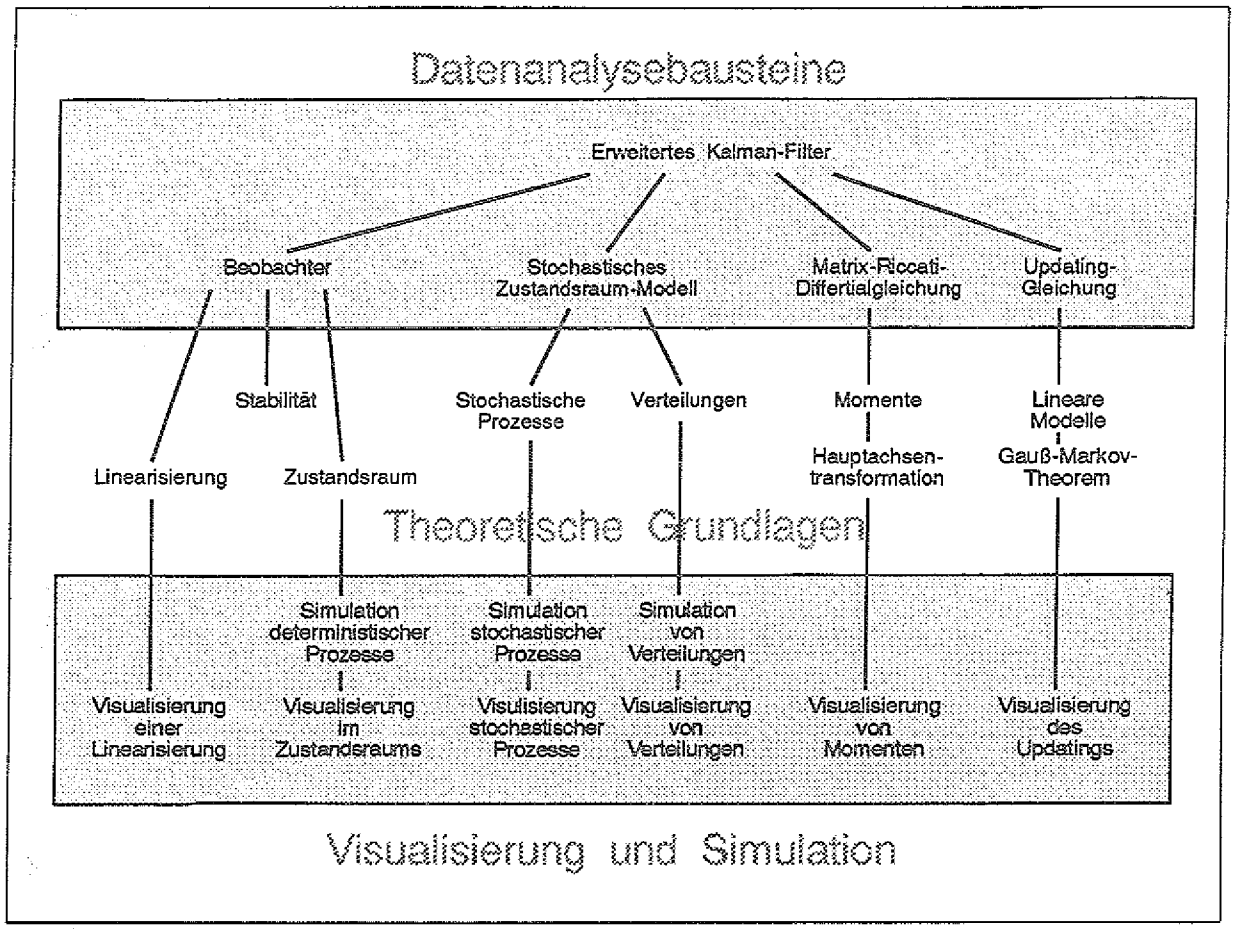


Abbildung 0.1: Abhängigkeitsbaum für die theoretischen Grundlagen des Erweiterten Kalman-Filters

statistischen Modelle auf dem Wege der Simulation eingeführt werden. Die Simulationstechniken selbst stellen Neuland für die Biotechnologie dar und müssen insofern ohnehin ausführlich erläutert werden.

Viele bekannte Methoden erscheinen im Rahmen des Datenanalysekonzepts in einem neuen Zusammenhang. So ist z.B. die Herleitung der Wachstumsmodelle mit Hilfe verallgemeinerter Reaktionsgleichungen und die erst im Anschluß erfolgende Einführung des stöchiometrischen Rechnens mit Hilfe von Erhaltungsgrößen im wesentlichen neu, obwohl die entstehenden Modellstrukturen bekannt sind. Unmittelbar verknüpft mit dieser unüblichen Zerlegung des Modells sind die Datenanalysetechniken der stationären und instationären Identifikation sowie der Hauptkomponentenanalyse. Analoges gilt für die Herleitung der Kalman-Filter-Gleichungen im Kontext der Beobachtertheorie, der Theorie stochastischer Prozesse und der statistischen Schätztheorie und die damit verbundene Zerlegung und mögliche Robustifizierung des Filters. Diese oft ungewohnte Reihenfolge der Grundlagen würde ein Vorgehen mit Literaturverweisen weiter erschweren.

0.2.3 Interdisziplinarität

Die Allgemeinheit der Fragestellung führt zwangsläufig zu einem größeren Umfang, da Methoden aus den Bereichen Modellierung, Regelungstechnik, Datenanalyse, Informatik, Mikrobiologie und Bioenergetik berücksichtigt werden müssen. Der Spezialist auf jedem dieser Gebiete wird eine Darstellung der Grundlagen für überflüssig halten, wogegen ihm andere Aspekte vollkommen fremd erscheinen werden. Ich möchte keinen der genannten Bereiche als vorrangig betrachten, denn ein interdisziplinäres Gebiet wie die Biotechnologie hat stets auch ein Verständigungsproblem: Chemiker, Mikrobiologen, Ingenieure, Mathematiker und Informatiker sprechen verschiedene Sprachen. Der Theoretiker hat in dieser Hinsicht die

größten Schwierigkeiten und seine Akzeptanz hängt entscheidend davon ab, wie gut er sein Arbeitsgebiet erläutern und motivieren kann. Ich habe mich zumindest in den einleitenden Kapiteln um Verständlichkeit auch für den Nicht-Theoretiker bemüht. Der weitgehende Verzicht auf Formalismen und die vielen Illustrationen erfordern allerdings auch einiges an Platz.

0.2.4 Didaktik

Das Datenanalysesystem zielt auf den regelungstechnisch interessierten Praktiker in der biotechnischen Verfahrensentwicklung. Er soll in die Lage versetzt werden, auch aufwendige Datenanalysetechniken selbst zu konfigurieren. Eine didaktische Aufbereitung der Methoden bildet daher einen notwendigen Bestandteil der Dissertation. Die Didaktik muß bei der Theorie anfangen und bis in die konkrete Realisierung hinein sichtbar bleiben. Der vorliegende Text hat daher nicht zuletzt auch die Funktion einer Einführung in die Methodik. Er stellt das "Tutorial" des Datenanalysesystems (im Gegensatz zum "Reference Manual") dar. Viele Beispiele und Illustrationen sollen dazu beitragen.

0.2.5 Visualisierung

Von zentraler Bedeutung sind die verschiedenen Visualisierungstechniken für die Datenanalyse. Wann immer dies möglich war, wurde eine graphische Darstellung numerischer Daten angestrebt. Die Wichtigkeit der Visualisierung muß auch in den Illustrationen der Arbeit ihre Entsprechung finden. Ich hoffe, daß mit Hilfe der Abbildungen viele Sachverhalte auch ohne das detaillierte Verständnis der Theorie zu erfassen sind.

0.2.6 Stoffaufbereitung

Schließlich sollte eine Doktorarbeit im Rahmen einer Arbeitsgruppe meines Erachtens nicht nur die eigene Leistung herausstellen, sondern auch die Funktion haben, einen Stoff für nachfolgende Doktoranden und Diplomanden aufzubereiten. Andernfalls geht ein großer Teil der gesammelten Erfahrungen für die Arbeitsgruppe verloren. Der nun fertiggestellte Text soll in den nächsten Jahren daher auch dazu verwendet werden, neuen Mitarbeitern den Einstieg zu erleichtern.

0.3 Zum Aufbau

Der Text richtet sich an ein breites Leserspektrum und ist sicherlich nicht dazu geeignet, am Stück gelesen zu werden. Vielmehr möchte ich abhängig vom Interessenprofil des Lesers einen Leseweg vorschlagen, der die jeweils interessanten Abschnitte miteinanander verbindet. Da die einzelnen Kapitel nicht ganz unabhängig voneinander sind, müssen dabei gelegentliche Lücken in der Argumentation in Kauf genommen werden.

0.3.1 Anwendungsorientierter Praktiker

Einige Kapitel, welche mögliche Anwendungen des Konzepts an Beispielen erläutern, sind weitgehend allgemeinverständlich gehalten. Dies sind vor allem die Kapitel 1, 2, 3, 4 sowie 21, 22. Sie geben einen eher informellen Überblick über die Möglichkeiten und den prinzipiellen Aufbau des Datenanalysesystems.

0.3.2 Modellierer und Datenanalytiker

Die Kapitel 5, 6, 7, 8 sind eigens der Modellierung biotechnischer Prozesse mit deterministischen, statistischen, stochastischen, logischen und heuristischen Modellen gewidmet. Auf die jeweils zugehörigen Simulationsmethoden und Datenanalysetechniken wird dann in Kapitel 9, 10, 11, 12 13, eingegangen. Hier findet der Theoretiker wahrscheinlich viele bekannte Ansätze, die überlesen werden können.

0.3.3 Programmierer

Die Programmiertechniken haben eine Sonderstellung im Rahmen der Arbeit, da sie nur für jemanden, der direkt mit der Programmierung befaßt ist, von Interesse sein dürften. Sie werden in den Kapiteln 14, 15, 16, 17, 18, 19, 20 abgehandelt. Für das Verständnis des Übrigen haben sie keine Bedeutung.

0.3.4 Eigenleistung

Zu einer Dissertation gehört die Kenntlichmachung der eigenen Leistung und die Abgrenzung von bereits bestehenden Resultaten. Aus den aufgeführten Gründen der Homegenitätswahrung ist dies bei der vorliegenden Arbeit nicht innerhalb des Textes geschehen. Als wesentliche Leistung möchte ich eben diese Integration eines weiten Methodenspektrums in ein einheitliches Gesamtkonzept ansehen. Welche Elemente im einzelnen neu, teilweise neu oder in einem neuen Zusammenhang erscheinen, entnehme man der Zusammenfassung.

Teil I

Problemstellung der Dissertation

Der erste Teil führt in die Problematik der biotechnischen Datenanalyse ein. Sein Ziel ist die Vermittlung eines Vorverständnisses der Konzeption eines interaktiven Datenanalysesystems. Die angewendeten Methoden werden deshalb zunächst auf einem informellen Niveau erläutert. Erst in den anschließenden Teilen erfolgt die systematische Zusammenstellung der theoretischen Voraussetzungen und die Präzisierung der Methodik.

Kapitel 1

Die experimentelle Situation in der Biotechnologie

Der Biotechnologe steht bei der Entwicklung eines Verfahrens im Labor- oder Technikumsmaßstab hinsichtlich der Meßdatenerfassung und -Auswertung sowie der Prozeßüberwachung und -steuerung vor folgenden Aufgaben:

- Modellierung des biologischen Systems unter Produktionsbedingungen.
- Ableitung biologisch relevanter Informationen aus den gemessenen Daten.
- Glättung und Filterung verrauschter Meßsignale.
- Regelung und Steuerung wichtiger Prozeßgrößen.
- Detektion von Störungen, Phasenübergängen und kritischen Zuständen.
- Automatisierung von Versuchsreihen.
- Steuerung des Gesamtablaufs der Produktion.

Diese Probleme sind anhand der vorliegenden Prozeßdaten zu bearbeiten. Dabei ist zusätzlich zu berücksichtigen, daß nur ein Teil der Informationen, die im Labor zur Verfügung stehen, auch unter Produktionsbedingungen vorliegen. Die Untersuchung verschiedener Fragestellungen führt zu Detaillösungen, die jeweils einen Aspekt der Prozeßüberwachung abdecken. Die eigentliche Problematik bei der Prozeßsteuerung im Produktionsmaßstab besteht nun darin, diese Teillösungen zu einem robusten und leistungsfähigen Überwachungssystem zusammenzufügen [1].

Die skizzierte Problematik erfordert ein flexibles Konzept der Datenanalyse und Prozeßüberwachung. Der erarbeitete Ansatz soll nun zunächst informell beschrieben werden.

1.1 Der experimentelle Zyklus

Die Abbildung 1.1a gibt den idealen Zyklus der experimentellen Informationsgewinnung (nicht nur in der Biotechnologie) wieder: Der Experimentator konzipiert ein Experiment, erhebt Meßdaten und gelangt über die Analyse der Daten zu konkreteren Vorstellungen und Vermutungen über das untersuchte System. Dies ermöglicht die gezielte Konzeption nachfolgender Versuche [2].

Der experimentelle Zyklus teilt sich (durchaus auch räumlich) in eine praktische Phase im Labor und eine theoretische Phase am Schreibtisch auf. Im Idealfall sollten beide Phasen gleiches Gewicht erhalten:

[1] Dieses Problem stellt sich im übrigen schon im Labor, wenn bestehende Erkenntnisse bei der weiteren Verfahrensentwicklung zum Tragen kommen sollen. Es ist denkbar, ganze experimentelle Abläufe durch Wiederverwendung und Neukombination von bestehender Problemlösungen zu automatisieren.

[2] Es soll zwischen *Versuchskonzeption* im Sinne einer zielgerichteten Informationsgewinnung und einer *Versuchsplanung* im strengen Sinne der Statistik [BR86] unterschieden werden. Eine Versuchskonzeption kann dabei durchaus auf einer statistischen Versuchsplanung beruhen.

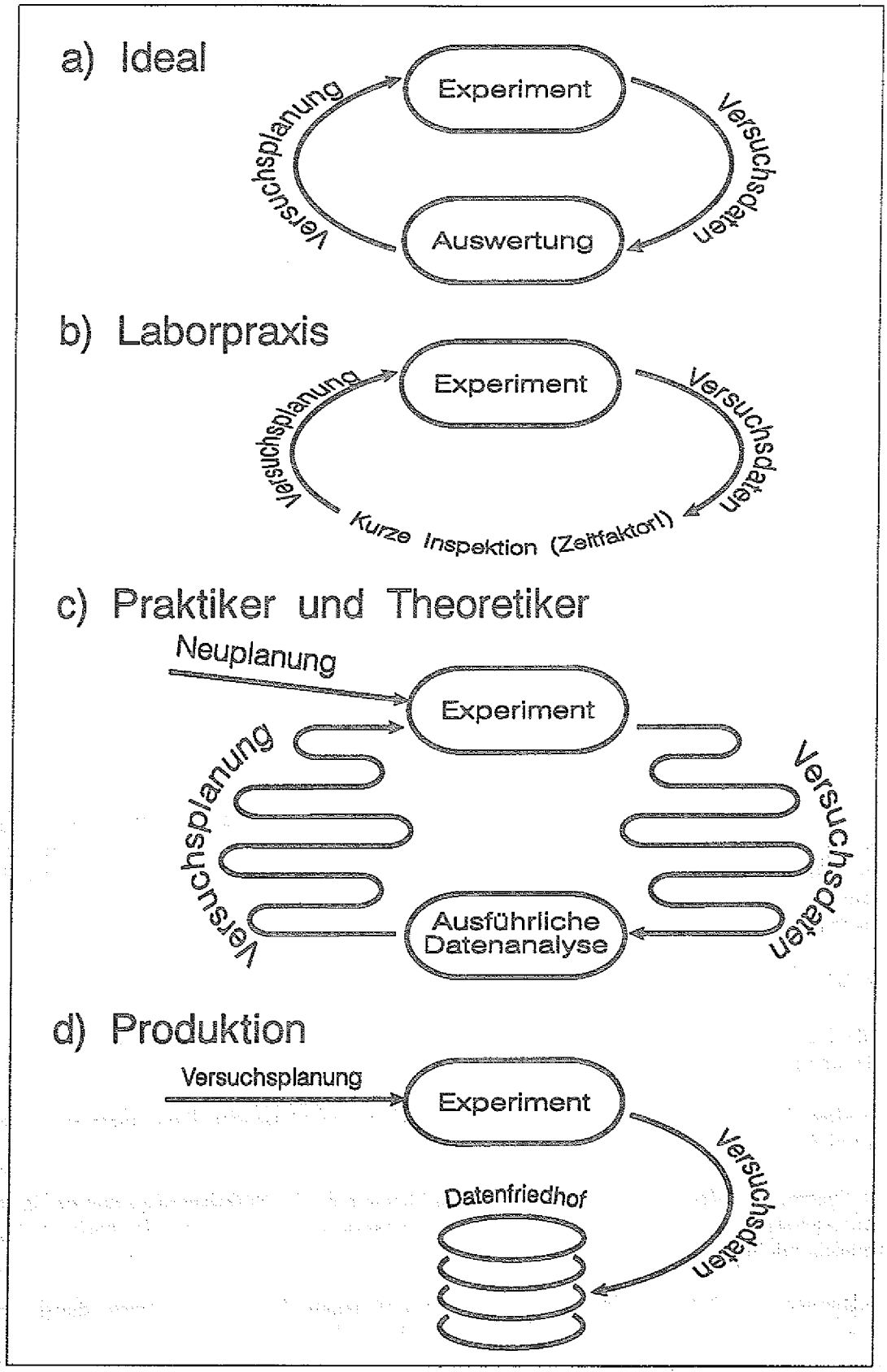


Abbildung 1.1: Varianten des experimentellen Zyklus in der Biotechnologie

Einerseits liefert eine schlechte experimentelle Basis nur geringe Systeminformation,

andererseits führt eine oberflächliche Analyse der Daten dazu, daß wichtige Details übersehen und ihr Informationsgehalt nicht voll ausgeschöpft wird.

In der Realität hat der experimentelle Zyklus oft eine stark verzerrte Gestalt. Es gibt einige typische Varianten:

1.1.1 Laborpraxis

Die Abbildung 1.1b schematisiert den Zyklus, wie er oft in der Praxis abläuft. Die Analyse der Daten kommt hier zu kurz. Sie beschränkt sich i.a. auf das In Augenschein Nehmen der Meßschreiberblätter. Eine quantitative Untersuchung ist häufig nicht mit vertretbarem Aufwand durchführbar, weil z.B.

- dem Experimentator die theoretischen Kenntnisse über komplexere Analysetechniken fehlen,
- erforderliche Software nicht erhältlich ist,
- der Versuchsablauf Unregelmäßigkeiten aufweist, die eine geschlossene Untersuchung des Datensatzes verhindern,
- die Verknüpfung von on-line gemessenen Daten, stark verrauschten Daten und Labordaten mit den gegebenen Mitteln nicht möglich ist.

In dieser Situation beschränkt man sich auf eine qualitative Begutachtung der Meßwerte in Hinblick auf die dem Experiment zugrunde liegende Fragestellung und schließt den nächsten Versuch alsbald an.

1.1.2 Kooperation von Theoretiker und Praktiker

Die Probleme des Praktikers legen es nahe, zu einer Zusammenarbeit zwischen einem Praktiker im Labor und einem Theoretiker, der einen Rechner bedient, überzugehen (Abb. 1.1c). Zwar stößt der theoretisch ausgebildete Mitarbeiter bei der Analyse der Daten auf ähnliche technische Probleme wie der Praktiker, jedoch kann er die bestehende Konfektionsoftware an die aktuellen Probleme anpassen. Die Schwierigkeiten treten jetzt an anderer Stelle auf:

- Der Zeitaufwand für die Datenanalyse kann sehr hoch sein.
- In dem Bestreben, ein Maximum an Information aus einem einzigen Datensatz zu gewinnen, verselbständigt sich der Theoretiker.
- Die Möglichkeit, schwach gestützte Hypothesen in einem nachfolgenden Experiment zu erhärten, wird übersehen.
- Der Theoretiker neigt zur Überinterpretation von Effekten, die der Praktiker aus seiner biologischen Systemkenntnis und aus den gemachten, aber nicht quantitativ protokollierten Beobachtungen heraus leicht erklären könnte.
- In Abwesenheit des Experimentators gelangt der Datenanalytiker zu Hypothesen, die sich zwar mathematisch begründen lassen, aber biologisch nicht haltbar sind.

Diese Effekte führen häufig dazu, daß man im Labor schon längst zu anderen Erkenntnissen gekommen ist, wenn die Auswertung eines länger zurückliegenden Experiments zum Abschluß kommt.

1.1.3 Industriepraxis in der Produktion

Ein einmal entwickeltes Verfahren wird im Produktionsmaßstab meist über Jahre hinweg nicht mehr verändert. Trotzdem wird oft eine Fülle von Daten aufgezeichnet, die aus betriebsstatistischen Gründen archiviert werden (Abb. 1.1d). Die Datenauswertung besteht — wenn sie überhaupt erfolgt — in einer Reduktion der Datensätze zur Speicherplatzersparnis [Car89]. Eine wirkliche Datenanalyse entfällt aus Personalgründen oder deshalb, weil eine Veränderung bestehender Betriebsparameter zu risikoreich erscheint.

Selbst dann, wenn man das letzte Argument gelten läßt, sprechen doch gewichtige Gründe für eine regelmäßige Datenauswertung:

- Verfolgung langfristiger Veränderungen der Parameter im biologischen oder technischen System (Qualitätssicherung).
- Diagnose und Früherkennung von Störungszuständen.
- Informationsgewinnung für die Projektierung zukünftiger Anlagen.

1.2 Die besondere Problematik bei biotechnischen Prozeßdaten

Das bisher Gesagte gilt in mehr oder weniger starker Ausprägung für jede experimentelle Situation. Es gilt nun herauszustellen, was die spezifisch biotechnischen Schwierigkeiten bei der Bioprozeßüberwachung sind. Dies wird zunächst an einem Beispiel illustriert:

1.2.1 Ein Beispiel für eine aufwendige Instrumentierung

Die Fermentation der methylotrophen Hefe *Candida boidinii* stellt hohe Anforderungen an die Instrumentierung. Ein Grobschema ihres Methanolstoffwechsels gibt Abbildung 1.2 wieder. Das intrazelluläre Enzym Formiat-Dehydrogenase ist für die NADH-Rezyklierung in Enzym-Membran-Reaktoren von großem Interesse, da es im Reaktionsgefäß die Energie für weitere NADH-gebundene Enzymreaktionen bereitstellen kann [Wan89]. Ziel einer im Institut für Biotechnologie 2 des Forschungszentrums Jülich begonnenen Verfahrensentwicklung ist deshalb die kontinuierliche Fermentation von *Candida boidinii* auf Methanol als C-Quelle mit anschließender Aufbereitung des intrazellulären Enzyms [WSW89].

Berücksichtigt man die im Labormaßstab zur Verfügung stehende Meßtechnik [Büh85] [WSE84] und die zu untersuchende Zielgröße, so kommen zur Informationsgewinnung die in Abbildung 1.3 versinnbildlichten Meßgeräte [3] in Frage.

Zunächst fällt die enorme Menge und die Heterogenität der Daten auf. Es ist nicht damit zu rechnen, daß diese Meßgrößen alle gleichzeitig gemessen werden. Potentiell kann jedoch jede Meßmethode im Verlauf des Experimentierens zum Einsatz kommen, um auf ihren Informationsgehalt hin geprüft zu werden. Ein Datenanalysesystem muß somit prinzipiell die Flexibilität zur Verarbeitung der ganzen Datenvielfalt haben. Um diese Flexibilität zu erreichen, ist es erforderlich herauszustellen, was die Meßwerte in Hinblick auf die Datenanalyse voneinander unterscheidet. Es sollen dazu einige Klassifikationsmerkmale bereitgestellt werden.

1.2.2 Klassifikation von Bioprozeßdaten in Hinblick auf die Datenanalyse

1.2.2.1 Abtastung

Unter dem Aspekt der Datenspeicherung im Rechner kann unterschieden werden zwischen (vgl. Abb. 1.4a und die Beispiele in Tabelle 1.1):

- quasikontinuierlicher Abtastung, die in — im Vergleich zu den Zeitkonstanten des biologischen Systems — kleinen Intervallen erfolgt (i.a. entspricht dies der Grundabtastrate der Meßdatenerfassung),

[3] High Performance Liquid Chromatography - Flüssigkeitschromatographie-Verfahren

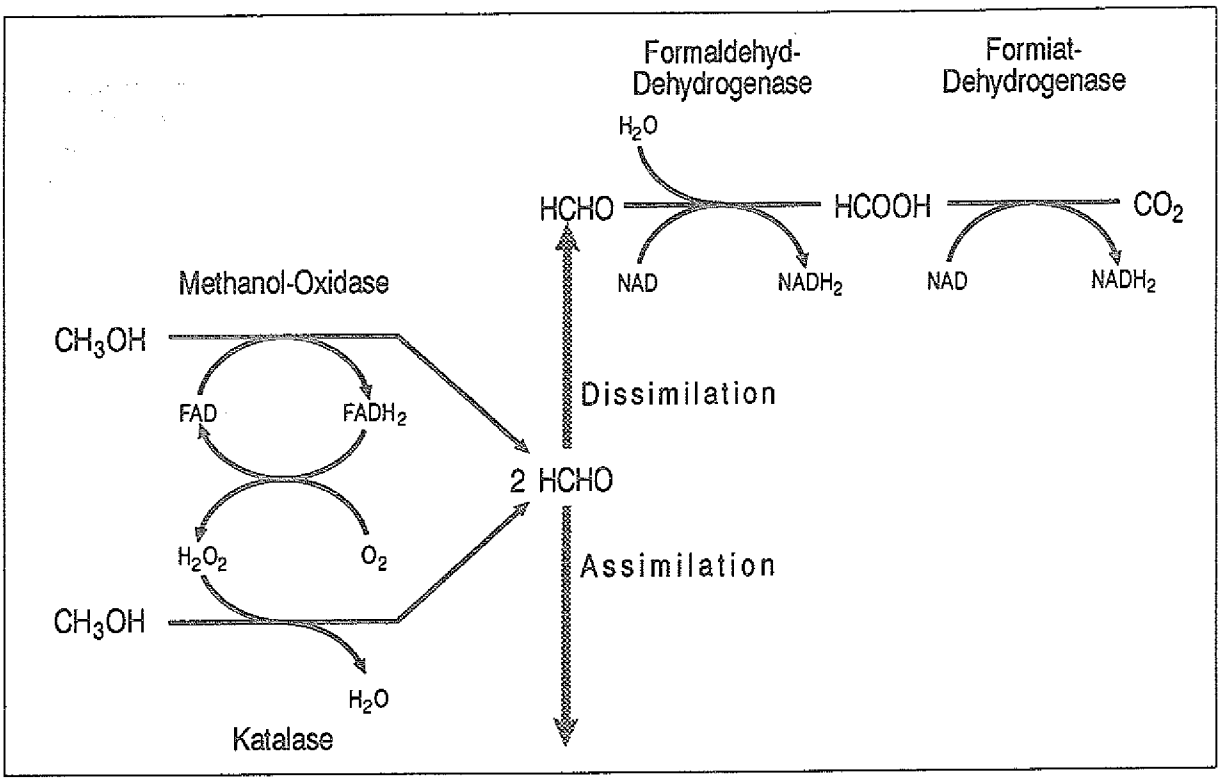


Abbildung 1.2: Grobschema des Methanolstoffwechsels von *Candida boidinii* (nach [WNSW90])

Art der Abtastung	Meßgröße	Abtastrate (Größenordnung)
quasikontinuielrich	Gasanalyse	1 sec
	Optische Dichte	1 sec
äquidistant	HPLC -Analysen	5 min
unregelmäßig	Laboranalysen	1 - 10 h
	Beobachtungen	10 -100 min

Tabelle 1.1: Beispiele für verschiedene Arten der Abtastung

- äquidistanter Abtastung, die in gleichen, jedoch so großen Zeitabständen erfolgt, daß nicht mehr problemlos zwischen benachbarten Punkten interpoliert werden kann,
- unregelmäßiger Abtastung.

1.2.2.2 Zeitverzögerung

Viele Messungen haben eine Verzögerungszeit zwischen Probennahme und Auswertung der Messung. Zu unterscheiden ist zwischen

- unmittelbar nach der Messung eintreffenden und
- zeitverzögert eintreffenden

Meßdaten (vgl. Abb. 1.4b und Beispiele in Tabelle 1.2). Eine Zeitverzögerung hat bei der Off-Line-Datenanalyse keine Bedeutung. Sie muß jedoch bei der Konfiguration von Datenanalysekomponenten für den On-Line-Einsatz berücksichtigt werden.

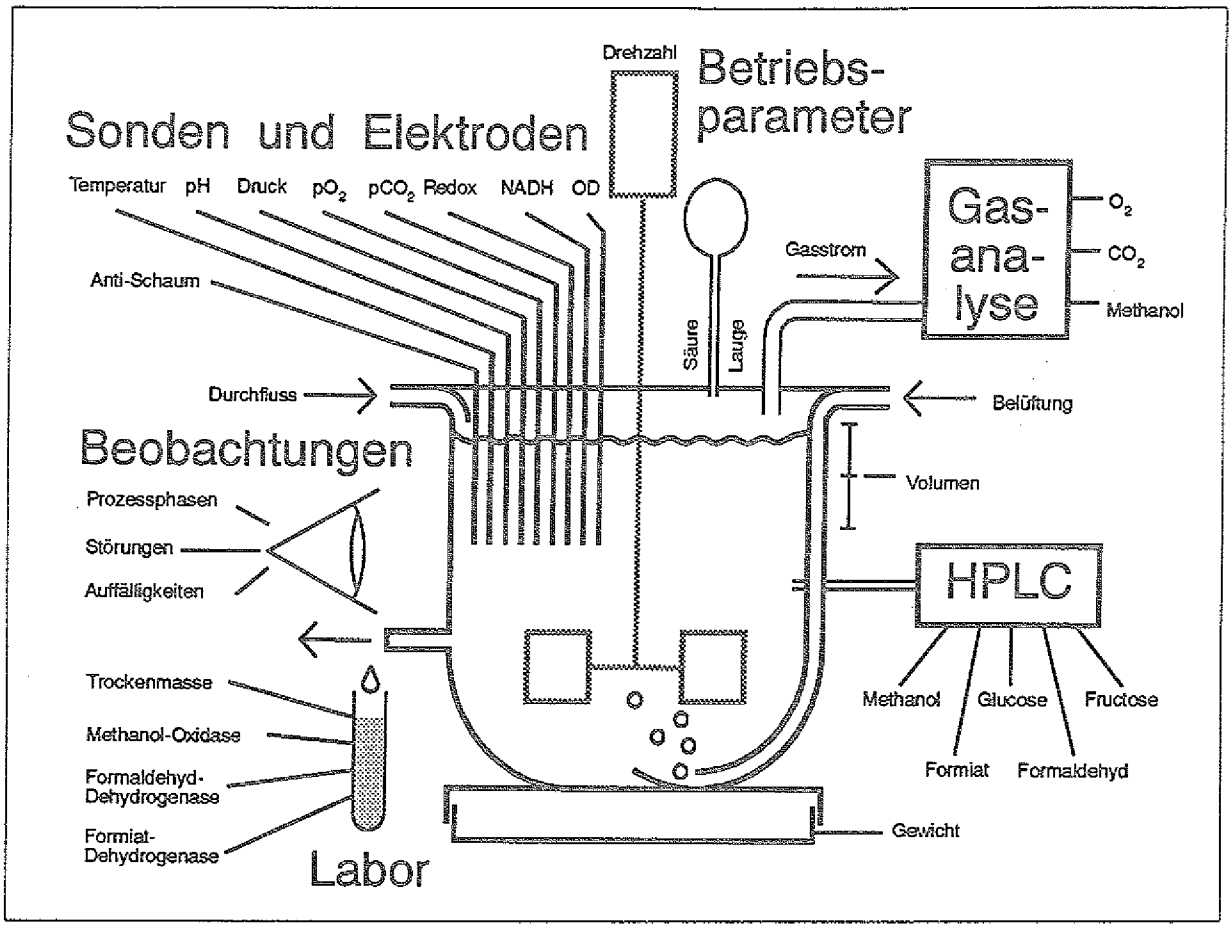


Abbildung 1.3: Meßgrößen bei einem aufwendig instrumentierten Fermenter

Art der Zeitverzögerung	Meßgröße	Verzögerungszeit (Größenordnung)
unmittelbar	Optische Dichte	0 sec
	Gasanalyse	1 sec
zeitverzögert	HPLC-Analysen	5-10 min

Tabelle 1.2: Beispiele für verschiedene Zeitverzögerungen

1.2.2.3 Genauigkeit

Die einzelnen Meßwerte unterliegen verschieden starken zufällige Schwankungen, so daß unterschieden werden muß zwischen

- quasi-exakten Meßwerten,
- konstant fehlerbehafteten Meßwerten,
- meßbereichsabhängig fehlerbehafteten Meßwerten.

(vgl. Abb. 1.4c und Tabelle 1.3). Der Fehler kann mit Hilfe eines Fehlermaßes quantifiziert werden und auf diese Weise mit in die Datenanalyse eingehen. Weiteres Zahlenmaterial ist in [Fly83] publiziert.

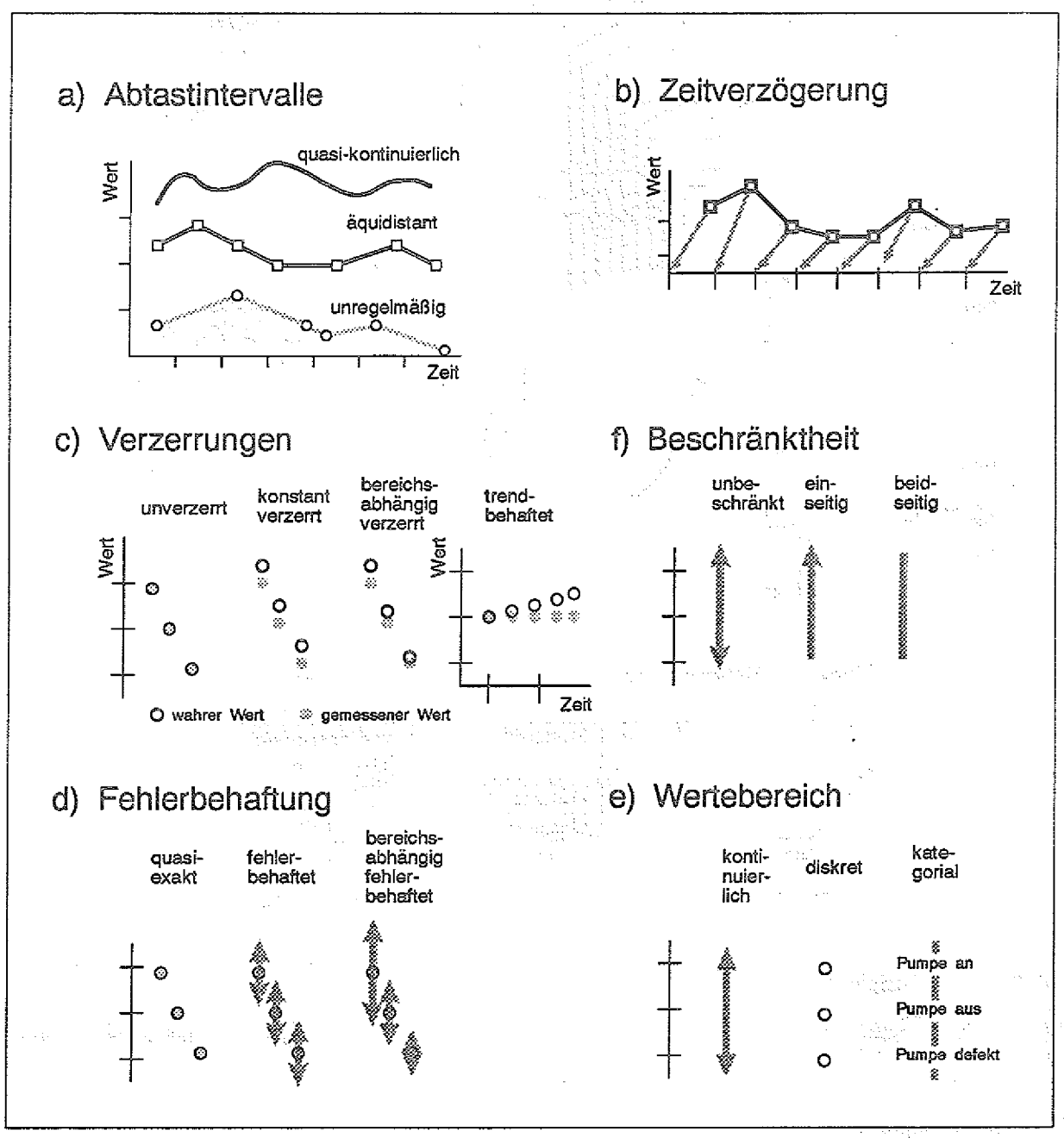


Abbildung 1.4: Charakterisierung biotechnischer Meßdaten

Art der Fehlerbehaftung	Meßgröße	Größenordnung des Fehlers
quasi-exakt	Gewicht	0.1 %
konstant fehlerbehaftet	Volumen (bei hohen Füllständen)	1 %
meßbereichsabhängig fehlerbehaftet	Gasanalyse pO_2-Messung	10 % 20 %

Tabelle 1.3: Beispiele für verschiedene Fehlerbehaftungen

Art der Verzerrung	Meßgröße	Genauere Spezifikation
unverzerrt	Gewicht	Waagen bedürfen nahezu keiner Kalibrierung.
konstant verzerrt	Volumen	Bei Füllstandsmessung mit nicht berücksichtigtem Totvolumen.
meßbereichsabhängig verzerrt	HPLC-Analyse	Chemische Analysen haben bedingt durch eine Verdünnung oft einen relativen Fehler.
trendbehaftet	pH-Messung	Drift von pH-Elektroden

Tabelle 1.4: Beispiele für verschiedene Verzerrungen

Wertebereich	Meßgröße
kontinuierlich	Gewicht
diskret	Pumpe (an/aus) Mehrwegeventil (Stellung 1/2/3)
kategorial	Beobachtungen Störungsmeldungen

Tabelle 1.5: Beispiele für verschiedene Wertebereiche

1.2.2.4 Verzerrungen

Eine Verzerrung ist ein systematischer Fehler in einer Messung, der den Meßwert um einen konstanten oder zeitabhängigen Wert verfälscht. Man unterscheidet (vgl. Abb. 1.4d und Beispiel-Tabelle 1.4):

- unverzerrte Meßwerte,
- konstant verzerrte Meßwerte,
- meßbereichsabhängig verzerrte Meßwerte,
- zeitabhängig verzerrte Meßwerte (Trend, Drift).

Informationen über auftretende Verzerrungen können in Form von Korrekturfunktionen berücksichtigt werden [Lun88].

1.2.2.5 Wertebereich

Jeder einzelne Meßwert kann Werte in einem

- kontinuierlichen,
- diskreten oder
- kategorialen

Wertebereich annehmen (vgl. Abb. 1.4e und Tabelle 1.5).

Bereich (zumindest theoretisch)	Meßgröße	Schranken
unbeschränkt	pH-Wert	$]-\infty, \infty[$
einseitig beschränkt	Gewicht	$[0, \infty[$
beidseitig beschränkt	pO_2	$[0\%, 100\%]$

Tabelle 1.6: Beispiele für verschiedene Bereichsschranken

1.2.2.6 Bereichsschranken

Die meisten Meßwerte und Systemparameter sind auf einen gegebenen Meßbereich oder auf einen physikalisch sinnvollen Bereich eingeschränkt. Man findet

- unbeschränkte,
- einseitig beschränkte und
- zweiseitig beschränkte

Wertebereiche. (vgl. Abb. 1.4f und Tabelle 1.6) Derartige Informationen können für Graphikskalierungen, Bereichs- und Plausibilitätstests herangezogen werden.

1.2.2.7 Physikalische Kategorie

Die Einteilung der Meßgrößen in eine physikalische Kategorie ist nicht nach klaren Kriterien durchführbar, da vielfach mehrere Kategorien in Frage kommen (vgl. auch [Fie84]). Trotzdem ist es sinnvoll, eine solche Einteilung für die Datenanalyse zu versuchen, da für die verschiedenen Kategorien von Daten oft auch verschiedene mathematische Modelle bzw. Komponenten von Modellen zur Anwendung kommen. Hier soll unterschieden werden zwischen

- thermodynamischen Systemparametern,
- mechanischen Betriebsparametern,
- Zusammensetzung der Gasphase,
- Zusammensetzung der Gelöst-Gasphase,
- Zusammensetzung des Mediums,
- physiko-chemische Daten des Mediums,
- biologisch-extrazellulären Daten,
- biologisch-intrazellulären Daten.

In der Abbildung 1.3 ist eine solche Einteilung zum Teil durch die Gruppierung der Meßgeräte angedeutet. Die Tabelle 1.7 zeigt diverse Beispiele.

1.2.2.8 Beziehung zu den Systemzustandsgrößen

Für die Modellierung ist weiterhin wichtig, ob eine Meßgröße

- in direkter oder
- in differentieller

Kategorie	Meßgröße
thermodynamische Systemparameter	Gewicht Druck Temperatur
mechanische Betriebsparameter	Durchfluß Rührerdrehzahl Pumpe an/aus
Zusammensetzung der Gasphase	O_2-Anteil CO_2-Anteil CH_4-Anteil
Zusammensetzung der Gelöst-Gasphase	pO_2 pCO_2
Zusammensetzung des Mediums	Stoffkonzentrationen
physiko-chemische Daten des Mediums	pH-Wert Redox-Potential
biologisch-extrazelluläre Daten	Biomasse Wachstumsrate Respirationsquotient Beobachtungen
biologisch-intrazelluläre Daten	Emzymkonzentrationen NADH-Fluoreszenz Mikroskopische Beobachtungen

Tabelle 1.7: Beispiele für verschiedene Meßwertkategorien

Beziehung	Meßgröße	assoziierte Zustandsgröße
direkt	Gewicht	Volumen
	Optische Dichte	Biomasse
differentiell	Durchfluß	Volumen
	Wachstum	Biomasse

Tabelle 1.8: Beispiele für verschiedene Beziehungen zu den Systemzustandsgrößen

Beziehung zu einer (kapazitiven) Zustandsgröße des Systems steht, d.h. ob diese Meßgröße mit der linken oder der rechten Seite einer Zustandsdifferentialgleichung

$$\dot{X} = f(X, Y, \ldots, \alpha, \beta, \ldots)$$

zusammenhängt [4]. Einige Beispiele finden sich in Tabelle 1.8.

1.2.2.9 Automatisierbarkeit

Nicht alle im Labor meßbaren Größen sind auch im Produktionsmaßstab mit vertretbarem Aufwand zu messen. Es ist daher zu unterscheiden zwischen (vgl. Tabelle 1.9)

- automatisierbaren,
- bedingt (d.h. nur mit einigem Entwicklungsaufwand) automatisierbaren und
- nicht automatisierbaren

Messungen.

[4] Die Wichtigkeit dieser Unterscheidung wird im Zusammenhang mit der Modellierung biotechnischer Systeme deutlicher werden 6.6.3.2.

Automatisierbarkeit	Meßgröße
automatisierbar	pH-Wert
bedingt automatisierbar	HPLC-Analyse
nicht automatisierbar	viele Laboranalysen

Tabelle 1.9: Beispiele für verschiedene Grade der Automatisierbarkeit

1.3 Anforderungen an ein Datenanalysesystem

Bei der im Abschnitt 1.2 geschilderten Datenlage wird klar, daß es ein allgemeines Verfahren zur Analyse biotechnischer Prozeßdaten nicht geben kann. Vielmehr muß ein solches System als ein "Werkzeugkasten" mit verschiedenen Datenanalysetechniken aufgefaßt werden. Es ergibt sich ein Anforderungskatalog für die Struktur eines biotechnischen Datenanalysesystems:

1.3.1 Vereinheitlichendes theoretisches Grundkonzept

Der Begriff "Werkzeugkasten" impliziert nicht, daß es sich um eine bloße Programmsammlung handeln soll. Vielmehr müssen alle Datenanalysetechniken in einem vereinheitlichenden theoretischen Grundkonzept zusammengefaßt sein. Ein solches Grundkonzept

- legt die logische Struktur des Datenanalysesystems fest,
- legt Methoden, die zur Anwendung kommen können, nahe,
- legt die Beziehung der Datenanalysebausteine untereinander fest.

1.3.2 Graphische Umgebung

Die Wichtigkeit einer graphischen Präsentation von Information wird immer noch weitgehend unterschätzt, obwohl schon bei elementaren Beispielen klar wird, wie stark die Erfassung des Gehaltes eines Datensatzes von der optischen Präsentation beeinflußt wird (vgl. Abb. 1.5). Eine graphische Benutzeroberfläche für die Datenanalyse kann unter mehreren Aspekten betrachtet werden:

1.3.2.1 Interaktive Bedieneroberfläche

Benutzeroberflächen werden heute in der Regel als graphische Oberflächen implementiert. Bei der Gestaltung einer solchen Oberfläche ist folgendes zu beachten:

- Ein vereinheitlichendes theoretisches Grundkonzept zieht insbesondere eine Vereinheitlichung der Bedienungselemente nach sich.
- Der Benutzer findet bei der Anwendung verschiedener Datenanalysebausteine jeweils verwandte Strukturen vor. Die Akzeptanz des Systems wird dadurch stark gefördert, die Einarbeitungszeit verkürzt sich.
- Aus der Einsicht, daß die Datenanalyse nicht nach einem allgemeinen Muster ablaufen kann, resultiert unmittelbar die Forderung nach einer interaktiven Arbeitsumgebung. Der Benutzer muß selbst wählen können, mit welcher Kombination von Analysetechniken er seine Daten behandelt.
- Dabei müssen Irrtümer korrigiert und alternative Wege eingeschlagen werden können.

1.3.2.2 Visualisierung von Daten

Es soll gefordert werden, daß sämtliche zahlenmäßige Information nach Möglichkeit unmittelbar graphisch dargestellt wird:

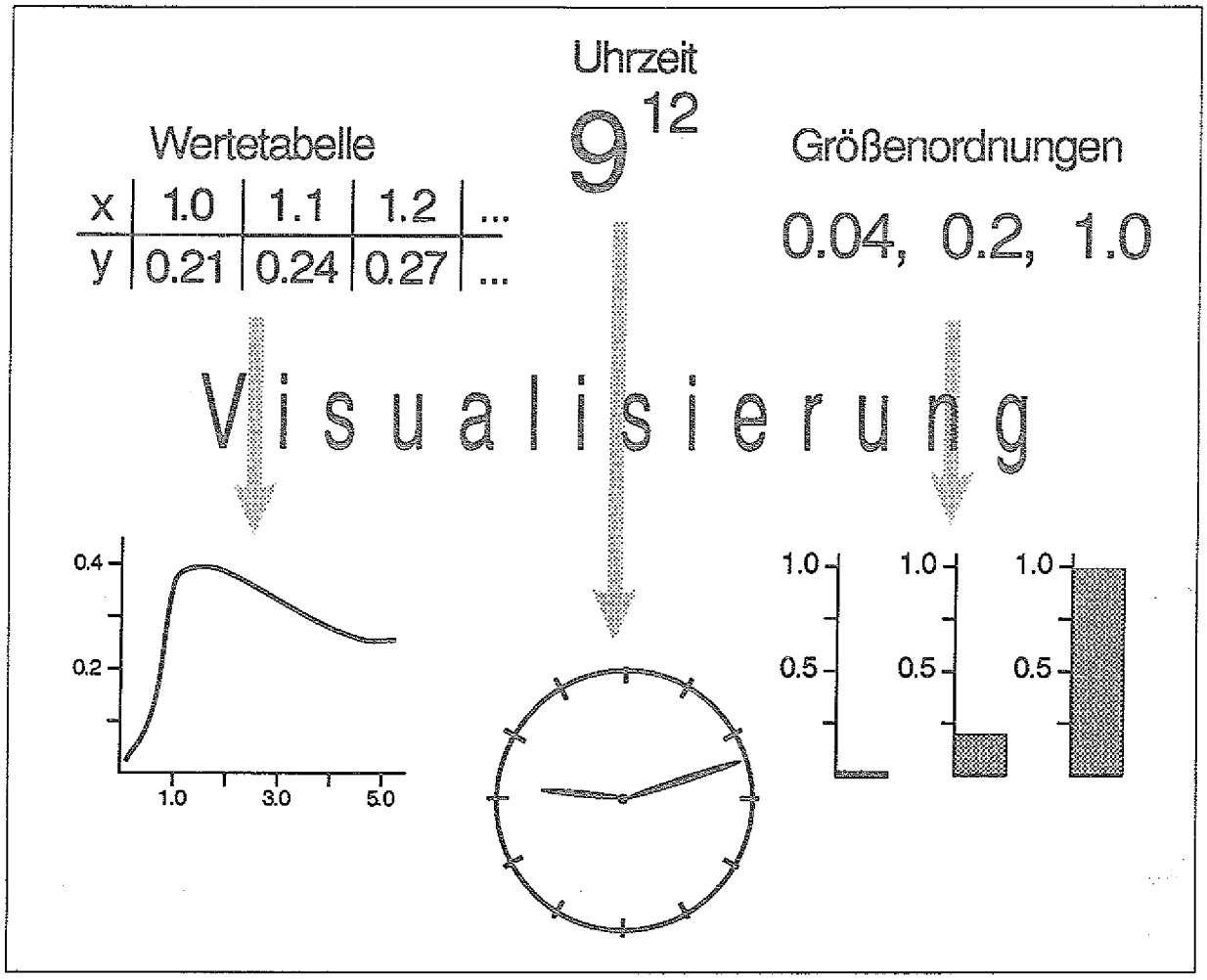


Abbildung 1.5: Graphische Visualisierung numerischer Daten

- Dazu stehen neben einfachen Funktionsgraphen noch eine Vielzahl anderer Präsentationstechniken zur Verfügung, die erst in den letzten Jahren entwickelt wurden (es sei hierzu auf das Kapitel 13 verwiesen).
- Auch die Eingabe und Veränderung von Meß- und Konfigurationsdaten kann graphisch realisiert werden.

In Kapitel 2 wird an einem Beispiel gezeigt, wie sich die Dauer einer Datenanalysesitzung durch die Benutzung graphischer Techniken drastisch verkürzen läßt!

1.3.2.3 Benutzerführung

Aus der Charakterisierung der experimentellen Situation in Abschnitt 1 ergibt sich die Forderung, daß im Idealfall der Experimentator selbst die Daten auswerten sollte. Zu einer Benutzerführung gehört

- eine sorgfältige didaktische Aufbereitung der theoretischen Grundlagen,
- eine Kollektion typischer Beispiele und
- die Bereitstellung von kontextsensitiven Hilfen.

1.3.2.4 Dokumentation

Zur Protokollierung der Ergebnisse wird benötigt:

- ein formatiertes Ablaufprotokoll der Datenanalysesitzung,
- die übersichtliche Ausgabe der Parametrisierung von Datenanalysebausteinen,
- die graphische Dokumentation der Resultate.

1.3.3 Werkzeuge

Die *Werkzeuge* des Datenanalysesystems sind die zur Verfügung gestellten Datenanalysemethoden. Bei der Zusammenstellung der Werkzeuge ist folgendes zu beachten:

1.3.3.1 Verarbeitung aller Datentypen

Das System muß prinzipiell in der Lage sein, alle im Abschnitt 1.2 bzgl. Abtastung, Zeitverzögerung, Genauigkeit, Verzerrungen und Wertebereich unterschiedenen Datentypen zu verarbeiten. Dazu sind die einzelnen Methoden auf eine möglichst allgemeine Situation anzupassen. So sind z.B. viele in der Literatur beschriebene Techniken nur für in äquidistanten Zeitabständen eintreffende Meßdaten entwickelt worden und müssen entsprechend erweitert werden.

1.3.3.2 Zukünftige Meßtechnik

Die Meßtechnik ist derzeit einer der in der Biotechnologie am stärksten expandierenden Bereiche [Büh85] [WSE84], d.h. Datenfülle und -heterogenität werden sich weiter vergrößern! Dieser Entwicklung hat die Konzeption des Datenanalysesystems durch entsprechende Flexibilität bzgl. der Datenrepräsentation Rechnung zu tragen.

1.3.3.3 Moderne Analyseverfahren

Die Bioprozeßsteuerung und -überwachung stellt ein anspruchsvolles Experimentierfeld für die Automatisierungs- und Regeltechnik dar. Es verwundert nicht, wenn die modernsten Methoden aus diesen Forschungsbereichen in der Biotechnologie erprobt werden. Auch hierbei ist die zu erwartende Entwicklung durch entsprechende Flexibilität bzgl. der Hinzunahme neuer Methoden zu berücksichtigen.

1.3.4 Datenverwaltung

Zur Datenverwaltung gehören:

- eine Datenbasis in Form eines Datenbanksystems, das den Anforderungen bzgl. Archivierung, Zugriff und Dokumentation des verfügbaren Datenmaterials Rechnung trägt.
- eine Datenverwaltung, welche die Verbindung zwischen Datenbasis und Datenanalysesystem herstellt und eine rechnerinterne Repräsentation der Daten zur Verfügung stellt.

1.3.5 Implementierung

Auch für die konkrete Realisierung des Systems gibt es Randbedingungen. Sie gewährleisten den Einsatz in einem breiten Anwendungsbereich:

1.3.5.1 Offene Architektur

Von einer *offenen Architektur* spricht man, wenn alle internen Daten des Systems transparent und über eine Programm-Schnittstelle dem Benutzer zugänglich gemacht sind. Solche Flexibilität wird bei einem komplexen System erreicht durch

- weitgehende Abstraktion und Formalisierung der Komponenten,
- starke Modularisierung.

Auf diese Weise wird die freie Anbindung weiterer Methoden durch einen Systemprogrammierer ermöglicht, so daß das System an den Bedarf angepaßt werden kann.

1.3.5.2 Automatisierbarkeit

"Automatisierbarkeit" der Datenanalyse bedeutet, daß ganze Abläufe der Datenanalyse in Form eines Protokolls oder eines Programms auf einer Meta-Ebene auch automatisch wiederholt werden können. Dadurch wird es möglich, erfolgreiche Analysetechniken erneut anzuwenden, ohne die Konfigurationsarbeit für die Einzelbestandteile erneut leisten zu müssen.

Insbesondere können solche automatisierten Bestandteile sofort in ein Prozeßüberwachungssystem eingebunden werden und stehen dann on-line zur Verfügung [5]. Auf diese Weise entsteht im Idealfall ein speziell angepaßtes Prozeßüberwachungssystem für den untersuchten Prozeß in einem Zuge mit der Verfahrensentwicklung!

1.3.5.3 Geschwindigkeit

Die Forderung nach einer kurzen Dauer der Datenanalyse resultiert aus dem im Abschnitt 1.1 über die Zeitplanung im experimentellen Zyklus Gesagten: Eine Datenanalysesitzung sollte höchstens in der zeitlichen Größenordnung der Experimentdauer liegen; andernfalls wird die Anwendung des Datenanalysesystems unattraktiv! Der Forderung nach Geschwindigkeit wird bereits durch die interaktive Umgebung mit Hilfestellung, die graphische Unterstützung und die Automatisierbarkeit ganzer Abläufe Rechnung getragen. Hinzu kommt die Aufgabe,

- zeiteffiziente Realisierungen der Methoden zu implementieren,
- aufwendigere Verfahren so in Einzelschritte zu zerlegen, daß der Ablauf dem Benutzer (z.B. mit graphischen Hilfsmitteln) transparent gemacht werden kann.

[5] Derartige Automatisierungstechniken werden oft unter dem Schlagwort *Rapid Prototyping* zusammengefaßt.

Kapitel 2

Beispiel einer Datenanalysesitzung

Das Ziel der vorliegenden Arbeit ist, eine Konzeption für die Datenanalyse bei Bioprozeßdaten vorzustellen, welche den Anforderungen aus Abschnitt 1.3 genügt. Bevor dieses Konzept auf abstrakter Ebene formuliert wird, soll anhand einer fiktiven Datenanalysesitzung vorgeführt werden, wie die Arbeit mit einem interaktiven Datenanalysesystem konkret aussehen könnte. Das auf diese Weise gewonnene Vorverständnis erleichtert die präzise Formulierung des Ansatzes.

2.1 Beispieldatensätze

2.1.1 Konzentrationsverläufe

Das Beispiel geht aus von den in Abbildung 2.1a dargstellten simulierten Prozeßdaten einer Batch-Fermentation. Dargestellt sind die Biomassekonzentration X, eine Substratkonzentration S und eine Produktkonzentration P. Der Zeitverlauf der Konzentrationen zeigt eine typische Wachstumsdiauxie:

Am Anfang steht eine kurze

Lag-Phase, gefolgt von einer

exponentiellen Wachstumsphase, in der der Organismus das Substrat S verbraucht und ein Produkt P ausscheidet, bis nach vollständiger Verwertung von S schließlich erneut eine

Lag-Phase eintritt. In dieser wird der Stoffwechsel für das Wachstum auf dem vorher gebildeten Produkt P umgestellt, so daß eine zweite

exponentielle Wachstumsphase beginnt, die schließlich nach Verbrauch aller Resourcen in der abschließenden

stationären Phase endet.

2.1.2 Bemerkungen zur Simulation

2.1.2.1 Modellannahmen

Die Daten aus diesem Beispiel wurden mittels Simulation auf der Basis eines Modells von Sweere et.al. für das diauxische Wachstum von Backhefe auf Glucose gewonnen. Die Parameter sind aus [SGB+88] und [SK86] zu entnehmen. Hinzugefügt wurde lediglich eine Lag-Phase von 2 Stunden am Anfang. Die Diauxie kommt hierbei zustande durch die beiden Stoffwechselwege

$$\text{Glucose} \xrightarrow{Gährung} \text{Biomasse} + \text{Äthanol}$$
$$\text{Äthanol} \xrightarrow{Atmung} \text{Biomasse} + CO_2$$

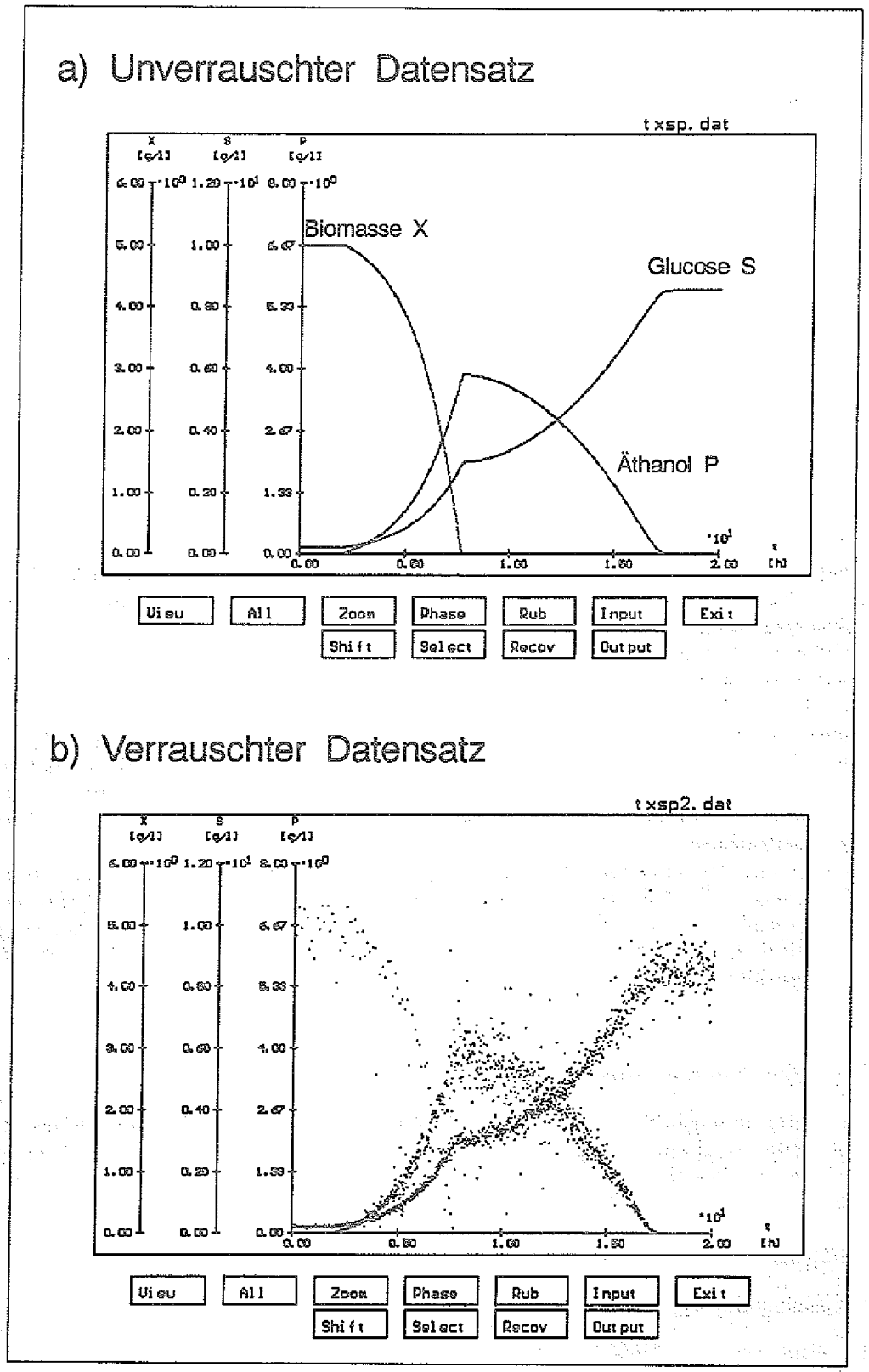


Abbildung 2.1: Diauxisches Wachstum von Backhefe auf Glucose

$$(\text{Biomasse} = X, \text{Glucose} = S, \text{Äthanol} = P),$$

wobei die Gärung bei Vorhandensein von Glucose bevorzugt wird. Dieser Effekt beruht darauf, daß für die Umsetzung von Äthanol spezielle Enzyme gebildet werden müssen. Eine vertiefende Diskussion des Modells sowie die Modellgleichungen finden sich in Abschnitt 5.2.4.5.

2.1.2.2 Datenerfassung

Es wird angenommen, daß die Messung der

Biomassekonzentration X kontinuierlich aufgrund einer Messung der optischen Dichte erfolgte (Abtastrate ca. 1 min), wogegen die Schätzung der

Produktkonzentration P auf einer Analyse der Abluft (jede Minute) beruht und die

Substratkonzentration S mittels einer On-Line-HPLC-Anlage gemessen wurde, und daher nur in äquidistanten Zeitintervallen (ca. 6 min) und mit entsprechender Zeitverzögerung vorliegt.

2.1.2.3 Meßrauschen

Schließlich wurde auf alle gemessenen Größen ein Meßrauschen aufgeschaltet, so daß die in Abbildung 2.1b dargestellten verrauschten Zeitverläufe entstanden sind. Um das Beispiel nicht zu kompliziert zu machen, wird davon ausgegegangen, daß die Meßdaten, die zunächst andere physikalische Einheiten tragen, bereits auf die Einheiten von X, S und P umgerechnet wurden [1]. Die damit eventuell verbundene Übertragung von Eichfehlern wird also vernachlässigt.

Die Simulation des Meßrauschens erfolgte mit Hilfe eines normalverteilten Weißen Rauschens (relativer Meßfehler, 5% bei Biomasse- und Glucosemessung bzw. 10 % bei der Äthanolmessung). Hinzu kommen Ausreißer mit einer vorgegebenen Wahrscheinlichkeit (5%), bei denen die Standardabweichung des Rauschens verdreifacht wurde. Diese verhältnismäßig einfache Methode zur Generierung von Rauschkomponenten wurde gewählt, um das Beispiel nicht mit den technischen Einzelheiten der stochastischen Simulation zu überfrachten.

Die Simulation der gasanalytischen Messung gelingt auf diese Weise recht gut, wie ein Vergleich mit realen Meßdaten zeigt (Abb. 2.2). Die Trübungsmessung wird dagegen sehr unrealistisch wiedergegeben (vgl. Abb. 6.5). Verfahren zur Erzeugung realistischer Rauschterme werden in Abschnitt 6.2 vorgestellt. Insbesondere wird in Abschnitt 6.3 gezeigt, wie eine Trübungsmessung simuliert werden kann. Für das Beispiel ist die gewählte Darstellung des Meßrauschens aber von Vorteil, da einige Effekte deutlicher zutage treten.

2.1.3 Ziel der Auswertung

Es geht nun darum, in möglichst kurzer Zeit ein Maximum an quantitativer Information aus den (verrauschten) Meßdaten zu gewinnen und im gleichen Zuge Bausteine für die On-Line-Prozeßüberwachung beim als nächstes zu planenden Fermenterlauf zu konfigurieren. Die beschriebenen Eigenschaften des Systems legen nahe, den Schwerpunkt auf

- die Schätzung von Wachstumsraten und Ertragskoeffizienten,
- die Erkennung und Eliminierung von Ausreißern und
- die Detektion von Übergängen zwischen den Prozeßphasen

zu legen.

[1] Beispielsweise müssen bei einer HPLC-Analyse Peakflächen in Konzentrationen umgerechnet werden.

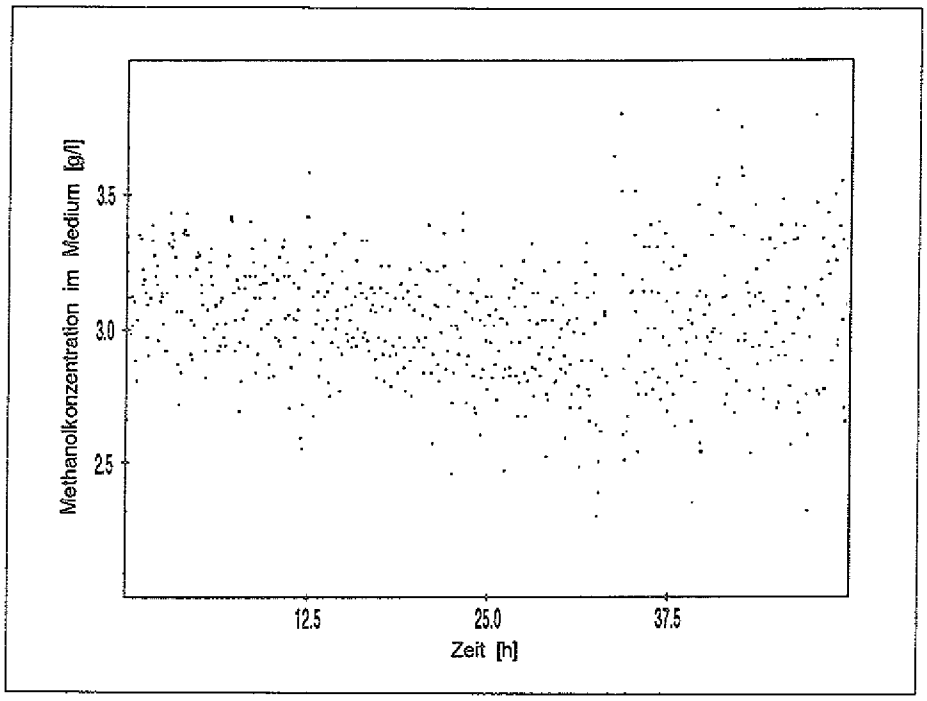


Abbildung 2.2: Typisches Signal einer gasanalytischen Messung (instationärer Prozeßverlauf)

2.1.4 Weitere Datensätze

Die Methoden zur Wachstumsratenschätzung, Ausreißerdetektion und Phasendetektion werden nicht an dem soeben beschriebenen — im folgenden als Datensatz A bezeichneten — Datensatz illustriert, sondern an drei weiteren Beispieldatensätzen. Dies hat zwei Gründe:

1. Es ist didaktisch sinnvoller, die Effekte an Beispielen aufzuzeigen, an denen sie klar hervortreten, wobei die Zahl der Einflußgrößen gering ist.
2. Die Beispiele werden im weiteren Verlauf der Arbeit noch benötigt. Es ist daher ökonomischer, sie bereits an dieser Stelle vorwegzunehmen.

Es wird jedoch klar werden, daß die beschriebenen Techniken prinzipiell auch auf den zur Eingangsmotivation verwendeten Backhefe-Datensatz angewendet werden können.
Die zusätzlichen Datensätze B,C und D entsprechen

B einem Standard-Modell für die Batch-Kultur mit inhibiertem Wachstum (vgl. 5.3.3.1 und 5.1.2.2) und einer anfänglichen Lag-Phase,

C einem einfachen exponentiellen Wachstumsmodell und

D einem Sprungmodell für den Übergang von der Lag-Phase in die exponentielle Wachstumsphase (vgl. 6.7).

In jedem Falle wurde ein Rauschen bzw. Ausreißer mit Hilfe der bereits skizzierten Techniken aufgeschaltet.

2.2 Datenaufbereitung mit einem graphischen Editor

Nach der Durchführung des (fiktiven) Experiments liegen die Prozeßdaten (Datensatz A) aus Abbildung 2.1b vor. Der Experimentator kann die Meßwertdatei auf den Datenanalyse-Rechner übertragen und auf dem Bildschirm visualisieren.

Bevor der Datenanalytiker beginnt, den Datensatz in seiner ganzen Komplexität zu untersuchen, sollte er zunächst eine grobe Aufbereitung und Strukturierung vornehmen. Im Anschluß wird er sich zunächst mit der Auswertung von Teilzusammenhängen beschäftigen. Erst danach kann aus den erhaltenen Teillösungen ein Gesamtkonzept erarbeitet werden.

Die Datenaufbereitung läßt sich mit Hilfe eines graphischen Daten-Editors in kurzer Zeit bewerkstelligen (vgl. die Abbildungssequenz 2.3). Nachdem der Datensatz in den Editor geladen worden ist (Menüeintrag "Input"), stehen die folgenden Funktionen zur Verfügung:

2.2.1 Phasensetzung

Über den Menüpunkt "Phase" wird ein graphischer Zeiger in Form eines beweglichen senkrechten Balkens angefordert (vgl. Abb. 2.3a), der auf der Zeitachse plaziert werden kann. Nach Arretierung des Zeigers als Phasengrenze wird automatisch ein neuer beweglicher Zeiger zur Verfügung gestellt. Eine bereits positionierte Phasengrenze kann duch erneute Anwahl wieder gelöscht werden.

2.2.2 Detaildarstellung

Die Datenfülle verwirrt, weil verschiedene Meßreihen sich überlagern oder stark verrauscht sind. Der Dateneditor bietet daher drei Möglichkeiten zur Detaildarstellung an, die frei kombiniert werden können:

- Darstellung einer einzelnen Phase (zwischen zwei Phasengrenzen) durch Auswahl von "Select" und Zeigen auf den gewünschten Bereich. In Abbildung 2.3b ist die zweite exponentielle Phase ausgewählt worden.
- Ausblenden einer oder mehrerer Meßreihen durch "Anklicken" der entsprechenden Menüpunkte "X","S" oder "P" in einem Submenü ("View") (vgl. Abb. 2.3c, wo "X" ausgewählt wurde).
- Ausschnittvergrößerung ("Zoom"), wobei ein Ausschnitt des Bildes in Form eines beweglichen Fensters ausgewählt und vergrößert dargestellt wird (Abb. 2.3d). Bei Überschreitung einer gewissen Auflösungsgrenze, erscheinen die einzelnen Datenpunkte als Kreuze markiert (Abb. 2.3e).

2.2.3 Ausreißerauslöschung

Die Ausreißerauslöschung ist mit einer "graphischen Bürste" [BC88] sehr einfach zu bewerkstelligen: Die Bürste (angefordert über den Menüpunkt "Brush") wird über das Bild gezogen und löscht die überstrichenen Datenpunkte aus (Abb. 2.3f). Wichtig ist dabei, daß die Daten nicht wirklich gelöscht, sondern nur inaktiviert werden, was im Bild durch den Übergang von Kreuzen zu Punkten angedeutet wird. Sie können daher mit derselben Technik auch wieder hervorgeholt werden (Menüpunkt "Recover").

Schließlich können die so bearbeiteten Daten phasenweise in Dateien abgespeichert werden ("Output"). Die Datei enthält dann exakt die Meßpunkte, welche aktuell auf dem Bildschirm zu sehen sind (WYSIWYG-Prinzip [2]).

2.3 Konfiguration eines Kalman-Filters

Nach der Aufbereitung der Daten geht es jetzt um die eigentliche Informationsgewinnung: Eine der wichtigsten Größen zur Charakterisierung des Wachstumsverhaltens von Mikroorganismen ist die spezifische Wachstumsrate μ. Als erste Aufgabe soll daher die Schätzung von μ während der (vermutlich) exponentiellen Phasen gestellt werden. Faßt man die Wachstumsrate zunächst als eine (auf unbekannte Weise) zeitabhängige Größe auf, so läßt sich dies in der bekannten Form

$$\dot{X} = \mu(t)\, X$$

ausdrücken. Im Prinzip berechnet sich daraus die Wachstumsrate mit

[2] WYSIWYG = "What you see is what you get"

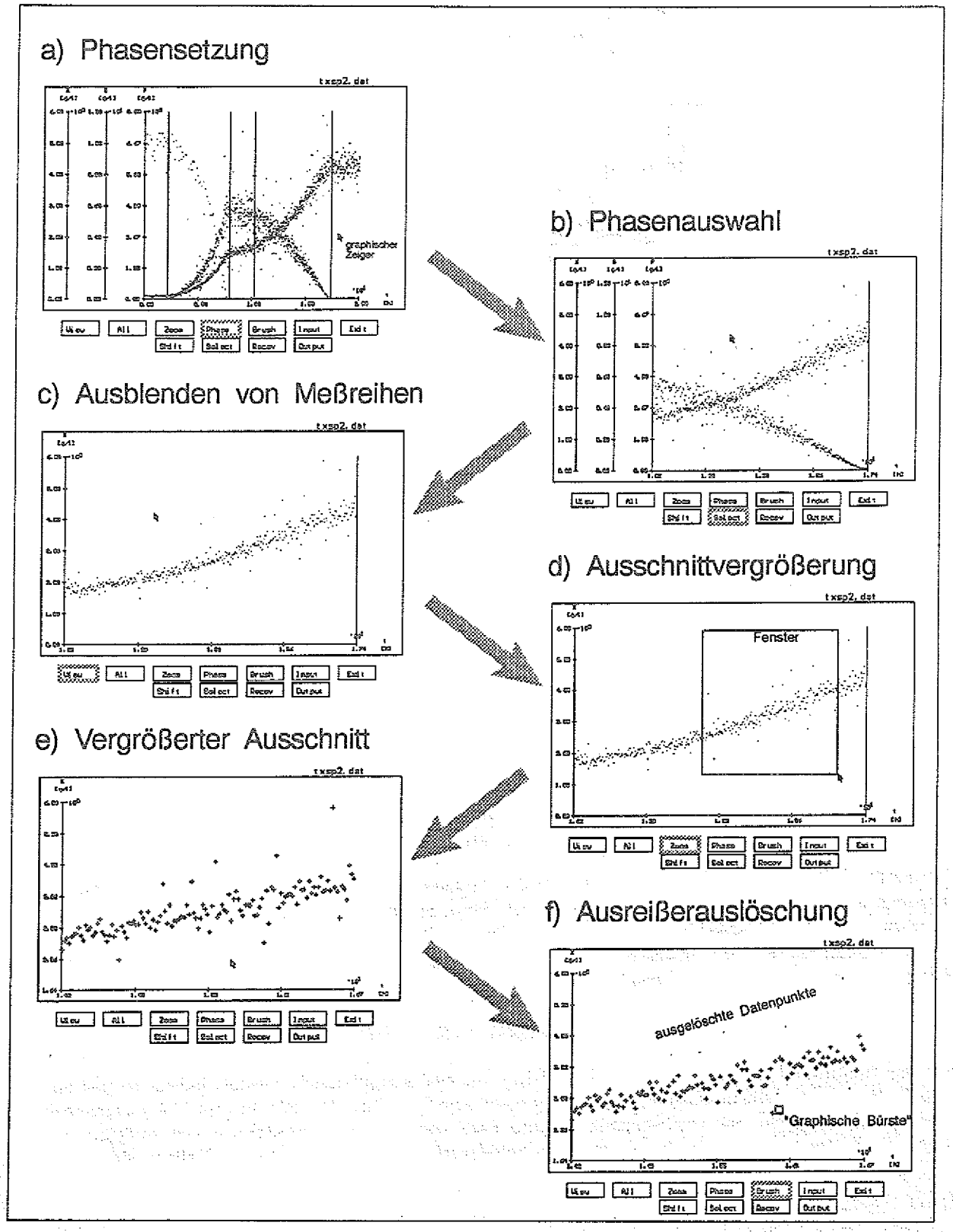


Abbildung 2.3: Graphischer Dateneditor

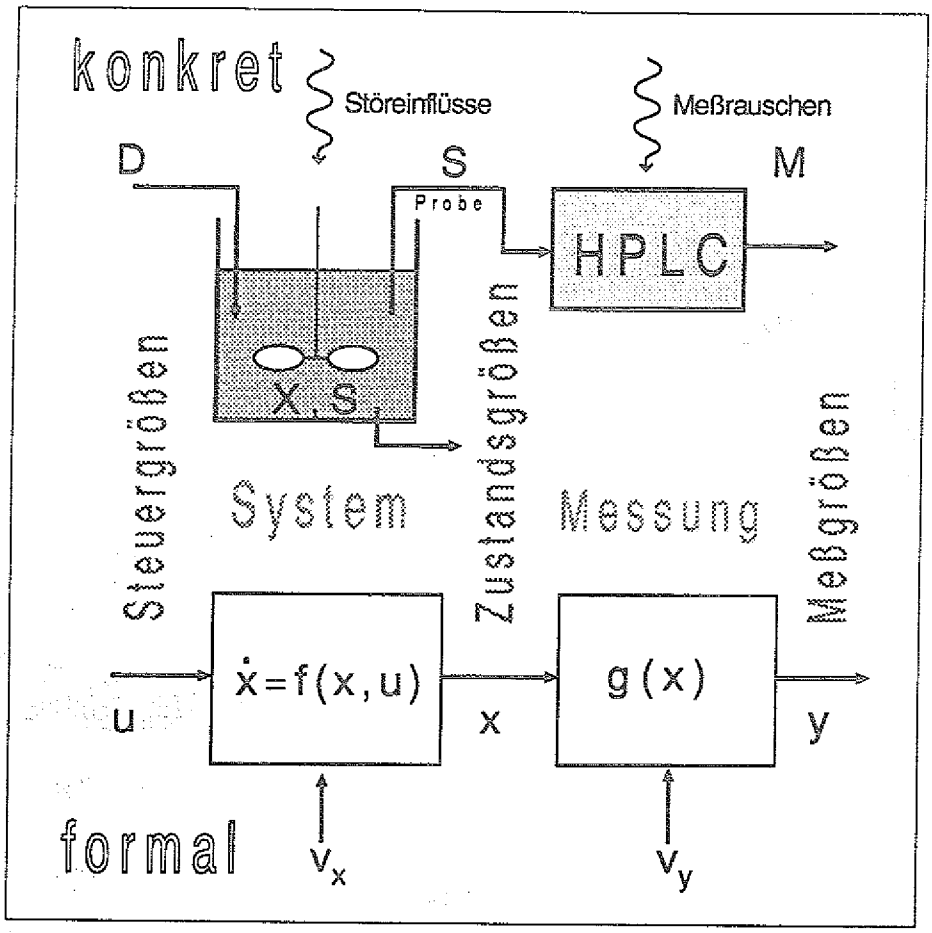


Abbildung 2.4: Struktur eines stochastischen Zustandsraummodells

$$\mu(t) = \frac{\dot{X}(t)}{X(t)} \quad .$$

Nun muß beachtet werden, daß $X(t)$ nicht etwa die Biomasse ist, sondern nur eine verrauschte Messung derselben darstellt. Die Messung der Biomasse soll daher präziser mit Y bezeichnet werden, die wahre (aber unbekannte) Biomasse mit X. Es stellt sich das Problem der Differentiation von $Y(t)$.

Wenn man das Blickfeld etwas erweitert, so stößt man schnell auf weitere Differentialgleichungs-Modelle, wie z.B. die Substratgleichung

$$\dot{S} = -Y_{S/X}\,\mu(t)X + (S_{in} - S)D$$

bei einer kontinuierlichen Kultur (vgl. z.B. 5.3.3.1), die eine kompliziertere Gestalt haben, so daß eine einfache Umformung wie im obigen Beispiel nicht mehr zum Ziel führt. Es stellt sich somit ganz allgemein das Problem der Schätzung von zeitveränderlichen Parametern in Differentialgleichungen aufgrund von verrauschten Messungen. Eines der elegantesten Schätzverfahren ist das sogenannte Kalman-Filter:

2.3.1 Prozeßmodell

Das Kalman-Filter geht von einem stochastischen, d.h. unscharfen Modell des Systems aus. Dabei wird zwischen dem eigentlichen Prozeß und der Messung unterschieden, wobei jeder Teil einem unbekannten Störeinfluß unterworfen ist (vgl. Abb. 2.4).

Die Messung wird beschrieben durch ihre Eichkurve und ein aufgeschaltetes Rauschen. Da die Umrechnung der physikalischen Einheiten im Beispiel vernachlässigt wurde (vgl. 2.1.2.3), entfällt die Eichfunktion und das Modell für die Messung der Biomasse ist:

$$M = X + w,$$

wobei w ein Störterm ist, der das Meßrauschen beschreibt. Das System selbst wird durch eine Differentialgleichung mit Rauschkomponenten beschrieben; z.B. so:

$$\begin{aligned} \dot{X} &= \mu X \\ \dot{\mu} &= 0 \quad + v \end{aligned}$$

D.h. μ verändert sich im wesentlichen nicht ($\dot{\mu} = 0$), bis auf einen Störterm v. X wächst dagegen exakt mit der jeweiligen Wachstumsrate μ ohne weitere Störung [3]. Wie eine solche stochastische Differentialgleichung genau zu verstehen ist, spielt jetzt noch keine Rolle. Wichtig ist nur, daß die Größe der Streuung von v und w über die Standardabweichungen

$$\sigma_M \text{ und } \sigma_v$$

vorgegeben werden kann.

Basierend auf dem Modell und der Vorgabe von σ_M und σ_v schätzt das Kalman-Filter on-line aus der Trübungsmessung die beiden nicht direkt zugänglichen Größen X und μ und gibt die beiden Schätzer

$$\hat{X}(t) \text{ und } \hat{\mu}(t)$$

aus.

2.3.2 Theorie

Ganz allgemein beruht das erweiterte Kalman-Filter auf einem stochastischen Modell der Form (vgl. auch Abb. 2.4):

$$\begin{aligned} \dot{x} &= f(t,x,u) + v_x \quad \text{(Systemgleichung)} \\ y &= g(t,x) + w_y \quad \text{(Meßgleichung)} \end{aligned}$$

$$\text{Im Beispiel ist: } \begin{aligned} x &= (X,\mu) & y &= (M) \\ v_x &= (0,v) & w_y &= (w) \end{aligned}$$

Das zugehörige erweiterte Kalman-Filter hat folgende Gleichungen (Sie werden in 11.3 hergeleitet):

A priori Zustandsschätzung:

$$\begin{aligned} \dot{\hat{x}} &= f(t,\hat{x},u) \\ \hat{y} &= g(t,\hat{x}) \end{aligned}$$

Matrix-Riccati-Differentialgleichungen:

$$\begin{aligned} \dot{P} &= f_x(\hat{x})\,P + P\,f_x^T(\hat{x}) + R \\ \dot{Q} &= g_x(\hat{x})\,P\,g_x^T(\hat{x}) + S \end{aligned}$$

mit den Kovarianzmatrizen:

$$\begin{aligned} P &= \mathrm{Cov}[x(t)] & Q &= \mathrm{Cov}[y(t)] \\ R &= \mathrm{Cov}[v_x(t)] & S &= \mathrm{Cov}[w_y(t)] \end{aligned}$$

und den Jacobi-Matrizen:

[3] Die stochastischen Wachstumsmodelle werden in Abschnitt 7.4 noch ausführlich diskutiert

$$f_x = \frac{\partial f}{\partial x} \quad \text{und} \quad g_x = \frac{\partial g}{\partial x}$$

Kalmansche Verstärkungsmatrix:

$$K = P\, g_x^T(\hat{x})\, S^{-1}$$

A posteriori Zustandsschätzer:

$$\begin{array}{lcll} P & := & P & - \; K\, g_x(\hat{x})\, P \\ \hat{x} & := & \hat{x} & + \; K\,(y - \hat{y}) \\ \hat{y} & := & g(\hat{x}) & \end{array}$$

Für ein tiefergehendes Verständnis der Gleichungen ist ein teilweises Studium der Gebiete

- Lineare Algebra,
- Wahrscheinlichkeitstheorie,
- Statistische Schätztheorie,
- Gewöhnliche Differentialgleichungen,
- Stochastische Differentialgleichungen und
- Kontrolltheorie

erforderlich! Auf diesen Aufwand ist es zurückzuführen, daß das Kalman-Filter zwar heute in der Bioverfahrenstechnik ein Begriff ist, aber bisher nur von Spezialisten angewendet wird. Es stellt sich die Frage:

1. Wieviel Grundwissen muß von einem Praktiker verlangt werden, damit er ein Kalman-Filter konfigurieren und die Ausgaben des Filters korrekt interpretieren kann ?
2. Wieviel davon kann automatisiert und damit dem Benutzer abgenommen werden ?

2.3.3 Datenflußschema

Auf der Basis der Lehrerfahrung [WSS88] soll die folgende Antwort gegeben werden: Vom Praktiker muß verlangt werden, daß er

1. die biotechnischen Standardmodelle in Form von Differentialgleichungen lesen kann.
2. die Fertigkeit zur sinnvollen Erweiterung und eventuellen Umformulierung einer derartigen Gleichung mit Hilfe von Rauschtermen hat.
3. eine Vorstellung von der Bedeutung und Größenordnung der Rauschkomponenten mitbringt.
4. Startwerte für die Initialisierung der Schätzwerte sowie Genauigkeitsangaben für diese Startwerte zur Verfügung stellen kann.

Punkt 1 sollte ein selbstverständlicher Teil der Ausbildung eines Biotechnologen sein. Das Wissen zu Punkt 2 läßt sich mit Fortbildungsmaßnahmen vermitteln wobei Simulationen und didaktisch angelegte Demonstrationen eine große Hilfe sind [4]. Punkte 3 und 4 sind Fragen, die sich ohnehin nur aus dem Gespür für das biologische System und der Erfahrung heraus beantworten lassen.

Mit den Angaben aus Punkt 1 - 4 läßt sich ein erweitertes Kalman-Filter vollständig beschreiben und parametrisieren. Man benötigt:

[4] Die Ausführungen in Kapitel 11 beruhen nicht zuletzt auf Erfahrungen im Zusammenhang mit der didaktischen Aufbereitung der Filtertheorie [WSS88].

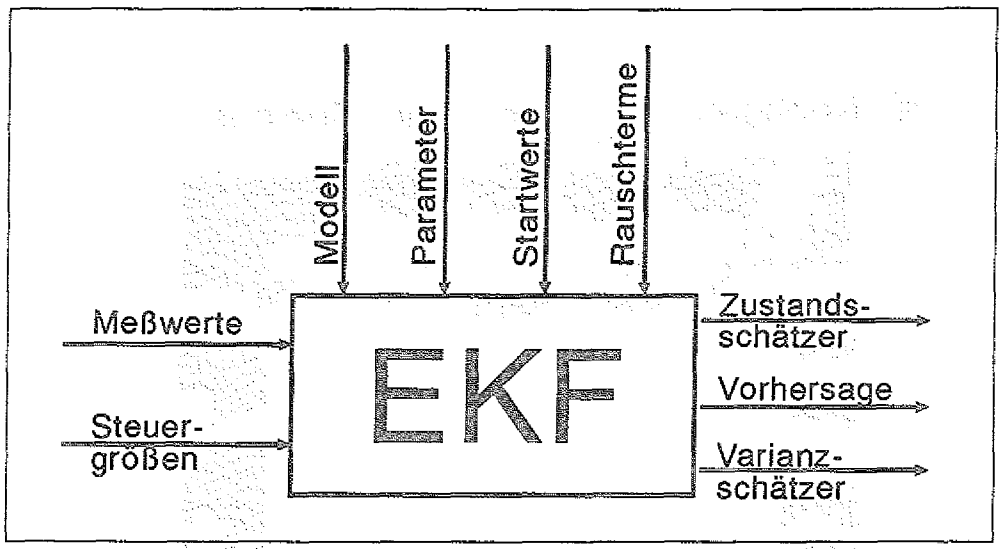


Abbildung 2.5: Datenflußschema eines Erweiterten Kalman-Filters

- Systemgleichungen in Form von Differentialgleichungen,
- Meßgleichungen in Form von Eichfunktionen,
- Startwerte für die Zustandsschätzer,
- Genauigkeitsmaße für die Startwerte in Form von Standardabweichungen,
- Streumaße für das Meßrauschen,
- Streumaße für das Systemrauschen.

Hierbei ist einzig das Streumaß für das Systemrauschen größenordnungsmäßig nicht genau bekannt. Den Fehler einzelner Meßgeräte und die Genauigkeit der Startwertschätzung kennt man hingegen meist recht gut.

Das Datenflußschema (Abb. 2.5) stellt die benötigten Konfigurationsangaben, Eingangs- und Ausgangsdaten des Filters noch einmal übersichtlich zusammen. Es abstrahiert das Filter zu einem Datenanalysebaustein.

2.3.4 Interaktive Konfiguration

Die Konfiguration des Kalman-Filters kann dem Benutzer über eine Benutzeroberfläche so einfach wie möglich gemacht werden (siehe Abb. 2.6): Modell und Parameter sind dabei in ein "Formular" einzutragen. Die Dimension des Zustandsraums und des Meßvektors werden interaktiv verändert, wobei sich alle Vektoren und Matrizen, die dieselbe Dimension haben, automatisch in der Größe anpassen. Ein Feld ohne Eintrag fällt daher sofort auf. Über eine Hilfe-Funktion kann bei Unschlüssigkeit kontextsensitive Hilfe angefordert werden.

Bemerkung 1: Dieser Ansatz unterscheidet sich wesentlich von der oft üblichen Verwendung einer Konfigurationsdatei. Bei der Erstellung einer solchen Datei muß der Benutzer eine Metasprache beherrschen und hat zudem keine kontextsensitive Hilfestellung zur Verfügung (vgl. auch [Bre83] [BSS84]).

Bemerkung 2: Eine denkbare Variante, die noch benutzerfreundlicher ist, wäre das graphische Anwählen der gewünschten Konfigurationspunkte in der schematischen Darstellung des Datenanalysebausteins (Abb. 2.5).

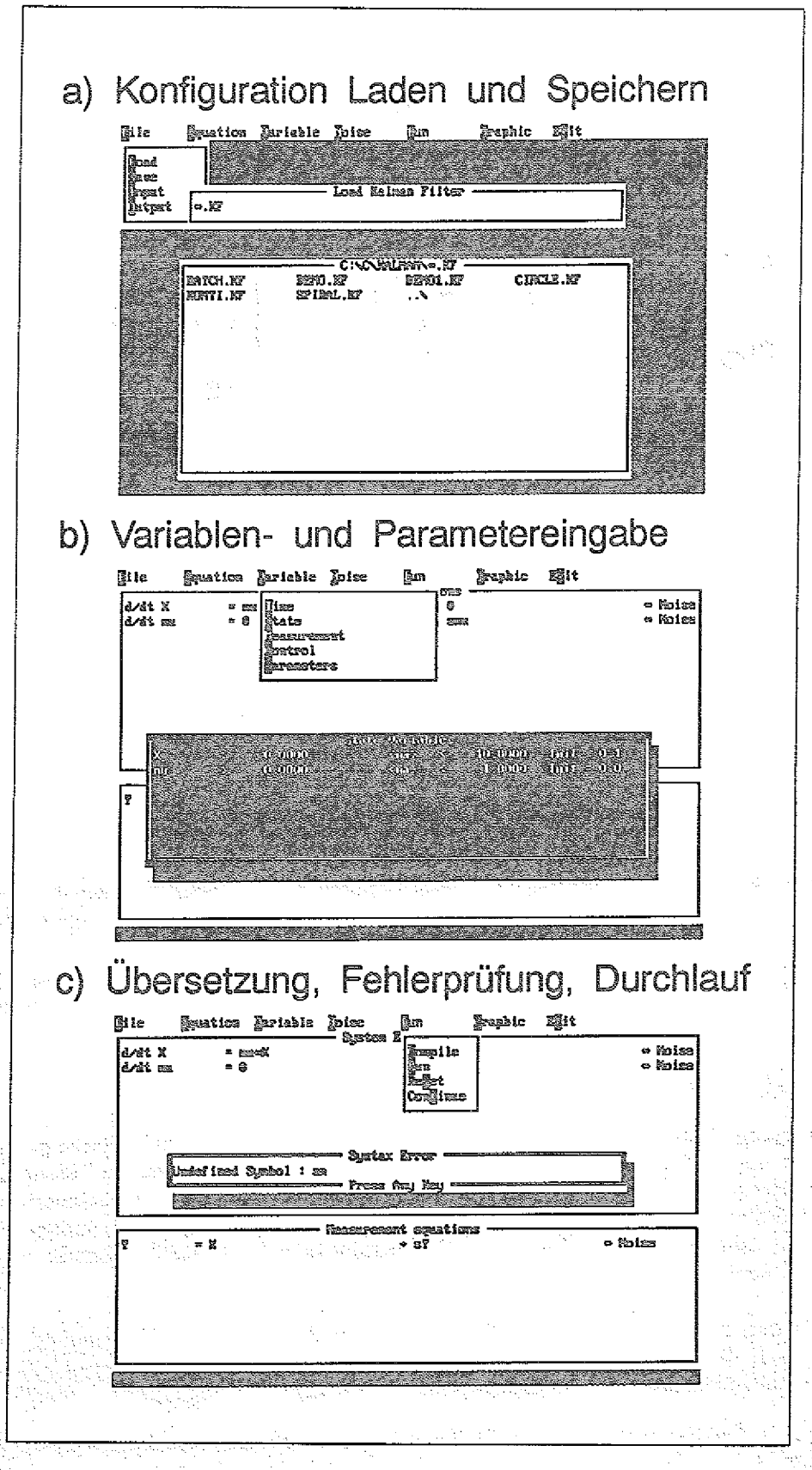


Abbildung 2.6: Interaktive Konfiguration eines Erweiterten Kalman-Filters

Ist das Filter fertig konfiguriert, so können alle Daten unter einem Namen (hier z.B. "Wachstumsrate") abgespeichert und mit einem Kommentar (z.B. "Wachstumsraten-Schätzer, angewendet am ... beim Prozeß ...") versehen werden. Mit der Zeit entsteht so eine Bibliothek von Verfahren, die bei Bedarf nur noch abgerufen werden müssen. Eine bestehende Parametrisierung kann jederzeit wieder verändert und unter einem neuen Namen abgespeichert werden.

2.3.5 Compiler, Differenzierer und numerische Realisierung

Es folgt der automatisierte Teil des Ablaufs:

- Ein Formelcompiler setzt alle vorkommenden arithmetischen Ausdrücke in die Umgekehrte Polnische Notation um (Parsing). Dabei werden Syntaxfehler erkannt und gemeldet (ggfs. mit Hilfestellung).
- Ebenso wichtig ist das automatische Differenzieren von Formeln zur Berechnung der Jacobi-Matrizen: Es ist eine oft gemachte Erfahrung [Boc83], daß sich bei manueller Ausführung des Differenzierens und der Eingabe arithmetischer Ausdrücke über die Tastatur Fehler einschleichen.

  Ein Beispiel: Der in der üblichen mathematischen Notation gegebene Ausdruck

  $$\frac{(x+y)^2}{-(a+b)\cdot \sin x} \quad \text{muß in der linearen Form} \quad (x+y)\uparrow 2/(-(a+b)*\sin(x)))$$

  eingegeben werden. Ein Klammerfehler ist dabei nur schwer zu lokalisieren! D.h. selbst dann, wenn die Differentiation — die im allgemeinen komplizierte Terme liefert — korrekt durchgeführt wurde, besteht immer noch das Risiko der falschen Eingabe.
- Ein entscheidender Rechenzeitfaktor bei der numerischen Realisierung ist die Formelauswertung. Ein Compiler übersetzt die Ausdrücke von Polnischer Notation in einen maschinennahen Code, um damit eine möglichst hohe Rechengeschwindigkeit zu erzielen.
- Die numerische Realisierung basiert auf Standardkomponenten zur Matrizenrechnung aus einem Numerik-Paket sowie einem Standard-Differentialgleichungs-Löser [5]. Entscheidend ist, daß der Algorithmus von vornherein für die Verarbeitung fehlender und verzögert eintreffender Meßdaten programmiert wird. Damit ist maximale Flexibilität erreicht (vgl. auch [HKK86]).

2.3.6 Durchlauf

Ansetzen des Filters auf den Beispieldatensatz B (inhibiertes Wachstum mit anfänglicher Lag-Phase) resultiert in der Abbildung 2.7a. (Dabei ist zur Verdeutlichung die der Simulation zugrunde liegende Wachstumsrate mit eingezeichnet). Es fällt sofort auf, daß bei dieser Parametrisierung

- eine Lag-Phase auftritt,
- eine Wachstumsinhibierung zu Anfang vorliegt,
- die Wachstumsrate μ nicht mit großer Genauigkeit zu schätzen ist.
- das Filter mit dieser Parametrisierung den sprunghaften Änderungen der Wachstumsrate nur träge folgt.

Diese Beobachtungen vermitteln einen realistischen Eindruck davon, was ein adaptives Filter vermag und was nicht. Die graphische Darstellung erweist sich auch hier als sehr nützlich, da sie die Grenzen der Automatisierbarkeit verdeutlicht. Oft werden an die Leistungsfähigkeit automatischer Verfahren überzogene Ansprüche gestellt!

[5] Zur Problematik steifer Differentialgleichungen im Zusammenhang mit der nichtlinearen Filterung sei verwiesen auf 9.1.0.3.

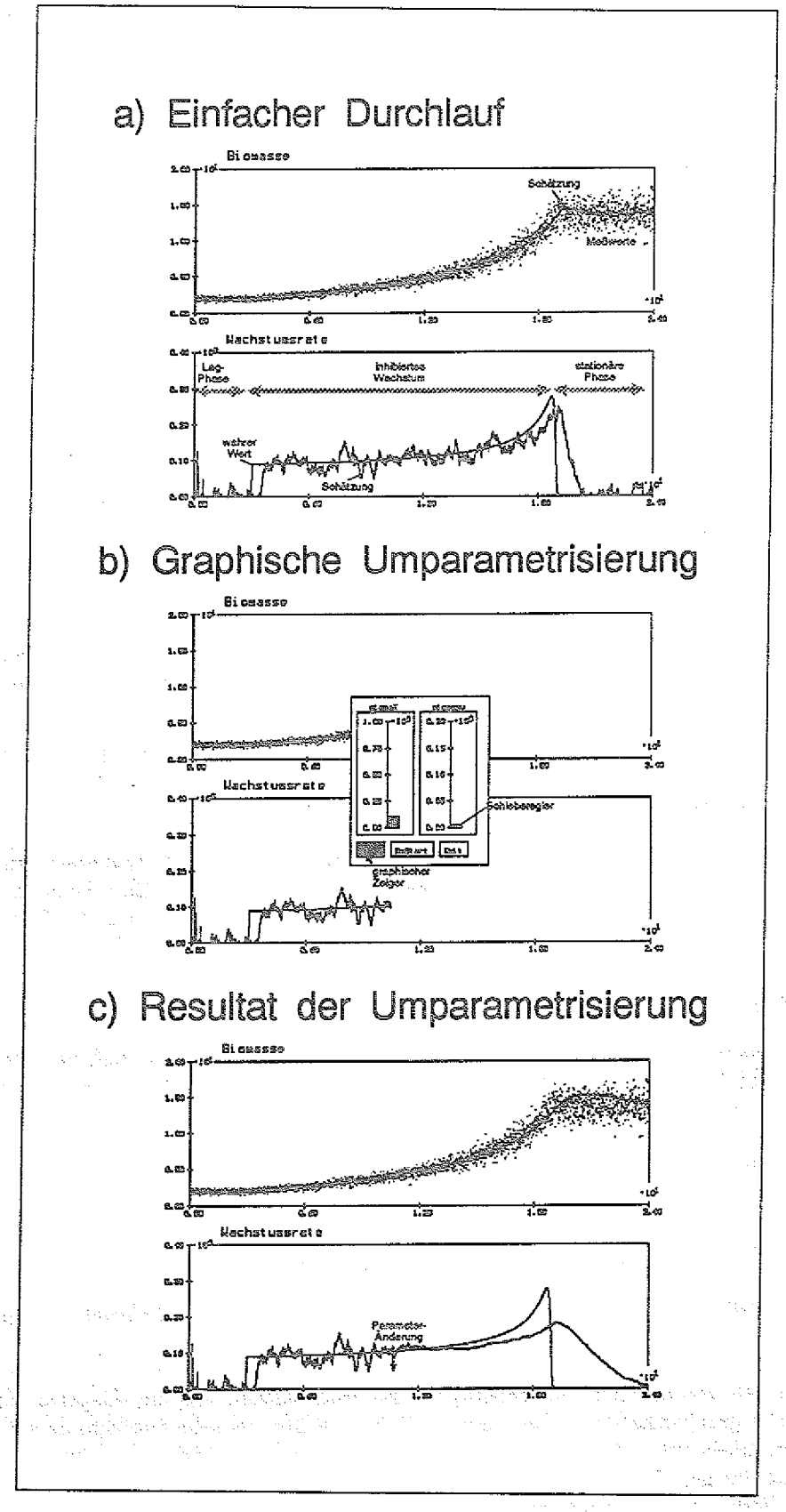


Abbildung 2.7: Durchlauf eines Erweiterten Kalman-Filters zur Schätzung der Wachstumsrate

2.3.7 Graphische Parametrisierung

Die zunächst gewählten Parameter waren sicherlich nicht sofort die bestmöglichen. Zur Parametrisierung muß vor allem der Einfluß der Rauschparameter σ_M und σ_v sowie der Startwerte $\hat{X}(0)$ und $\hat{\mu}(0)$ systematisch untersucht werden [6]. Die Theorie macht hierzu einige Vorgaben:

- σ_M beschreibt die Größe des Meßfehlers bei der Biomassemessung und muß entsprechend der Streuung der Trübungsmessung gewählt werden.
- σ_v beschreibt die durchschnittliche Geschwindigkeit der Veränderung von μ, wie das Modell

$$\dot{\mu} = 0 + v$$

  besagt. Dieser Parameter ist ohne Vorwissen nur grob schätzbar. Aus dem Modell folgt, daß σ_v die Trägheit des Filters bestimmen wird.
- $\hat{X}(0)$ ist die Anfangsbiomassekonzentration und mit Hilfe einer Analyse recht genau bestimmbar.
- $\hat{\mu}(0)$ ist die Anfangswachstumsrate, also 0.

Die Überlegungen zeigen, daß man vor allem den Einfluß von σ_v untersuchen sollte. Ein Gefühl für die richtige Parametrisierung entsteht am besten dann, wenn man die Parameter auf graphische Schieberegler legt, die on-line mit einem Zeiger verstellt werden können (siehe Abb. 2.7b). Auf diese Weise kann ein Umschalten zwischen zwei Parametrisierungen sogar innerhalb eines Durchlaufs vorgenommen werden. Abb. 2.7c zeigt ein solches Beispiel. Der Einfluß der relevanten Parameter ist nun auf interaktivem Wege schnell untersucht:

- Entscheidender Parameter ist σ_v. Er beeinflußt wie erwartet die Trägheit des Filters. Dies wird übrigens in Abb. 2.7c recht deutlich illustriert, wo dieser Parameter während eines Laufs stark verändert wurde
- Wichtig ist nur die Größenordnung der Parameter; kleine Veränderungen machen sich wenig bemerkbar
- Das Filter kann nicht gleichzeitig eine ruhige Wachstumsraten-Schätzung liefern und schnellen Veränderungen der Wachstumsrate gut folgen, es sei denn man verändert den Trägheitsparameter σ_v kontinuierlich mit der Substratkonzentration.

2.3.8 Erweiterung des Filters

Das Modell kann mit Hilfe eines zeitabhängigen Ertragskoeffizienten $Y(t) = Y_{S/X}(t)$ um eine Substratgleichung erweitert werden:

$$\dot{S} = -Y(t) \cdot \mu(t) \cdot X$$

Ganz analog konstruiert man daraus ein Kalman-Filter, das eine ruhige Schätzung der Substratkonzentration S und des Ertragskoeffizienten Y liefert (Abb. 2.8 zeigt einen Durchlauf in einer Ausschnittvergrößerung).

Bemerkenswert ist, daß das Filter auch mit Zeitverzögerungen und verschiedenen Abtastraten arbeiten kann (vgl. die Annahmen über die Messung von S in 2.1.2.2). Dabei wird zunächst eine Vorwärtsschätzung gemacht, die durch einen eintreffenden — rückwärts zu datierenden — Meßwert korrigiert wird und sich deshalb im Takt der Abtastung (allerdings kaum merklich) sprunghaft ändert.

Nach der Erweiterung des Modells kann auch das oben angesprochene Problem der dynamischen Veränderung von σ_v angegangen werden. Dies soll an dieser Stelle jedoch nicht mehr geschehen, da die prinzipielle Vorgehensweise jetzt genügend weit klar sein dürfte.

[6] Zwei weitere manipulierbare Parameter sind die Startwerte $\sigma_\mu(0)$ und $\sigma_X(0)$ für die Standardabweichungen der Schätzer $\hat{\mu}$ und $\hat{X}$. Sie haben einen verhältnismäßig geringen Einfluß auf das Resultat und werden hier der Übersichtlichkeit halber unterschlagen.

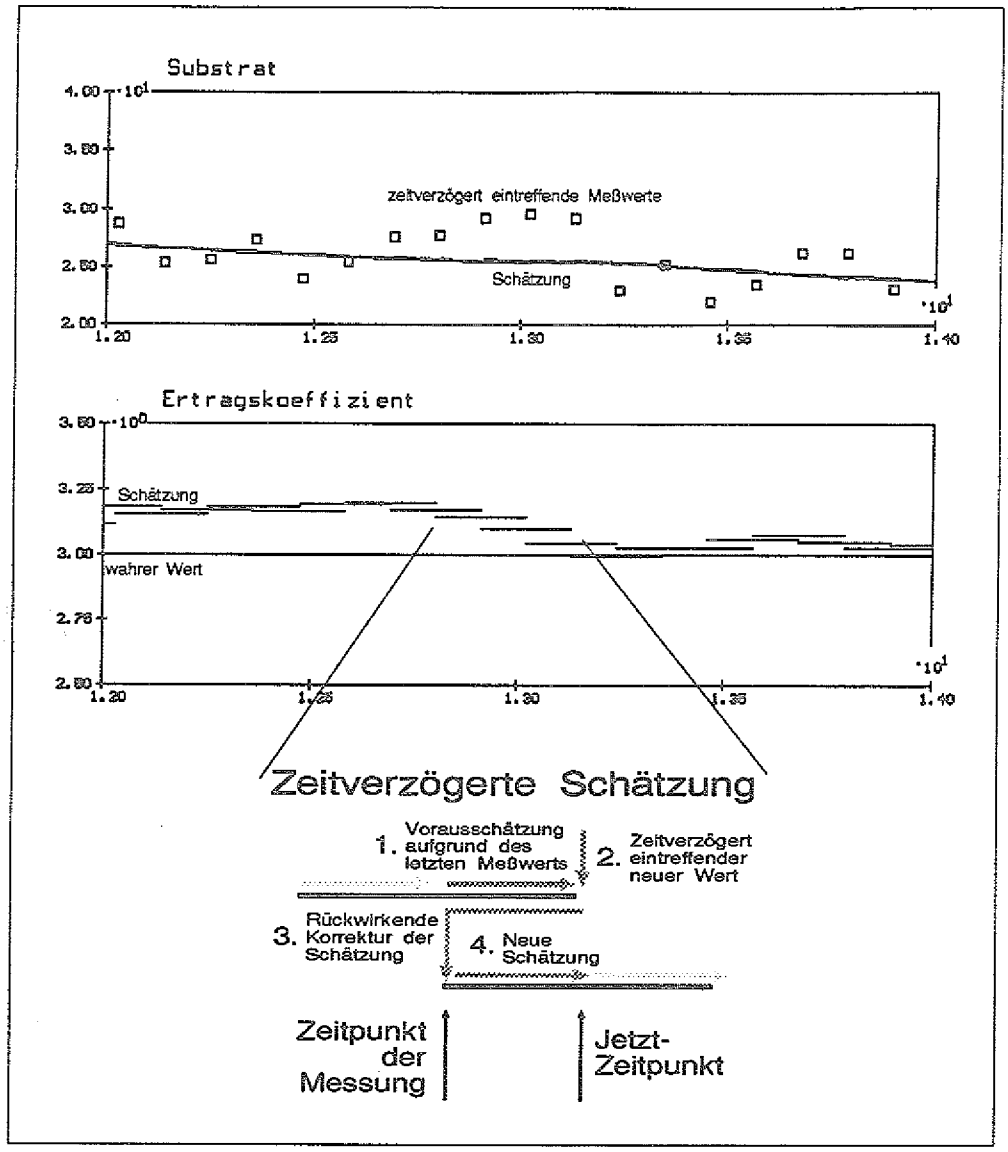


Abbildung 2.8: Schätzung von Substratkonzentration und Ertragskoeffizient bei Vorliegen einer Zeitverzögerung (stark ausschnittvergrößert)

2.4 Ausreißer-Erkennung

Das nächste Problem ist die Erkennung der Ausreißer. Sie wurden zu Anfang manuell entfernt, um das Kalman-Filter nicht unnötig zu stören. Am laufenden Prozeß ist dies natürlich nicht möglich. Gesucht wird also nach einer Methode zur automatischen Ausreißererkennung. Bei der Konzeption eines solchen Verfahrens muß man sich zunächst fragen, wie subjektiv ein Ausreißer erkannt wird. Dies führt zu der vorläufigen Formulierung:

> "Ein Ausreißer ist ein Meßwert, der im Verhältnis zu einer vorgegebenen Toleranz stark vom mittleren Kurvenverlauf abweicht."

2.4.1 Signifikanztests

Im Prinzip ist dies die Charakterisierung für einen Signifikanztest auf Abweichung von einem vorgegebenen Wert. Ein solcher Test benötigt:

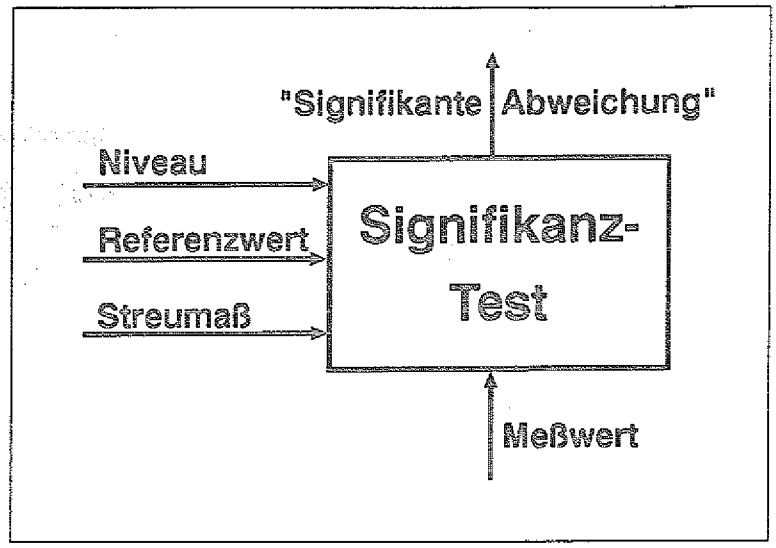


Abbildung 2.9: Datenflußschema eines Signifikanztests

- einen Referenzwert (hier der mittlere Kurvenverlauf),
- einen zu testenden Wert (nämlich den potentiellen Ausreißer),
- ein Streumaß [7] (hier die mittlere Streuung des Rauschens),
- ein Signifkanzniveau (bestimmt durch einen Faktor, der angibt, was unter einer "starken" Abweichung im Verhältnis zum Streumaß verstanden werden soll).

Die einfachste Realisierung eines Signifikanztests ist gegeben durch (Eine exakte Begründung findet sich in 10.5.2):

$$|\text{Referenzwert} - \text{Testwert}| < (\text{Streumaß} \cdot \text{Niveaufaktor})$$

Er kann dem Benutzer in Form eines weiteren Datenanalysebausteins zur Verfügung gestellt werden. Das zugehörige Datenflußschema zeigt die Abbildung 2.9.

2.4.2 Verschaltung von Datenanalysebausteinen

Eine weitere nützliche Eigenschaft des Kalman-Filters wurde noch nicht erwähnt: Es liefert nicht nur die Schätzer $\hat{X}(t)$ und $\hat{\mu}(t)$ sondern zugleich auch ein Streumaß dazu in Form der Standardabweichungen $\sigma_X(t)$ und $\sigma_\mu(t)$. Dabei werden alle Werte zusammen mit den Streumaßen stets auch für eine kleine Zeitspanne in die Zukunft prognostiziert; jedenfalls bis zum Eintreffen des nächsten Meßwertes.

Damit sind die Eingangsdaten für einen Signifikanztest gegeben, mit dem getestet wird, ob der gerade eintreffende Meßwert zu stark vom vorhergesagten Wert abweicht. Am Beispiel der Biomassemessung ist:

- der Referenzwert die Schätzung $\hat{X}(t)$ für die derzeitige Biomassekonzentration,
- das Streumaß gleich der Standardabweichung $\sigma_X(t)$,
- der Testwert der gerade gemessene Wert $M(t)$ für die Biomassekonzentration und
- das Signifikanzniveau α ein geeignet einzustellender Parameter.

Die Verbindung von Kalman-Filter und Signifikanztestbaustein ist ein erstes Beispiel für die Verschaltung von Datenanalysebausteinen zu komplexeren Verfahren. Die Schaltung gibt Abbildung 2.10 wieder. Erklärung des Ablaufs:

[7] Oft wird das Streumaß durch eine ganze Verteilungsfunktion verallgemeinert.

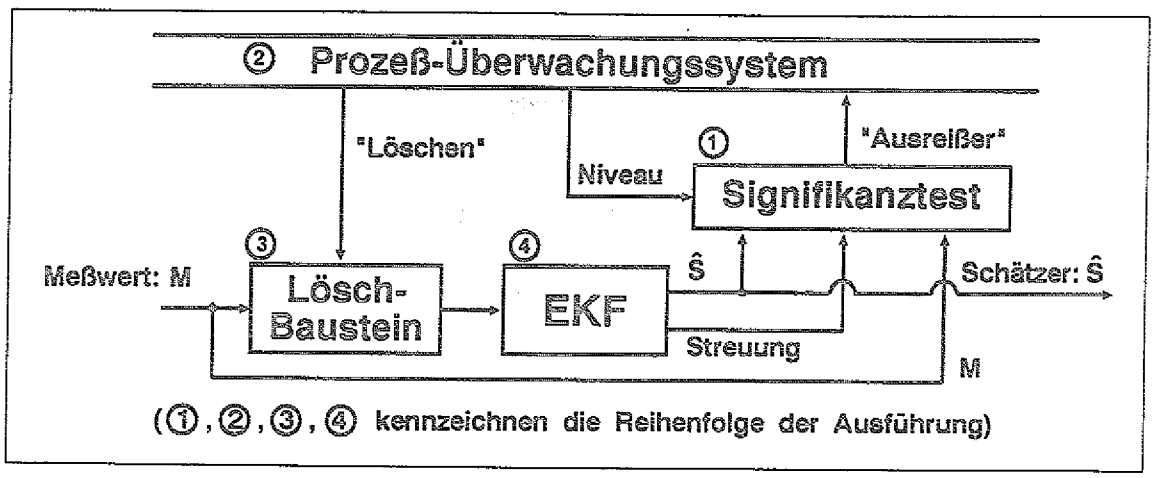


Abbildung 2.10: Datenflußschema eines Ausreißer-Detektors

1. Zunächst wird der eintreffende Biomasse-Meßwert im "Bypass" um das Kalman-Filter herumgeführt, wo ihn der Signifikanztest mit der aktuellen Prognose vergleicht.
2. Bei positivem Ergebnis erfolgt eine Meldung an das Prozeßüberwachungssystem, welches nun feststellen kann, ob Zusatzinformationen vorliegen, die gegen das Auftreten eines Ausreißers sprechen.
3. Ist dies nicht der Fall, so wird ein weiterer Baustein — ein Löschelement — damit beauftragt, den Meßwert nicht als Eingabe zum Kalman-Filter weiterzuleiten.
4. Das Filter muß nun für ein Zeitintervall ohne neuen Meßwert auskommen, was für eine allgemein programmierte Version des Filters kein Problem darstellt.

2.4.3 Durchlauf

Die Parametrisierung des Tests geschieht erneut auf graphischem Wege. Die Abbildung 2.11 zeigt einen solchen Durchlauf beim einfachen exponentiellen Wachstumsmodell (Beispieldatensatz C), wobei die simulierten Ausreißer zur Verdeutlichung mit eingezeichnet sind. Nach der Hälfte der Zeit wurde das Signifikanzniveau geändert. Es ist zu erkennen, daß

- einige Meßwerte fehlerhaft als Ausreißer detektiert werden (Fehler der 1.Art).
- einige "Ausreißer" deshalb nicht erkannt werden, weil der fehlerhaft gemessene Wert zufällig genau in den vorhergesagten Bereich fällt (Fehler der 2.Art).
- bei großem Signifikanzniveau α nur wenige Ausreißer erkannt werden, bei kleinem Niveau jedoch zu viele.

Ein in der Literatur oft genannter Erfahrungswert für die Wahl des Niveaus ist die 3σ-Umgebung des Referenzwertes [Sac84].

2.4.4 Makro-Programmierung und Prozeßanbindung

Der so konfigurierte Wachstumsraten-Schätzer mit Ausreißerdetektion soll nun an den Prozeß angebunden werden, um für das nächste Experiment on-line zur Verfügung zu stehen. Dazu wird zunächst die vollständige Konfiguration jedes Bausteins in einer Konfigurationsdatei abgelegt (vgl. 2.3.4). Sodann muß die Vernetzung in einer geeigneten Form festgelegt werden, z.B. wie folgt:

```
% Konfigurationsdateien
```

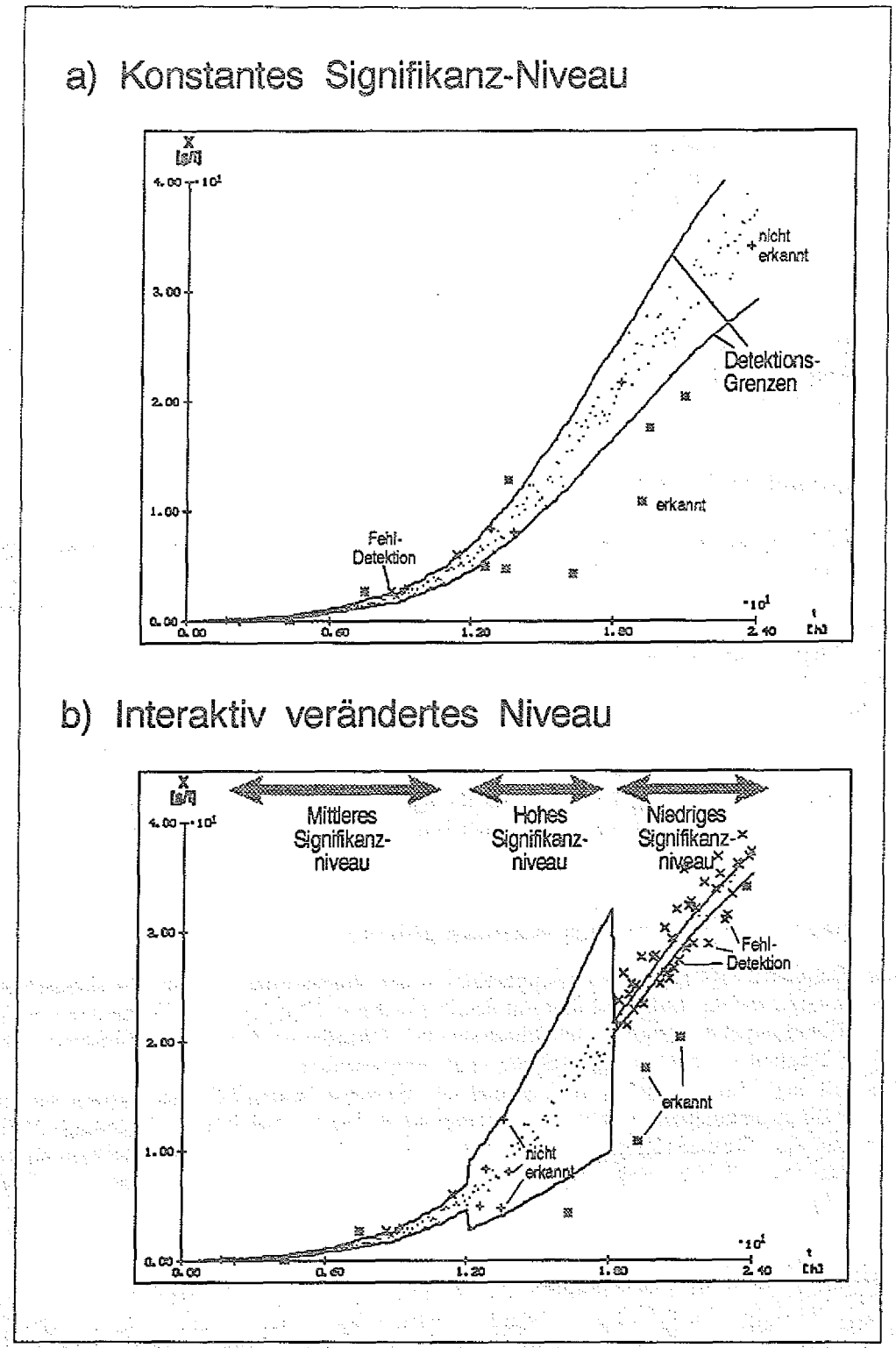


Abbildung 2.11: Durchlauf eines Aussreißer-Detektors für die Messung der Biomassekonzentration

Kalman-Filter : Wachstum
Signifikanztest : Ausreißer
Löschbaustein : Löschen

% Signalflüsse

Extern.M → Ausreißer.Meßwert
Wachstum.$\hat{X}$ → Ausreißer.Referenz
Extern.α → Ausreißer.Niveau
Ausreißer.Erkannt→ Löschen.Bedingung
Extern.M → Löschen.Eingang
Löschen.Ausgang → Wachstum.M

Auch die Konfiguration des Signalflusses ließe sich auf graphischem Wege mit einem CAD-ähnlichen System [8] durchführen. Einige Systeme zur Prozeßsimulation verfügen bereits über solche am Blockschaltbild orientierten Eingabemöglichkeiten [Sch90d].

2.5 Phasendetektion

Eine weitere Willküroperation beim graphischen Editieren der Daten war die Phasensetzung. Der Benutzer hat aufgrund des optischen Eindrucks und seiner Sachkenntnis die Einteilung in Lag-Phasen und exponentielle Phasen vorgenommen. Vielfach ist es auch am laufenden Prozeß sinnvoll, eine solche Phasenumschaltung vorzunehmen. So z.B. dann, wenn ein modellgestütztes Verfahren mit dem Beginn der exponentiellen Phase aufgesetzt werden soll, wobei die Länge der Lag-Phase nicht genau bekannt ist (vgl. [MP89]). Deshalb soll zum Abschluß am Beispieldatensatz D gezeigt werden, wie man das Ende der ersten Lag-Phase automatisch erkennen kann.

Ein intuitives Vorverständnis für die Sprungdetektion gibt die folgende Formulierung wieder:

> "Der Übergang von der ersten Lag-Phase in die exponentielle Phase ist durch einen sprunghaften Anstieg des Wachstums vom Nullwachstum zum exponentiellen Wachstum gekennzeichnet."

2.5.1 Umparametrisierung des Kalman-Filters

Die vorläufige Definition legt nahe, zur Sprungdetektion den Wachstumsratenschätzer einzusetzen. Die Abbildung 2.12a zeigt, wie das Kalman-Filter mit den in Abschnitt 2.3 gefundenen Parametern die Wachstumsrate beim Übergang in die exponentielle Phase schätzt. Offenbar ist dieser Wachstumsraten-Schätzer aufgrund seiner Trägheit für die Sprungdetektion völlig unbrauchbar.

Parametrisiert man dasselbe Filter so um, daß der Wachstumsraten-Schätzer extrem sensibel auf Änderungen der Biomassekonzentration reagiert, so ergibt sich das in Abbildung 2.12b gezeigte Verhalten. Dieser Schätzer ist zwar für eine ruhig verlaufende Schätzung der Wachstumsrate vollkommen ungeeignet, jedoch ist er in der Lage, dem Sprung (im Mittel) sehr schnell zu folgen. Man kann also versuchen, diese Information zur Sprungdetektion auszunutzen.

2.5.2 Kumulative-Summen-Tests

Die sogenannten Kumulativen Summen ermöglichen die Schätzung des Zeitpunkts einer sprunghaften Veränderung in einem Signal $y(t)$ und zugleich der Höhe des Sprungs. Zudem bilden sie ein Maß, mit dem getestet werden kann, ob der detektierte Sprung signifikant ist. In der einfachsten Version für die Erkennung eines Sprungs

[8] Computer Aided Design - rechnergestützte Erstellung technischer Zeichnungen

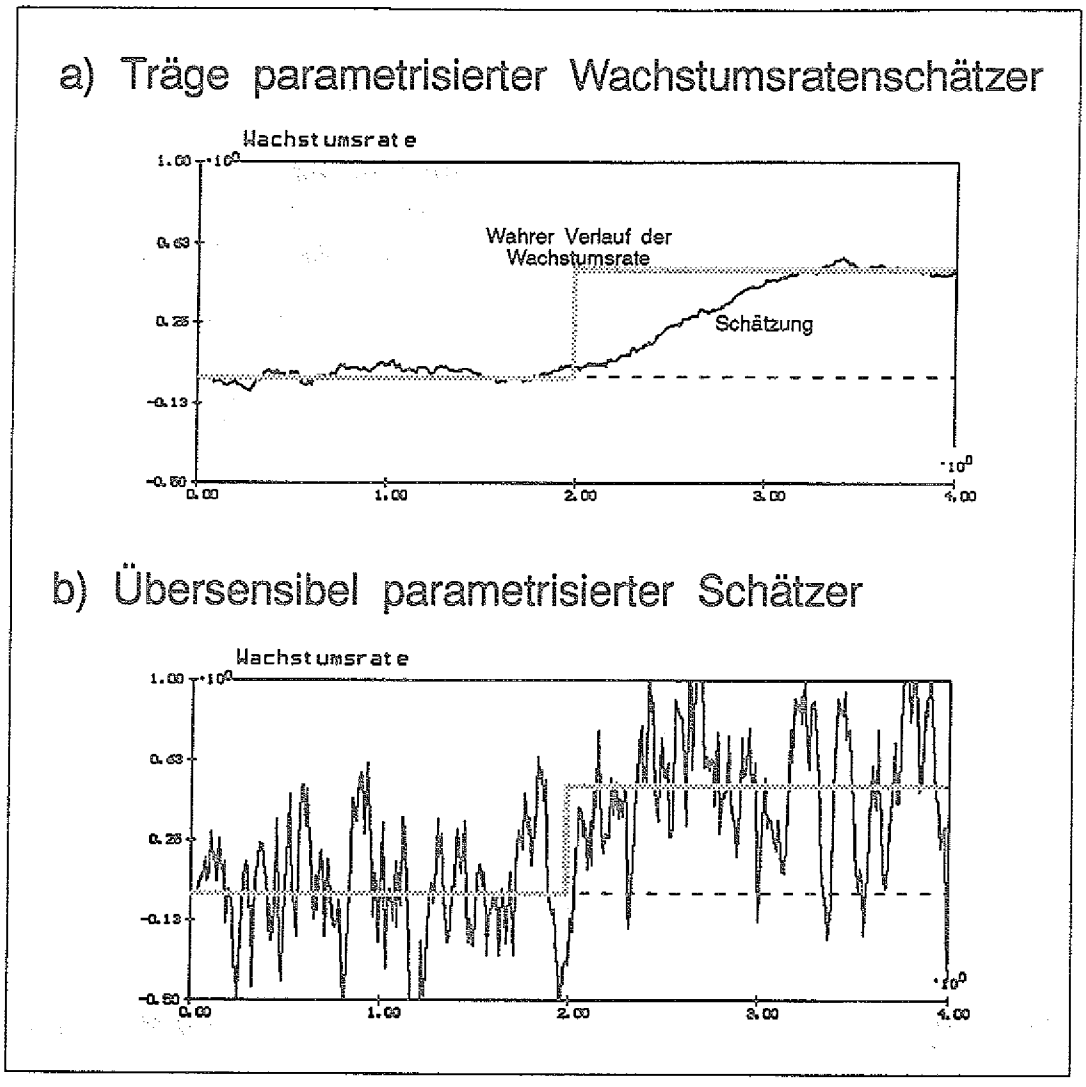


Abbildung 2.12: Träge und übersensibel parametrisiertes Kalman-Filter zur Wachstumsratenschätzung

im Signal	$y(t)$
vom Wert	$y_< = 0$
auf den Wert	$y_\geq = y_< + \Delta y$
zum Zeitpunkt	t_{step}

hat der Kumulative-Summen-Test die Form:

Eingehende Meßwerte:

$$y_i = y(i \cdot \Delta t),\ i = 1, ..., n$$

Schätzung der Sprunghöhe Δy, falls dieser zum Zeitpunkt $s \cdot \Delta t$ erfolgt ist:

$$\hat{\Delta y}(s) = \frac{1}{n - s + 1} \sum_{i=s}^{n} (y_i - y_<)$$

Kumulative Summe (Signifikanzmaß für einen Sprung zum Zeitpunkt $s \cdot \Delta t$):

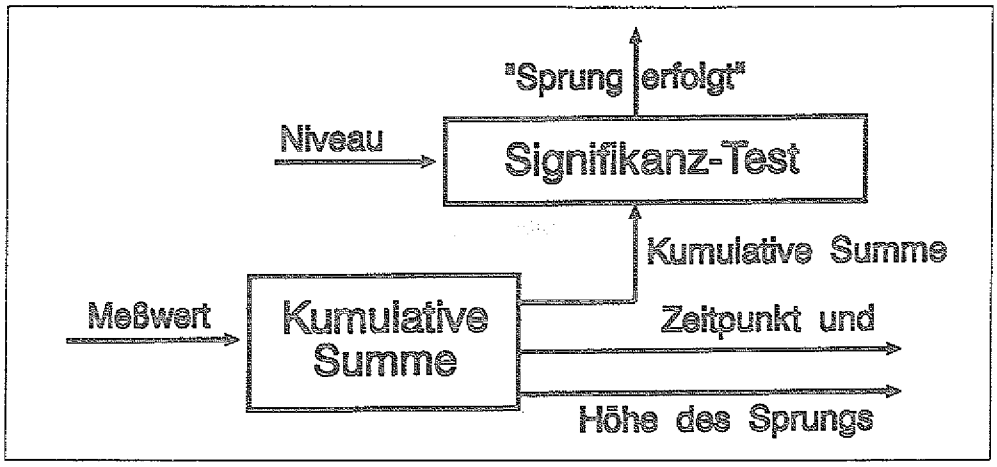


Abbildung 2.13: Datenflußschema für einen Sprung-Detektor

$$CS(s) = \hat{\Delta y}(s) \cdot \sum_{i=s}^{n} (y_i - y_< - \frac{1}{2}\hat{\Delta y}(s))$$

Schätzung des Sprungzeitpunkts $\hat{t}_{step} = \hat{s} \cdot \Delta t$*:*

$$\hat{s} = \arg \max_{s=1,\ldots,n} \hat{\Delta y}(s) \sum_{i=s}^{n} (y_i - y_< - \frac{1}{2}\hat{\Delta y}(s))$$

Schätzung der Sprunghöhe zum geschätzten Sprungzeitpunkt:

$$\hat{\Delta y} = \hat{\Delta y}(\hat{s})$$

Er wird auf der Basis eines Likelihood-Quotienten-Tests aus einem einfachen Sprung-Modell hergeleitet (vgl. dazu die detaillierteren Ausführungen in 10.2.4 und 10.5.3). Der Test, ob ein Sprung erfolgt ist, ist wieder ein Signifikanztest:

$$CS \stackrel{def}{=} CS(\hat{s}) > \lambda.$$

2.5.3 Datenflußschema und Verschaltung

Auch mit der Theorie zum Kumulative-Summen-Test muß der Anwender nicht belastet werden. Er läßt sich als Datenanalysebaustein mit Eingang $\hat{\mu}$ (Wachstumsraten-Schätzung) und Ausgängen $\hat{t}_{step}$ (Sprungzeitpunkt), $\Delta\hat{\mu}$ (Sprunghöhe) und CS (kumulative Summe) realisieren. Verbindet man diesen Baustein mit dem (übersensibel parametrisierten) Kalman-Filter für die Wachstumsraten-Schätzung und einem nachgeschalteten Signifikanztest zum Niveau α, so ist der Sprungdetektor komplett (Abb. 2.13).

Wird der Sprung dem Prozeßüberwachungssystem gemeldet, so kann dieses die Kontrollstrategie ändern, Modellwechsel vornehmen etc. Dabei ist die geschätzte Sprunghöhe $\hat{\mu}_{step}$ zugleich auch ein Anhaltspunkt für die Startwertwahl weiterer Bausteine (z.B. für ein träge parametrisiertes Kalman-Filter).

2.5.4 Durchlauf

Das Ergebnis eines Durchlaufs beim idealen Sprugmodell zeigt die Abbildung 2.14. Man erkennt, daß der Detektor (bei günstiger Parametrisierung) sehr schnell und zuverlässig den Sprungzeitpunkt erkennt. Der Detektionszeitpunkt liegt etwas hinter dem wahren Sprungzeitpunkt, da der Test erst die erforderliche Information aus dem System aufnehmen muß.

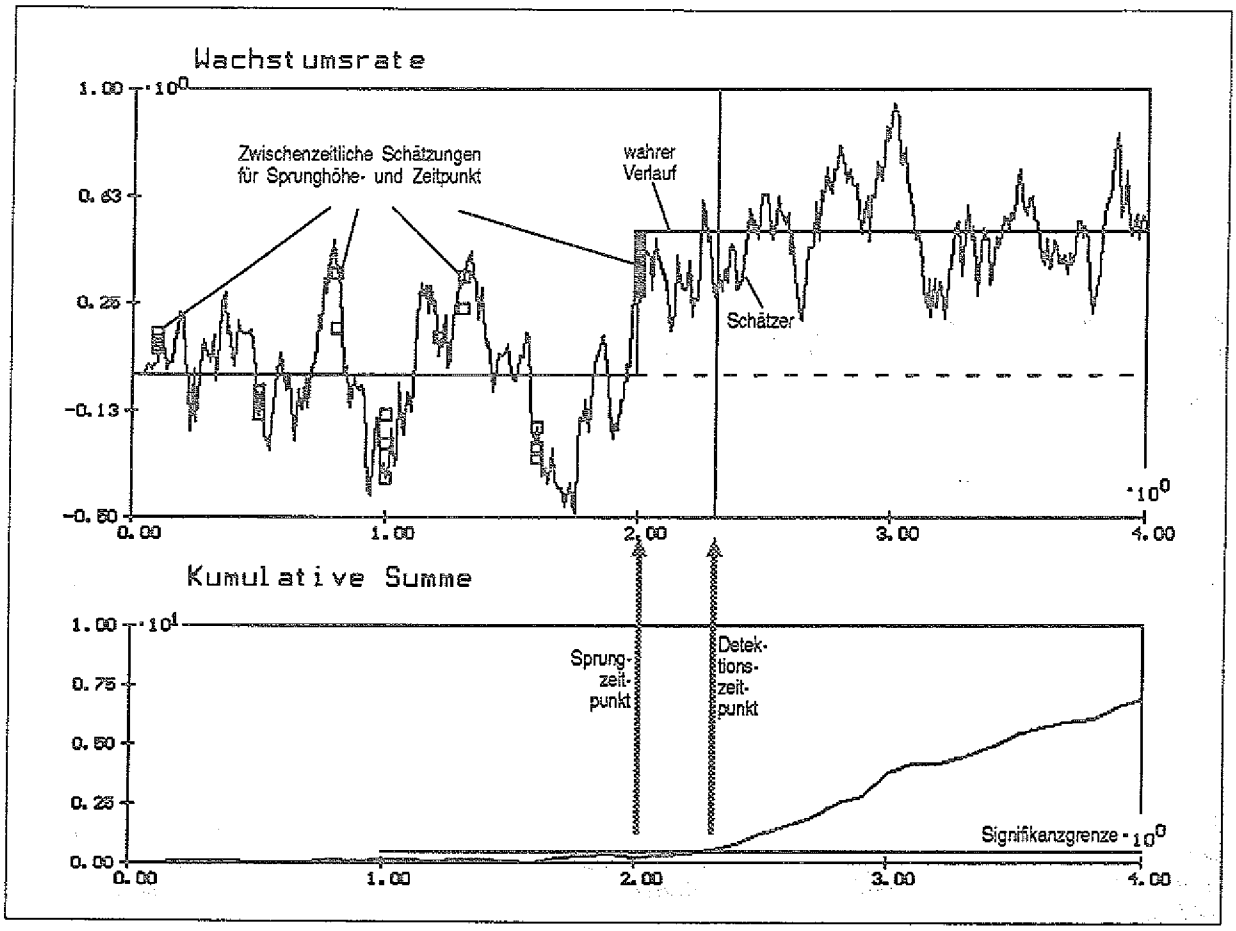


Abbildung 2.14: Durchlauf eines Sprung-Detektors zur Lag-Phasen-Erkennung

Die Abbildung enthält zudem den Zeitverlauf der kumulativen Summe. Hier kann schnell abgelesen werden, wie das Signifikanzniveau zu wählen ist, um

a) eine Frühdetektion zu unterbinden und gleichzeitig

b) eine vertretbar kurze Verzögerungszeit zu gewährleisten.

Das Prozeßüberwachungssystem muß in der Lage sein, bei Detektion des Sprunges eine Umkonfiguration rückwirkend zum Sprungzeitpunkt vorzunehmen — ein weiterer Grund, zu verlangen, daß alle Bausteine auch mit zeitverzögert eintreffenden Daten arbeiten können!

Kapitel 3

Konkretisierung des Datenanalysekonzeptes

Nach der Untersuchung der Datenlage und der Erstellung des Anforderungskatalogs in Kapitel 1 kann nun mit der systematischen Konzeption eines interaktiven Datenanalysesystems begonnen werden. Dabei ist das in Kapitel 2 gewonnene Vorverständnis hilfreich. Das Konzept wird ausgehend von den Grundbausteinen des Systems hin zur Gesamtablaufsteuerung entwickelt.

3.1 Ein Spektrum von Datenanalysemethoden

Beim Studium der biotechnischen Literatur, stößt man auf eine Fülle von Prozeßüberwachungsbausteinen, die von "klassischen" statistischen Methoden der Datenanalyse und Signalverarbeitung bis zu speziell auf die Biotechnologie zugeschnittenen modellgestützten Verfahren reicht. Hinzu kommen noch die verschiedenen graphischen Visualisierungstechniken (vgl. 3.4).

Für die Programmiertechnik stellt die Implementierung der modellgestützten Verfahren und der Visualisierungsmethoden das schwierigste Problem dar. Trotzdem muß angestrebt werden, auch die Standardverfahren in ein einheitliches Konzept zu integrieren. Die folgende Zusammenstellung von Methoden erhebt keinen Anspruch auf Vollständigkeit. Vielmehr soll sie die Vielfalt der Möglichkeiten und die damit verbundenen Ansprüche an die Computerimplementierung vor Augen führen:

3.1.1 Modellgestützte Verfahren

Hierunter sollen Methoden verstanden werden, die auf einem Differentialgleichungsmodell des betrachteten Systems beruhen. Im allgemeinen Fall ist dies ein stochastisches Zustandsraummodell der Form (vgl. Abb. 2.4):

$$\begin{aligned} \dot{x} &= f(\alpha, x, u, v_x) \quad \text{(Systemgleichung)} \\ y &= g(\alpha, x, \quad v_y) \quad \text{(Meßgleichung)} \end{aligned}$$

mit	dem Systemzustand	x
	den Steuergrößen	u
	den Meßgrößen	y
	dem Systemrauschen	v_x
	dem Meßrauschen	v_y
und	den Systemparametern	α

Die modellgestützten Methoden lassen sich wie folgt aufschlüsseln:

3.1.1.1 Deterministische Simulation

Simulationen sind zwar keine direkten Datenanalysemethoden, sie sind jedoch insofern für den Informationsgewinn wichtig, als sie das Verhalten eines Modellsystems und die Einflüsse der Parameter deutlich machen. Viele Modellansätze können schon im Vorfeld der Datenanalyse aufgrund von rein qualitativen

Verhaltensweisen (z.B. Instabilität oder Insensitivität bzgl. relevanter Parameter) ausgegrenzt werden. Ferner wird die deterministische Simulation zum Test der Stabilität von Zustandsbeobachtern unbedingt benötigt (vgl. 9.1).

3.1.1.2 Stochastische Simulation

Vor allem die modellgestützten Filter basieren auf stochastischen Differentialgleichungen mit Rauschtermen. Da stochastische Differentialgleichungen nicht zum Grundwissen eines Praktikers in der Biotechnologie gehören, muß die Bedeutung von Parametern (wie z.B. Erwartungswerten und Varianzen von stochastischen Zustandsvariablen) mit Hilfe von Simulationen verdeutlicht werden. Des weiteren benötigt man die stochastische Simulation zur Aussonderung unbrauchbarer Modellansätze und zum Test von Filterbausteinen (vgl. 10.1 und 11.1).

3.1.1.3 Kurvenanpassung

Methoden zur Parameteranpassung gehören in vielen Bereichen zum Standard. Das Problem des Auffindens optimaler Parameter bzgl. eines Minimierungskriteriums bleibt dabei weiterhin Gegenstand der Forschung (vgl. 10.2). Insbesondere bei den nichtlinearen Systemen zeichnen sich erst in den letzten Jahren grundsätzliche Neuentwicklungen ab [BS86b]. Trotzdem ist die Anwendung der Kurvenanpassung in der Biotechnologie mit Vorsicht zu betrachten, da prinzipiell keine stochastischen Systeme mit Systemunschärfen behandelt werden können (vgl. dazu 10.3). Ferner ist zu beachten, daß es sich zunächst um eine Off-Line-Technik handelt. Erweiterungen auf den On-Line-Einsatz befinden sich in der Erprobungsphase [JJM86] [Gol89] [MP89].

3.1.1.4 Zustandsbeobachter

Zu den vielversprechendsten Methoden der Prozeßüberwachung gehören die Zustandsbeobachter [Lue64], da sie es gestatten, aufgrund einer Modellvorstellung aus Meßgrößen andere relevante, aber nicht direkt meßbare Größen abzuleiten (z.B. die Substratkonzentration aus der Biomassekonzentration und den Gasanalysewerten). Für die Biotechnologie sind vor allem die nichtlinearen Zustandsbeobachter von Bedeutung, deren Stabilität jedoch auf dem Wege einer Simulation getestet werden sollte (siehe Kapitel 9.2).

3.1.1.5 Modellgestützte Filter

Der prominenteste Verteter ist das Kalman-Filter (vgl. 11.3) [Kal60], das rein formal gesehen ein Zustandsbeobachter mit einer nichtlinearen und zeitabhängigen Feedback-Beziehung ist. Die zugrunde liegende Theorie führt jedoch in das Gebiet der stochastischen Differentialgleichungen. Dabei werden prinzipiell nicht modellierbare Rauscheinflüsse auf das System berücksichtigt. Obwohl dieser Ansatz dem Praktiker sehr nahe stehen dürfte, liegt die Hemmschwelle für das Erlernen dieses Verfahrens recht hoch (vgl. auch 2.3.2). Die didaktische Aufbereitung und Nutzbarmachung modellgestützter Filter ist eines der Hauptanliegen dieser Arbeit.

3.1.2 Lineare Systemtheorie

Lineare und zeitabhängig lineare Systemmodelle sind spezielle Zustandsraummodelle der Gestalt:

$$\begin{aligned} \dot{x} &= Ax + Bu + Rv_x \quad \text{(Systemgleichung)} \\ y &= Cx + Sv_y \quad \text{(Meßgleichung)} \end{aligned}$$

mit Matrizen A, B, C, R, S, die von Parametern abhängen können. Man spricht oft auch von Black-Box-Modellen, da sie oft keine detaillierte Systemkenntnis benötigen.

Da sich biologische Systeme im allgemeinen nichtlinear verhalten, ist eine Anwendung von linearen Modellen nur in einem Arbeitsbereich zulässig, in dem sich das System annähernd linear verhält. Zudem ist zu beachten, daß selbst bei einem gelungenen Einsatz eines Black-Box-Verfahrens oft nur auf kompliziertem Wege von den Parametern des Black-Box-Modells auf biologisch relevante Parameter geschlossen werden kann [Eyk90]. Man kann grob zwei lineare Techniken unterscheiden:

3.1.2.1 Lineare Filter und Beobachter

Sie unterscheiden sich nur durch die Art des zugrunde liegenden Modells von den oben beschriebenen Filtern und Beobachtern (vgl. 11.2). Allerdings lassen sich die linearen Filter und Beobachter oft numerisch effizienter realisieren, was vor allem zu einer großen Verbreitung bei der Überwachung technischer Prozesse mit schneller Dynamik geführt hat [LS83] [Lju87] [Ise87].

3.1.2.2 Pole-Placement-Techniken

Beim Pole-Placement geht es um die systematische Beeinflussung der Stabilität eines linearen Systems durch die Veränderung der Lage der Eigenwerte in der komplexen Ebene [BC85] [Göl87]. Diese Technik geht mit einer Transformation des Systemmodells in den Frequenzbereich einher. Im Frequenzbereich gewinnen die Pole dann oft eine konkrete Interpretation für das Systemverhalten [Göl87].

3.1.3 Regler und Steuerungen

Methoden zur modellgestützten Prozeßkontrolle scheinen in einem Datenanalysesystem zunächst fehl am Platze zu sein. Bei genauerer Betrachtung zeigt sich jedoch, daß Regler und Steuerungen meist genau auf den Modellen aufsetzen, die zuvor im Rahmen einer Datenanalyse erstellt wurden (vgl. [BC85]). Simulationsläufe lassen sich innerhalb des Datenanalysesystems durchführen, weil entsprechende Simulationskomponenten ohnehin vorhanden sein müssen.

3.1.3.1 Dualität zu den Beobachtern und Filtern

Es soll auf die Dualität zwischen Reglern und Beobachtern hingewiesen werden: Jedem Regler entspricht über den Dualitätssatz der Kontrolltheorie [BC85] unmittelbar ein Beobachter und umgekehrt. Ein ähnlicher Satz gilt für die optimale Steuerung und die Filterung.

Dabei ist die Dualität dergestalt, daß für einen Regler dieselben Verfahren benötigt werden wie für den dualen Beobachter. Folglich ist nach Programmiererung eines Beobachters zugleich auch die Hauptarbeit für die Umsetzung des dualen Reglers geleistet worden. Insbesondere entsprechen sich:

- der allgemeine lineare Beobachter und der allgemeine lineare Regler [1].
- das lineare Filter und der lineare stochastische Regler.
- das Kalman-Filter und der Minimal-Varianz-Regler.

3.1.3.2 Optimale Kontrolle

Eine Bemerkung zur Anwendung der Theorie der optimalen Kontrolle [MS82] scheint angebracht: Solche Techniken lassen sich erst dann anwenden, wenn ein gutes Modell für das betrachtete System vorliegt. D.h. an eine optimale Kontrolle (im strengen Sinne) ist erst nach Abschluß der Systemidentifikation zu denken. Sie gehört nicht mehr in den Anwendungsbereich der Datenanalyse.

3.1.4 Nichtparametrische Methoden

Die nichtparametrischen Methoden der Kurvenglättung und -differentiation gehören zum Standardumfang jedes Statistik-Programmpakets. Sie helfen bei der Visualisierung von Prozeßdaten und geben Hinweise für erste Ansätze zur Modellierung von Systemen, bei denen wenig Vorinformation vorliegt. Dazu gehören:

3.1.4.1 Ausgleichsverfahren und glättende Splines

Hierbei werden global (bei den Ausgleichsverfahren) oder stückweise (bei den glättenden Splines) definierte Ausgleichskurven einfachen Typs (meist Polynome) an die Meßreihen angepaßt, wozu es in der Regel geschlossene algebraische Lösungen gibt [WS79] [EMR90]. Im deterministischen Grenzfall hat man Interpolationsfunktionen und (gewöhnliche) Splines.

[1] Ein "einfacher" Spezialfall des linearen Reglers ist der bekannte PID-Regler [Ise87].

3.1.4.2 Fensterschätzer

Ein Fensterschätzer bewirkt eine Kurvenglättung und -differentiation durch Faltung mit einem geeigneten Integralkern (dem "Fenster") [GM84] [GMM84]. Im Gegensatz zu den glättenden Splines sind die Fensterschätzer theoretisch einfacher zu analysieren, so daß konkrete Anhaltspunkte für eine geeignete Parametrisierung vorliegen.

3.1.4.3 Frequenzanalytische Methoden

Sie beruhen stets auf einer Fourier-Transformation des Signals [Pri81] [PFTV88]. Durch Wegfiltern hoher Frequenzanteile und anschließende Rücktransformation kann oft eine sehr gute Glättung erreicht werden. Auf der Basis des Frequenzspektrums lassen sich schließlich brauchbare lineare Filter für den On-Line-Einsatz konstuieren.

Bei biotechnischen Systemen treten oft periodische Störungen auf (z.B. durch eine schlechte Regelung von Pumpen). Deren Frequenz könnte mit Spektralmethoden identifiziert und damit die Fehlerquelle aufgefunden werden.

3.1.5 Tests und Klassifikationsverfahren

Statistische Tests und Klassifikatoren benutzt man zur Entscheidung, ob eine vorliegende Beobachtung einer bestimmten Kategorie (z.B. Bereichsüberschreitung, Ausreißer oder Störung) angehört.

3.1.5.1 Ein- und Mehrfachhypothesentests

Mit einem Test kann mit vorgegebener Irrtumswahrscheinlichkeit festgestellt werden, ob eine gegebene Beobachtung eine bestimmte Hypothese stützt oder nicht [Sac84]. Tests werden im allgemeinen aus konkreten Annahmen für die Verteilung der zufälligen Größen konstruiert, sind also theoretisch begründet (vgl. 10.5).

3.1.5.2 Klassifikatoren

Klassifikatoren dienen ebenfalls der Einordnung von Beobachtungen in eine oder mehrere Klassen [Nie83]. Im Unterschied zu den Testverfahren werden sie meist auf der Grundlage einer Reihe von Lernstichproben heuristisch konstruiert (Mustererkennung). Ein Klassifikator bietet den Vorteil der Lernfähigkeit und damit die Möglichkeit zur Anpassung an veränderliche reale Begebenheiten (vgl. 10.7).

3.1.6 Schätzverfahren

Vielen Tests liegt eine vorangehende Schätzung von Zufallsvariablen aufgrund von Meßdaten zugrunde. Hier kommt die statistische Schätztheorie zur Anwendung, bei der es um die optimale Schätzung von Systemgrößen aus verrauschten Meßdaten mit vorgegebenen Wahrscheinlichkeitsverteilungen geht [Sac84]. Schätzer sind meist auf spezielle Anwendungen zugeschnitten.

3.1.6.1 Schätzer für Erwartungswerte, Varianzen und Verteilungen

Diese Schätzer gehören zum Standardumfang eines Datenanalysesystems. Aus der Zeitreihenanalyse sollten auch Schätzer für die Auto- und Kreuz-Kovarianzen von Signalen berücksichtigt werden [Pri81].

3.1.6.2 Lineare Schätzer

Lineare Modelle (vgl. 6.5.1.1, [FH84]) treten in der Biotechnologie stets im Zusammenhang mit der Stöchiometrie des Systems auf (vgl. 6.6) [Fei77]. Eingangsdaten sind die Stoff- und Elementflüsse am Ein- und Ausgang des Systems; der Reaktionsmechanismus legt dazu lineare Nebenbedingungen fest. Auf dieser Grundlage können nicht gemessene Komponenten geschätzt und die Konsistenz der Meßwerte geprüft werden (vgl. 10.2).

Insbesondere können überbestimmte Gleichungen vorkommen. Die darin enthaltene Redundanz kann zur Verbesserung der Schätzung, aber auch zur Auffindung fehlerhafter Meßwerte benutzt werden. Sowohl im stationären als auch im instationären Betrieb bilden diese Techniken die Grundlage für die Stoffbilanzierung.

3.1.6.3 Hauptkomponentenanalyse

Selbst für den Fall, daß ein Reaktionsmechanismus nicht oder nur teilweise bekannt ist, kann versucht werden, aus einem Satz von Meßergebnissen einen sogenannten "empirischen" Reaktionsraum zu bestimmen [Lia89], der die durch die (unbekannte) Stöchiometrie gegebenen Nebenbedingungen widergibt. Dazu wird eine Hauptkomponentenanalyse [Übe71] durchgeführt, mit der man herausfinden kann, ob eine Punktwolke in einem höherdimensionalen Raum flächig angeordnet ist (vgl. Kapitel 10.7).

3.1.6.4 Kumulative Summen

Der Detektion einer sprunghaften Veränderung der Systemdynamik geht meist die Schätzung eines Maßes für die Sprunghaftigkeit voraus. Die bekanntesten Schätzer stellen die sogenannten Kumulativen Summen dar [BB86], die ein Maß für die Wahrscheinlichkeit eines erfolgten Sprungs angeben (vgl. 10.5.3).

3.1.7 Transformationen und Algorithmen

Zu den elementaren Operationen mit Meßreihen gehören Transformationen und wechselseitige Verknüpfungen des Typs:

$$y = f(x_1, \ldots, x_n)$$

3.1.7.1 Transformationen

Eine allgemeine Transformation berechnet eine Verknüpfung verschiedener Meßdaten (z.B. die Biomassekonzentration aus einer Messung der optischen Dichte und einer vorgegebenen Eichkurve). Die Rechenvorschrift ist in einer Formelschreibweise anzugeben.

3.1.7.2 Algorithmen

Kompliziertere funktionale Zusammenhänge (z.B. Iterationen und Fallunterscheidungen) zwischen Meßwerten können besser in Form von kurzen Algorithmen formuliert werden. Dazu kann eine einfache Programmiersprache (vgl. C-- in 18.1) verwendet werden.

3.1.8 Visualisierungsmethoden

Die Visualisierung bildet die Schnittstelle zwischen System und Benutzer. Sie erweist sich formal als eine spezielle Art von Prozeßüberwachungs-Baustein, der den Datenfluß zwischen Mensch und Maschine regelt ("Man Machine Interface"). Die interaktive Computergraphik hat eine Fülle von Möglichkeiten entwickelt, die in Abschnitt 3.4 und Kapitel 13 zusammengestellt sind.

3.2 Das Modularisierungsschema für einen Datenanalysebaustein

Sucht man bei der Fülle der unter 3.1 erwähnten Methoden nach Gemeinsamkeiten, die eine einheitliche Umsetzung in ein Programm ermöglichen, so scheint der kleinste gemeinsame Nenner zunächst nur in dem Datenflußschema bestehend aus

- Eingangs-Daten,
- Ausgangs-Daten und

- Konfigurationsdaten

zu bestehen. Der eigentliche Kern stellt jeweils ganz verschiedene Anforderungen. Bei näherer Betrachtung wird allerdings deutlich, daß viele Bestandteile wie z.B. Visualisierungsmethoden, numerische Komponenten, Techniken der Datenspeicherung, Gestaltung von Anwenderoberflächen und Konfigurationsdateien in wechselnden Kombinationen immer wieder auftreten. Dies führt zur Bereitstellung einer Kollektion von Elementarmethoden [2], aus denen jeder Datenanalysebaustein zusammengesetzt werden kann.

Um die Frage zu beantworten, welche Elementarmethoden benötigt werden, ist es sinnvoll, ein allgemeines Schema für die Realisierung von Methoden zu entwickeln. Das Modularisierungsschema aus Abbildung 3.1 unterscheidet die Strukturelemente

- Theorie und Methodik,
- Konfiguration,
- Compiler,
- Benutzerführung,
- Visualisierung,
- Numerik,
- Datenfluß.

Es stellt einen Leitfaden für den systematischen Aufbau neuer *Datenanalysebausteine* dar und gleichzeitig einen Anforderungskatalog für die benötigten Elementarmethoden.

Bemerkung:3 Nicht in den Aufgabenbereich eines einzelnen Datenanalysebausteins gehören Probleme der

- Datenablage und -archivierung,
- Speicherverwaltung,
- Ablaufsteuerung und
- Steuerung des Benutzerdialogs.

Diese Aufgaben werden von der Gesamt-Ablaufsteuerung des Systems übernommen (siehe dazu 3.5 und 4).

Die einzelnen Komponenten des Modularisierungsschemas sollen etwas eingehender erläutert werden:

3.2.1 Theorie und Methodik

Zunächst wird eine Methode und ihre theoretische Grundlage nach folgenden Gesichtspunkten analysiert und gegebenenfalls modifiziert:

- Eine möglichst allgemeine Klasse von Eingangsdaten soll akzeptiert werden (z.B. nicht äquidistante Abtastpunkte, Rauschen, Zeitverzögerung, fehlende Werte).
- Die On-Line-Einsetzbarkeit sollte ermöglicht werden.
- Das Verfahren sollte insbesondere iterativ ablaufen können, um eine Synchronisation von außen und damit die Einbindung in ein Netzwerk von Datenanalysebausteinen zu ermöglichen.
- Möglichst große Teile der analytischen Arbeit (wie z.B. Differenzieren) sollten automatisierbar sein.
- Dies impliziert eine saubere didaktische Aufbereitung der Theorie, um die Anwenderfreundlichkeit zu steigern.

[2] Der Programmierer spricht von sogenannten *Tools*, die in *Toolboxen* zusammengefaßt sind.

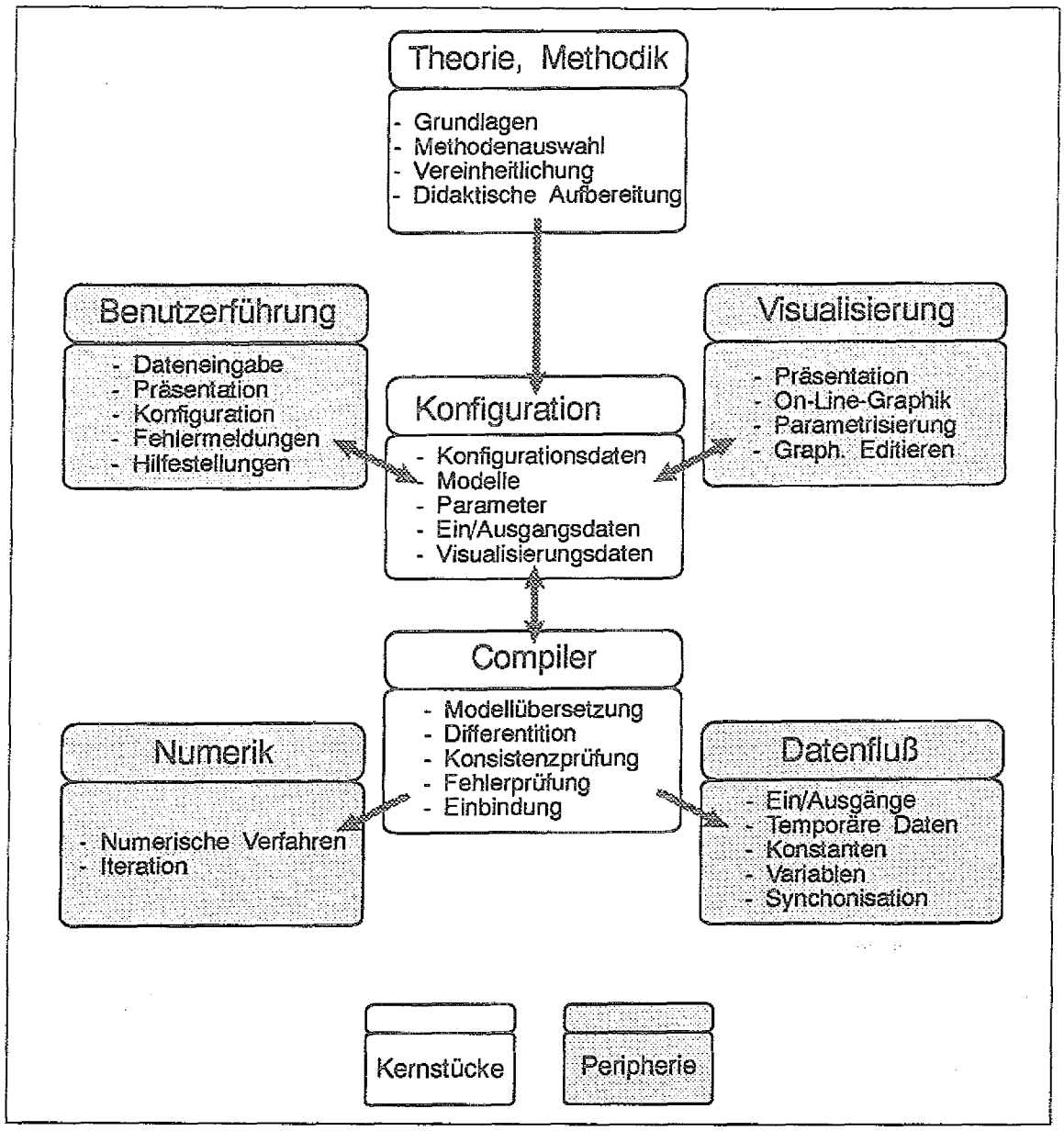


Abbildung 3.1: Modularisierungsschema für einen Datenanalysebaustein

3.2.2 Konfiguration

Die *Konfiguration* eines Bausteins ist die Kollektion aller Angaben, die zu seiner Initialisierung benötigt werden. Dazu gehören:

- alle Einstellparameter und Dimensionierungsangaben für die auftretenden Variablen [3],
- eine genaue Spezifikation der Datentypen von Ein- und Ausgängen,
- gegebenenfalls Modelle oder ganze Lerndatensätze,
- Angaben über assoziierte Visualisierungstechniken (vgl. die Bemerkungen zur Visualisierung in 3.2.5).

[3] Im regelungstechnischen Umfeld spricht man von der *Parametrisierung*.

Alle diese Angaben sind in einer geeigneten formalen Notation in einer *Konfigurationsdatei* abzulegen. Die Konfiguration ist somit die "Seele" eines Datenanalysebausteins, welcher ohne diese nur eine leere Hülle wäre. Es sollten möglichst viele Daten vom statischen Programmcode in die dynamisch veränderbare Konfigurationsdatei verlagert werden. Auf diese Weise entsteht auf der Basis ein und desselben Programmcodes mit der Zeit eine flexible Bibliothek von Datenanalysebausteinen.

Aber auch die in Textform zu notierenden Einträge in der Konfiguration sollten nicht statischer Natur sein, sondern können selbst wieder Anweisungen sein, die erst zum Zeitpunkt der Initialisierung umgesetzt werden können (z.B. Angaben über die geeignete Wahl von Startwerten auf der Basis des bisherigen Prozeßverlaufs). Zur Formulierung solcher Anweisungen ist eine geeignete Metasprache zu definieren.

3.2.3 Compiler

Wenn die Konfiguration die "Seele" einer Methode darstellt, so ist der Compiler der "Kopf", da er dafür sorgt, daß die in der Konfiguration abgelegten Informationen zur Ausführung kommen. Aufgaben des Compilers sind

- die Prüfung der Konfigurationsdatei in Hinblick auf syntaktische Fehler und Inkonsistenzen.
- die Bereitstellung von methodenspezifischen Hilfestellungen.
- die Übersetzung metasprachlicher Ausdrücke in eine ausführbare Form [4].
- insbesondere die Formelübersetzung und -differentiation [5].
- die Einbindung des Datenanalysebausteins in den Datenfluß, wozu die Anforderung von Speicherplatz bei der Verwaltung des Systems gehört.

Wenn hier die Möglichkeiten des Compilerbaus (vgl. Kapitel 18, [ASU88]) nicht voll ausgeschöpft werden, bleiben die tatsächlichen Möglichkeiten des Datenanalysesystems gegenüber den Anforderungen stets zurück.

3.2.4 Benutzerführung

Die Benutzerführung kann mit Hilfe einer graphischen Standardoberfläche erfolgen. Dabei müssen alle Konfigurationsdaten auch für den Benutzer transparent sein. Zu einer professionellen Oberfläche gehören [HDF+85]:

- dynamische Strukturen, die interaktiv veränderbar sind,
- die Einhaltung von Querbeziehungen zwischen den visualisierten Objekten (*Constraint Propagating*),
- kontextsensitive Hilfestellungen,
- Dokumentationsvorrichtungen.

Wenn es auch nicht die Aufgabe einer Forschungsarbeit sein kann, ein professionelles System zur Datenanalyse zu erstellen, welches diesen Anforderungen genügt, so ist es doch sinnvoll, diese Probleme zumindest auf dem Papier anzugehen, weil Inkonsistenzen in der theoretischen und didaktischen Aufbereitung neuer Methoden hierbei deutlich hervortreten.

[4] Ideal wäre die Definition einer kleinen Programmiersprache (vgl. C-- in 18.1)

[5] Die Übersetzung der Formeln sollte nach Möglichkeit sogleich in den Maschinencode erfolgen, da die Formelauswertung iterativ erfolgt und daher zeitkritisch ist.

3.2.5 Visualisierung

Bei Verwendung bestimmter Methoden liegen oft bestimmte Formen der graphischen Darstellung nahe (z.B. sollte bei einem Kalman-Filter die Möglichkeit zur Darstellung von Fehlerbalken möglich sein oder bei einem Klassifikator die Visualisierung des Merkmalsraums). Streng genommen sind Visualisierungsmethoden selbst nur Datenanalysebausteine, die sich formal nicht von den anderen Komponenten unterscheiden, jedoch wird der Bedarf einer Visualisierung oft erst nach dem Entwurf der eigentlichen Methoden klar.

Jeder Datenanalysebaustein kann Methoden aus einer Kollektion von Visualisierungstechniken verwenden, wie z.B.

- verschiedene Darstellungen in Koordinatensystemen (Meßpunkte, Verbindungslinien, Fehlerbalken),
- die Darstellung von Phasenräumen,
- die Darstellung von Histogrammen und Verteilungen,
- die Darstellung von Punktwolken,
- graphische Parametereingabe,
- Abtastmethoden,
- Auslöschfunktionen für Ausreißer.

3.2.6 Numerik

Die Numerik schließlich ist das "Herz" eines Verfahrens, da sie die eigentliche Arbeit ausführt. Sie stellt allerdings programmtechnisch das kleinste Problem dar, da auf diesem Bereich bereits viele Verfahren vorkonfektioniert sind. Aus Effizienzgründen sollte soweit wie möglich auf diese Programm-Bibliotheken zurückgegriffen werden. Zu beachten ist dazu:

- daß die Datenrepräsentation "konservativ" gehalten wird, d.h. daß Vektoren, Matrizen und Datenreihen als sequentiell im Speicher abgelegte numerische Werte realisiert werden.
- daß zeitintensive Teile wie z.B. die Auswertung von Formeln in zeitoptimierten Code (i.a. Maschinencode) umzusetzen sind.

3.2.7 Datenfluß

Die Fähigkeit der Datenanalysebausteine, untereinander zu kommunizieren, stellt die eigentliche Stärke des Systems dar. Deshalb muß versucht werden, für eine möglichst allgemeine Verwendbarkeit zu sorgen. Dies betrifft:

- den Eingangsdatenfluß,
- den Ausgangsdatenfluß,
- den Konfigurationsdatenfluß (auch während der Laufzeit),
- die Synchronisation verschiedener Methoden.

Diese Aufgaben werden nicht von einem Baustein selbst wahrgenommen, sondern von der Datenflußverwaltung des Gesamtsystems. Trotzdem muß jeder einzelne Baustein in der Lage sein, mit der Verwaltungseinheit zu kommunizieren. Dazu wird ein Protokoll vorgegeben, das bausteinspezifisch zu implementieren ist.

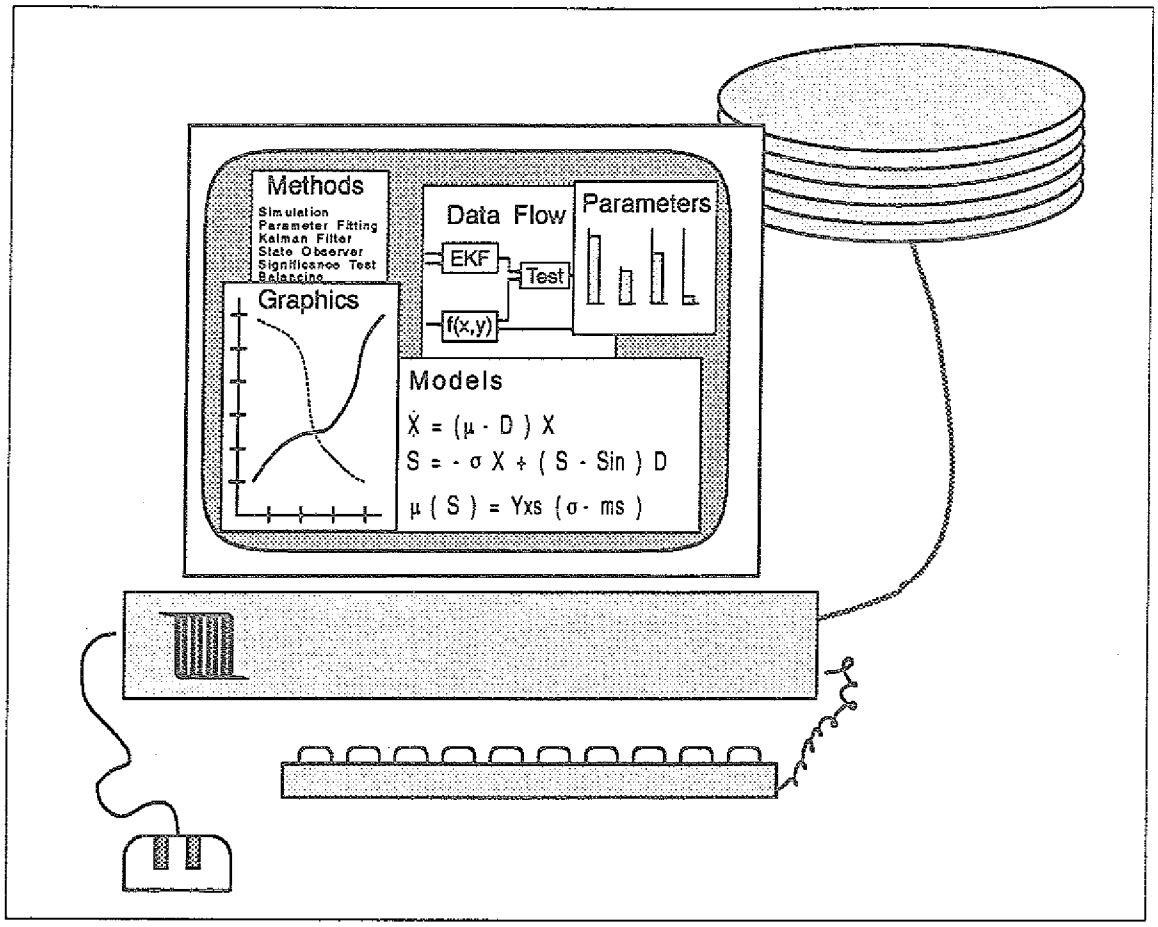


Abbildung 3.2: Oberfläche eines interaktiven Datenanalysesystems

3.3 Das Arbeiten mit einem interaktiven Datenanalysesystem

Es stellt sich nun die Frage, wie die einzelnen Bausteinen innerhalb des Gesamtsystems organisiert sind und vom Benutzer frei verschaltet werden können. Die Anforderungen aus der Sicht des Benutzers sollen noch einmal kurz zusammengefaßt werden:

3.3.1 Funktionalität

Unter der Oberfläche des Datenanalysesystems (vgl. Abb. 3.2) vereinen sich

- die Bibliothek der Datenanalysemethoden,
- die Bibliothek der Visualisierungsmethoden,
- die Datenbasis und
- die Ablaufsteuerung.

Die Funktionalität des Systems ist durch folgende Eigenschaften gekennzeichnet:

- interaktive Steuerung in allen Teilen,
- extensive Verwendung von Visualisierungsmethoden,
- intelligente Abwicklung von Routineaufgaben der Verwaltung,

- weitgehende Automatisierung komplexer Abläufe,
- Bedienung auf einer hohen metasprachlichen Ebene,
- kontextsensitive Hilfestellung,
- Erweiterbarkeit bzgl. des Methodenangebotes (offene Architektur),
- Lernfähigkeit in bezug auf vergangene Datenanalysesitzungen.

Dies erlaubt ein kreatives und exploratives Arbeiten mit dem Datenmaterial mit Methoden, die an die menschlichen Fähigkeiten möglichst gut angepaßt sind (vgl. hierzu auch 3.4).

3.3.2 Analogie zu einem Werkzeugkasten

Die Art des Arbeitens mit dem Datenanalysesystem soll anhand der Analogie zum Werkzeugkasten eines Handwerkers, der für ein recht breites Aufgabenfeld gerüstet sein muß, verdeutlicht werden: Bei der Zusammenstellung des Werkzeugkastens kommt es darauf an, daß die Werkzeuge

- nach funktionalem Gesichtspunkt ausgewählt werden, d.h. alle Aufgaben müssen sich mit einer Kombination der elementaren Hilfsmittel lösen lassen (z.B. wird stets ein Hammer gebraucht aber eine Schraubzwinge kann fehlen, wenn eine Gripzange vorhanden ist).
- wohlsortiert sind, d.h. nach Größe und Anwendungsbereichen, so daß mit einem Griff das passende Werkzeug zur Hand ist (z.B. sind Schraubendreher, Maulschlüssel und Ringschlüssel gestaffelt und an separaten Plätzen angeordnet).
- kombinierbar sind, d.h. es wird auf maximale Flexibilität geachtet (z.B. kann eine Bohrmaschine mit diversen Vorsätzen ausgerüstet werden).
- Routinearbeiten erleichtern, d.h. immer wiederkehrende Abläufe werden automatisiert (z.B. ist ein Akkuschrauber vorhanden).

So ausgerüstet kommt es auf das Improvisationstalent und die Geschicklichkeit des Handwerkers an, was er mit dem Werkzeugkasten macht. So können z.B. Lastwagenfahrer in Entwicklungsländern oft mit Hilfe eines Hammers, eines Schraubendrehers und einer verstellbaren Zange Reparaturarbeiten erledigen, die hierzulande nur in wohlausgestatteten Werkstätten ausgeführt werden (allerdings mit weit größerer Zeiteffizienz!).

Ganz allgemein findet man eine Schere zwischen Einfachheit, kurzer Anlernphase und hohem Zeitaufwand zum Lösen der gestellten Aufgaben einerseits sowie Kompliziertheit, langer Einarbeitungszeit und effektiver Problemlösung andererseits. Dazwischen ist ein Mittelweg zu finden: Ein gutes System ist schrittweise erlernbar ist und bietet neue Werkzeuge erst mit der Zeit an.

3.4 Interaktive Computergraphik und Explorative Datenanalyse

Obwohl die interaktiven graphischen Methoden eine zentrale Stellung innerhalb des Datenanalysekonzepts haben, sind sie nicht durch ein mathematisches oder formales Modell motiviert. Die Grundlage für ihre Anwendung ist vielmehr die durch Erfahrung begründete Erkenntnis, daß das Auge unser leistungsfähigstes Organ zur schnellen Erfassung komplexer Strukturen ist! Dies begründet die Suche nach Techniken zur Datenpräsentation, die der visuellen Wahrnehmung besonders entgegenkommen (vgl. auch Abb. 1.5). Ein Exkurs in die Forschungsgebiete der interaktiven und Explorativen Datenanalyse ist deshalb angebracht. Die dort etablierten Techniken wurden auf ihre Tauglichkeit im biotechnischen Umfeld geprüft oder dienten als Anregung für neue Entwürfe.

3.4.1 Interaktive Computergraphik

Die *interaktive Computergraphik* beschäftigt sich mit dem Entwurf interaktiver graphischer Mensch-Maschine-Schnittstellen für die Darstellung von Datenmaterial [BHM+88] [Gro81]. In einem gewissen Gegensatz dazu steht eine vorkonfektionierte graphische Datenpräsentation und auf jeden Fall die nichtvisuelle Darstellung.
Problemstellungen der interaktiven Computergraphik sind:

- die Visualisierung von Datensätzen,
- die graphische Präsentation und Hervorhebung von wesentlichen Eigenschaften der Daten,
- die Untersuchung der menschlichen Wahrnehmung in bezug auf das präsentierte Material,
- die Entwicklung interaktiver graphischer Techniken zur Dateneingabe und Steuerung des Bildaufbaus (Layout, Raumanordnung),
- die interaktive graphische Veränderung von Daten (Ausradieren, Anfärben, Extrahieren).

Diese Entwicklungen stehen im Kontext breitgefächerter Bemühungen um die Konstruktion effizienter Mensch-Maschine-Schnittstellen [Joh87]. Andere Beispiele für solche Schnittstellen sind die diversen graphischen Oberflächen, CAD-Systeme, Desktop-Applikationen und Textsatzsysteme der letzten Jahre. Erst in jüngster Zeit wird allerdings die Nützlichkeit solcher Entwicklungen im Bereich der Bioprozeßsteuerung erkannt [CIF89] [CFB90] [HMN+90] [FPS85], obwohl gerade hier eine fest vorgegebene Datenpräsentation eher hinderlich ist: Der Biotechnologe kann zum Zeitpunkt des Versuchsaufbaus oft nicht genau sagen, für welche On-Line-Informationen er sich besonders interessieren wird!

3.4.2 Explorative Datenanalyse

Die *Explorative Datenanalyse* [Tuk77] [Boc80] beschäftigt sich mit Techniken zur Aufdeckung unbekannter Strukturen in großen und heterogenen Datensätzen. Sie steht damit in einem Gegensatz zur klassischen konfirmatorischen Datenanalyse, die versucht, eine vorgegebene Hypothese statistisch abzusichern. Während die klassische Statistik weitgehend damit beschäftigt ist, Modelle zu verifizieren oder zu falsifizieren, versucht die Explorative Datenanalyse, sinnvolle Modelle für empirische Daten überhaupt erst einmal zu finden.

Unentbehrliche Grundlage für die Explorative Datenanalyse ist die interaktive Computergraphik, die ein kreatives Arbeiten mit den Daten erlaubt. Im Grunde beruht die Explorative Datenanalyse auf der Ausnutzung der enormen Strukturerkennungs- und Differenzierungsfähigkeiten des menschlichen Auges. Es muß daher stets angestrebt werden, das Datenmaterial geeignet für das Auge aufzubereiten. Einige Techniken der Explorativen Datenanalyse sind:

- Projektionsverfahren für hochdimensionale Daten in den zwei- und dreidimensionalen Raum,
- dynamische Visualisierung im Raum,
- weiterentwickelte Glättungsverfahren (glättende Splines, nichtparametrische Regression),
- weiterentwickelte multivariate statistischer Verfahren (Hauptkomponentenanalyse, Regressionanalyse),
- Clusteranalyse und Klassifikation.

Bestehende Systeme, die explorative Techniken anbieten, sind z.B. die Systeme S [BC84] und ISP [ISP88] sowie diverse Versuchssysteme [Stu87] [CM88].

3.4.3 Explorative und konfirmatorische Datenanalyse

Das Verhältnis zwischen Explorativer und konfirmatorischer Datenanalyse ist kein ausschließliches [Vic80] [Ihm80]. Vielmehr sollte die Explorative Datenanalyse die Grundlage für eine anschließende statistische Absicherung mit klassischen Methoden sein. Wer mit der Modellierung biologischer Systeme befaßt ist, weiß allerdings, welche herausragende Rolle hier das Auffinden der Struktur von Modellen spielt. Erst wenn eine brauchbare Struktur gefunden ist, lohnt sich die Schätzung von Modellparametern. Ein kreativer Umgang mit dem Datenmaterial spielt somit eine wichtige Rolle.

Trotzdem muß betont werden, daß die Explorative Datenanalyse die Gefahr der Überinterpretation oder Fehlinterpretation von Daten in sich birgt. Bekanntlich ist das menschliche Gefühl für den Zufall nicht sehr ausgeprägt, so daß wir dazu neigen, sogar in zufällige Datensätze Strukturen hineinzudeuten [6] [ZN80]. Man muß sich darüber bewußt sein, daß explorativ gewonnene Ergebnisse

- im strengen Sinne a posteriori nicht mehr statistisch abzusichern sind.
- nicht auf einer eigens dafür konzipierten Versuchsplanung beruhen.
- auf einem nicht als statistisches Entscheidungsverfahren formulierbaren Weg gewonnen wurden.
- oft auf einer subjektiv ausgewählten Teilmenge der Daten basieren.

Diese Probleme machen die Explorative Datenanalyse zu einer sehr umstrittenen Methode, vor allem in Bereichen, wo die Datenerhebung abgeschlossen ist oder die Schlußfolgerungen weitreichende Konsequenzen haben (medizinische Anwendungen, bevölkerungsstatistische Erhebungen).

Andererseits können mit explorativen Methoden noch Probleme behandelt werden, die weit außerhalb des Spektrums klassischer Methoden liegen! Es ist zu betonen, daß gerade auf einem Arbeitsgebiet wie der Biotechnologie, wo Versuchsbedingungen nur schwer reproduzierbar sind, jedoch stets große Datenmengen anfallen, ein sehr fruchtbares Feld für explorative Methoden besteht: Die Aufstellung einer falschen Hypothese hat in der Regel keine negativen Folgen, da der nachfolgende Versuch entsprechend ausgerichtet werden kann. Schlimmer ist allerdings das Nichtauffinden eines guten Modells (vgl. dazu auch die Diskussion der Modellierungskonzepte in Abschnitt 10.4)!

3.4.4 Programmiertechnische Entwicklungen

Die interaktive Computergraphik und die Explorative Datenanalyse stellen extreme Anforderungen an die Flexibilität der Software, die zu einigen Weiterentwicklungen in den letzten Jahren führten. Die Bereiche, zu denen Beiträge geleistet wurden, sollen in Stichworten genannt werden (vgl. [BHM+88]):

- graphische Oberflächen,
- schnelle graphische Algorithmen,
- Dialogsprachen, Protokollgenerierung und -analyse,
- Nutzung verteilter Rechenkapazitäten,
- Constraint Propagating,
- objektorientierte Datenrepräsentation,
- deklarative Beschreibung graphischer Daten.

3.5 Gesamt-Ablaufsteuerung des Systems

Nach der Klärung der Funktionalität bleibt festzulegen, wie die Steuerung eines solchen Systems zu implementieren ist:

[6] ... wofür die Astrologie ein klassisches Beispiel gibt!

3.5.1 Eine generische Architektur für informationsverarbeitende Systeme

Die Konzeption des Datenanalysesystems orientiert sich an der Architektur anderer informationsverarbeitender Systeme. Es kann sich hierbei um statistische Datenanalysesysteme, Systeme der Regelungstechnik, aber auch um Datenbanken oder Kommunikationssysteme handeln. Auf abstrakter Ebene läßt sich für alle diese Systeme eine generische Architektur angeben, welche die logische Struktur spezifiziert [Mar90]. Diese Architektur ist in Abbildung 3.3 schematisiert [7].

Der Aufbau des Anforderungskatalogs für die biotechnische Datenanalyse aus Abschnitt 1.3 und das Modularisierungsschema für die Datenanalysebausteine aus Abschnitt 3.2 ist bereits stark an die allgemeine logische Struktur angelehnt. Die zugehörigen Lösungskonzepte sollen nun in das allgemeine Gerüst eingeordnet werden.

3.5.2 Theoretische Grundlagen und Methodik

Im Vorfeld des Entwurfs informationsverarbeitender Systeme muß Klarheit über die theoretischen Grundlagen und darauf aufbauend über die zu bei der Problemlösung verwendende Methodik bestehen. Dies erfordert zunächst die Erarbeitung eines einheitlichen theoretischen Grundkonzepts. Man kann nicht erwarten, ein homogen strukturiertes System zu erhalten, wenn bereits die Basis eine Sammlung von unzusammenhängenden Modellen und Methoden ist!

3.5.2.1 Eine vereinheitlichende Theorie der biotechnischen Datenanalyse

Leider sind gerade im Bereich der biotechnischen Datenanalyse bisher eine Fülle von theoretischen Ansätzen zu verzeichnen. Bislang existiert keine Vereinheitlichung dieser Ansätze, die eine allgemeine Rechnerimplementation ermöglicht. Die Ausführungen in den Kapiteln 5, 6, 7 und 8 stellen den Versuch einer einheitlichen Theorie der biotechnischen Datenanalyse dar. Dabei wurden bestehende Teilkonzeptionen aus der Literatur (vor allem [Roe83] [Fei77] [vDMdMW88] [Lia89] [NS89] [Red88]) zugrunde gelegt und — nach einer geeigneten Verallgemeinerung — zu einem umfassenden Konzept zusammengefügt.

Zusätzlich zur Vereinheitlichung der Theorie ist eine entsprechende didaktische Aufbereitung des Stoffs erforderlich. Auf diesen Aspekt wird in den genannten Kapiteln besonderer Wert gelegt.

3.5.2.2 Auswahl der Methoden

Das theoretische Fundament eines Datenanalysesystems kann zum größten Teil mit Hilfe von mathematischen Modellen formuliert werden. Dadurch werden die anzuwendenden Methoden bereits weitgehend festgelegt, da Methoden der Datenanalyse meist unmittelbar auf einer Modellstruktur aufsetzen. Die Korrespondenz zwischen Modellen und Methoden wird durch die Anlage der Kapitel 9, 10, 11 und 12 betont, in denen zu den verschiedenen Modellstrukturen jeweils die entsprechenden Datenanalysetechniken vorgestellt werden.

3.5.2.3 Homogenität des Methodenspektrums

Ein Indiz für eine gelungene Vereinheitlichung der Theorie ist gegeben, wenn die auf den theoretischen Modellen basierenden Methoden selbst wieder eine homogene Einheit bilden [8]. Dieser Aspekt wird für die vorliegende Konzeption in Kapitel 4 diskutiert und an verschiedenen Stellen durch Beispiele illustriert (vor allem Kapitel 2, 21 und 22).

3.5.3 Oberfläche

3.5.3.1 Kommunikation mit den Werkzeugen

Die Oberfläche des Datenanalysesystems besorgt die Kommunikation zwischen dem Benutzer und den Werkzeugen des Systems. Sie ist mit keinerlei Intelligenz bzgl. der einzelnen Methoden ausgestattet,

[7] Für die Anregung danke ich meinem Doktorvater, Herrn Priv. Doz. Dr. W. Martin von der Deutschen Software AG, Darmstadt.

[8] Der Mathematiker würde von einem Homomorphismus zwischen Theorie und Realisierung sprechen.

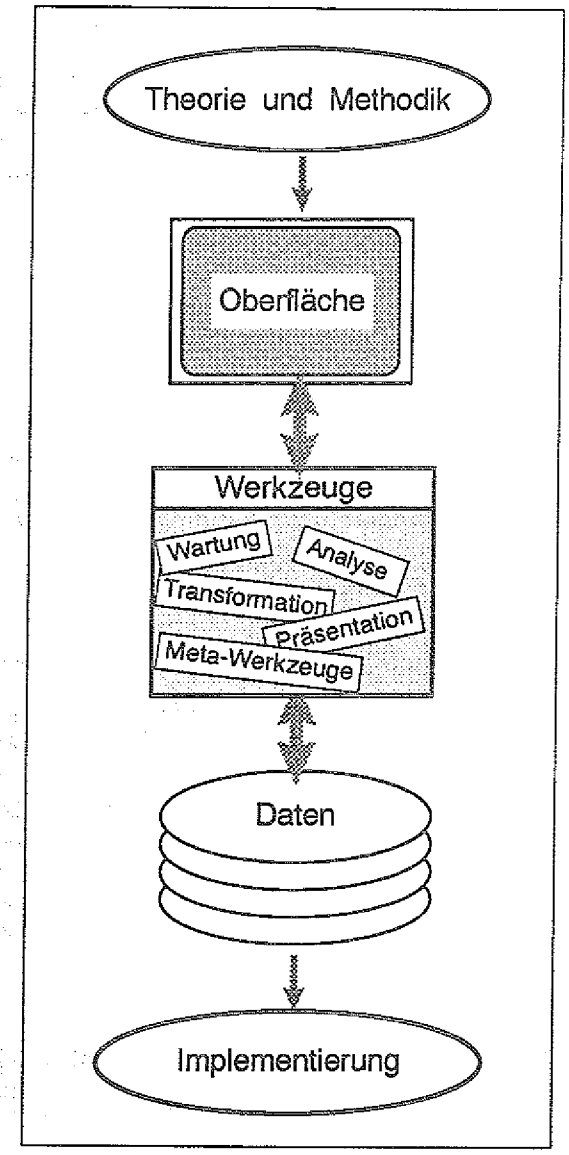


Abbildung 3.3: Generische Architektur für informationsverarbeitende Systeme

sondern stellt lediglich eine Ein-/Ausgabe-Einheit dar. Es erfolgt nur eine oberflächliche Analyse auf Korrektheit der Eingaben (korrekte Zahleneingaben, Existenz von ausgewählten Dateien etc.). Parameter, Modelle und Steueranweisungen werden nach Eingabe durch den Benutzer an die Werkzeuge weitergegeben. Diese prüfen die Korrektheit im Detail sowie die Konsistenz der Daten.

3.5.3.2 Weitere Aufgaben

Weitere Aufgaben der Oberfläche, die unmittelbar mit dem Benutzerdialog zusammenhängen und ggfs. auch über die Schnittstelle mit dem zugrunde liegenden Werkzeug abgewickelt werden müssen, sind

- die Übermittlung von Fehlermeldungen,
- kontextsensitive Hilfestellung,
- die Erstellung und Veränderung von Konfigurationsdateien, die alle Konfigurationsdaten eines Datenanalysebausteins enthalten,

- die Dokumentation der Konfiguration eines Werkzeugs.

3.5.3.3 Vereinheitlichung und Akzeptanz

Obwohl die Haupt-Verwaltungsarbeit an die Werkzeuge weitergegeben wird, stellt die Erstellung der Oberfläche die Hauptprogrammierarbeit dar (vgl. dazu auch 14.1.2). Dies liegt darin begründet, daß der Benutzerdialog das eigentlich interaktive Element bei einer Datenanalysesitzung ist. Die Realisierung solcher interaktiven Komponenten ist erfahrungsgemäß mit einem erheblichen Zeitaufwand verbunden!

Auf der anderen Seite ist zur Realisierung der Oberfläche bei der geschilderten Aufgabenverteilung zwischen Oberfläche und Werkzeugen keine Detailkenntnis über die Methoden erforderlich. Die dadurch ermöglichte Arbeitsteilung bei der Programmierung erlaubt die einheitliche Gestaltung aller Oberflächen mit entsprechenden Konsequenzen für die Akzeptanz beim Benutzer.

3.5.3.4 Realisierung

Für die Erstellung graphischer Oberflächen stehen heute unter den verschiedenen Betriebssystemen diverse Hilfsmittel zur Verfügung. Trotzdem hat die Datenanalyse einige spezifische Anforderungen an die Benutzerführung, die mit der bestehenden Konfektionssoftware — die oft für Verwaltung und Geschäftsführung entwickelt wurde — nur umständlich zu lösen sind. Deshalb werden im Kapitel 16 einige zentrale Aspekte der Oberflächengestaltung und Benutzerführung behandelt.

3.5.4 Werkzeuge

Zu den Werkzeugen eines informationsverarbeitenden Systems gehören zum einen die aus der Methodik erwachsenen Datenanalysetechniken zum anderen aber auch Hilfsmittel für die Verwaltung des Datenbestandes. Für die interaktive Datenanalyse erscheint die nachfolgende Klasseneinteilung der Werkzeuge sinnvoll [9]:

3.5.4.1 Wartung und Analyse

Mit *Wartung* wird die elementare Pflege (Speichern, Laden, Löschen etc.) der Daten und Konfigurationsdateien bezeichnet [10]. Die *Analyse* des Datenbestands umfaßt Hilfsmittel zur Auffindung gewünschter Informationen (z.B. Vergleichsdatensätze oder frühere Ergebnisse und Konfigurationen) [11]. Die Anforderungen an diese Werkzeuge werden erst dann präzise formulierbar, wenn geklärt ist, wie die genaue Struktur der Datenbasis und die rechnerinterne Datenrepräsentation aussehen (einige Details finden sich in 20). Viele Bedürfnisse entstehen zudem erst bei der praktischen Erprobung eines Systems. Das folgende Beispiel soll dies verdeutlichen:

> Jeder Biotechnologe verwendet bei der Prozeßüberwachung für die verschiedenen Meßdatenkanäle seiner Apparatur eigene Bezeichnungen (z.B. BM, BTM oder X für die Bio(trocken)-masse). Ein für den Fermenter A konfigurierter Datenanalysebaustein ist folglich — selbst bei identischer Instrumentierung — nicht unmittelbar am Fermenter B einsetzbar. Benötigt wird ein Umbenennungswerkzeug, welches die Korrespondenz zwischen den Bezeichnern herstellt. Es muß auf alle Datenanalysebausteine anwendbar sein.

3.5.4.2 Transformation

Transformationen sind Operationen, die aus dem gegebenen Datenmaterial neue Daten und Datensätze gewinnen. Im Kontext der Datenanalyse ist dies genau die Charakterisierung eines Datenanalysebausteins. Der Aufbau eines solchen Bausteins wurde in Abschnitt 3.2 behandelt; eine Zusammenstellung verschiedener Methoden die so realisiert werden können, bietet Abschnitt 3.1.

---

[9] Bei anderen Aufgabenstellungen — z.B. bei Datenbanksystemen — kann eine hiervon verschiedene Klassifikation angemesssen sein.

[10] Die Programmierer-Fachsprache hat für solche Werkzeuge das treffende Wort "Utilities" geprägt

[11] In der Datenbank-Fachsprache wird dies als *Retrieval* bezeichnet

3.5.4.3 Graphische Präsentation

Die *Graphische Präsentation* umfaßt die verschiedenen Visualisierungstechniken. Die Methoden sind Gegenstand des Abschnitts 3.4 sowie von Kapitel 13.

3.5.4.4 Meta-Werkzeuge

Eine besondere Stellung unter den Werkzeugen hat die Ablaufsteuerung für Netzwerke von Datenanalysebausteinen. Sie umfaßt

- die Konfiguration der Netzwerke,
- die zeitliche Synchronisation der Bausteine,
- die Steuerung der Kommunikation zwischen den Bausteinen sowie
- die Automatisierung der Abläufe

und stellt damit ein "Meta-Werkzeug" über der Ebene der Transformations- und Präsentationswerkzeuge dar [12] . Die Ablaufsteuerung läßt sich besser im Kontext der Bioprozeßüberwachung behandeln, so daß hier nicht weiter darauf eingegangen wird. Detaillierte Ausführungen finden sich in den Kapiteln 4 und 20.

3.5.5 Daten

Die Datenlage in der Biotechnologie wurde in Kapitel 1 geklärt. Zu lösen ist jetzt das Problem der Datenrepräsentation und der Datenablage.

3.5.5.1 Datenrepräsentation und Speicherverwaltung

Auf die Darstellung der Daten im Speicher des Rechners wird in Kapitel 20 genauer eingegangen. Besonderer Wert wird hier auf weitgehende Abstraktion der Datentypen gelegt, um der Heterogenität der Eingangsdaten sowie der in Zukunft zu erwartenden Meßtechnik gerecht zu werden.

3.5.5.2 Datenablage

Bzgl. der Datenablage und Archivierung ist eine Entscheidung für ein bestimmtes Datenbanksystem erforderlich, die hier noch nicht gefällt werden kann. Sobald eine konkrete Datenbank ausgewählt ist, ergibt sich ein spezieller Bedarf an Werkzeugen für Wartung und Analyse des Datenbestandes. Dieses Problem wird im Rahmen der vorliegenden Arbeit ausgegrenzt.

3.5.6 Implementierung

3.5.6.1 Programmiertechniken

Die konkrete Implementierung eines Datenanalysesystems erfordert eine ausgereifte Programmiertechnik, wovon Effizienz, Ausführungsgeschwindigkeit und Übersichtlichkeit des erzeugten Codes entscheidend abhängen. Ein zentrales Thema des Kapitels 14 ist daher

- die Objektorientierte Programmierung, die bei der Realisierung komplexer Systeme von kommunizierenden Komponenten unumgänglich wird und eine offene Konzeption des Systems ermöglicht. Ferner werden Aspekte
- der deklarativen Programmierung,
- der Modularisierung und
- der Programmierung paralleler Prozesse behandelt.

[12] Der Begriff *Meta-Werkzeug* ist bei Systemen zum Computer Aided Software Engineering (CASE) nicht unbekannt.

3.5.6.2 Datentypen

Das Kapitel 15 geht dann auf die objektorientierte Realisierung immer wiederkehrender Strukturen wie

- Strings,
- Vektoren und Matrizen,
- sequentielle Speicher und die
- Fehlerbehandlung ein.

3.5.6.3 Weitere Gebiete der Programmierung

In einigen Gebieten der Programmierung sind spezielle Techniken entwickelt worden, die bei der Datenanalyse von Wichtigkeit sind. Erläutert werden Konzepte aus den Bereichen

- Benutzerführung (Kapitel 16),
- Graphik (Kapitel 17),
- Compilerbau (Kapitel 18) und
- Numerik (Kapitel 19).

3.6 Formulierung des Arbeitsziels

Der Arbeitsaufwand zur Erstellung eines kompletten Datenanalysesystems ist um ein Vielfaches zu hoch, um im Rahmen einer Dissertation abgehandelt werden zu können [13]. Vielmehr handelt es sich um eine Studie, die dazu dient, die Anforderungen festzustellen und exemplarische Teillösungen zu programmieren. Trotzdem darf das große Ziel nicht aus den Augen verloren werden, damit wesentliche Komponenten nicht mehrfach programmiert werden müssen.

3.6.1 Vorhandene Arbeitsumgebung

Vor der Formulierung des Arbeitsziels muß kurz die Arbeitsumgebung beschrieben werden, die zur Verfügung stand. Sie wurde im Kontext der obigen Erwägungen unter dem Gesichtspunkt der Weiterverwendbarkeit zusammengestellt:

Mit den Mittel des BMFT-Stipendiums "Biotechnologie und angewandte Mikrobiologie" sowie mit großzügiger Unterstützung von Herrn Prof. Wandrey (IBT2 der KFA Jülich) und Herrn Prof. Alt (Abteilung Theoretische Biologie der Universität Bonn) konnten die folgenden Anschaffungen gemacht werden:

3.6.1.1 Hardware

- PC (HP Vectra RS/20, 80386 Prozessor, 20 MHz Taktrate),
- Mathematischer Koprozessor (80387),
- 4 MB Speicher,
- 100 MB Festplatte,
- Hochauflösender Farbgraphik-Adapter (1024x768 Pixel, 16 Farben),
- 16 Zoll Farbmonitor,

[13] Professionelle Systeme werden meist von Ingenieurbüros mit ca. 10-20 Beschäftigten in mehrjähriger Entwicklungszeit erstellt.

- Maus.

Diese Konfiguration repräsentierte zum Zeitpunkt des Kaufs den höchsten Leistungsstandard im PC-Bereich. Die Fähigkeiten einer kleinen Workstation sind im Prinzip gegeben, jedoch stößt die Rechengeschwindigkeit und Speicherkapazität bald auf ihre Grenzen. Der Rechner kann später zu einem X-Windows-Server ausgebaut werden und als Terminal von einer Workstation mitbetreut werden.

3.6.1.2 Software

- Betriebssystem DOS 3.3,
- MS Windows Oberfläche,
- MicroSoft C, Turbo C und Turbo Pascal Compiler,
- Turbo-Assembler und -Debugger,
- MathPak87-Paket [MP886] mit maschinencodierten elementaren Numerik-Routinen.

Es bleibt zu bemerken, daß derzeit die OS/2 und UNIX-Systeme für den PC noch nicht ausgereift sind. D.h. die unter DOS chronische Speicherplatznot mußte in Kauf genommen werden, so daß manche Problemlösungen auf einen späteren Zeitpunkt verschoben werden.

3.6.2 Arbeitsziel

Ziel der Arbeit ist es,

- die Notwendigkeit eines interaktiven Ansatzes für die Analyse biotechnischer Daten zu begründen.
- die Datenlage in der Biotechnologie zu analysieren und einen Anforderungskatalog für ein interaktives Datenanalysesystem zu erstellen.
- die bestehenden Entwicklungen aus den verschiedenen Bereichen zu sammeln und zu verbinden.
- eine vereinheitlichende Theorie für die Analyse biotechnischer Prozeßdaten zu schaffen.
- die bestehenden Modellansätze und Datenanalysemethoden in diese Theorie einzuordnen.
- die grundlegenden Methoden didaktisch aufzubereiten,
- die Programmierwerkzeuge bereitzustellen.
- die Leistungsfähigkeit des Ansatzes an einer Pilotversion auf einem beschränkten Anwendungsbereich vorzuführen.
- eine Reihe von (nicht-interaktiven) Anwendungsbeispielen zur Demonstration der Vielseitigkeit des Methodenspektrums zu programmieren.
- eine On-Line-Version eines modellgestützten adaptiven Filters an ein konkretes Meßdatenerfassungssystem (MEAS vom IBT2 der KFA-Jülich [Sch89]) anzubinden.

In diesem Kontext und aufgrund der gegebenen Möglichkeiten wird es nicht Ziel dieser Arbeit sein,

- die Bausteinverwaltung und die Vernetzung von Methoden interaktiv zu realisieren. Stattdessen wird eine prozedurale Programmierweise gewählt. Damit bleibt der Datenfluß zwar statisch, jedoch können alle wesentlichen Konzepte bereits verdeutlicht werden.
- die Datenbasis flexibel zu gestalten (was unter DOS zu mühsam wäre).
- die zugrunde liegenden numerischen Verfahren neu zu entwickeln oder zu verbessern.
- die Modellierung konkreter biotechnischer Systeme zu betreiben. Stattdessen werden bekannte Beipiele zur Hilfe genommen und unter neuen Gesichtspunkten beleuchtet.

Kapitel 4

Einordnung in das Umfeld der Bioprozeßüberwachung

Es sollen in diesem Kapitel zwei Thesen aufgestellt und mit Argumenten unterstützt werden:

1. Jedes Prozeßüberwachungssystem beruht auf einem (meist komplexen) Modell des zu überwachenden bzw. zu kontrollierenden Systems.
2. Aus einem Systemmodell kann umgekehrt stets auch ein Prozeßüberwachungssystem konstruiert werden.

Die erste These ist nicht neu: In weiten Teilen der Automatisierungstechnik, Informatik, Regelungstechnik und Systemtheorie wird seit langem von Modellen für das betrachtete System gesprochen. Allerdings werden sehr unterschiedliche Modellbegriffe zugrunde gelegt, die im folgenden noch etwas näher untersucht werden sollen.

Interessanter ist die zweite These: Die Konsequenz dieser Aussage wäre ein Rezept zur Generierung von Prozeßüberwachungssystemen. Es wird sich zeigen, daß die zweite These in engem Zusammenhang mit dem vorgestellten Datenanalysekonzept steht.

4.1 Ein erweiterter Modellbegriff

Die in dieser Arbeit vorgestellten Methoden sind in erster Linie sogenannte *modellgestützte Verfahren* im Sinne der Systemtheorie, d.h. solche Verfahren, die auf einem mathematischen Systemmodell in Form von Gleichungen aufbauen. Der Modellbegriff kann aber durchaus weiter gefaßt werden und auch heuristische und logische Systembeschreibungen aufnehmen.

> Ganz allgemein soll unter einem *Modell* jegliche Form der formalen Beschreibung eines gegebenen Systems verstanden werden. Darunter fallen nicht nur Gleichungs-Modelle sondern auch heuristische, in einen Formalismus gefaßte Beschreibungen oder logische Ablaufbeschreibungen.

4.1.1 Eine Modellhierarchie

Ausgegangen werden soll von fünf Modellklassen, wobei die Grenzen nicht scharf zu ziehen sind. Die Klasseneinteilung dient als Arbeitsplattform für die weitere Gliederung dieser Arbeit. Andere Klassifikationsansätze finden sich z.B. in [MS88] [LG87] [PUB83] [Sch82b].

4.1.1.1 Deterministische Modelle

Unter einem *deterministischen Modell* soll ein (deterministisches) Differentialgleichungs-Modell verstanden werden, das meist in Form eines sogenannten Zustandsraummodells notiert wird (vgl. 5.4). Deter-

ministische Modelle dienen zur Beschreibung dynamischer Vorgänge, bei denen alle Beschreibungsgrößen und Mechanismen genau bekannt sind.

4.1.1.2 Stochastische Modelle

Jedes mit Hilfe von stochastischen Prozessen formulierte Modell für die Dynamik eines Systems sei in diesem Rahmen als *stochastisches Modell* bezeichnet. Darunter fallen u.a. stochastische Differentialgleichungen (z.B. die stochastischen Zustandsraummodelle, vgl. Abb. 2.4) und viele Modelle aus der Zeitreihenanalyse. Ein stochastisches Modell wird benötigt, wenn ein dynamisches System nur bis auf Unschärfen beschreibbar ist.

4.1.1.3 Statistische Modelle

Unter den Sammelbegriff *statistische Modelle* sollen alle Beschreibungsformen der klassischen mathematischen Statistik fallen, die nicht bereits als stochastisches System klassifiziert sind. Dies sind die Modelle der Schätz- und Testtheorie sowie der multivariaten und nichtparametrischen Statistik.

Die hier verwendete Bedeutung der Begriffe "statistisch" und "stochastisch" ist nicht allgemein üblich. Allerdings differiert die Verwendung der Begriffe von Autor zu Autor (verwiesen sei auf [Sac84] oder ein Konversationslexikon). Aus diesem Grunde sei die hier vollzogene Neudefinition gestattet. Die Trennung zwischen stochastischen und statistischen Modellen ist im Rahmen der biotechnischen Datenanalyse sehr scharf: "Stochastisch" sind alle (zufallsbeeinflußten) Modelle, bei denen die Zeit im Datensatz vorkommt, und "statistisch" die übrigen.

4.1.1.4 Logische Modelle

Unter einem *logischen* oder *heuristischen Modell* soll jede in einen Formalismus gefaßte Beschreibung eines Systems verstanden werden, die nicht zu den bisher genannten Modellklassen gehört. Solche Modelle müssen dann eingesetzt werden, wenn zwar eine Vorstellung von der Funktionsweise des modellierten Systems vorliegt, diese aber quantitativ nicht genau faßbar ist.

Eine formale Beschreibung soll dabei als *logisches Modell* aufgefaßt werden, wenn sie objektivierbare Tatsachen widerspiegelt. Ein Beispiel: Die Aussage

> "Bei einem Ausfall des Rührers verringert sich der Sauerstoffeintrag in das Medium"

ist ein logisches Modell für einen Störungszustand im Reaktor.

4.1.1.5 Heuristische Modelle

Die Formalisierung von Erfahrungswerten und -strategien wird als *heuristisches Modell* angesehen. Die Aussage

> "Bei einem Ausfall des Rührers verringert sich der Sauerstoffeintrag
> in das Medium um ca. 95%"

ist ein derartiges heuristisches Modell, sofern die Zahl 95% als Daumenregel eingesetzt wurde und nicht einer konkreten Versuchsreihe entspringt. In der Praxis ist eine Trennlinie zwischen logischen und heuristischen Aussagen kaum scharf zu ziehen, wie auch die beiden oben angeführten Beispiele zeigen.

4.1.2 Vergleich der verschiedenen Modelltypen

Man stellt in der Reihe von den deterministischen über die stochastischen und statistischen zu den logischen und heuristischen Modellen einen geringer werdenden "Erklärungswert" der Modelle fest. Der Begriff *Erklärungswert* läßt sich naiv definieren als der Grad, in dem ein Modell den wirklichen Mechanismus, der dem System zugrunde liegt, erklärt und damit Vorhersagen für das zukünftige Systemverhalten zuläßt.

Eine vertiefende Diskussion des Begriffs "Erklärungswert" führt sehr schnell zu allgemeinen philosophischen Betrachtungen über den naturwissenschaftlichen Erkenntnisbegriff. Hier sei die Lektüre von Karl

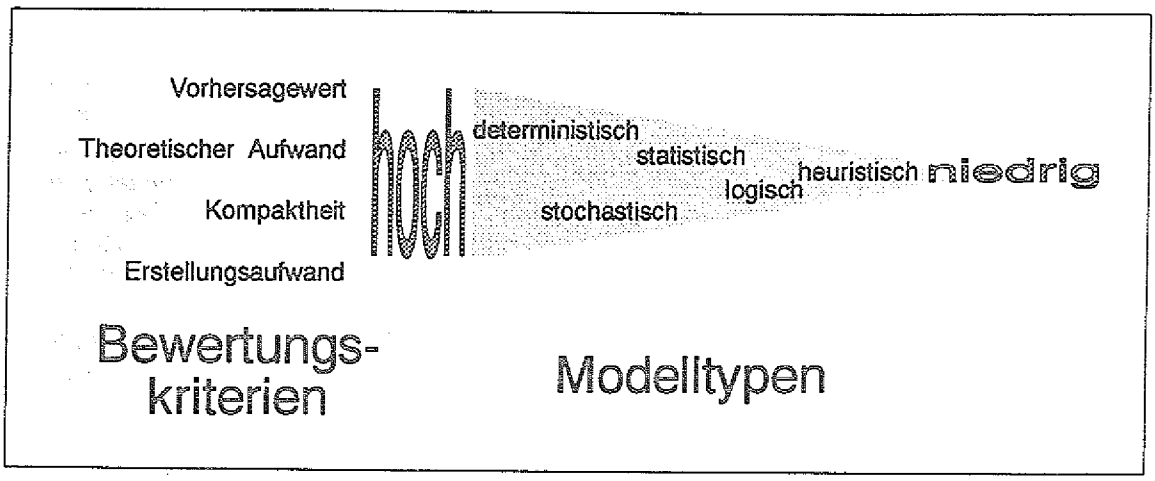


Abbildung 4.1: Bewertung von Modellierungsansätzen unter verschiedenen Kriterien

Poppers Buch "Objektive Erkenntnis — Ein evolutionärer Entwurf" [nPo84] empfohlen, der von einer pragmatischen Definition des Erkenntnisbegriffs ausgeht. Zentral ist die Feststellung, daß naturwissenschaftliche Aussagen, die auf Empirie beruhen, niemals verifizierbar, sondern lediglich falsifizierbar sind! D.h. die Güte einer Modellierung läßt sich nur daran messen, wie sehr sie einer kritischen Überprüfung standhält. Ein Modell, daß einer Überprüfung nicht standgehalten hat, muß nicht verworfen, sondern in seinem behaupteten Gültigkeitsbereich eingeschränkt werden. Die Frage, ob ein Modell die "Wirklichkeit" beschreibt, erscheint bei dieser Sichtweise als unsinnig!

Vom biotechnischen Standpunkt aus gesehen, muß die Überprüfung und Bewertung von Modellen vor allem unter Gesichtspunkten der Ökonomie und Realisierbarkeit geschehen. Einige Bewertungskriterien sind (vgl. auch [Koh81] [FM82])

der Vorhersagebereich, d.h. der zeitliche Bereich, in dem ein Modell präzise Vorhersagen für den zukünftigen Prozeßverlauf macht,

der zeitliche und personelle Aufwand, der für das Erlernen der theoretischen Grundlagen betrieben werden muß,

die Kompaktheit eines Modells, d.h. das Verhältnis zwischen Informationsgehalt und Umfang der formalen Beschreibung,

der Zeitaufwand zur Erstellung und Validierung eines Modells.

Die verschiedenen Klassen von Modellen lassen sich nach diesen Kriterien grob in eine Skala einordnen, die schematisch in Abbildung 4.1 dargestellt ist.

Die Auswahl eines Modelltyps hängt von der Aufgabenstellung ab: Z.B. sind für die Regelung und Steuerung Modelle mit großem Vorhersagebereich anzustreben, wogegen zur Erkennung von Störungszuständen oft bereits einfache logische Modelle genügen.

Von besonderer Bedeutung ist die Schere zwischen Vorhersagewert und Zeitaufwand: Zur Erstellung eines guten deterministischen Modells für ein biotechnisches System muß erfahrungsgemäß ein Zeitaufwand in der Größenordnung von einem Jahr und mehr veranschlagt werden. Man stößt hier sehr schnell an die Grenzen der zeitlichen und personellen Ressourcen!

4.2 Expertensysteme und klassische Regelungstechnik

In den letzten Jahren läßt sich bei der Bioprozeßüberwachung eine Kontroverse zwischen der Anwendung klassischer Regelungstechnik auf der einen Seite und Expertensystem-Techniken auf der anderen Seite feststellen. Es wird sich zeigen, daß diese Kontroverse auf unterschiedlichen Modellbegriffen beruht.

4.2.1 Vergleich der Ansätze

Es ist nach den Ausführungen zur Datenlage in der Biotechnologie (1.2) klar geworden, daß streng auf mathematischen Modellansätzen basierende klassische Regelungskomponenten allein nicht in der Lage sind, einen Bioreaktor zu steuern und zu überwachen: Zum einen ist die Komplexität des Systems mit einfachen Modellen nur in Teilaspekten beschreibbar und zum anderen muß seiner Störanfälligkeit mit speziellen Techniken begegnet werden. In der Praxis hat dies zu einer generellen Skepsis bzgl. der Tauglichkeit mathematischer Modellansätze geführt. Diese Skepsis verstärkt sich nicht zuletzt auch dadurch, daß immer aufwendigere Methoden aus der Theorie in die Praxis übertragen werden und die Modelle bis zu einer nicht mehr validierbaren Komplexität anwachsen [1].

Die Folge dieser Entwicklung sind verstärkte Bemühungen auf dem Gebiet der Steuerung und Überwachung von Bioprozessen mit logischen und heuristischen Ansätzen. Die so entstandenen Expertensysteme versuchen, das Verhalten des Betreibers bei der Überwachung eines Bioreaktors zu simulieren. Dazu ist es erforderlich, den Erfahrungsschatz des Praktikers in einer Wissensbasis zu formalisieren ("Bedienermodell" [FPS85]). Hier liegen also im wesentlichen logisch-heuristische Modellvorstellungen zugrunde.

4.2.2 Kritikpunkte an der heuristischen Vorgehensweise

Die Problematik des Einsatzes heuristischer Methoden wird auf mehrfache Weise deutlich:

4.2.2.1 Informationsdichte

Wenn für ein Teilsystem eines Bioreaktors ein gutes mathematisches Modell gefunden werden kann, so stellt dies die kürzeste und präziseste formale Beschreibung dar, die derzeit denkbar ist. Insbesondere kann aus dem Modell die Parametrisierung von Reglern und Beobachtern direkt durch Rechnung abgeleitet werden. Diese Informationsdichte ist mit einer heuristischen Beschreibung nicht annähernd erreichbar.

4.2.2.2 Übertragbarkeit

Bisherige heuristische Ansätze sind äußerst prozeßspezifisch, da sie aus den Erfahrungen an einem konkreten Prozeß entwickelt wurden [LHKH+89] [KH89] [QSQJC89]. Um eine Übertragbarkeit auf andere Systeme zu gewährleisten, muß eine Modularisierung in Teilmodelle erfolgen, die bei einem evolutiv am Prozeß entstandenen Modell aufgrund seines hohen Vernetzungsgrades nachträglich kaum noch durchführbar ist.

4.2.2.3 Implizite Numerik

Die Programmierung von Heuristiken in einer geeigneten formalen Sprache verführt zur Verwendung von Konstrukten der folgenden Art:

> "Falls Gelöst-Sauerstoffkonzentration $< 5\%$, so liegt Sauerstoffmangel vor"
>
> "Falls pH < 7.03, zwei Einheiten NaOH nachdosieren."
>
> "Falls
> 1. $X(t-3\,\Delta t), X(t-2\,\Delta t), X(t-\Delta t)$ um weniger als 5% schwanken
> 2. $X(t), X(t+\Delta t), X(t+2\,\Delta t)$ um weniger als 5% schwanken und
> 3. die beiden Gruppen um mehr als 20% voneinander abweichen, so liegt ein Sprung im Signal X vor."

In die mathematische Sprache übersetzt, handelt es sich hier um einen Signifikanztest, einen schaltenden Regler und einen Sprungdetektor. Würde man diese Ausdrücke durch entsprechende regelungstechnische Bausteine ersetzen, so wäre nicht nur eine Effektivitätssteigerung, sondern auch eine statistisch begründbare Wahl der Parameter möglich!

[1] Erinnert sei auch an die Weltmodelle des Club of Rome [Gol70].

4.2.2.4 Magische Konstanten

Zahlenwerte wie 7.03, 5% etc. werden im Kontext einer Programmiersprache gern als *magische Konstanten* [KR83] bezeichnet, da sie beim späteren Lesen des Programms völlig unmotiviert im Text auftauchen und die Begründung für die Wahl des Zahlenwertes längst abhanden gekommen ist. Die Gefahr, daß sich in heuristische Modelle magische Konstanten und ähnliche Konstrukte einschleichen, ist recht groß. Sie stellen ein weiteres Hemmnis für die Übertragbarkeit auf andere Systeme dar.

4.2.3 Forderungen an heuristische Modelle

Die dargestellten Probleme lassen sich beheben, wenn man zwei sehr weitreichende Forderungen an die Modelle stellt:

1. Heuristische Modelle müssen genau wie mathematische Modelle strikt modular für Teilsysteme aufgebaut werden. Eine heuristische Überwachung des Gesamtsystems entsteht erst im Zusammenspiel der Komponenten.
2. Ein logisches oder heuristisches Modell darf keinerlei numerische Werte oder arithmetische Ausdrücke enthalten. Alle Ein- und Ausgaben sowie interne Variablen müssen daher entweder logischer oder kategorialer Natur sein.

Verbannt man aufgrund dieser Forderungen alle numerischen Komponenten aus den Heuristiken, so ist man zu einer Synthese zwischen klassischer Regelungstechnik und heuristischen Ansätzen gezwungen.

Ein Beispiel: Ein heuristischer Modul zur Regelung der Sauerstoffkonzentration darf nicht die obige Formulierung

"Falls `Gelöst-Sauerstoffkonzentration` < 5%, so liegt Sauerstoffmangel vor"

enthalten, sondern muß vielmehr die Regel in der Form

"Falls `Gelöst_Sauerstoffkonzentration_Niedrig`, so liegt Sauerstoffmangel vor"

aufnehmen, wobei `Gelöst_Sauerstoffkonzentration_Niedrig` eine logische Variable ist, die ihren Wert z.B. aus einem statistischen Test (also einem numerischen Baustein) beziehen kann.

Der im folgenden vorgestellte systematische Aufbau ermöglicht jetzt die streng modulare und portierbare Implementierung von Prozeßüberwachungssystemen:

4.3 Versuch einer Synthese

4.3.1 Eine Hierarchie von Prozeßüberwachungskomponenten

Das Hauptproblem bei der Synthese zwischen klassischen regelungstechnischen Ansätzen und Expertensystem-Techniken ist die exakte Definition der Schnittstellen und Informations-Übergabekonventionen. Dieses Problem läßt sich im Rahmen der vorgestellten Modellhierarchie sehr elegant lösen, da der Hierarchie von Modellen unmittelbar eine Hierarchie von Prozeßüberwachungskomponenten entspricht, welche auf diesen Modellen aufbauen. Dies legt eine 3-Ebenen-Architektur für die Bioprozeßüberwachung nahe (vgl. auch Abb. 4.2):

a) Prozeßdatenfluß:

- Die obere Hierarchieebene wird gebildet durch heuristische und logische Bausteine, die gemäß der obigen Forderung frei von jeglichen numerischen Elementen sind. Ein- und Ausgänge sind entweder logisch oder kategoriell.
- Die untere Hierarchieebene enthält ausschließlich Komponenten, die die eigentliche Signalverarbeitung besorgen. Hierzu gehören die klassischen modellbasierten regelungstechnischen Komponenten. Sie haben numerische Ein- und Ausgänge.

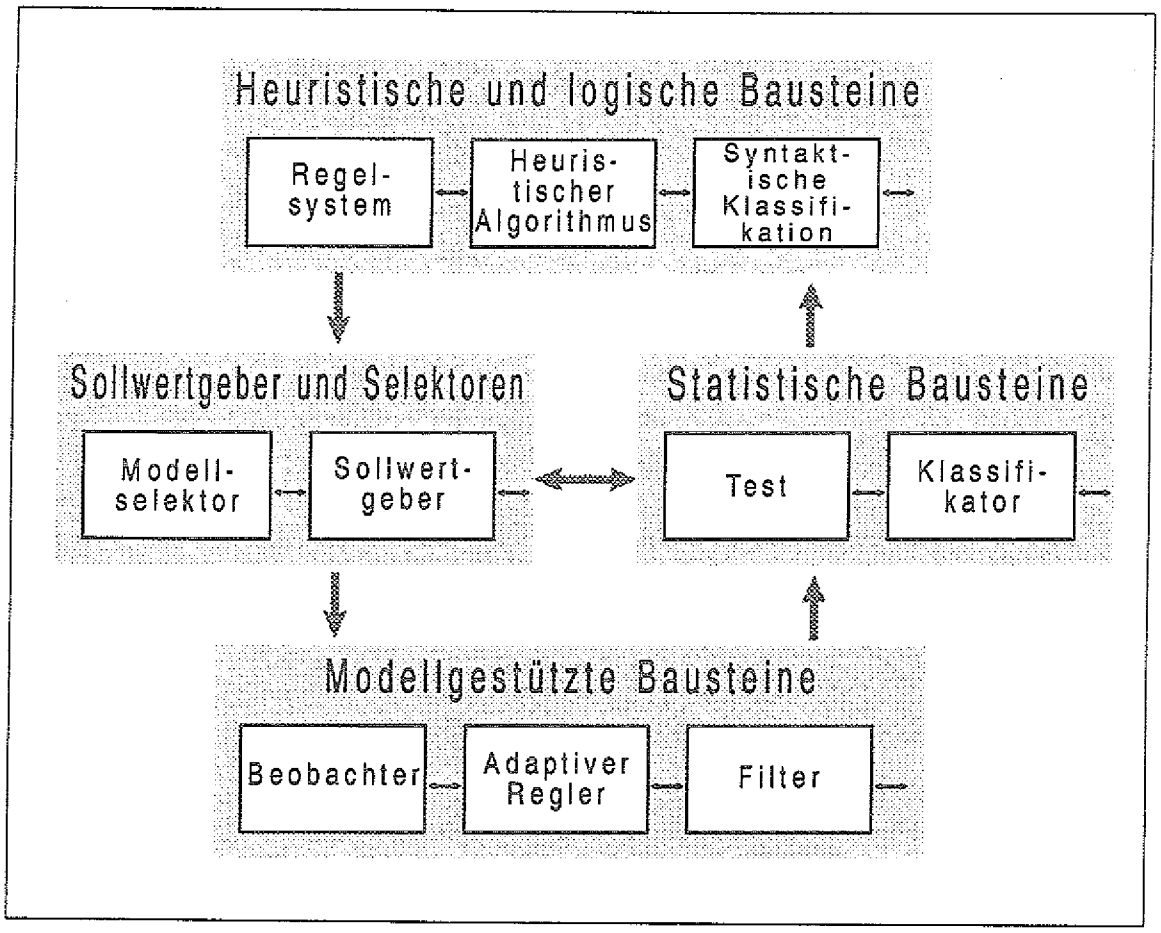


Abbildung 4.2: 3-Ebenen-Architektur für die Bioprozeßüberwachung

- Auf der mittleren Ebene sind die Komponenten angesiedelt, die die Verbindung zwischen der oberen und unteren Ebene herstellen. Die Ein- und Ausgänge nach oben sind logischer und kategorieller Natur, die Ein- und Ausgänge nach unten sind numerisch. Beispiele sind statistische Tests und Klassifikatoren (Datenfluß von unten nach oben) sowie Ausreißerfilter und Stellgrößengeber (Datenfluß von oben nach unten).

b) Kontrolldatenfluß:
Der Kontrolldatenfluß regelt die Konfiguration des Gesamtsystems. Er ist ausschließlich logischer und kategorialer Natur und stets von oben nach unten gerichtet. Jeder Baustein hat einen Eingang für Konfigurationsdaten.

4.3.2 Beispiele

Die bisher vorgestellten Beispiele für die Verschaltung von Datenanalysekomponenten sind alle nach diesem Schema aufgebaut. Die Abbildungen 2.10 und 2.13 lassen sich in diesem Zusammenhang neu lesen. Weitere Beispiele finden sich in Kapitel 22.

4.3.3 Realisierung der Schnittstellen

Solange man mit logischen und kategorialen Daten arbeitet, sind die anzuwendenden Techniken seit langem etabliert (Expertensystem-Shells, Sprachen der künstlichen Intelligenz, Logische Kalküle). Dasselbe kann man für die Arbeit mit numerischen Daten behaupten (Systemsimulatoren, Systeme der Regelungstechnik). Problematisch wird erst der Entwurf einer Schnittstelle zwischen den beiden Ebenen, da hier

Modellvorstellungen aufeinanderstoßen, die mit einer völlig unterschiedlichen Begrifflichkeit formuliert sind. Genau für dieses Problem stellen die hier entwickelten Datenanalysebausteine einen Lösungsansatz dar. Dabei besorgen bzgl. des Prozeßdatenflusses die Bausteine der mittleren (statistischen) Ebene die Verbindung zwischen der numerischen und der logisch/kategorialen Ebene, wogegen bzgl. des Kontrolldatenflusses von vornherein jeder Baustein einen Eingang für Konfigurationsdaten vorsieht.

Wie solche Bausteine zu realisieren sind, wird später noch genau beschrieben und am Beispiel des erweiterten Kalman-Filters auch explizit umgesetzt. Damit ist ein wesentlicher Schritt für die Einbindung modellgestützter Komponenten der Regelungstechnik in Expertensysteme zur Prozeßüberwachung geleistet worden (vgl. auch [Wie89] [HW88] [BP88b]).

4.3.4 Frame-basierte Expertensysteme

Expertensysteme, die eine derartige Vorgehensweise unterstützen, heißen *frame-basiert*[PM87] : Die einzelnen Komponenten sind jeweils in einem "Rahmen" zusammengefaßt; der Informationsaustausch zwischen den verschiedenen Frames geschieht über sogenannte *Slots*. Innerhalb jedes Rahmens kann eine beliebige Beschreibungsform (mathematisch, logisch, heuristisch) zur Wissensrepräsentation gewählt werden. Wie ein solcher Frame intern zu realisieren ist, wird vollkommen offen gelassen; das System übernimmt lediglich die Verwaltung des Informationsaustauschs und die Ablaufsteuerung.

Die Bausteine des in dieser Arbeit entwickelten Datenanalysesystems legen auf natürliche Weise eine Implementierung in ein solches frame-basiertes Expertensystem nahe. Sie liefern gewissermaßen das "Innenleben" der Frames.

4.4 Von der Datenanalyse zur Prozeßüberwachung

Es kann nun die zweite Teilaussage der anfänglich aufgestellten These diskutiert werden: Die Erzeugung eines Prozeßüberwachungssystem aus einem gegebenen Modell. Das Datenanalysekonzept bietet die Möglichkeit, ein Prozeßüberwachungssystem im Zuge mit Verfahrensentwicklung evolutiv zu entwickeln. Nach Abschluß der technischen Entwicklung ist damit zugleich ein Systemmodell und ein Prototyp für ein Überwachungssystem erstellt. Dieser Arbeitsgang schließt sich im normalen Entwicklungszyklus jetzt erst an!

4.4.1 Der Entwicklungszyklus

Der Entwicklungszyklus geht zunächst von einem geringen Vorwissen aus. Darauf aufbauend wird das Modell sukzessive erweitert und verbessert (vgl. Abb. 4.3) [2].

1. Jeder Bestandteil des Systems ist a priori durch ein Modell mit möglichst hohem Vorhersagebereich zu beschreiben. Dies werden in der Regel heuristische Modelle sein, es können aus der Literatur oder Erfahrungen mit verwandten Systemen aber auch schon mathematische Modellansätze bestehen. Insbesondere können bestehende Module aus anderen Problemlösungen übernommen werden. Begonnen wird also mit einem "Flickenteppich".

2a. Auf der Grundlage der bestehenden Modellierung lassen sich Experimente zur Informationsgewinnung oder auch Reglerentwürfe ausführen.

2b. Im Laufe einer Verfahrensentwicklung sind sukzessive Komponenten mit kleinem Vorhersagebereich durch verbesserte Modelle zu ersetzten. Damit werden wiederum leistungsfähigere Methoden zur weiteren Informationsgewinnung durchführbar.

3. Am (idealen) Schlußpunkt der Entwicklung fügen sich alle Teilmodelle zu einem einzigen Systemmodell mit großem Vorhersagebereich bzgl. aller Systemkomponenten zusammen.

In der zur Verfügung stehenden Zeit wird die Entwicklung irgendwo in der Mitte zum Stehen kommen. Man verfügt dann über ein aus deterministischen, stochastischen und heuristischen Komponenten zusammengesetztes Prozeßüberwachungssystem.

[2] Der Ansatz ist stark angeregt durch [Pop71].

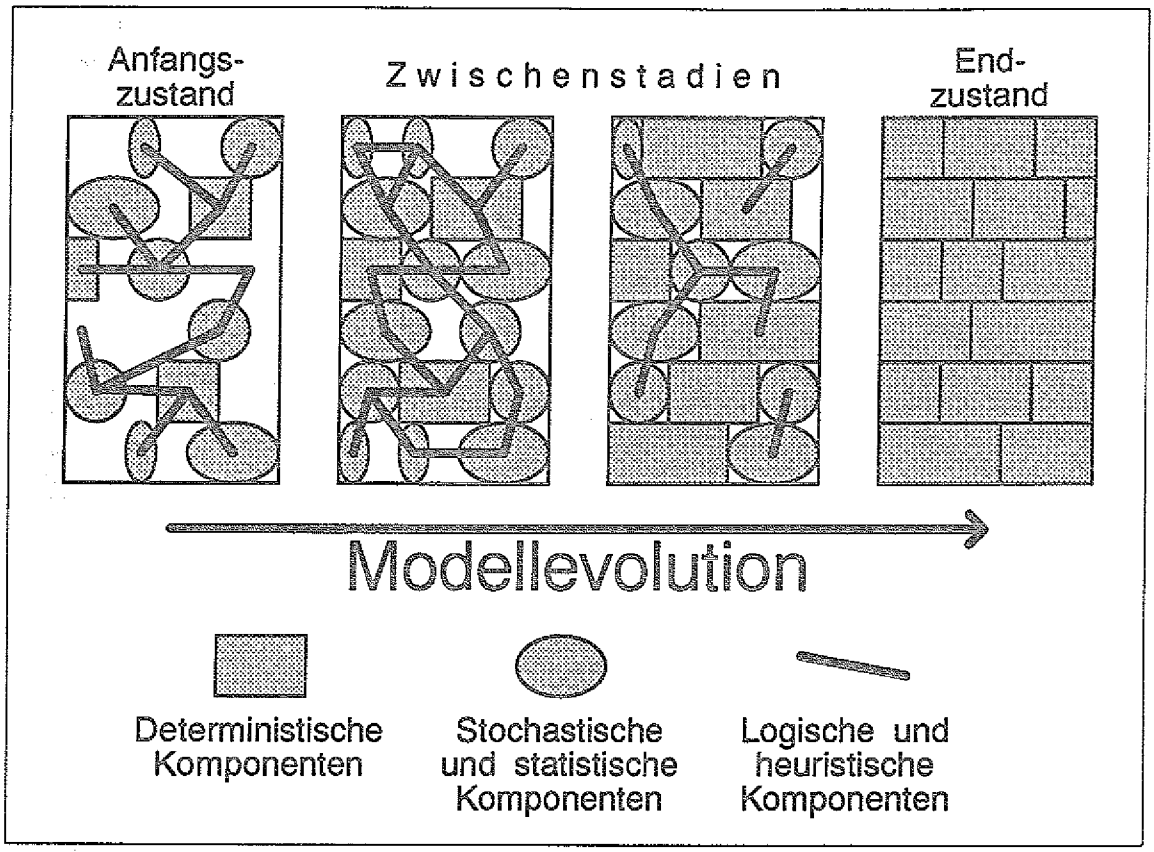


Abbildung 4.3: Evolutionäre Strategie zur Modellbildung bei biotechnischen Prozessen

4.4.2 Praktische Konsequenzen für den Laboreinsatz

Der evolutive Ansatz ist keineswegs der Standard. Vielmehr wird oft versucht, von vornherein — gewissermaßen "aus dem Stand" — strukturierte Differentialgleichungsmodelle an den Prozeßverlauf anzupassen! In Labor und Technikum gelangt man dann schnell in eine typische Situation: Um ein (Langzeit-)Experiment mit dem Ziel der Informationsgewinnung überhaupt durchführen zu können, muß eine vorläufige Kontrollstrategie schon vorhanden sein. Ein differenziertes Modell steht zu Anfang aber gerade nicht zur Verfügung! Die oben skizzierte Vorgehensweise bietet die Möglichkeit, zunächst mit einfachen Mitteln das Experiment grob zu steuern, um dann mit Hilfe der gewonnenen Informationen Modelle und Kontrollstrategie zu verfeinern (vgl. auch das Beispiel in 22.1).

Allerdings ist der oben skizzierte Ansatz mit den derzeit vorhandenen Prozeßüberwachungssystemen nur rudimentär durchführbar. Hierzu die Mittel bereitzustellen, ist eines der Ziele dieser Arbeit.

4.5 Zur Problematik des On-Line-Einsatzes

Die Prozeßdatenanalyse muß sich stets am Anspruch des On-Line-Einsatzes messen lassen. Bei dem vorgestellten Konzept steht dem On-Line-Einsatz im Prinzip nichts im Wege. Schließlich kann in diesem Rahmen auch der Fermenter selbst als ein Baustein mit Eingängen, Ausgängen und Konfigurationsdaten betrachtet werden [3] (vgl. dazu Abb. 4.4).

[3] Diese Vorstellung ist im allgemeinen eher problematisch, da ein biologisches System von seiner Komplexität her nicht mit einen "gewöhnlichen" technischen System verwechselt werden sollte.

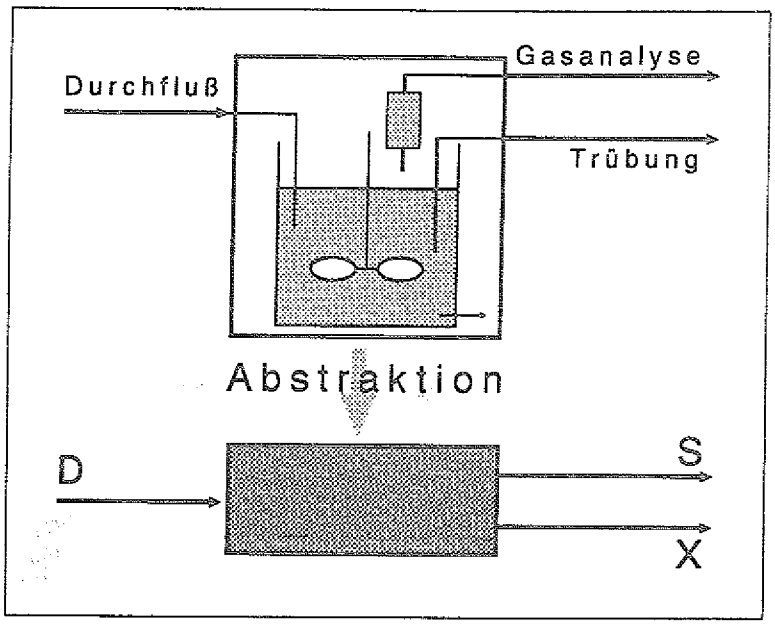


Abbildung 4.4: Ein-/Ausgangsschema eines Bioprozesses

4.5.1 Besonderheiten der On-Line-Anbindung

Einige technische Unterschiede müssen jedoch berücksichtigt werden :

- das biologische System bewegt sich in der natürlichen Zeit, d.h.
  - der Zeittakt ist vorgegeben.
  - es kann nur auf Vergangenheitswerte zugegriffen werden.
  - Zeitverzögerungen sind zu beachten.
- der Datenfluß vom und zum Fermenter läuft über eine Hardwareschnittstelle
  - Meß- und Steuerdaten müssen über eine elektronische Schnittstelle in den Rechner geführt werden.
  - Konfigurationsdaten (z.B. Elektrodenaustausch, Störungsbeseitigung) müssen an den Betreiber weitergegeben werden.
  - Beobachtungen und Meldungen werden über die Konsole eingegeben.

Diese Unterschiede sind nicht gravierend, da

- einerseits die einzelnen Datenanalysebausteine von vornherein für den On-Line-Einsatz und eine Synchronisation von außen ausgelegt werden,
- andererseits die Hardwareschnittstelle durch heutige Meßdatenerfassungssysteme vorbildlich verwaltet wird, so daß das Datenanalysesystem nur auf der Softwareseite mit der Meßdatenerfassung gekoppelt werden muß.

4.5.2 Methoden zur Anbindung von Datenanalysebausteinen an den Prozeß

Für den On-Line-Einsatz gibt es prinzipiell zwei Alternativen :

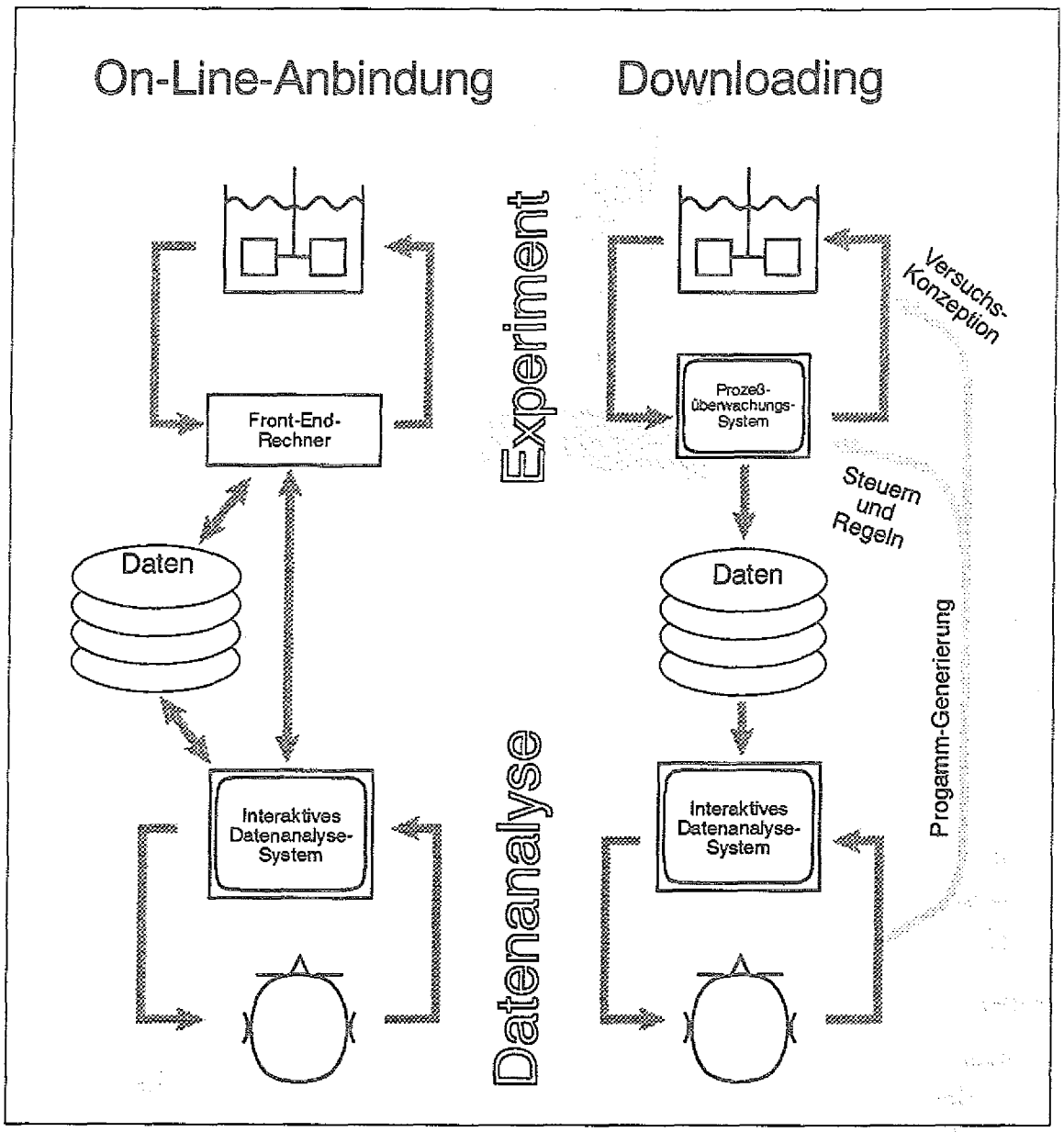


Abbildung 4.5: Methoden zur Anbindung eines Datenanalysesystems an den Prozeß

Das *Downloading,* bei dem eine erfolgreiche Datenanalysestrategie in prozeduraler Programmierung (off-line) automatisiert und vor Ort an das Meßdatenerfassungssystem angebunden wird, welches hierfür eine Softwareschnittstelle haben muß (vgl. Abb. 4.5a) [4].

Die *Direkte Ankopplung,* (4.5b) bei welcher der Front-End-Prozessor und der Datenanalyserechner über ein Computer-Netzwerk on-line miteinander verbunden werden. Dafür müssen sowohl das Meßdatenerfassungssystem als auch die Analysestation das Protokoll für die Datenübertragung im Netzwerk beherrschen.

[4] Ein Downloading-Ansatz führt somit zur Erstellung eines CASE-Tools für die Bioprozeßüberwachung (vgl. [Str88] [Gre89]).

4.5.3 Vergleich der Methoden

Die Vorteile des Downloadings sind unmittelbar auch die Nachteile der direkten Anbindung und umgekehrt. Es genügt also, die Vorteile aufzulisten.

- Vorteile des Downloadings :
  - Einsetzbarkeit bei Meßdatenerfassungssystemen verschiedener Hersteller mit nur geringen Anpassungsproblemen.
  - Entkopplung von Front-End-Prozessor und Datenanalyse-Rechner, welcher mit Prozeßsteuerungsaufgaben nicht belastet wird.
  - Kostengünstige Lösung, da der teure Analyserechner nur einmal angeschafft werden muß.
- Vorteile der direkten Anbindung :
  - Die Umsetzung von Prozeßüberwachungstrategien in eine prozedurale Form entfällt.
  - Regler und Kontrollstrategien können im geschlossenen Regelkreis erprobt werden.
  - Rechenintensive Kontrollstrategien können nur von leistungsstarken Rechnern durchgeführt werden.
  - Die Überwachung mehrerer Prozesse gleichzeitig ist möglich.

Eher problematisch ist die Möglichkeit zum interaktiven Eingriff in den Prozeß. Biologische Systeme reagieren sehr empfindlich und oft mit langfristigen Folgen auf Parameteränderung. Daher ist ein "unpersönlicher" Zugriff auf den Prozeß von einer weiter entfernten Konsole aus nicht unbedingt wünschenswert und verleitet zum "Spielen" mit den Parametern, was jede Versuchsplanung zunichte machen kann. Daher sollte dem Benutzer ein interaktiver Zugriff nur in einem fest umschriebenen Rahmen gestattet werden.

Teil II

Modelle für biotechnische Prozesse

Unter einem Modell soll — wie bereits in Kapitel 4 beschrieben — jegliche Form von mathematischer oder formaler Beschreibung eines gegebenen Systems verstanden werden. Eine Einteilung wurde vorgenommen in die Klassen:

- deterministische Modelle,
- statistische Modelle,
- stochastische Modelle,
- logische Modelle,
- heuristische Modelle.

Diese Modellklassen sollen nun in einer vereinheitlichenden Weise abgehandelt werden.

Kapitel 5

Deterministische Modelle

Unter einem deterministischen Modell soll in diesem Kontext ein deterministisches Differentialgleichungs-System verstanden werden, das meist in Form eines sogenannten Zustandsraummodells notiert wird. Die Grundzüge der Beschreibung von biotechnischen Prozessen mit Hilfe von deterministischen Modellen werden als bekannt vorausgesetzt (siehe z.B. [Ber83] [Baz83]).

Es hat in den letzten Jahren diverse Ansätze zur Beschreibung des mikrobiellen Stoffwechsels gegeben, die sich zwar nicht grundsätzlich voneinander unterscheiden, das Problem jedoch aus verschiedenen Blickwinkeln betrachten. Zu nennen wären unter anderem:

- die klassische Beschreibung durch *chemische Reaktionssysteme* [Fei77] (Reaktionsgleichungen und Stöchiometrie),
- die Methoden der *Enzymkinetik* [CB76] (Reaktionsgleichungen mit Stationaritätsannahmen),
- die *Metabolic Control Theory* [Red88] (Untersuchung geschwindigkeitsbestimmender Schritte auf enzymkinetischer Grundlage),
- das *Pathway Modelling* [Kac88] (Annahme konkreter Stoffwechselwege mit bekannter Stöchiometrie),
- die *phänomenologische Modellierung* [Ber83] (globale Wachstumsbesschreibung ohne konkrete Annahme von Stoffwechselwegen),
- die Formulierung *globaler Wachstumsgleichungen* [Roe83] (Gesamtstöchiometrie des Wachstums),
- die Beschreibung mit den Begriffen der *Thermodynamik* [WvD87] (Stoff- und Energieflüsse).

Jede Sichtweise eines Systems zieht eine entsprechende Methodik zur Datenanalyse nach sich. Für eine Rechner-Implementierung ist es jedoch erforderlich, die Methodenvielfalt auf die wesentlichen Ansätze zu reduzieren. Dazu wird eine unifizierende Theorie der Stoffwechselmodellierung benötigt, die in der Lage ist, die genannten Ansätze in einem einheitlichen Rahmen abzuhandeln. Erst auf der Basis einer solchen Unifizierung können Datenanalysemethoden auf benutzergerechte Weise implementiert werden.

Die nachfolgenden Ausführungen stellen den Versuch einer vereinheitlichenden Theorie dar, die auf der Begriffsbildung der *verallgemeinerten Reaktionsgleichung* sowie der Annahme *abstrakter zellinterner Intermediate* aufbaut. Sie ermöglicht die Einordnung bestehender Modellierungs- und Datenanalysemethoden. Insbesondere ergibt sich

- die Möglichkeit, alle Prozeßgrößen mit ihren jeweiligen physikalischen Dimensionen einzubeziehen,
- eine algebraische Formulierung der Zusammenhänge, die der automatischen Behandlung mit einem Computer-Algebra-System zugänglich ist,
- eine Herleitung der bekannten phänomenologischen Standardmodelle auf der Basis weniger Grundannahmen,

- eine Trennung der Stoffwechselbeschreibung von der Bilanzierung,
- eine saubere Unterscheidung zwischen den Begriffen des theoretischen, stöchiometrischen und empirischen Reaktionsraums.

Auf der Grundlage des allgemeinen Modells läßt sich im weiteren ein wohlsortiertes Methodenspektrum für den Bereich der Stoffwechselmodellierung begründen. Damit ist ein Fundament für die "computergerechte" Behandlung biotechnischer Modelle gelegt.

5.1 Modellierung biochemischer Reaktionen

5.1.1 Chemische Reaktionsgleichungen

5.1.1.1 Beschreibung von Einzelreaktionen

Chemische Umsetzungen lassen sich quantitativ durch ihre Reaktionsgleichungen in der Form

$$\mu_1 \cdot E_1 + \cdots + \mu_r \cdot E_r \xrightarrow{R} \eta_1 \cdot P_1 + \cdots + \eta_s \cdot P_s$$

	E_i	$, i = 1, \ldots, r$	– Edukte,
	P_j	$, j = 1, \ldots, s$	– Produkte,
mit	μ_i	$, i = 1, \ldots, r$	– (ganzzahlige) stöchiometrische Koeffizienten der E_i,
	η_i	$, j = 1, \ldots, s$	– (ganzzahlige) stöchiometrische Koeffizienten der P_j,
	R		– Reaktionsrate

beschreiben [1]. Kennzeichnet man die Edukte durch negative und die Produkte durch positive Koeffizienten, so erhält man die mathematisch leichter handhabbare Form:

$$\nu_1 \cdot X_1 + \cdots + \nu_{r+s} \cdot X_{r+s} \xrightarrow{R} \quad .$$

Für ein biochemisches Reaktionssystem soll zunächst angenommen werden, daß alle beteiligten Substanzen $X_1, \ldots, X_n$ in einer wässrigen Lösung homogen verteilt vorliegen. Die für die Reaktionsgeschwindigkeit entscheidenden Größen sind dann die Konzentrationen, die hier — ohne Einführung einer gesonderten Notation — erneut mit

$$X_1, X_2, X_3, \ldots$$

bezeichnet werden sollen. In der organischen Chemie sind i.a. die Reaktanden oder zumindest wichtige Zwischenprodukte genau bekannt. Die physikalischen Einheiten ergeben sich dann auf natürliche Weise aus der Vorstellung, daß die Reaktanden teilchenweise miteinander reagieren, d.h.

X_i	hat die Einheit	$\mathrm{Mol} = \frac{\mathrm{mol}}{\mathrm{l}}$,
ν_i	ist eine ganze Zahl und	
R	hat die Einheit	$\frac{\mathrm{Mol}}{\mathrm{h}} = \frac{\mathrm{mol}}{\mathrm{l \cdot h}}$.

Wenn nur die eine Reaktion $\nu_1 \cdot X_1 + \cdots + \nu_n \cdot X_n \xrightarrow{R}$ stattfindet, so ist folglich die Änderung der Konzentrationen beschreibbar durch:

$$\begin{aligned} \dot{X}_1 &= \nu_1 \cdot R \\ &\vdots \\ \dot{X}_n &= \nu_n \cdot R. \end{aligned}$$

Die Einheiten der $\dot{X}_i$ sind jeweils $\frac{\mathrm{Mol}}{\mathrm{h}}$.

[1] Die Reaktion kann durchaus auch in beiden Richtungen ablaufen. Dieser Fall läßt sich durch Annahme einer negativen Reaktionsrate R abdecken.

5.1.1.2 Verallgemeinerte Reaktionsgleichungen

In der Biochemie und Mikrobiologie werden als "Reaktanden" formal auch hoch- und nichtmolekulare Komponenten zugelassen. Einige Beispiele:

- Eine enzymatisch katalysierte Umsetzung unter Bildung eines Enzym-Substrat-Komplexes wird beschrieben durch [Eng81] (E - Enzym, S - Substrat, ES - Enzym-Substrat-Komplex, P-Produkt):

$$E + S \longleftrightarrow ES \longrightarrow E + P.$$

  Vom (aktiven) Enzym ist in der Praxis meist weder die Konzentration in Mol, noch die genaue molekulare Struktur oder der Reaktionsmechanismus bekannt.

- Die Glykolyse hat die Summengleichung [Leh82]

$$C_6H_{12}O_6 \longrightarrow 2\,C_3H_6O_3 + \Delta G.$$

  mit der freien Reaktionsenthalpie ΔG als nichtmolekularem "Produkt" der Reaktion. Die freie Reaktionsenthalpie unter Standardbedingungen ist dabei [Leh82]:

$$\Delta G^{0\prime} = 197.3\ \frac{\text{kJ}}{\text{Mol}}$$

- Formuliert man dieselbe Gleichung mit Hilfe des biologischen "Energiequantums" ATP, so lautet die Summengleichung:

$$C_6H_{12}O_6 + 2\,P + 2\,ADP \longrightarrow 2\,C_3H_6O_3 + 2\,ATP.$$

  In dieser Gleichung ist die Energiebilanz nicht mehr richtig, da die beiden ATP-Moleküle nur einen Teil der freiwerdenden Energie binden. Interessiert man sich nur für die "wesentlichen" Komponenten, so schreibt man dies z.B. in der Form:

$$C_6H_{12}O_6 \longrightarrow 2\,C_3H_6O_3 + 2\,ATP.$$

  Jetzt stimmt auch die Stöchiometrie nicht mehr!

- Das Wachstum eines Mikroorganismus läßt sich durchaus auch als chemische Umsetzung auffassen [Roe83]:

$$\text{Substrat} + \text{Stickstoffquelle} + O_2 \longrightarrow \text{Biomasse} + \text{Produkt} + CO_2 + H_2O.$$

  Die Biomasse ist eine nichtmolekulare Komponente und wird i.a. in der Einheit g/l gemessen.

Offenbar wird bei solchen *verallgemeinerten Reaktionsgleichungen* die molare Betrachtungsweise sinnlos. In der Praxis koexistieren die Konzentrationseinheiten:

- Mol = mol/l für definierte chemische Substanzen,
- g/l für wägbare Komponenten,
- g Trockenmasse/l, g Naßgewicht/l und C-Mol [2] für Biomasse,
- Units/l für Enzyme [3],
- % für flüssige Komponenten (z.B. Methanol oder Äthanol),
- Joule/l, Mol ATP und Mol NADH als Energiedichte-Einheiten.

Diese Einheiten sind oft nur schwierig oder — wie bei den Einheit Units/l — gar nicht ineinander umrechenbar.

[2] Ein C-mol ist die Menge an Biomasse, welche ein mol C-Atome enthält.

[3] Eine sehr problematische Einheit, da "Unit" eine Einheit für die Enzymaktivität und damit nur indirekt ein Mengenmaß ist [ALS77].

5.1.1.3 Verallgemeinerte stöchiometrische Koeffizienten

Um auch die verallgemeinerten Reaktionsgleichungen mit in der Praxis vorkommenden Einheiten handhaben zu können, müssen die stöchiometrischen Koeffizienten $\nu_1, \ldots, \nu_n$ und die Reaktionsrate R in der Gleichung

$$\nu_1 \cdot X_1 + \cdots + \nu_n \cdot X_n \xrightarrow{R}$$

verallgemeinert werden. Dies soll so geschehen, daß die Differentialgleichungen

$$\begin{aligned} \dot{X}_1 &= \nu_1 \cdot R \\ &\vdots \\ \dot{X}_n &= \nu_n \cdot R \end{aligned}$$

ihre Gültigkeit bewahren. Zu diesem Zweck müssen die physikalischen Einheiten so gewählt werden, daß die Einheitengleichungen

$$[\dot{X}_i] = \frac{1}{\mathrm{h}} \cdot [X_i] = [\nu_i] \cdot [R] \quad , i = 1, \ldots, n$$

erfüllt sind. Als physikalische Einheit für R kommt dabei z.B. in Frage:

$\frac{\mathrm{mol}}{\mathrm{h}}$ Dann ist R eine auf die Stoffmengen bezogene absolute Reaktionsgeschwindigkeit.

$\frac{\mathrm{mol}}{\mathrm{l.h}} = \frac{\mathrm{Mol}}{\mathrm{h}}$ Dann ist R eine auf die Stoffkonzentrationen bezogene absolute Reaktionsgeschwindigkeit.

$\frac{1}{\mathrm{h}}$ Dann ist R eine auf eine Stoffmengeneinheit bezogene spezifische Reaktionsgeschwindigkeit.

Weitere Einheiten sind denkbar. Kennzeichnend ist lediglich, daß stets "die Zeit im Nenner steht".

Ist die Einheit von R gewählt, so folgt:

$$[\nu_i] = \frac{1}{\mathrm{h}} \cdot \frac{[X_i]}{[R]} \quad , i = 1, \ldots, n$$

D.h. die Beschränkung auf ganze Zahlen für die ν_i (vgl. 5.1.1.1) muß bei der Verallgemeinerung aufgegeben werden. Aufgrund der verschiedenen Einheiten werden jedoch die in der Chemie üblichen stöchiometrischen Beziehungen für die Elemente C, O, H, N, S, P, ...im allgemeinen sinnlos!

5.1.1.4 Normierung von Koeffizienten

Es ist zu beachten, daß die verallgemeinerten stöchiometrischen Koeffizienten nicht eindeutig sind, so daß Normierungen vorgenommen werden können. Die Normierung muß derart erfolgen, daß

$$\frac{[X_j]}{[\nu_j] \cdot [R]} \quad , i = 1, ..., n \quad \text{eine Zeiteinheit ist.}$$

Hierfür bestehen zwei Möglichkeiten (C eine beliebige Konstante):

a) Normierung von R und ν_i:

$$\frac{X_i}{\nu_i \cdot R} = \frac{X_i}{\frac{\nu_i}{C} \cdot (R \cdot C)} \quad , i = 1, \ldots, n$$

b) Normierung von X_i und R:

$$\frac{X_i}{\nu_i \cdot R} = \frac{\frac{X_i}{C}}{\nu_i \cdot \frac{R}{C}} \quad ,i = 1, \ldots, n$$

c) Normierung von X_i und ν_i:

$$\frac{X_i}{\nu_i \cdot R} = \frac{\frac{X_i}{C}}{\frac{\nu_i}{C} \cdot R} \quad ,i = 1, \ldots, n$$

Pro Reaktionsgleichung kann so ein stöchiometrischer Koeffizient auf 1 normiert werden.

5.1.1.5 Beispiel

Biomasseproduktion (die Zahlen entsprechen etwa denen der Backhefeproduktion [Del87], gTM steht für "Gramm Trockenmasse"):

$$180 \text{ g Glucose} + 180 \text{ g Sauerstoff} \xrightarrow{R} 97 \text{ gTM Biomasse} + 250 \text{ g } CO_2$$

Die Einheiten sind:

$$[\text{Glucose}] = [\text{Sauerstoff}] = [CO_2] = \text{g}, \quad [\text{Biomasse}] = \text{gTM} \quad \text{und} \quad [R] = \frac{1}{\text{h}}$$

a) Normierung von R und ν_i mit $C = 180$ g:

$$1.00 \text{ Glucose} + 1.00 \text{ Sauerstoff} \xrightarrow{R_a} 0.54 \frac{\text{gTM}}{\text{g}} \text{ Biomasse} + 1.39 \, CO_2$$

Einheiten:

$$[\text{Glucose}] = [\text{Sauerstoff}] = [CO_2] = \frac{\text{g}}{\text{l}}, \quad [\text{Biomasse}] = \frac{\text{gTM}}{\text{l}} \quad \text{und} \quad [R] = \frac{1}{\text{h}}$$

b) Normierung von X und R_a aus a) mit $C = \frac{1}{\text{l}}$:

$$1.00 \text{ Glucose} + 1.00 \text{ Sauerstoff} \xrightarrow{R_a} 0.54 \frac{\text{gTM}}{\text{g}} \text{ Biomasse} + 1.39 \, CO_2$$

Einheiten:

$$[\text{Glucose}] = [\text{Sauerstoff}] = [CO_2] = \frac{\text{g}}{\text{l}}, \quad [\text{Biomasse}] = \frac{\text{gTM}}{\text{l}} \quad \text{und} \quad [R] = \frac{\text{g}}{\text{l} \cdot \text{h}}$$

c) Normierung von X und ν_i aus der ersten Gleichung mit $C = 180 \frac{1}{\text{l}}$:

$$1.00 \frac{\text{g}}{\text{l}} \text{ Glucose} + 1.00 \frac{\text{g}}{\text{l}} \text{ Sauerstoff} \xrightarrow{R_c} 0.54 \frac{\text{gTM}}{\text{l}} \text{ Biomasse} + 1.39 \frac{\text{g}}{\text{l}} \, CO_2$$

Einheiten:

$$[\text{Glucose}] = [\text{Sauerstoff}] = [CO_2] = \frac{\text{g}}{\text{l}}, \quad [\text{Biomasse}] = \frac{\text{gTM}}{\text{l}} \quad \text{und} \quad [R] = \frac{\text{g}}{\text{h}}$$

5.1.2 Reaktionssysteme

5.1.2.1 Beschreibung eines Reaktionssystems

Betrachtet wird ein Reaktionssystem mit n beteiligten Substraten und m (verallgemeinerten) Reaktionsgleichungen:

$$\begin{aligned} \nu_{11} \cdot X_1 + \cdots + \nu_{1n} \cdot X_n &\xrightarrow{R_1} \\ &\vdots \\ \nu_{m1} \cdot X_1 + \cdots + \nu_{mn} \cdot X_n &\xrightarrow{R_m} . \end{aligned}$$

Kommt eine Komponente in einer Gleichung nicht vor, so ist der entsprechende Koeffizient 0 zu setzen. Wird eine Komponente von außen zugeführt, so kann man das schreiben als

$$\nu \cdot X \xrightarrow{R} \quad \text{und analog} \quad \xrightarrow{R} \nu \cdot X$$

für eine Komponente, die das System verläßt. Die Änderung der Konzentrationen wird jetzt beschrieben durch

$$\begin{aligned} \dot{X}_1 &= \nu_{11} \cdot R_1 + \cdots + \nu_{m1} \cdot R_m \\ &\vdots \\ \dot{X}_n &= \nu_{1n} \cdot R_1 + \cdots + \nu_{mn} \cdot R_m \end{aligned}$$

oder in der Matrizenschreibweise

$$\begin{pmatrix} \dot{X}_1 \\ \vdots \\ \dot{X}_n \end{pmatrix} = \begin{pmatrix} \nu_{11} & \cdots & \nu_{m1} \\ \vdots & \ddots & \vdots \\ \nu_{1n} & \cdots & \nu_{mn} \end{pmatrix} \cdot \begin{pmatrix} R_1 \\ \vdots \\ R_m \end{pmatrix} .$$

Kurz:

$$\dot{X} = \nu \cdot R$$

mit:
- X – Konzentrationsvektor,
- ν – Matrix der stöchiometrischen Koeffizienten,
- R – Vektor der Reaktionsraten.

In anderen Worten: ν beschreibt den Reaktionsmechanismus, R die Kinetik und X den Systemzustand [4].

5.1.2.2 Reaktionskinetiken

Über die Reaktionsgeschwindigkeiten wurde bisher keinerlei Annahme gemacht. Sie werden im homogenen System als Funktionen der einzelnen Konzentrationen sowie der Konzentrationen weiterer eventuell im System vorhandener Komponenten

$$Y_1, \ldots, Y_q$$

[4] Diese Zerlegung in eine konstante Koeffizientenmatrix und einen zustandsabhängigen Vektor bildet die Grundlage der *Metabolic Control Theory*, die unter anderem versucht, allgemeine Eigenschaften des Reaktionssystems aus den Eigenschaften der Matrix abzuleiten [KW86] [Red88] .

angesehen. Mit dem Vektor

$$Y = (Y_1, \ldots, Y_q)$$

ist also

$$R = R(X,Y) = (R_1(X,Y), \cdots, R_m(X,Y)).$$

Beispiele für Kinetiken [ALS77] [CB76]:

- Aus der Stoßtheorie chemischer Reaktionen erhält man die "Massenwirkungskinetiken" (vgl. Abb. 5.1a)

  $$R(X_1, \ldots, X_n) = c \cdot X_1^{p_1} \cdot \ldots \cdot X_n^{p_n} \quad \text{(Kinetik } (p_1 + \cdots + p_n)\text{-ter Ordnung)}.$$

  Speziell bei nur einer beteiligten Komponente:

  $$R(X) = c \cdot X^p \quad \text{(Kinetik } p\text{-ter Ordnung)}.$$

- Aus der Enzymkinetik kennt man die Sättigungskinetik (Abb. 5.1b)

  $$R(X) = R_X \cdot \frac{X}{X + K_X}.$$

  Es handelt sich um eine Kinetik "gemischter Ordnung", da

  $$R(X) \approx \begin{cases} \frac{R_X}{K_X} \cdot X & \text{für} \quad X \sim 0 \\ R_X & \text{für} \quad X \gg 0. \end{cases}$$

- Eine Verallgemeinerung ist die sigmoide Kinetik (Abb. 5.1d)

  $$R(X) = R_X \cdot \frac{X^p}{X^p + K_X^p}.$$

- Eine Inhibition durch ein zweites Substrat beschreibt man oft durch einen multiplikativen Hemmterm in der Form

  $$R(X,Y) = R_X(X) \cdot \frac{I_Y}{Y + I_Y},$$

  wobei $R_X(X)$ eine "gewöhnliche" Kinetik ist. Dem liegt die Vorstellung zugrunde, daß Substrat und Inhibitor unabhängig voneinander einwirken. Speziell für den Fall, daß $X = Y$ ist und R_X eine Sättigungskinetik, entsteht die bekannte Inhibierungskinetik (Abb. 5.1c)

  $$\begin{aligned} R(X) &= R_X \cdot \frac{X}{X + K_X} \cdot \frac{I_X}{X + I_X} \\ &= R_X \cdot \frac{X}{\frac{1}{I_X} X^2 + \frac{I_X + K_X}{I_X} \cdot X + K_X} \\ &\sim R_X \cdot \frac{X}{\frac{1}{I_X} X^2 + X + K_X}, \qquad \text{sofern } I_X \gg K_X . \end{aligned}$$

Weitere Ansätze finden sich z.B. in [CB76] [Eng81] [TY76] [Bla81] [Lev80] [Koc82].

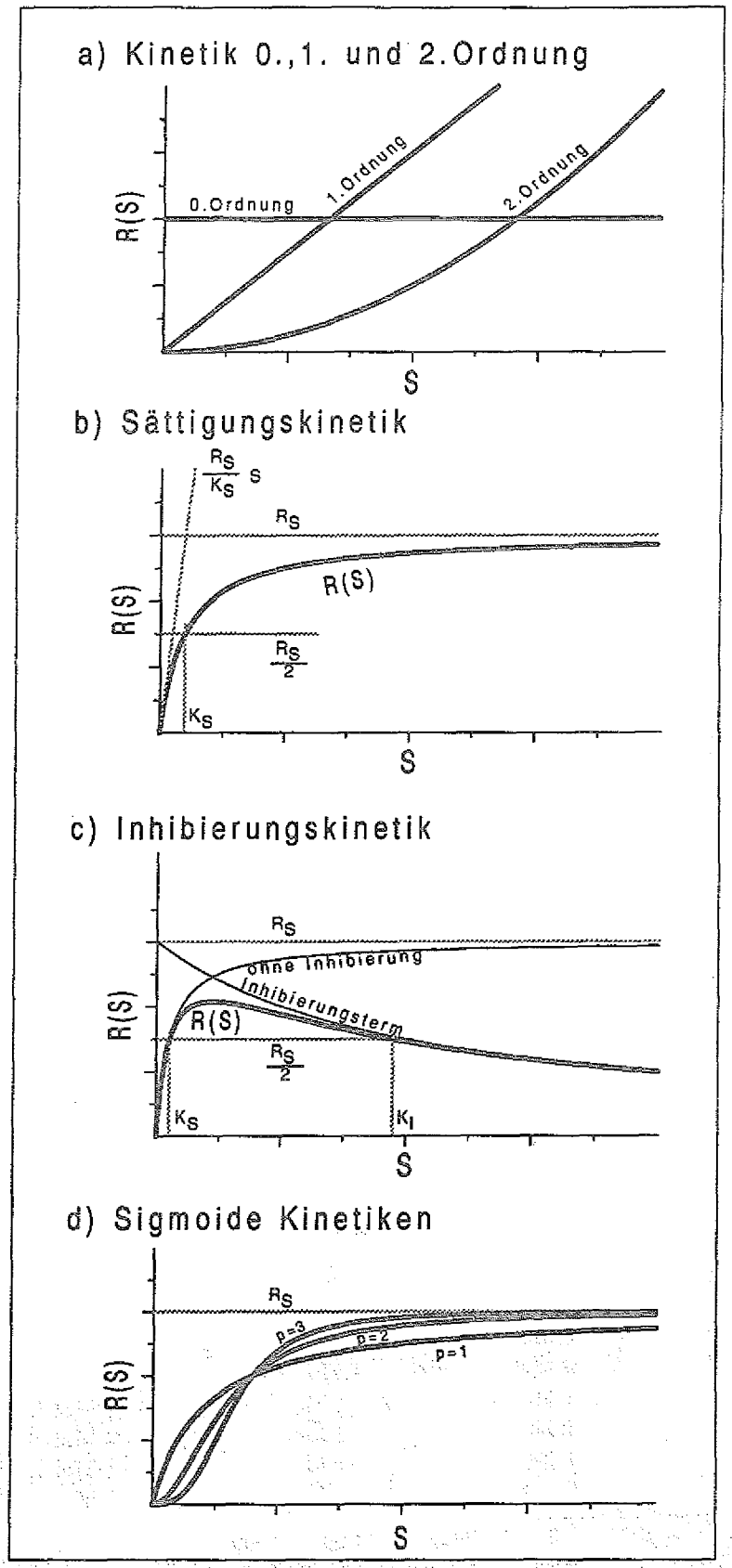


Abbildung 5.1: Verschiedene Kinetiken

5.1.2.3 Der theoretische Reaktionsraum

Das Differentialgleichungssystem $\dot{X} = \nu \cdot R$ für ein Reaktionssystem kann man anders schreiben als

$$\begin{pmatrix} \dot{X}_1 \\ \vdots \\ \dot{X}_n \end{pmatrix} = \begin{pmatrix} \nu_{11} \\ \vdots \\ \nu_{1n} \end{pmatrix} \cdot R_1 + \ldots + \begin{pmatrix} \nu_{m1} \\ \vdots \\ \nu_{mn} \end{pmatrix} \cdot R_m$$

oder

$$\dot{X} = \sum_{j=1}^{m} \nu_j \cdot R_j \quad \text{mit konstanten Vektoren} \quad \nu_j = \begin{pmatrix} \nu_{j1} \\ \vdots \\ \nu_{jn} \end{pmatrix}.$$

Aus der Gleichung folgt:

$$\begin{aligned} \frac{d}{dt}(X(t) - X(0)) &= \sum_{j=1}^{m} \nu_j \cdot R_j \\ \Rightarrow \qquad X(t) &= \sum_{j=1}^{m} [\nu_j \cdot \int_0^t R_j \, d\tau] + X(0) \end{aligned}$$

Somit liegt $X(t)$ zu allen Zeiten in dem von

$$X(0) \text{ und } \nu_1, \ldots \nu_m \text{ aufgespannten affinen Unterraum des } \Re^n.$$

Dieser rang(ν)-dimensionale [5] Unterraum des *Zustandsraums* $\Re^n$ heißt der *theoretische Reaktionsraum* des Reaktionssystems [6]. Er kann alternativ durch $(n - \text{rang}(\nu))$ lineare Gleichungen beschrieben werden [Fis75] und überträgt das Konzept der stöchiometrischen Bilanzen auf verallgemeinerte Reaktionsgleichungen.

5.1.2.4 Ein Beispiel

Die bereits in Abschnitt 5.1.1.2 erwähnte Enzym-Reaktion

$$1 \text{ Mol } E + 1 \text{ Mol } S \overset{\alpha_+, \alpha_-}{\longleftrightarrow} 1 \text{ Mol } ES \xrightarrow{\beta} 1 \text{ Mol } E + 1 \text{ Mol } P \quad (\alpha_+, \alpha_-, \beta \geq 0)$$

mit den von E, S, ES und P abhängigen Kinetiken

$$\alpha_+, \alpha_-, \beta \geq 0$$

setzt sich aus drei Teilreaktionen zusammen, die auf die Gleichung

$$\begin{aligned} \begin{pmatrix} \dot{E} \\ \dot{S} \\ \dot{ES} \\ \dot{P} \end{pmatrix} &= \begin{pmatrix} -1 \text{ Mol} & 1 \text{ Mol} & 1 \text{ Mol} \\ -1 \text{ Mol} & 1 \text{ Mol} & 0 \text{ Mol} \\ 1 \text{ Mol} & -1 \text{ Mol} & -1 \text{ Mol} \\ 0 \text{ Mol} & 0 \text{ Mol} & 1 \text{ Mol} \end{pmatrix} \cdot \begin{pmatrix} \alpha_+ \\ \alpha_- \\ \beta \end{pmatrix} \\ &= \begin{pmatrix} -1 \text{ Mol} \\ -1 \text{ Mol} \\ 1 \text{ Mol} \\ 0 \text{ Mol} \end{pmatrix} \cdot \alpha_+ + \begin{pmatrix} 1 \text{ Mol} \\ 1 \text{ Mol} \\ -1 \text{ Mol} \\ 0 \text{ Mol} \end{pmatrix} \cdot \alpha_- + \begin{pmatrix} 1 \text{ Mol} \\ 0 \text{ Mol} \\ -1 \text{ Mol} \\ 1 \text{ Mol} \end{pmatrix} \cdot \beta \end{aligned}$$

[5] Der Rang einer Matrix ν ist gleich der Zahl der linear unabhängigen Spalten.

[6] [Lia89] verwendet bei einer exakt bekannten Stöchiometrie den Begriff des *stöchiometrischen Reaktionsraums*. Hier soll jedoch verallgemeinernd vom theoretischen Reaktionsraum gesprochen werden. Der Begriff des stöchiometrischen Reaktionsraums (im engeren Sinne) wird dann in Abschnitt 6.6 eingeführt.

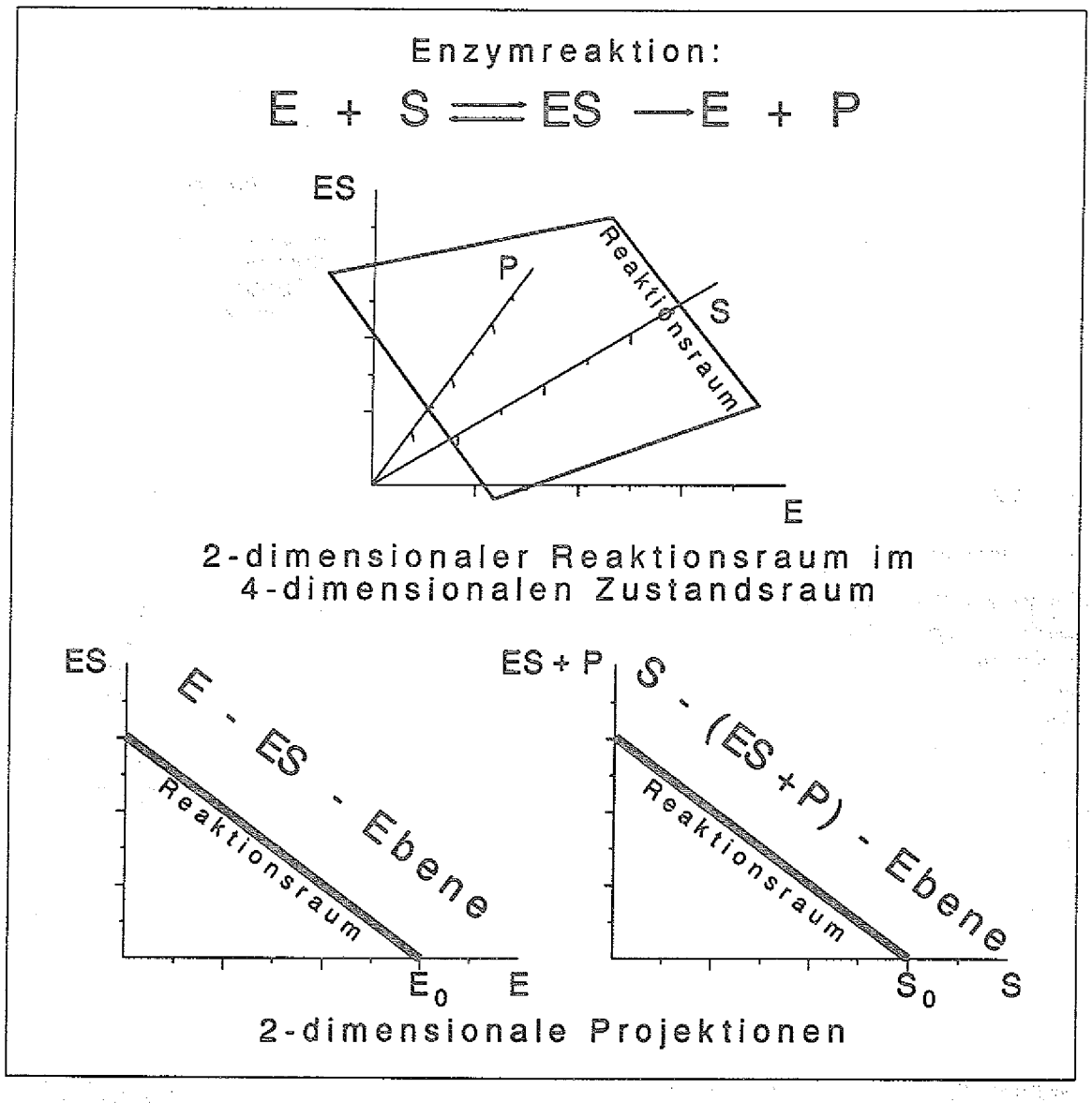


Abbildung 5.2: Theoretischer Reaktionsraum bei einer Enzymreaktion

führen. Der Rang der Matrix ist — aufgrund der linearen Abhängigkeit von Spalte 1 und 2 — gleich 2, so daß zu jedem Zeitpunkt t gilt:

$$\begin{pmatrix} E(t) \\ S(t) \\ ES(t) \\ P(t) \end{pmatrix} = \begin{pmatrix} E(0) \\ S(0) \\ ES(0) \\ P(0) \end{pmatrix} + \begin{pmatrix} -1\ \text{Mol} \\ -1\ \text{Mol} \\ 1\ \text{Mol} \\ 0\ \text{Mol} \end{pmatrix} \cdot c_\alpha(t) + \begin{pmatrix} 1\ \text{Mol} \\ 0\ \text{Mol} \\ -1\ \text{Mol} \\ 1\ \text{Mol} \end{pmatrix} \cdot c_\beta(t)$$

mit gewissen zeitabhängigen Koeffizienten $c_\alpha(t)$, $c_\beta(t)$. Der 2-dimensionale theoretische Reaktionsraum kann alternativ beschrieben werden durch die beiden Gleichungen

$$\begin{array}{lcll} E + ES & = & E(0) + ES(0) & \text{(Enzymbilanz)} \\ S + ES + P & = & S(0) + ES(0) + P(0) & \text{(Substratbilanz)} \end{array}$$

Abbildung 5.2 verdeutlicht diese Zusammenhänge.

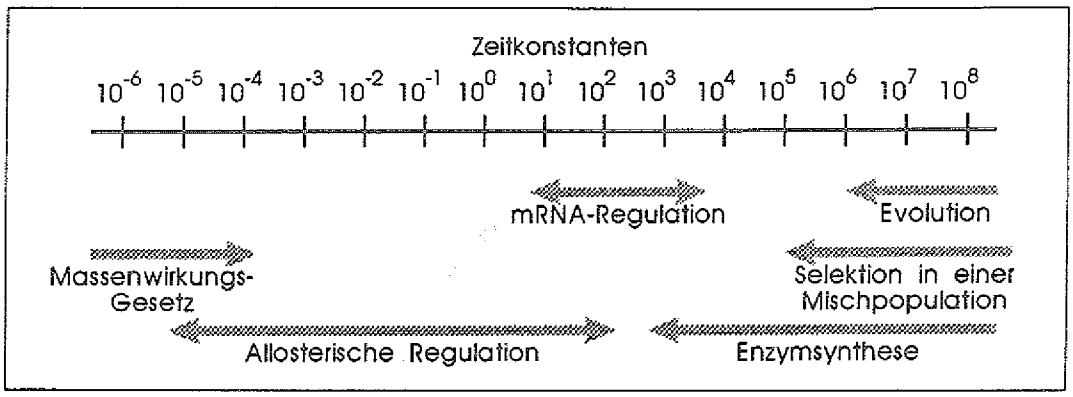


Abbildung 5.3: Verschiedene Zeitkonstanten bei biologischen Systemen (nach [Roe83])

5.1.3 Annahmen zur Modellvereinfachung

5.1.3.1 Separierung der Zeitskalen bei biologischen Systemen

Bei biologischen Systemen liegt meist eine starke Separierung der Zeitskalen vor, d.h. die verschiedenen dynamischen Vorgänge haben Zeitkonstanten mit sehr verschiedenen Größenordnungen. Das Spektrum der Zeitkonstanten reicht von den Zeitkonstanten des Massenwirkungsgesetzes über die Konstanten der allosterischen und mRNA-Regulation bis hin zu den Größenordnungen der biotischen Evolution [Roe83] (vgl. Abb. 5.3).

Bei der Modellbildung beschränkt man sich auf einen bestimmten Bereich der Zeitskala. Prozesse mit größeren Zeitkonstanten können dann als (im betrachteten Zeitraum) konstant angesehen werden, während Prozesse mit kleineren Zeitkonstanten als stationär betrachtet werden (wobei stabile stationäre Zustände vorliegen müssen, vgl. auch [LL88]). Zwei Beispiele:

- Reaktionswege in der Zelle sind im allgemeinen regulierte Enzymsysteme, die in mehreren Stufen ablaufen:

$$S_0 \rightarrow S_1 \rightarrow S_2 \rightarrow \cdots \rightarrow S_{k-1} \rightarrow S_k$$

  Bei schnellen Änderungen der Substratkonzentrationen weisen sie oft ein kompliziertes dynamisches Verhalten auf (Schwingungen, Zustandsübergänge etc.). Ändert sich die Substratkonzentration — so wie in einem Bioreaktor — jedoch nur langsam, so kann angenommen werden, daß sich das System stets in einem stationären Zustand befindet. In diesem Fall ist

$$S_k = f(S_0)$$

  eine Funktion der Eingangskonzentration S.

- Aus diesem Grund kann bei vielen Mikroorganismen das spezifische Wachstum $\mu(S)$ auf einem Substrat S durch die Monod-Kinetik

$$\mu(S) = \mu_S \frac{S}{S + K_S}$$

  beschrieben werden. μ_S und K_S können sich durch Mutation durchaus verändern, werden jedoch für den Zeitraum eines einzelnen Experiments als konstant angesehen.

5.1.3.2 Stationaritätskonzepte

Die Annahme einer Separierung der Zeitskalen entspricht gewissen Stationaritätskonzepten für Reaktionssysteme, deren Unterschiede verdeutlicht werden müssen. Betrachtet wird ein Glied S in einer Reaktionskette

$$\ldots \longleftrightarrow R \overset{\alpha_+,\alpha_-}{\longleftrightarrow} S \overset{\beta_+,\beta_-}{\longleftrightarrow} T \longleftrightarrow \ldots \quad (\alpha_+,\alpha_-,\beta_+,\beta_- \geq 0).$$

Für die Komponente S gilt dann (wenn — der Einfachheit halber — alle stöchiometrischen Koeffizienten 1 sind):

$$\dot{S} = \alpha_+ - \alpha_- - \beta_+ + \beta_-.$$

Man kann nun drei verschieden starke Stationaritätsannahmen treffen:

1. Fließgleichgewicht (im folgenden einfach als *Stationarität* bezeichnet) bei dem sich alle Hin- und Rückreaktionen aufwiegen:

$$\alpha_+ - \alpha_- = \beta_+ - \beta_-$$

2. *Quasi-Stationarität*, bei der angenommen wird, daß eine Seite der Reaktionskette vernachlässigbar kleine Reaktionsraten gegenüber der anderen hat. Ist dies z.B. die rechte Seite (d.h. $\beta_\pm \gg \alpha_\pm$), so ergibt sich:

$$\beta_+ - \beta_- = 0$$

   Meist wird eine solche Annahme als Approximation zur Vereinfachung von Rechnungen benutzt.

3. *Konstanz*, bei der alle Reaktionsraten vernachlässigbar sind:

$$\alpha_+,\alpha_-,\beta_+,\beta_- = 0$$

Im Falle der Stationarität und der Konstanz gilt die Beziehung

$$\dot{S} = 0,$$

wogegen die Quasistationarität sich nicht auf einen Reaktionspartner sondern auf eine Hin- und Rückreaktion bezieht.

5.1.3.3 Ein allgemeiner Modellierungsansatz für Reaktionssysteme

Die Modellbildung für Reaktionssysteme kann nun ganz allgemein auf folgendes Schema zurückgeführt werden:

1. Man formuliere das (verallgemeinerte) Reaktionssystem und stelle das zugehörige Differentialgleichungssystem

$$\dot{X} = \nu \cdot R(X,Y)$$

   auf (vgl. 5.1.2.1).

2. Für alle dynamischen Vorgänge mit sehr großen Zeitkonstanten nehme man Konstanz an. Es folgt

$$X_i = \text{const} \quad \textit{(Konstanzprinzip)}$$

   für die langsam veränderlichen Komponenten.

3. Hingegen seien alle Vorgänge mit kleinen Zeitkonstanten stationär, d.h.:

$$\dot{X}_i = 0 \iff 0 = \nu_{1i} \cdot R_1 + \cdots + \nu_{mi} \cdot R_m \quad \textit{(Stationaritätsprinzip)}.$$

Es resultieren Gleichungen, mit denen einige der R_i eliminiert werden können. Die Annahme der Quasi-Stationarität, welche für einzelne Reaktionen getroffen werden muß, hat rechnerisch ähnliche Konsequenzen (Elimination einiger Reaktionsterme).

4. Man berechne den theoretischen Reaktionsraum. Die sich dabei ergebenden Gleichungen können zur Elimination einiger X_i-Terme benutzt werden.

5.1.3.4 Ein Beispiel

Bei dem Enzym-Reaktionssystem (vgl. 5.1.2.4)

$$S + E \overset{\alpha_+,\alpha_-}{\longleftrightarrow} ES \xrightarrow{\beta} E + P$$

mit den Massenwirkungskinetiken

$$\begin{aligned} \alpha_+(E,S) &= k_+ \cdot E \cdot S \\ \alpha_-(ES) &= k_- \cdot ES \\ \beta(ES) &= k_\beta \cdot ES \end{aligned}$$

sieht das wie folgt aus:

1. Differentialgleichungssystem der Reaktion (hier einheitslos notiert):

$$\begin{pmatrix} \dot{E} \\ \dot{S} \\ \dot{ES} \\ \dot{P} \end{pmatrix} = \begin{pmatrix} -1 & 1 & 1 \\ -1 & 1 & 0 \\ 1 & -1 & -1 \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} \alpha_+ \\ \alpha_- \\ \beta \end{pmatrix}$$

2. Konstanzannahme für k_+, k_- und k_β:

$$k_+ = \text{const.}, \quad k_- = \text{const.}, \quad k_\beta = \text{const.},$$

(was bei nachlassender Enzym-Aktivität infolge einer Degenerierung durchaus nicht zu gelten braucht [7]).

3. Stationaritätsannahme für ES:

$$\dot{ES} = 0.$$

Es folgt

$$\alpha_+ - \alpha_- - \beta = 0 \quad \Leftrightarrow \quad \alpha_+ = \alpha_- + \beta,$$

d.h. α_+, α_- und β sind linear abhängig. Bei den speziell angenommenen Kinetiken folgt

$$(k_- + k_\beta) \cdot ES = k_+ \cdot E \cdot S \iff ES = \frac{k_+}{k_- + k_\beta} \cdot E \cdot S.$$

[7] Eine Enzymdegenerierung wird oft auch durch Annahme einer weiteren Reaktion $E \rightarrow E_{deg}$ beschrieben [HS86].

4. Der theoretische Reaktionsraum wurde bereits vorher ausgerechnet:

$$\begin{aligned} E + ES &= E(0) + ES(0) \\ S + ES + P &= S(0) + ES(0) + P(0). \end{aligned}$$

Es folgt z.B.:

$$E = E(0) + ES(0) - ES.$$

Aus den so abgeleiteten Gleichungen berechnet man ES als Funktion von S und daraus die bekannte Michaelis-Menten-Kinetik:

$$v(S) = \dot{P} = k_\beta \cdot ES = k_\beta \cdot (E(0) + ES(0)) \cdot \frac{S}{S + \frac{k_- + k_\beta}{k_+}}.$$

In der Regel startet man eine Enzym-Reaktion durch Zugabe des Substrates, so daß ferner der Startwert

$$ES(0) = 0$$

angenommen werden kann.

Bemerkung: Mit der stärkeren Quasi-Stationaritätsannahme für die erste Reaktion erhält man zunächst das Gleichgewicht

$$ES = \frac{k_+}{k_-} \cdot E \cdot S$$

und daraus durch Elimination von ES und Ausnutzung der stöchiometrischen Beziehung:

$$v(S) = k_\beta \cdot ES = k_\beta E(0) \frac{S}{S + \frac{k_-}{k_+}}$$

Diese (historisch frühere) Herleitung der Michaelis-Menten-Kinetik liefert somit eine Approximation an die Gleichung, die man unter der Stationaritätsannahme erhält.

5.2 Modellierung des Wachstums von Mikroorganismen

5.2.1 Ein einfaches Wachstumsmodell

5.2.1.1 Unstrukturierte Modelle

Bei der unstrukturierten (oder phänomenologischen) Betrachtungsweise wird die Biomasse nicht weiter in Kompartimente unterteilt. Das Wachstum wird als verallgemeinerte chemische Reaktion

$$\text{Substrat} + \text{Sauerstoff} + \text{Stickstoffquelle} \xrightarrow{R} \text{Biomasse} + \text{Produkt} + CO_2 + H_2O$$

aufgefaßt, die durch die Biomasse katalysiert wird (Autokatalyse). Alle internen Komponenten der Zelle werden als im Fließgleichgewicht befindlich angenommen, so daß ein einfaches durch Funktionen beschreibbares Eingangs/Ausgangs-Schema für Substrate und Produkte entsteht (siehe Abb. 5.4).

Der unstrukturierte Ansatz erweist sich somit als ein stationärer Modellierungsansatz für ein komplexes Reaktionssystem. Diese Vorstellung ist mit Sicherheit zu einfach, sobald Vorgänge wie Reservestoffwechsel oder Enzyminduktion ins Spiel kommen. Allerdings sollte man bedenken, daß der Biotechnologe stets bemüht ist, seine Zellen in einer optimalen Umgebung zu kultivieren, und somit Limitierungen weitestgehend zu unterbinden versucht. Man kann also hoffen, daß phänomenologische Modelle einen Mikroorganismus in einem gewissen Arbeitsbereich zufriedenstellend beschreiben.

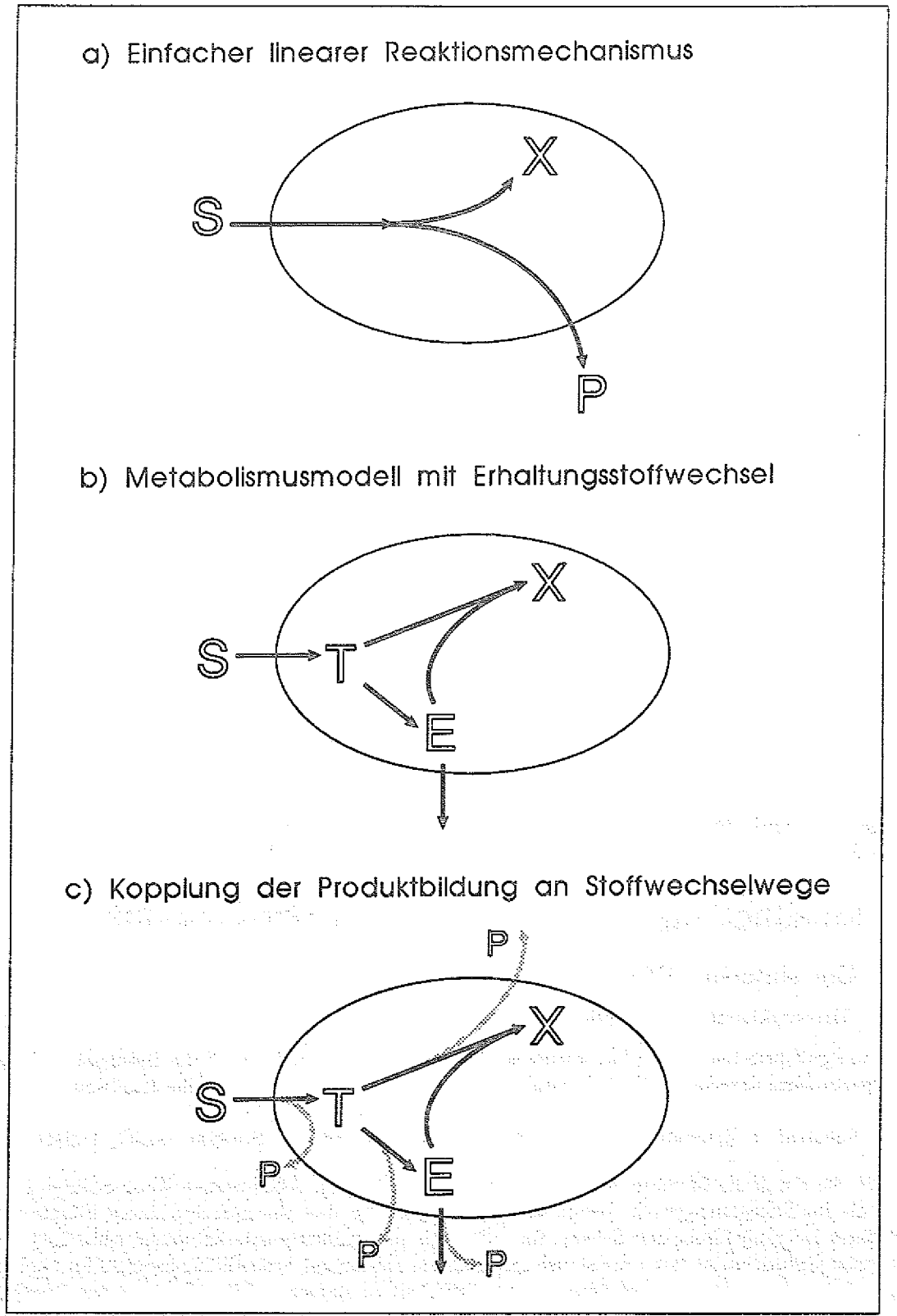


Abbildung 5.4: Verschiedene Stoffwechselmodelle

5.2.1.2 Das allgemeine unstrukturierte Wachstumsmodell

Alle Überlegungen beziehen sich zunächst wieder auf das homogen gerührte geschlossene System (Batch-Reaktor). Mit der allgemein üblichen Bezeichnungsweise (vgl. Abb. 5.4a)

$$\begin{array}{lll} X & - & \text{für Biomasse,} \\ S & - & \text{für Substrat,} \\ P & - & \text{für Produkt} \end{array}$$

sowie der — recht starken — Stationäritätsannahme, daß das System mit diesen Variablen vollständig beschrieben werden kann, schreibt man aus der Reaktionsgleichung

$$\nu_S \cdot S \xrightarrow{R} \nu_X \cdot X + \nu_P \cdot P$$

sofort das allgemeine Modell

$$\begin{aligned} \dot{X} &= \nu_X \cdot R \\ \dot{S} &= \nu_S \cdot R \\ \dot{P} &= \nu_P \cdot R \end{aligned}$$

für die Komponenten X, S und P hin. Nach einer entsprechenden Normierung kann weiterhin $\nu_x = 1$ gewählt werden (vgl. 5.1.1.4).

5.2.1.3 Spezifische Wachstums-, Substrataufnahme- und Produktbildungsrate

Man pflegt in der Biotechnologie alle Reaktionsraten auf die Biomasse (=Katalysator) zu beziehen [8] . Dies ist sinnvoll, da der Durchsatz jeder im System auftretenden Reaktion proportional zur Biomassekonzentration ist. Man gelangt damit

$$\begin{array}{lll} \text{zur } \textit{spezifischen Wachstumsrate} & \mu = \frac{R}{X}, \\ \text{zur } \textit{spezifischen Substrataufnahmerate} & \sigma = \frac{R \cdot \nu_S}{X} & [9] \\ \text{zur } \textit{spezifischen Produktbildungsrate} & \pi = \frac{R \cdot \nu_P}{X} \end{array}$$

und erhält das in der Biotechnologie allgemein übliche unstrukturierte Modell für den Batch-Reaktor:

$$\begin{aligned} \dot{X} &= \mu \cdot X \\ \dot{S} &= \sigma \cdot X \\ \dot{P} &= \pi \cdot X. \end{aligned}$$

In dieser speziellen Situation gilt weiter:

$$\begin{aligned} \mu &= \frac{1}{\nu_S} \cdot \sigma \overset{def}{=} Y_{X/S} \cdot \sigma \\ \pi &= \frac{1}{\nu_P} \cdot \sigma \overset{def}{=} Y_{P/S} \cdot \sigma. \end{aligned}$$

Die Koeffizienten $Y_{X/S}$ und $Y_{P/S}$ heißen *Ertragskoeffizienten* für X und P und geben an, wieviele Einheiten Biomasse bzw. Produkt aus einer Einheit Substrat entstehen.

[8] Formal gesehen ist dies eine Normierung von R im Sinne von Abschnitt 5.1.1.4

[9] Es sollte trotzdem nicht vergessen werden, daß dem Wachstumsmodell ein Reaktionsmechanismus zugrunde liegt, der mit nicht-spezifischen Größen formuliert ist. Dieser ist bei stöchiometrischen Rechnungen und Bilanzen oft einfacher zu handhaben!

5.2.1.4 Formalkinetik

Über die Kinetik $R = \mu X$ der Reaktion wurde bisher nichts ausgesagt. Sie kann eine Funktion aller beteiligter Komponenten sein. Meist nimmt man eine Abhängigkeit von S, P und oft auch von der Sauerstoffkonzentration O oder der Biomassekonzentration X [Con59] an:

$$R = R(X, S, P, O) = \mu(S, P, O, X) \cdot X.$$

Für die Annahme weiterer Abhängigkeiten (z.B. von X bei extrem hoher Zelldichte [JMG89]) bestehen jedoch keine Einschränkungen.

Der Grundgedanke bei der Suche nach geeigneten Kinetiken ist nun der, daß gewisse Enzyme im Stoffwechsel der Mikroorganismen für das Wachstum limitierend sind (Schlüsselenzyme), wogegen alle anderen Reaktionen stationär ablaufen [10]. Dies legt nahe, Kinetiken zu verwenden, die aus der Enzym- und Transportkinetik (vgl. 5.1) entnommen sind [But83]. Diese Übertragungsmethode wird als *Formalkinetik* bezeichnet [Mos81].

Im obigen Modell ist nur noch die spezifische Substrataufnahmerate σ frei wählbar, da μ und π linear mit dieser verknüpft sind. Die einfachste Enzymkinetik ist die Michaelis-Menten-Kinetik, die in der Biotechnologie als *Monod-Kinetik* bezeichnet wird, und auf das klassische Monod-Modell

$$\begin{aligned} \dot{X} &= \mu(S)\, X \\ \dot{S} &= \sigma(S)\, X \end{aligned} \quad \text{mit} \quad \mu(S) = Y_{X/S} \cdot \sigma(S)$$

sowie die Substrataufnahme-Kinetik

$$\sigma(S) = \sigma_S \frac{S}{S + K_S}$$

führt. Beim Auftreten einer Substratinhibierung setzt man stattdessen

$$\sigma(S) = \sigma_S \frac{S}{\frac{1}{I_S} S^2 + S + K_S},$$

beim Auftreten einer Produktinhibierung (falls P unabhängig von S wirkt, vgl. 5.1.2.2):

$$\sigma(S, P) = \sigma_S \frac{S}{S + K_S} \cdot \frac{I_P}{P + I_P}.$$

Ganz allgemein wird man fordern:

$$\sigma = \sigma(S, P, \ldots) \geq 0 \quad \text{und} \quad \sigma(0) = 0.$$

Weitere Formalkinetiken werden in der Folge noch vorgestellt.

5.2.2 Ein Stoffwechselmodell

Das einfache Modell ist im allgemeinen nicht ausreichend, da es die Stoffwechselvorgänge in einem Mikroorganismus zu stark vereinfacht und einen unverzweigten Reaktionsweg vom Substrat zu Biomasse und Produkt anninmt. Erst bei Berücksichtigung der wichtigsten zellulären Mechanismen gelangt man zu brauchbaren Modellen, die von Organismus zu Organismus eine unterschiedliche Struktur haben können. Einige wichtige Ansätze werden in diesem und den nächsten Abschnitten an Beispielen vorgeführt.

5.2.2.1 Energie-, Aufbau- und Erhaltungsstoffwechsel

Eines der am häufigsten eingesetzten Modelle geht von folgenden Annahmen aus (vgl. Abb. 5.4b):

[10] Eng mit der Vorstellung von Schlüsselenzymen verknüpft ist das Konzept der *Master Equation* [BHLD89].

1. Das Substrat S gelangt über einen *Transport*mechanismus in die Zelle. Daran anschließen können sich noch weitere Umwandlungsschritte bis hin zu einem *zellinternen Intermediat* T. Zwischen S und T dürfen keine anderen Stoffwechselwege abzweigen. So könnte man z.B. die Glykolyse mit zum Transportmechanismus zählen und das entstehende Pyruvat als das eigentliche Substrat ansehen [11].

   In der Zelle treten zwei wesentliche Stoffwechselvorgänge auf:

2. *Energiegewinnung* durch Dissimilation des Substrats. Dabei wird das interne Substrat T abgebaut und Energie E produziert, die in biologischen Energieträgern (ATP, NADH) gespeichert wird [12].

3. *Wachstum* durch Assimilation des Substrats. Das Substrat S wird zusammen mit weiteren Stoffen, wie z.B. einer N-Quelle oder Spurenelenten zur Synthese neuer Biomasse X verwendet. Dabei wird ein Teil der Energie E verbraucht.

   Hier liegt die weitere Annahme zugrunde, daß dasselbe Substrat sowohl als Energiequelle als auch als C-Quelle dient, was bei biotechnischen Produktions-Prozessen fast immer der Fall ist.

4. Als letzter Faktor kommt der *Erhaltungsstoffwechsel* hinzu: Die Zelle benötigt Energie für (vgl auch [Del87])

   - Reparaturfunktionen (z.B. an der Zellmembran),
   - die Neusynthese von Molekülen mit kurzer Lebensdauer (Enzyme, mRNA),
   - die Aufrechterhaltung von Ionengradienten an Grenzschichten,
   - die Zellbewegung (bei manchen Organismen),
   - oder sogenannte *Futile Cycles*, die ohne eine daran gekoppelte Produktbildung Energie verbrauchen.

   Die Zellerhaltung ist somit ein reiner Energieverbraucher. Kann die Zelle die zur Erhaltung benötigte Minimalenergie nicht mehr zur Verfügung stellen, so kann es z.B. zur Lysis kommen.

5.2.2.2 Spezifische Raten

Das Reaktionssystem soll diesmal von vornherein mit den auf die Biomasse bezogenen spezifischen Raten formuliert werden:

$$
\begin{array}{lrcll}
(1) & S & \xrightarrow{\sigma \cdot X} & \nu_{\sigma T} \cdot T & \text{(Substrataufnahme)} \\
(2) & T & \xrightarrow{\rho \cdot X} & \nu_{\rho E} \cdot E & \text{(Energiegewinnung)} \\
(3) & \nu_{\mu T} \cdot T + \nu_{\mu E} \cdot E & \xrightarrow{\mu \cdot X} & X & \text{(Biomasseproduktion)} \\
(4) & E & \xrightarrow{m_E \cdot X} & & \text{(Erhaltungsstoffwechsel)}
\end{array}
$$

Dabei wurde von der Möglichkeit, Koeffizienten auf 1 zu normieren, (vgl. 5.1.1.4) Gebrauch gemacht, was die nachfolgenden Rechnungen übersichtlicher hält. Die spezifischen Reaktionsraten werden bezeichnet als:

$$
\begin{array}{rll}
 & \textit{spezifische Substrataufnahme} & \sigma, \\
 & \textit{spezifische Wachstumsrate} & \mu \\
\text{und} & \textit{spezifischer Erhaltungsterm} & m_E.
\end{array}
$$

Man beachte, daß verallgemeinerte Reaktionsgleichungen vorliegen:

[11] Beim Wachstum auf Glucose vernachlässigt diese Annahme die Tatsache, daß ein Teil der Glucose unmittelbar zur Synthese von Polysacchariden verwendet wird (Zellwandaufbau). Die Berücksichtigung dieses Mechanismus würde die Annahme eines weiteren Reaktionsweges erfordern. Dabei treten keine wesentlich neuen Modellstrukturen zutage, da man zu demselben Ergebnis gelangt, wenn man formal von einer *Gluconeogenese* $T \rightarrow S \rightarrow X$ (ohne Energiebedarf) ausgeht, die mit unter dem Wachstum subsummiert werden kann.

[12] Bei dem später behandelten Backhefe-Modell wird (vgl. 5.2.4.1) die Energiegewinnung durch die Atmung konkretisiert.

- Die physikalischen Einheiten können verschieden sein.
- Die Biomasse ist eine nichtmolekulare Substanz.
- Die Energie kann als ein abstraktes zellinternes Intermediat angesehen werden, dessen physikalische Einheit willkürlich wählbar ist [13].
- Die Energiebilanz stimmt nicht, da ein Teil der gewonnenen Energie nicht biologisch gebunden oder vollständig wieder abgegeben werden kann.
- Die Stoffbilanz stimmt nicht, da gar nicht alle für das Wachstum wichtigen Substrate vorkommen. Es fehlen N-Quelle, Spurenelemente, Vitamine, etc..

5.2.2.3 Auswertung der Stationaritäts-Annahmen

Das Stoffwechselmodell resultiert in den Gleichungen:

$$\begin{array}{rcl} \dot{S} &=& \left(\quad -\sigma \qquad\qquad\qquad\qquad\qquad\quad \right) \cdot X \\ \dot{T} &=& \left(\nu_{\sigma T} \cdot \sigma \qquad -\rho \quad -\nu_{\mu T} \cdot \mu \qquad\quad \right) \cdot X \\ \dot{E} &=& \left(\qquad\quad \nu_{\rho E} \cdot \rho \quad -\nu_{\mu E} \cdot \mu \quad -m_E \right) \cdot X \\ \dot{X} &=& \left(\qquad\qquad\qquad\qquad\quad \mu \qquad\quad \right) \cdot X \end{array}$$

Der unstrukturierte Ansatz verlangt, daß alle intrazellulären Konzentrationen im Fließgleichgewicht stehen:

$$\begin{array}{rcl} \dot{T} &=& 0 \quad \Rightarrow \quad \rho - \nu_{\sigma T} \cdot \sigma + \nu_{\mu T} \cdot \mu \;\; = 0 \\ \dot{E} &=& 0 \quad \Rightarrow \quad m_E - \nu_{\rho E} \cdot \rho + \nu_{\mu E} \cdot \mu \;\; = 0 \end{array}$$

Diesen Gleichungen entsprechen zwei Annahmen über die zelluläre Regulation:

- Der Transport von Substrat in die Zelle ist genau so hoch wie der Verbrauch, d.h. das interne Substrat T wird nicht akkumuliert. Eine naheliegende Annahme, wenn man z.B. weiß, daß die Glykolyse über den ATP-Pool und damit indirekt über den Energiebedarf reguliert wird [Sch85]
- Es wird nur soviel Energie produziert, wie durch Wachstum und Erhaltungsstoffwechsel auch wieder verbraucht werden kann. Dies setzt einen entsprechenden Regulationsmechanismus für die Verzweigungsstelle Assimilation/Dissimilation voraus. Insbesondere wird keine Energie, z.B. in Form von Speicherstoffen akkumuliert.

Die Stationären Annahmen ermöglichen die Elimination zweier Reaktionsraten. Man sollte sich überlegen, welche Terme man sinnvollerweise behält:

- Die Substrataufnahme ist die "treibende Kraft" des Wachstums.
- Das Wachstum ist zwar die Zielgröße des Systems, aber eben deshalb sollte es sich als Folge der übrigen Vorgänge in der Zelle ergeben.
- Auch die Energieproduktion richtet sich nach dem zellinternen Bedarf, wogegen
- der Erhaltungsstoffwechsel vor allem durch äußere Faktoren (z.B. Zellmembranschädigung) bedingt ist.

Auf diesen Gründen beruht die Entscheidung, die stationären Gleichungen nach σ und m_E freizustellen. Zunächst ergibt sich

$$\begin{array}{rcl} \mu &=& \dfrac{\nu_{\sigma T}\, \nu_{\rho E} \cdot \sigma - m_E}{\nu_{\rho E}\, \nu_{\mu T} + \nu_{\mu E}}, \\ \rho &=& \nu_{\sigma T} \cdot \sigma - \nu_{\mu T} \cdot \mu. \end{array}$$

[13] In [BvdHH90] wird beispielsweise die Energie E durch die ATP-Konzentration konkretisiert.

und weiter

$$\begin{aligned}
\dot{S} &= -\sigma \cdot X \\
\dot{X} &= \frac{\nu_{\sigma T}\,\nu_{\rho E} \cdot \sigma - m_E}{\nu_{\rho E}\,\nu_{\mu T} + \nu_{\mu E}} \cdot X \\
&= \frac{\nu_{\sigma T}\,\nu_{\rho E}}{\nu_{\rho E}\nu_{\mu T} + \nu_{\mu E}} \cdot (\sigma - \frac{1}{\nu_{\sigma T}\,\nu_{\rho E}} \cdot m_E) \cdot X \\
&\stackrel{def}{=} Y_{X/S} \cdot (\sigma - m_S) \cdot X.
\end{aligned}$$

Der *Maintenance*-Term

$$m_S = \frac{1}{\nu_{\sigma T}\nu_{\rho E}} \cdot m_E$$

läßt sich wie folgt interpretieren: m_S ist die Menge Substrat S, die abgezweigt werden muß, um die Zellerhaltung mit Energie zu versorgen. Das Wachstum ist dann direkt proportional zum verbleibenden Substratfluß

$$\mu = Y_{X/S} \cdot (\sigma - m_S)$$

mit dem *(wahren) Ertragskoeffizienten*

$$Y_{X/S} = \frac{\nu_{\sigma T}\,\nu_{\rho E}}{\nu_{\rho E}\nu_{\mu T} + \nu_{\mu E}}$$

der angibt, wieviele Einheiten Biomasse — nach Abzug der Erhaltungsenergie — pro Einheit Substrat entstehen. In $Y_{X/S}$ "kondensieren" offenbar eine Reihe von stöchiometrischen Koeffizienten, die "von außen" nicht einzeln beobachtet werden können. Im Gegensatz dazu gibt es auch einen *scheinbaren Ertragskoeffizienten*

$$Y^*_{X/S} = -\frac{\dot{X}}{\dot{S}} = Y_{X/S}\,(1 - \frac{m_S}{\sigma})\,,$$

der zwar direkt einer Messung zugänglich, aber für $m_S \neq 0$ nicht konstant ist [14]. [Pir75] [And89].

5.2.2.4 Formalkinetiken für den Erhaltungsstoffwechsel

Das fertige Modell in üblicher Schreibweise ist:

$$\begin{aligned}
\dot{X} &= \mu X \\
\dot{S} &= -\sigma X
\end{aligned} \quad \text{mit} \quad \mu = Y_{X/S}\,(\sigma - m_S)$$

Die zellinternen Komponenten T und E treten in diesem Modell nicht mehr auf; lediglich die Kinetiken σ und m_S sind jetzt noch mit Inhalt zu füllen. Für σ gilt das bereits vorher Gesagte: Es können Sättigungskinetiken, Substratinhibierungs-Kinetiken, Produktinhibierungskinetiken usw. Verwendung finden. In bezug auf den Maintenance-Term m_S muß man zunächst feststellen, daß in den meisten Systemen:

$$m_S \ll \sigma(S) \text{ über einen weiten Bereich bzgl. } S$$

gilt. D.h. der Erhaltungsstoffwechsel, der von außen nur in Form des Defizits

$$m_S(S) = \frac{1}{Y_{X/S}}\mu(S) - \sigma(S)$$

[14] Eine alternative Beschreibungsform des Erhaltungsstoffwechsels, sind die *Variable-Yield-Modelle*, die von einer solchen variablen Stöchiometrie ausgehen (vgl. 5.2.6.5).

sichtbar ist, läßt sich experimentell nur schwer bestimmen. Dementsprechend muß man sich mit einer groben Beschreibung des Erhaltungsstoffwechsels begnügen. Sinnvollerweise sollte gelten:

$$m_S = m_S(S, P, \ldots) \geq 0$$

Häufig wird angenommen, daß $m_S(S)$ einen festen Grundbedarf zur Zellerhaltung repräsentiert. Dann ist:

$$m_S = m_{S,0} = \text{ const}$$

Es kann aber durchaus auch sein, daß eine Komponente des Mediums die Zelle schädigt. Z.B. greift Methanol, das als C-Quelle für *Candida boidinii* dient, zugleich auch die Zellmembranen an (vgl. auch [BOHR81]). In dieser Situation sollte im einfachsten Fall eine Proportionalität

$$m_S = m_{S,1} \cdot S$$

bzw. in Kombination mit einem Grundbedarf eine lineare Beziehung

$$m_S = m_{S,1} \cdot S + m_{S,0}$$

gewählt werden [15]. Vielfach wird eine Inhibierung durch eine unmittelbare Wachstumshemmung, d.h. durch eine Inhibierungskinetik, erklärt. Hierbei ist nicht berücksichtigt, daß eine Inhibierung auch durch einen anwachsenden Maintenance-Term beschreibbar ist. Die beiden entstehenden Modelle unterscheiden sich fundamental voneinander (vgl. 10.4.2.4).

5.2.3 Produktbildung

5.2.3.1 Modellannahmen für die Produktbildung

Das bestehende Modell läßt sich sehr einfach um eine *spezifische Produktbildungsrate* π erweitern. Dazu muß man lediglich wissen, an welchen Soffwechselvorgang die Produktbildung gekoppelt ist. In bezug auf das Stoffwechselmodell sind mehrere Mechanismen denkbar (vgl. Abb. 5.4c):

1. Kopplung an den Transportschritt:
   Im allgemeinen sind in diesem Fall mit der Substrataufnahme auch einige energieerzeugende Schritte verbunden. Ein Beispiel ist die Glykolyse, die im Falle einer Gärung Energie gewinnt und gleichzeitig Äthanol als Produkt freisetzt. In Erweiterung des obigen Reaktionssystems gelangt man zu:

   $$(1') \quad S \xrightarrow{\sigma \cdot X} \nu_{\sigma T} \cdot T + \nu_{\sigma E} \cdot E + \nu_{\sigma P} \cdot P$$

2. Kopplung an die Energiegewinnung (ein eher unwahrscheinlicher Fall):

   $$(2') \quad T \xrightarrow{\rho \cdot X} \nu_{\rho E} \cdot E + \nu_{\rho P} \cdot P$$

3. Kopplung an das Wachstum:

   $$(3') \quad \nu_{\mu T} \cdot T + \nu_{\mu E} \cdot E \xrightarrow{\mu \cdot X} X + \nu_{\mu P} \cdot P$$

4. Kopplung an den Erhaltungsstoffwechsel (ebenfalls eher unwahrscheinlich):

   $$(4') \quad E \xrightarrow{m_E \cdot X} \nu_{mP} \cdot P$$

[15] In [Del87] wird auch eine mögliche lineare Beziehung mit negativer Steigung begründet.

5. Annahme eines eigenen Stoffwechselweges für die Produktbildung:
   Es kann eingewendet werden, daß die Natur sich einen solchen Luxus ohne einen daran gekoppelten Gewinn nicht erlauben wird. Bei näherer Betrachtung finden sich jedoch gleich mehrere biotechnische Beispiele:

   - Die Produktion von Antibiotika, dient der Abwehr von Futterkonkurrenten und setzt (zumindest bei natürlich vorkommenden Stämmen) bei Substratmangel ein. Die Antibiotikaproduktion profitiert von diesem Effekt [Del87].
   - Stoffwechselmutanten werden gerade auf ihre Stoffwechseldefekte hin künstlich selektioniert.
   - Pflanzliche und tierische Zellen produzieren Stoffe, die im natürlichen Zellverband eine Funktion hätten.

   Die entsprechende Reaktionsgleichung ist:

$$(5)\quad \nu_{\pi E} \cdot E + \nu_{\pi S} \cdot S \xrightarrow{\pi \cdot X} P$$

5.2.3.2 Auswertung der Modellannahmen

1. Kopplung an den Transportschritt :

$$\dot{P} = \nu_{\sigma P} \cdot \sigma \cdot X \implies \pi = \nu_{\sigma P} \cdot \sigma = Y_{P/S} \cdot \sigma,$$

   mit dem *Produkt-Ertragskoeffizienten* $Y_{P/S}$.

2. Kopplung an die Energiegewinnung:

$$\dot{P} = \nu_{\rho P} \cdot \rho \cdot X \implies \pi = \nu_{\rho P} \cdot \rho$$

3. Kopplung an das Wachstum:

$$\dot{P} = \nu_{\mu P} \cdot \mu \cdot X \implies \pi = \nu_{\mu P} \cdot \mu$$

4. Kopplung an den Erhaltungsstoffwechsel:

$$\dot{P} = \nu_{mP} \cdot m_E \cdot X \implies \pi = \nu_{mP} \cdot m_E$$

5. Annahme eines eigenen Stoffwechselweges für die Produktbildung:

$$\dot{P} = \pi \cdot X$$

5.2.3.3 Zusammenfassung

Die verschiedensten Modelle sind denkbar. Die Annahmen 1 - 4 haben nach entsprechender Umformung gemeinsam, daß sich π als Linearkombination von σ und m_E darstellen läßt (vgl. 5.2.2.3):

$$\pi = Y_{P/S} \cdot \sigma + Y_{P/E} \cdot m_E \quad \text{mit den Ertragskoeffizienten } Y_{P/S},\ Y_{P/E}$$

Spezialfälle sind:

- $\pi = Y_{P/S} \cdot \sigma$ (Annahme 1)
- $\pi = Y_{P/E} \cdot m_E$ (Annahme 4)
  Speziell für eine konstante Maintenance $m_E = m_{E,0}$ ist π eine Konstante.
- Gemischter Fall (Annahme 2,3)

Unter Annahme 5 ist die Kinetik π unabhängig von σ und m_E wählbar. Häufig wählt man dann als einfachste Beschreibungsform den linearen Ansatz

$$\pi(S) = \pi_1 \cdot S + \pi_0 .$$

Ganz allgemein muß dann beachtet werden:

$$\pi = \pi(X, S, P, ...) \geq 0$$

5.2.4 Wachstum auf zwei Substraten

Um zu zeigen, daß sich auch kompliziertere Modelle im Rahmen verallgemeinerter Reaktionssysteme ableiten lassen, soll das Beispiel der Backhefe *Saccharomyces cerevisia* herangezogen werden.

5.2.4.1 Stoffwechselwege bei *Saccharomyces cerevisia*

Der Stoffwechsel von *Saccharomyces cerevisia* ist durch folgende Eigenschaften charakterisiert:

- Die Backhefe vermag sowohl auf dem Substrat Glucose (S) als auch auf dem Substrat Äthanol (A) zu wachsen, wobei Glucose bei Angebot beider Substrate vorgezogen wird.
- Auf der anderen Seite wird Glucose unter bestimmten Bedingungen (Sauerstoffmangel, Substratüberschuß) zu Äthanol vergoren [16] und nicht auf oxidativem Wege vollständig zu CO_2 verbrannt. Äthanol ist somit sowohl ein Substrat als auch ein Produkt des Stoffwechsels.
- Als zellinternes Intermediat wird eine Substanz T angenommen (vgl. auch [PKGB77]), die mit dem Pyruvat als Produkt der Glykolyse identifiziert werden könnte (vgl. [Bar90]). Im Falle einer Gärung wird T zu Äthanol umgesetzt und ausgeschieden, im Falle einer Atmung dagegen vollständig verbrannt.
- Beim Wachstum auf Äthanol wird angenommen, daß A zunächst wieder in T zurückverwandelt und dann veratmet wird [17]. Ein Stoffwechselweg, der dabei vernachlässigt wird, ist die Gluconeogenese: Bei Wachstum auf Äthanol muß die Zelle ihren Bedarf an Glucose durch Neusynthese sättigen. Die Hinzunahme dieses Effekts produziert noch längere Terme, jedoch am Ende keine andere Modellstruktur (vgl. hierzu auch 5.2.5.4).
- Ein Erhaltungsstoffwechsel wird ebenfalls nicht angenomen, um den wesentlichen Aspekt des Modells deutlicher hervortreten zu lassen.

5.2.4.2 Formalisierung

Das den obigen Annahmen entsprechende Stoffwechselmodell ist in Abbildung 5.5 schematisiert. Der Ablauf der Modellierung entspricht dem bei den bisher vorgeführten Stoffwechselmodellen:

1. Formulierung des verallgemeinerten Reaktionssystems (mit entsprechenden Normierungen, vgl. 5.1.1.4):

$$\begin{array}{lrcll}
(1) & S & \xrightarrow{\sigma \cdot X} & \nu_{\sigma T} \cdot T + \nu_{\sigma E} \cdot E & \text{(Glykolyse)} \\
(2) & T & \xrightarrow{\rho \cdot X} & \nu_{\rho E} \cdot E & \text{(Atmung)} \\
(3a) & \nu_{\pi T} \cdot T & \xrightarrow{\pi \cdot X} & A & \text{(Gärung)} \\
(3b) & A + \nu_{\eta E} \cdot E & \xrightarrow{\eta \cdot X} & \nu_{\eta T} \cdot T & \text{(Äthanol-Aufnahme)} \\
(4) & \nu_{\mu T} \cdot T + \nu_{\mu E} \cdot E & \xrightarrow{\mu \cdot X} & X & \text{(Wachstum)}
\end{array}$$

[16] Die Bierherstellung beruht auf diesem Effekt.

[17] Diese Annahme ist nicht ganz korrekt. Tatsächlich wird Äthanol über Acetaldehyd und Acetyl-CoA — dem direkten Nachfolger von Pyruvat auf dem Dissimilationsweg — in den Zitronensäure-Zyklus eingeschleust [SGD71].

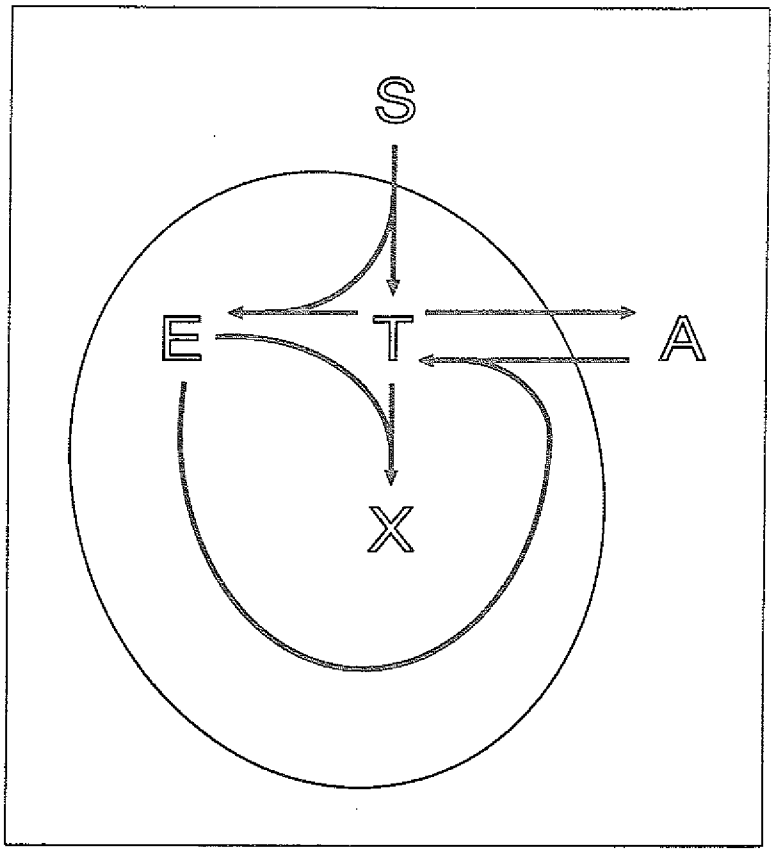


Abbildung 5.5: Stoffwechselschema der Backhefe *Saccharomyces cerevisia*

2. Aufstellung des zugehörigen Differentialgleichungssystems:

$$\begin{array}{rcl}
\dot{S} &=& \left(\quad -\sigma \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \right) \cdot X \\
\dot{T} &=& \left(\nu_{\sigma T} \cdot \sigma \qquad -\rho \quad -\nu_{\pi T} \cdot \pi \quad +\nu_{\eta T} \cdot \eta \quad -\nu_{\mu T} \cdot \mu \right) \cdot X \\
\dot{A} &=& \left(\qquad\qquad\qquad\qquad\qquad\qquad \pi \quad -\eta \qquad\qquad\qquad \right) \cdot X \\
\dot{E} &=& \left(\nu_{\sigma E} \cdot \sigma \quad +\nu_{\rho E} \cdot \rho \qquad\qquad\qquad -\nu_{\eta E} \cdot \eta \quad -\nu_{\mu E} \cdot \mu \right) \cdot X \\
\dot{X} &=& \left(\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \mu \qquad\qquad \right) \cdot X
\end{array}$$

3. Stationaritäts-Annahmen:

$$\begin{array}{rcl}
\dot{T} = 0 &\iff& \nu_{\sigma T} \cdot \sigma \qquad -\rho \quad -\nu_{\pi T} \cdot \pi \quad +\nu_{\eta T} \cdot \eta \quad -\nu_{\mu T} \cdot \mu \quad = 0 \\
\dot{E} = 0 &\iff& \nu_{\sigma E} \cdot \sigma \quad +\nu_{\rho E} \cdot \rho \qquad\qquad\qquad -\nu_{\eta E} \cdot \eta \quad -\nu_{\mu E} \cdot \mu \quad = 0
\end{array}$$

4. Auswertung der Annahmen:
Drei Kinetiken sind frei wählbar. Es ist sinnvoll, die Wachtumsrate μ (als Zielgröße des Stoffwechsels) und die Energieproduktion ρ (meßtechnisch nicht zugänglich) als Funktionen der Glucose-Aufnahme-Rate σ, der Äthanol-Aufnahme-Rate η und der Produktbildungsrate π darzustellen. Eine Rechnung liefert:

$$\begin{array}{rcl}
\mu &=& \dfrac{\nu_{\rho E}\nu_{\sigma T} - \nu_{\sigma E}}{\nu_{\rho E}\nu_{\mu T} + \nu_{\mu E}} \cdot \sigma + \dfrac{\nu_{\rho E}\nu_{\eta T} - \nu_{\eta E}}{\nu_{\rho E}\nu_{\mu T} + \nu_{\mu E}} \cdot \eta - \dfrac{\nu_{\rho E}\nu_{\pi T}}{\nu_{\rho E}\nu_{\mu T} + \nu_{\mu E}} \cdot \pi \\
\rho &=& -\dfrac{\nu_{\sigma E}\nu_{\mu T} - \nu_{\mu E}\nu_{\sigma T}}{\nu_{\rho E}\nu_{\mu T} + \nu_{\mu E}} \cdot \sigma + \dfrac{\nu_{\mu E}\nu_{\eta T} + \nu_{\eta E}\nu_{\mu T}}{\nu_{\rho E}\nu_{\mu T} + \nu_{\mu E}} \cdot \eta - \dfrac{\nu_{\mu E}\nu_{\pi T}}{\nu_{\rho E}\nu_{\mu T} + \nu_{\mu E}} \cdot \pi
\end{array}$$

Durch Einführung von Konstanten

$$\mu_\sigma,\ \mu_\eta,\ \mu_\pi \geq 0 \quad \text{und} \quad \rho_\sigma,\ \rho_\eta,\ \rho_\pi \geq 0$$

stellen sich μ und ρ als Linearkombinationen von σ, η, π dar:

$$\begin{aligned} \mu &\stackrel{def}{=} +\mu_\sigma \cdot \sigma + \mu_\eta \cdot \eta - \mu_\pi \cdot \pi \\ \rho &\stackrel{def}{=} -\rho_\sigma \cdot \sigma + \rho_\eta \cdot \eta - \rho_\pi \cdot \pi \end{aligned}$$

5.2.4.3 Der Flaschenhals-Effekt

Bisher sind σ, η und π frei wählbar, was mit Sicherheit nicht den realen Gegebenheiten entspricht. Die entscheidende Nebenbedingung für das System wird durch den sogenannten *Flaschenhals-Effekt* bestimmt, der besagt:

> Die oxidative Kapazität des Systems ist — selbst bei ausreichender O_2-Versorgung — beschränkt:
>
> $$\text{(A)} \quad \rho = -\rho_\sigma \cdot \sigma + \rho_\eta \cdot \eta - \rho_\pi \cdot \pi \leq \rho_{max}$$
>
> mit einer Konstanten $\rho_{max} > 0$.

Weitere Nebenbedingungen sind durch die (stets implizit mitgeführten) Forderungen

$$\text{(B)} \quad \sigma, \eta, \pi, \rho, \mu \geq 0 \ \text{ und } \ \nu_{ij} \geq 0 \quad (i = \sigma, \eta, \pi;\ \ j = S, T, A, E, X)$$

gegeben. Mit diesen Einschränkungen sind noch acht Wertekombinationen für σ, η, π möglich, die nun auf ihre biologische Bedeutung hin untersucht werden können:

Fall $+++$ $(\sigma > 0,\ \eta > 0,\ \pi > 0)$:
Da die Zellen nicht gleichzeitig Äthanol ausscheiden und wieder aufnehmen, muß dieser Fall ausgeschlossen werden:

$$\text{(C)} \quad \pi > 0 \ \Rightarrow \ \eta = 0$$

Mit (B) folgt hieraus auch: $\eta > 0 \ \Rightarrow \ \pi = 0$

Fall $++0$ $(\sigma > 0,\ \eta > 0,\ \pi = 0)$:
Beschreibt die gleichzeitige Veratmung von Glucose und Äthanol. Obwohl dieser Fall theoretisch denkbar wäre, nimmt die Hefe selbst dann fast ausschließlich Glucose auf, wenn der Flaschenhals nicht ganz ausgelastet ist. Dieses Verhalten läßt sich durch eine Glucose-Hemmung der Äthanol-Aufnahme beschreiben (vgl. 5.1.2.2):

$$\text{(D)} \quad \eta = \eta' \cdot \frac{I_S}{S + I_S} \ \text{ mit einem gewissen } \eta'$$

Diese Bedingung schließt nicht aus, daß sehr kleine Mengen von Äthanol zusätzlich verstoffwechselt werden.

Fall $+0+$ $(\sigma > 0,\ \eta = 0,\ \pi > 0)$:
Beschreibt das gärende Verhalten der Zelle: Glucose wird bis an die Kapazitätsgrenze veratmet, der Überschuß als Äthanol ausgeschieden. Formal:

$$\text{(E)} \quad \pi > 0 \ \Rightarrow \ \rho = \rho_{max}$$

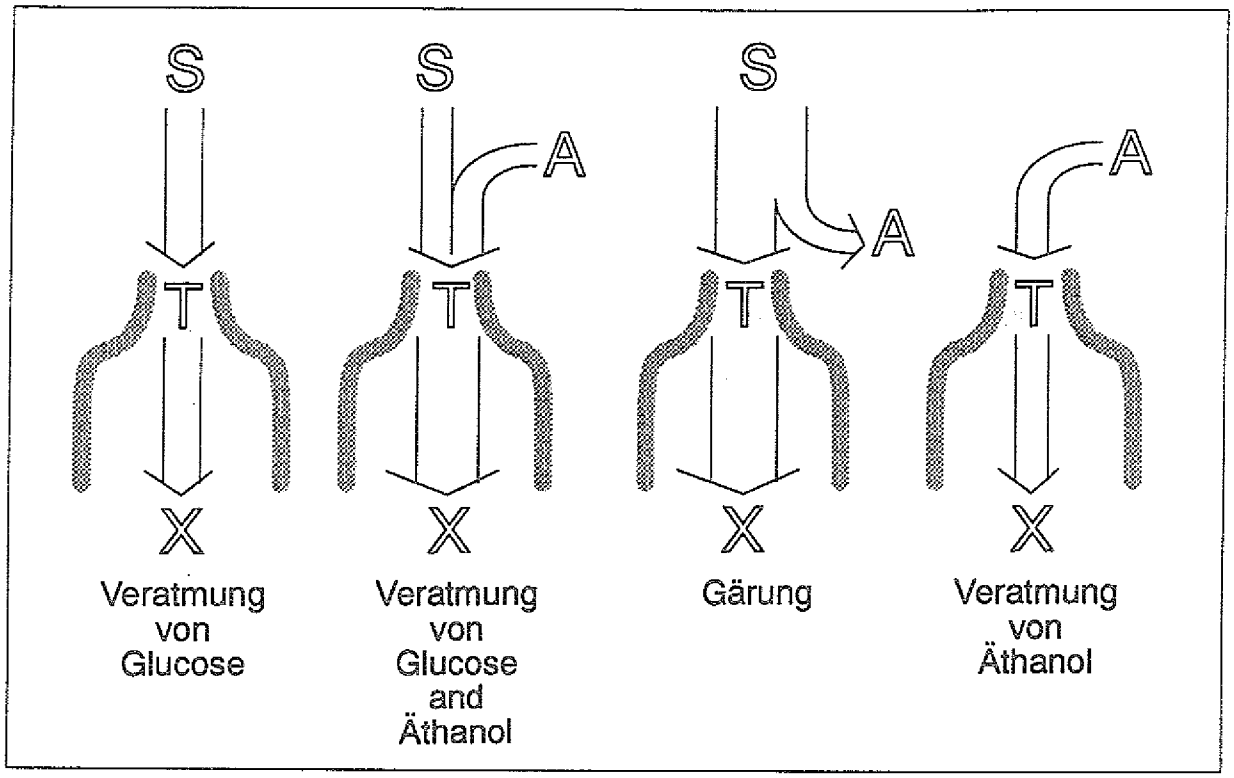


Abbildung 5.6: Mögliche Arbeitsweisen des Flaschenhalses bei *Saccharomyces cerevisia*

Fall $+\,0\,0$ $(\sigma > 0,\ \eta = 0,\ \pi = 0)$:
Vollständige Veratmung von Glucose. Der Flaschenhals ist nicht ausgelastet. Aus (A),(B) und (D) folgt in diesem Fall: $\rho < \rho_{max} \ \Rightarrow \ \pi = 0$

Fall $0\,+\,+$ $(\sigma = 0,\ \eta > 0,\ \pi > 0)$:
Ist ausgeschlossen aufgrund von (C).

Fall $0\,+\,0$ $(\sigma = 0,\ \eta > 0,\ \pi = 0)$:
Veratmung von Äthanol ohne Kapazitätsauslastung (analog zum Fall $+\,0\,0$).

Fall $0\,0\,+$ $(\sigma = 0,\ \eta = 0,\ \pi > 0)$:
Ist aufgrund von (A) und (B) ausgeschlossen.

Fall $0\,0\,0$ $(\sigma = 0,\ \eta = 0,\ \pi = 0)$:
Der triviale Fall.

Die vier nichttrivialen möglichen Arbeitsweisen des Flaschenhalses sind in Abbildung 5.6 schematisiert. Die Forderungen (A)-(E) können als "Axiomatisierung" des Hefestoffwechsels verstanden werden. Sie lassen für die Wahl von σ, η und π nur noch geringe Freiheit.

5.2.4.4 Wahl der Kinetiken

Die Kinetiken σ, η und π können folgendermaßen gewählt werden:

1. Man nehme eine beliebige Kinetik $\sigma > 0$ an.
2. Die Produktbildung π folgt zwingend aus (A),(C) und (E):

$$\pi = \begin{cases} 0 & , \text{falls} \quad \rho_\sigma \cdot \sigma \leq \rho_{max} \\ \frac{\rho_\sigma \cdot \sigma - \rho_{max}}{\rho_\pi} & , \text{falls} \quad \rho_\sigma \cdot \sigma > \rho_{max} \end{cases}$$

3. Man nehme eine beliebige Kinetik $\tilde{\eta} > 0$ an, die bei Nichtvorhandensein des Flaschenhalses und der Glucose-Hemmung Gültigkeit hätte.

4. Dann resultiert die tatsächliche Äthanol-Aufnahme-Kinetik η mit:

$$\eta = \frac{I_S}{S + I_S} \cdot \begin{cases} 0 & , \text{falls} \quad \pi > 0 \\ \tilde{\eta} & , \text{falls} \quad \pi = 0 \quad \text{und} \quad \rho_\eta \cdot \tilde{\eta} + \rho_\sigma \cdot \sigma \leq \rho_{max} \\ \frac{\rho_{max}}{\rho_\eta} - \frac{\rho_\sigma}{\rho_\eta} \cdot \sigma & , \text{falls} \quad \pi = 0 \quad \text{und} \quad \rho_\eta \cdot \tilde{\eta} + \rho_\sigma \cdot \sigma > \rho_{max} \end{cases}$$

Die Setzungen erfüllen die Axiome (A)-(E), wie man leicht nachweisen kann. Umgekehrt kann man sich überlegen, daß die Bedingungen an σ, η und π nur auf diese Weise erfüllbar sind.
In der Praxis wählt man als einfache Kinetiken meist Sättigungskinetiken:

$$\sigma(S) = \sigma_S \frac{S}{S + K_S} \quad \text{und} \quad \tilde{\eta}(A) = \eta_A \frac{A}{A + K_A}$$

5.2.4.5 Das Modell von Sonnleitner und Käppeli

Das folgende Modell stellt den derzeitigen "Stand der Technik" bei der Modellierung des Hefewachstums dar [SK86]. Es wird hier mit der oben verwendeten Terminologie widergegeben:

Glucose-Aufnahme:	σ	$=$	$\sigma_S \frac{S}{S+K_S}$
Oxidativer Glucose-Abbau:	σ_{ox}	$=$	$\min(\sigma,\ Y_{S/O} \cdot \rho_{max})$
Reduktiver Glucose-Abbau:	σ_{red}	$=$	$\sigma - \sigma_{ox}$
Ethanol-Aufnahme:	η'	$=$	$\eta_A \frac{A}{A+K_A} \frac{I_S}{S+I_S}$
Oxidativer-Äthanol-Abbau:	η_{ox}	$=$	$\min(\eta',\ (\rho_{max} - Y_{O/S} \cdot \sigma_{ox}) \cdot Y_{A/O})$
Äthanol-Produktion:	π	$=$	$Y_{A/S} \cdot \sigma_{red}$
Wachstum:	μ	$=$	$Y^{ox}_{X/S} \cdot \sigma_{ox} + Y^{red}_{X/S} \cdot \sigma_{red} + Y_{X/A} \cdot \eta_{ox}$

Es wäre interessant, die genaue Korrespondenz zwischen dem Modell von Sonnleitner und Käppeli und dem oben deduktiv hergeleiteten Modell herzustellen. Dies soll hier aus Platzgründen nicht mehr geschehen. Hingewiesen sei lediglich auf zwei Unterschiede in der Betrachtungsweise:

- Als unabhängige Kinetiken werden $\sigma_{ox}, \sigma_{red}, \eta$ gewählt, im Gegensatz zu σ, η, π in Abschnitt 5.2.4.2 (vgl. dazu auch die Gegenüberstellung in 5.2.6.3).

- Das Modell von Sonnleitner und Käppeli zerlegt die Substrataufnahme σ in:

  - den Anteil σ_{ox}, der die Veratmung von Glucose zusammen mit dem daraus resultierenden Substratbedarf für das Wachstum beschreibt und
  - den Anteil σ_{red}, der den vergärten Anteil der Glucose zusammen mit der entsprechenden Wachstumskomponente enthält.

  Diese beiden Anteile sind von außen getrennt beobachtbar, indem man anaerobe Bedingungen ($\rho = 0$) bzw. areobe Bedingungen mit geringer Glucosekonzentration ($\pi = 0$) herstellt. Im Gegensatz dazu können die Raten σ und π des anderen Modells nicht experimentell entkoppelt werden. Das Modell von Sonnleitner und Käppeli beschreibt den Stoffwechsel "von außen nach innen" beschreibt, wogegen der deduktive Ansatz "von innen nach außen" vorgeht.

- Es besteht ein Unterschied bzgl. der Äthanolaufnahme: Das letztgenannte Modell verhindert die Äthanol-Aufnahme für den Fall der Auslastung des Flaschenhalses durch die Glucose-Aufnahme ($\pi > 0$) nicht. Es ist somit das Axiom (C) verletzt: Äthanol wird aufgenommen, kann dann aber

nicht veratmet werden und verschwindet aus den Bilanzen! Dieses "Leck" ist aufgrund der Glucose-Hemmung von η verschwindend klein, so daß mit dem Modell trotzdem gute Anpassungsergebnisse erzielt werden konnten [SK86] [18].

5.2.5 Methodische Bemerkungen

5.2.5.1 Allgemeiner Ansatz

Der Modellierungsansatz für mikrobielle Stoffwechselvorgänge auf der Basis verallgemeinerter Reaktionssysteme erweist sich als äußerst leistungsfähig. Er ist sowohl in der Lage, die biotechnischen Standardmodelle herzuleiten als auch komplizierte Stoffwechselmodelle, wie z.B. das Hefe-Wachstumsmodell. Ferner umfaßt er die Methoden des Pathway Modeling, die weniger abstrakt formuliert werden können, weil sie eine konkrete Stöchiometrie der Stoffwechselwege sowie konkrete zellinterne Intermediate zugrunde legen.

5.2.5.2 Deduktive und evolutive Modellierungstechniken

Das Backhefe-Modell von Sonnleitner und Käppeli steht in einer langen Entwicklungsreihe, aus der hier nur [BK72] [PBCL74] [BH77] [BdKR80] [SK86] und [SGB+88] genannt seien. Hieran wird sehr schön deutlich, wie biotechnische Modelle vielfach entstehen, nämlich durch evolutives Verändern und Erweitern bestehender Ansätze. Der deduktive Ansatz baut im Gegensatz dazu jedes Modell neu aus Grundannahmen über den Stoffwechsel auf. Man könnte auch von einer "Bottom-Up-Methode" im Gegensatz zu einer "Top-Down-Methode" sprechen. Bei sorgfältigem Vorgehen lassen sich logische Fehler des Modells vermeiden (vgl. die Ausführungen in 5.2.4.5 und 5.2.5.4).

Unstrukturierte Modelle werden oft auch als *phänomenologische Modelle* bezeichnet. Das Wort "phänomenologisch" drückt hier einen nur beschreibenden und nicht begründenden Charakter des Modells aus. Der phänomenologische Ansatz birgt die Gefahr, daß Modelle erst aufgestellt und nachträglich dann begründet werden.

Ein weiteres Beispiel für diesen Unterschied stellen die verschiedenen Modelle zur Wachstumsinhibierung dar. Hier wurden im Laufe der Jahre durch Modifikation des einfachen Substratinhibierungs-Modells (vgl. 5.1.2.2) diverse Kinetiken formuliert [Lev80] [Bla81] [Luo87], die sich bei einem deduktiven Ansatz durch Veränderung wenigen Grundannahmen ergeben würden (vgl. dazu auch die Ausführungen in 10.4.2).

5.2.5.3 Grundannahmen

Einer der wesentlichen Vorteile des Ansatzes ist, daß die Annahmen, die in ein Modell eingehen, klar hervortreten. Dies macht ein solches Modell auch mit "Nicht-Modellierern" diskutierbar. Sind die Grundannahmen akzeptiert, so folgt das Modell mit Notwendigkeit. Andernfalls können die Annahmen der Reihe nach (auch experimentell) auf Stichhaltigkeit geprüft und gegebenenfalls verändert werden.
Die Art der Grundannahmen sei noch einmal kurz zusammengefaßt:

- Die Stoffwechselwege sind vorzugeben. Sie fassen den Stoffwechsel zu großen Teilabschnitten zusammen [19].
- Ein solcher Stoffwechselweg hat eine feste Stöchiometrie. Bei dieser Annahme sollte man vor allem eines bedenken: Die biologische Evolution beruht entscheidend darauf, daß durch die Bildung von biologischen Membranen die Schaffung abgeschlossener Reaktionsräume in einer äußeren Umgebung ermöglicht wurde [Leh82]. Die intrazelluläre Regulation hat die Aufgabe, die Bedingungen in der Zelle in weiten Bereichen konstant zu halten. Daher ist es nicht unsinnig anzunehmen, daß die

[18] Eine Simulation der Backhefeproduktion auf der Basis dieses Modells ist ein hervorragendes didaktisches Hilfsmittel zur Erläuterung des Flaschenhals-Effektes [SW88].

[19] Die Auswahl geeigneter Stoffwechselwege erfordert eine gute biologische Systemkenntnis. Insbesondere wird die Vernachlässigbarkeit von Effekten immer wieder zur Diskussion stehen!

chemischen Umsetzungen eines Stoffwechselweges unter verschiedenen äußeren Bedingungen mit konstantem Wirkungsgrad ablaufen [20].

- Alle zellinternen Konzentrationen werden als stationär betrachtet. D.h. keiner dieser Stoffe wird akkumuliert oder abgebaut [21].
- Aus den auftretenden Kinetiken für jeden Reaktionsweg ist eine linear unabhängige Teilmenge sinnvoll auszuwählen. Hierfür bieten sich oft die Kinetiken an, die am Anfang der Reaktionswege stehen.
- Für die linear unabhängigen Kinetiken sind Funktionsterme anzunehmen, wobei einfache Modelle zu bevorzugen sind.
- Für Verzweigungsstellen mit Nebenbedingungen sind diese in Form von Gleichungen und Ungleichungen zu formulieren. Die Auswertung der Nebenbedingungen führt zu Fallunterscheidungen in den Modellgleichungen.

5.2.5.4 Ablaufschema

Sehr interessant ist, daß man den Ablauf der Modellierung weitgehend automatisieren kann (vgl. Abb. 5.7 [22].): Im Prinzip genügt die Angabe

- des Reaktionssystems,
- der stationären Konzentrationen,
- der linear unabhängigen Kinetiken,
- der Nebenbedingungen und
- der kinetischen Terme,

um mit Hilfe eines programmierbaren Computeralgebra-Systems (wie z.B. Reduce oder Mathematica [Wol88]) alle erforderlichen Rechnungen automatisch durchzuführen [23]. Das Algebra-System gibt weiterhin Auskunft darüber, welche stöchiometrischen Koeffizienten sich in welchen (von außen sichtbaren) Systemkonstanten niederschlagen, und welche davon nicht direkt beobachtbar sind.

Der eigentliche Vorteil dieser Vorgehensweise tritt dann zutage, wenn ein Modell modifiziert oder erweitert werden muß. Die Neuberechnung des Modells per Hand wäre dann äußerst zeitraubend! Beispielsweise läßt sich relativ schnell herausfinden, welche Konsequenzen die Hinzunahme der Gluconeogenese für die bisher diskutierten Modelle hat.

5.2.5.5 Didaktische Konsequenzen

In Lehrveranstaltungen [SW89] hat sich wiederholt gezeigt, wie schwierig eine stichhaltige Herleitung der biotechnischen Standardmodelle ist. Bei einer rein phänomenologischen Erklärung bleibt in jedem Fall ein unbefriedigender Eindruck in bezug auf die dem Modell zugrunde liegenden Annahmen zurück. Die deduktive Methode bietet hier den Vorteil, daß klare Grundannahmen am Anfang stehen, woraus sich das Modell durch reine Rechenarbeit ergibt.

[20] Diese Sichtweise schließt nicht aus, daß sich die Gesamtstöchiometrie des Wachstums ändern kann. Was sich ändert, sind aber nicht die Koeffizienten der einzelnen Stoffwechselwege, sondern die Reaktionsraten, die die Regulation beschreiben. In einem gewissen Gegensatz dazu stehen die Variable-Yield-Modelle, die rein phänomenologischer Natur sind (vgl. 5.2.6.5).

[21] Reicht diese Annahme nicht aus, so kann man zu strukturierten Modellen übergehen (vgl. 5.2.7).

[22] Man vergleiche vor den in Abbildung 5.7 schematisierten Ablauf mit dem in [Roe83] vorgestellten. Die Unterschiede werden in 5.2.6.6 und 6.6.4.3 diskutiert.

[23] Die beschriebenen Modelle wurden halbautomatisch mit Hilfe des DERIVE-Systems [Der89] erstellt. In [Mel91] ist ein System zur Behandlung chemischer Reaktionsgleichungen mit dem REDUCE-System vorgestellt.

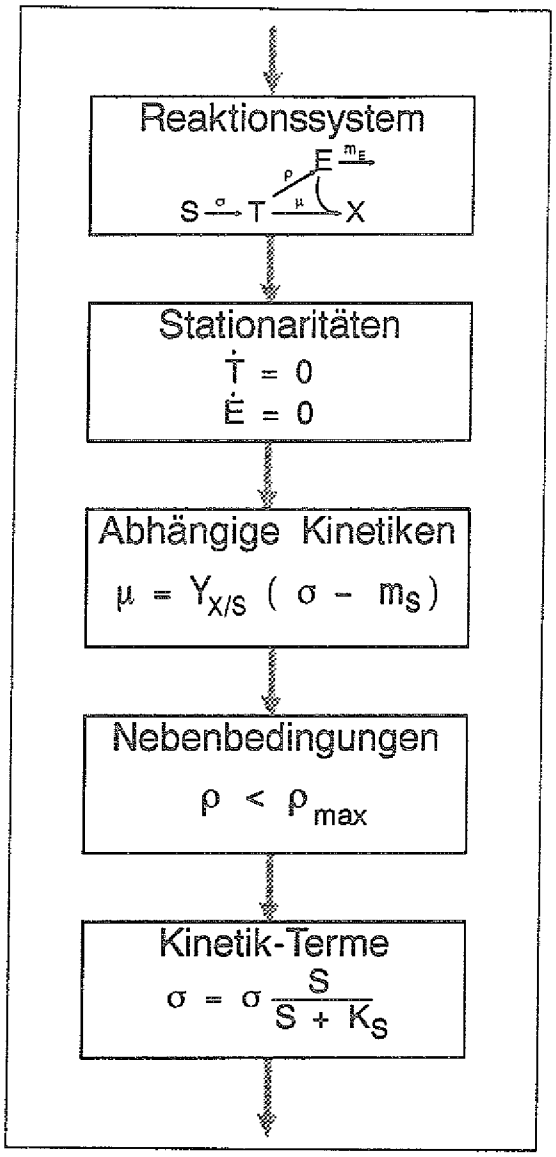


Abbildung 5.7: Ablaufschema für die Stoffwechsel-Modellierung

5.2.5.6 Konsequenzen für die Datenanalyse

Das in dieser Arbeit entwickelte Datenanalyse-Konzept für biotechnische Prozeßdaten tritt mit dem Anspruch auf allgemeine Anwendbarkeit auf. Da die modellgestützten Methoden im Mittelpunkt stehen, ist der Nachweis erforderlich, daß das System allen Situationen gerecht wird. Ein solcher Nachweis ist zwar nicht schlüssig zu führen, jedoch zeigen die Beispiele, daß sich bisher beschriebene (auch komplexere) Modelle mühelos einordnen lassen (vgl. auch die Einleitung von Kapitel 5).

Als Nebenprodukt tritt — in Form verallgemeinerter Reaktionssysteme — eine allgemeine Struktur der Stoffwechselmodelle zutage. Datenanalysetechniken wie z.B. Zustandsschätzung und Bilanzierung können direkt auf diesen Modellen aufsetzen, ohne daß sie eigens dafür vorbereitet werden müssen. Wichtig ist, daß alle Größen mit den Einheiten auftreten, mit denen der Biotechnologe sie im Fermenter vorfindet. Meßdaten können daher ohne weitere Umrechnungen zur Datenanalyse eingesetzt werden [24].

Als Nebeneffekt wurde klar, daß sogar der rein algebraische Aufwand der Modellbildung automatisierbar ist und in den Datenanalyse-Zyklus eingebunden werden kann.

[24] Die Umrechnung von Einheiten kann im übrigen ebenfalls automatisiert werden, wenn dies erforderlich wird.

5.2.6 Eine allgemeine Modellstruktur

5.2.6.1 Formalisierung der Vorgehensweise

Die Stoffwechselmodellierung auf der Basis von verallgemeinerten Reaktionssystemen führt zu einer einheitlichen formalen Modellstruktur, auf der alle Datenanalyseverfahren aufsetzen können (vgl. hierzu auch die Abb. 6.10):

1. Ein verallgemeinertes Reaktionssystem für die Anwendung in der Biotechnologie hat die Gestalt (vgl. 5.1.3.3 und 5.2.1.3):

$$\begin{pmatrix} \dot{X}_1 \\ \vdots \\ \dot{X}_n \end{pmatrix} = \begin{pmatrix} \nu_{1,1} & \cdots & \nu_{1,m} \\ \vdots & \ddots & \vdots \\ \nu_{n,1} & \cdots & \nu_{n,m} \end{pmatrix} \cdot X_1 \cdot \begin{pmatrix} r_1 \\ \vdots \\ r_m \end{pmatrix}$$

   Dabei bezeichnet X_1 stets die Biomasse und die r_i darauf bezogene *spezifische* Reaktionsraten.

2. Wählt man die Indizierung so, daß $X_{n-s+1}, \ldots, X_n$ die stationären Komponenten sind, so folgt:

$$0 = \begin{pmatrix} \dot{X}_{n-s+1} \\ \vdots \\ \dot{X}_n \end{pmatrix} = \begin{pmatrix} \nu_{n-s+1,1} & \cdots & \nu_{n-s+1,m} \\ \vdots & \ddots & \vdots \\ \nu_{n,1} & \cdots & \nu_{n,m} \end{pmatrix} \cdot X_1 \cdot \begin{pmatrix} r_1 \\ \vdots \\ r_m \end{pmatrix}$$

3. Unter entsprechenden Forderungen an den Rang der Koeffizientenmatrix $((\nu_{i,j}))$, die hier nicht erörtert werden sollen [25], leitet man daraus ab:

$$\begin{pmatrix} r_{m-s+1} \\ \vdots \\ r_m \end{pmatrix} = M \cdot \begin{pmatrix} r_1 \\ \vdots \\ r_{m-s} \end{pmatrix}$$

   mit einer gewissen $s \times (m-s)$-Matrix M. D.h. die abhängigen Kinetiken $r_{m-s+1}, \ldots, r_m$ ergeben sich als Linearkombinationen der unabhängigen Kinetiken $r_1, \ldots, r_{m-s}$

4. Durch Eliminierung der abhängigen Kinetiken folgt die allgemeine Modellstruktur

$$\begin{pmatrix} \dot{X}_1 \\ \vdots \\ \dot{X}_{n-s} \end{pmatrix} = \begin{pmatrix} \nu_{1,1} & \cdots & \nu_{1,m} \\ \vdots & \ddots & \vdots \\ \nu_{n-s,1} & \cdots & \nu_{n-s,m} \end{pmatrix} \begin{pmatrix} 1 \\ M \end{pmatrix} \cdot X_1 \cdot \begin{pmatrix} r_1 \\ \vdots \\ r_{m-s} \end{pmatrix},$$

   wobei $(1\ M)^T$ die aus M und einer $(m-s) \times (m-s)$-Einheitsmatrix zusammengesetzte Matrix bezeichnet.

Die resultierende Modellstruktur läßt sich kurz schreiben als:

$$\dot{X} = N \cdot X_1 \cdot r \quad \textit{(Allgemeines Stoffwechselmodell)}$$

mit

- dem *Kinetik-Vektor*

$$r = (r_1, \ldots, r_{m-s})^T$$

- und der *stöchiometrischen Matrix*

$$N = \begin{pmatrix} \nu_{1,1} & \cdots & \nu_{1,m} \\ \vdots & \ddots & \vdots \\ \nu_{n-s,1} & \cdots & \nu_{n-s,m} \end{pmatrix} \cdot (1\,M)^T$$

[25] Exakte Formulierungen der Voraussetzungen. und Beweise für solche Matrizen-Umformungen finden sich in [Red88], [Gie88].

5.2.6.2 Basisdarstellung des theoretischen Reaktionsraums

Es soll bei der Notation $\dot{X} = N \cdot X_1 \cdot r$ im folgenden stets davon ausgegangen werden, daß daß die Operationen zur Dimensionsreduktion soweit durchgeführt worden sind bis

1. die Kinetiken linear unabhängig sind.
2. die Matrix N Maximalrang hat.

Ferner soll gelten [26]

3. $m = \dim r \;<\; n = \dim X$.

Unter diesen Bedingungen bilden die Spaltenvektoren $N_1, \ldots, N_{m-s}$ der Matrix N gerade eine Basis des theoretischen Reaktionsraums (vgl. 5.1.2.3):

$$\operatorname{im} N = \operatorname{span}\{N_1, \ldots, N_{m-s}\}$$

Diese Basisdarstellung ist für die statistischen Modelle und Methoden von fundamentaler Bedeutung (vgl. 6.6).

Bemerkung: Um für einen konkreten Prozeßverlauf den Reaktionsraum (als affinen Unterraum des $\Re^{n-s}$) zu bestimmen, müssen zusätzlich die Startwerte $X_0 = X(0)$ hinzu addiert werden (vgl. 5.1.2.3). Im folgenden soll diese Unterscheidung nicht mehr erfolgen und es wird bereits bei $\operatorname{im} N$ vom *theoretischen Reaktionsraum* gesprochen.

5.2.6.3 Einige Beispiele

Bei den oben vorgeführten Beispielen ergibt sich:

- Einfaches Wachstumsmodell (vgl. 5.2.1.3):

$$\begin{pmatrix} \dot{X} \\ \dot{S} \end{pmatrix} = \begin{pmatrix} Y_{X/S} \\ -1 \end{pmatrix} \cdot X \cdot \begin{pmatrix} \sigma(S) \end{pmatrix}$$

- Wachstum mit Erhaltungsstoffwechsel (vgl. 5.2.2.3):

$$\begin{pmatrix} \dot{X} \\ \dot{S} \end{pmatrix} = \begin{pmatrix} Y_{X/S} & -Y_{X/S} \\ -1 & 0 \end{pmatrix} \cdot X \cdot \begin{pmatrix} \sigma(S) \\ m_s(S) \end{pmatrix}$$

- Wachstum mit wachstumsgekoppelter Produktbildung (vgl. 5.2.3.3):

$$\begin{pmatrix} \dot{X} \\ \dot{S} \\ \dot{P} \end{pmatrix} = \begin{pmatrix} Y_{X/S} & -Y_{X/S} \\ -1 & 0 \\ Y_{P/S} & -Y_{P/S} \end{pmatrix} \cdot X \cdot \begin{pmatrix} \sigma(S) \\ m_s(S) \end{pmatrix}$$

- Backhefe-Modell in der Formulierung aus Abschnitt 5.2.4.2:

$$\begin{pmatrix} \dot{X} \\ \dot{S} \\ \dot{A} \end{pmatrix} = \begin{pmatrix} \mu_\sigma & -\mu_\eta & \mu_\pi \\ -1 & 0 & 0 \\ 0 & -1 & 1 \end{pmatrix} \cdot X \cdot \begin{pmatrix} \sigma(S) \\ \eta(S,A) \\ \pi(S) \end{pmatrix}$$

  mit den Koeffizienten μ_σ, μ_η, μ_π aus Abschnitt 5.2.4.2.

- Backhefe-Modell in der Formulierung von Sonnleitner und Käppeli (vgl. 5.2.4.5):

$$\begin{pmatrix} \dot{X} \\ \dot{S} \\ \dot{A} \end{pmatrix} = \begin{pmatrix} Y^{ox}_{X/S} & Y^{red}_{X/S} & Y_{X/A} \\ 1 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \cdot X \cdot \begin{pmatrix} \sigma_{ox}(S) \\ \sigma_{red}(S) \\ \eta \quad (S,A) \end{pmatrix}$$

[26] Die Forderung wird der Einfachheit halber gestellt, um im folgenden unnötige Fallunterscheidungen zu vermeiden. Sie ist bei praktischen Anwendungen im allgemeinen erfüllt.

5.2.6.4 Ein Vergleich mit bestehenden Ansätzen

Die allgemeine Modellstruktur $\dot{X} = N \cdot X_1 \cdot r$ ist für die Methode des Pathway Modelling bekannt [NS89] [ST89]. Die wesentliche Erweiterung ist, daß auf der Basis verallgemeinerter Reaktionssysteme keine Annahmen mehr über die konkrete Stöchiometrie gemacht werden müssen.

Das Modell beschreibt die Stoffwechselvorgänge durch eine konstante Stöchiometrie (repräsentiert durch die Matrix N) und eine zustandsabhängige Kinetik (repräsentiert durch $r(X)$). Auf dieser Zerlegung beruht auch die *Metabolic Control Theory*, die versucht, Eigenschaften von Reaktionssystemen auf Eigenschaften von N und $r(X)$ zurückzuführen [Red88] [KW86].

Im Falle einer bekannten Stöchiometrie der einzelnen Reaktionen (Pathway Modeling) ist die Matrix N a priori bekannt. In diesem Fall können entsprechend leistungsfähige Methoden zur Datenanalyse angewendet werden. Der Vorteil des allgemeinen Modells ist aber der, daß ein Grundstock von Methoden auch bereits bei unbekanntem N anwendbar ist. Von besonderer Bedeutung ist dabei die in [Lia89] vorgeschlagene Hauptkomponentenanalyse zur empirischen Bestimmung des Reaktionsraums. Wie in Abschnitt 6.6 gezeigt wird, kann die Stöchiometrie auch nachträglich berücksichtigt werden. Dies führt zu einer Trennung der Methoden zur Datenalyse in solche, die die Stöchiometrie bereits voraussetzen und solche, die dies nicht tun. Diese Trennung ermöglicht die Einordnung bestehender Methoden in einen wohlgeordnetes Instrumentarium sowie die kritische Bewertung ihrer Verwendbarkeit (vgl. hierzu die Ausführungen in Kapitel 10 und 11).

In [BHLD89] ist ein weiterer Zugang zur Modellbildung vorgestellt, der ebenfalls auf dem allgemeinen Modell aufbaut. Dabei wird auf die Modellierung konkreter Stoffwechselwege völlig verzichtet und eine phänomenologische Beschreibung der stöchiometrischen Zusammenhänge gewählt. Der hier beschriebene Ansatz kann somit methodisch zwischen dem Pathway Modelling, das oft aufgrund einer zu geringen Systeminformation nicht möglich ist, und der in [BHLD89] beschriebenen Technik, die einen weniger erklärenden Charakter hat, angesiedelt werden (vgl. dazu auch 4.1.2).

Ein sehr allgemeiner Ansatz ist der von Westernhoff und van Dam [WvD87], der vollständig auf den Begriffen der Thermodynamik aufbaut. Diese Systematik zeichnet sich durch große Konstistenz aus. Insbesondere findet sich in [vDMdMW88] eine alternative Herleitung des Erhaltungsstoffwechselmodells mit thermodynamischen Begriffen. Die Theoriebildung in [WvD87] geht weit über die biotechnische Anwendbarkeit hinaus. Insbesondere erschwert die thermodynamische Begrifflichkeit die Anwendung auf konkrete Meßdaten.

5.2.6.5 Modelle mit veränderlicher Stöchiometrie

Eine Variante der beschriebenen allgemeinen Stoffwechselmodelle stellen die *Variable-Yield-Modelle* dar [CLT76] [HSRD78]. Sie beschreiben die Veränderungen in der globalen Stöchiometrie des Wachstums nicht durch zusätzlich eingeführte Stoffwechselwege mit konstanter Stöchiometrie, sondern durch Annahme variabler stöchiometrischer Koeffizienten.
Dies soll am Beispiel des Erhaltungsstoffwechselmodells

$$\begin{pmatrix} \dot{X} \\ \dot{S} \end{pmatrix} = \begin{pmatrix} Y_{X/S} & -Y_{X/S} \\ -1 & 0 \end{pmatrix} \cdot X \cdot \begin{pmatrix} \sigma(S) \\ m_s(S) \end{pmatrix}$$

kurz illustriert werden: Die erste Gleichung kann umgeformt werden zu (vgl. 5.2.2.3)

$$\dot{X} = Y^*_{X/S}(S) \cdot \sigma(S) \cdot X$$

mit dem zustandsabhängigen Ertragskoeffizienten

$$Y^*_{X/S}(S) = Y_{X/S} \cdot (1 - \frac{m_s(S)}{\sigma(S)}) \; .$$

Das Modell erhält die alternative Struktur

$$\begin{pmatrix} \dot{X} \\ \dot{S} \end{pmatrix} = \begin{pmatrix} Y^*_{X/S}(S) \\ -1 \end{pmatrix} \cdot X \cdot \sigma(S) \quad ,$$

entsprechend der allgemeinen Modellstruktur

$$\dot{X} = N(X) \cdot X_1 \cdot r(X) \quad .$$

In dieses Modell gehen $N(X)$ und $r(X)$ beide als Funktionen von X ein und sind daher nicht mehr qualitativ voneinander unterschieden. Auf linearen Modellen beruhende Datenanalysetechniken sind darauf nicht mehr in jedem Fall anwendbar. Aus diesem Grunde sollte die Verwendung von Variable-Yield-Modellen zur Stoffwechselmodellierung vermieden werden. Stattdessen kann man versuchen, derartige Modelle durch Einführung zusätzlicher kinetischer Terme in Modelle mit konstanter stöchiometrischer Matrix zu überführen.

5.2.6.6 Globale Wachstumsgleichungen

Der von Roels in [Roe83] vorgestellte Ansatz geht von einer *globalen Wachstumsgleichung* der Form

$$\nu_S(S,\ldots) \cdot S + \nu_O(S,\ldots) \cdot O + \nu_N(S,\ldots) \cdot N \longrightarrow X + \nu_P(S,\ldots) \cdot P + \nu_H(S,\ldots) \cdot H + \nu_C(S,\ldots) \cdot C$$

(S - Substrat, O - Sauerstoff, N - Stickstoffquelle, P - Produkt, H - H_2O, C - CO_2) aus. Dem entspricht das allgemeine Modell

$$\frac{d}{dt}\begin{pmatrix} S \\ O \\ N \\ X \\ P \\ H \\ C \end{pmatrix} = \begin{pmatrix} \nu_S(S,\ldots) \\ \nu_O(S,\ldots) \\ \nu_N(S,\ldots) \\ 1 \\ \nu_P(S,\ldots) \\ \nu_H(S,\ldots) \\ \nu_C(S,\ldots) \end{pmatrix} \cdot X \cdot \mu(S,P,\ldots)$$

Es kann in dieser Form als Variable-Yield-Modell gelesen werden. Der Zusammenhang zwischen den einzelnen Stoffwechselwegen und der globalen Stöchiometrie tritt dabei nicht explizit hervor, was die praktische Anwendung auf ein konkretes biologisches System erschwert.

5.2.7 Strukturierte Modelle

5.2.7.1 Instationäre zellinterne Größen

Strukturierte Modelle ergeben sich im Kontext des allgemeinen Modellierungsansatzes durch Wegnahme einer oder mehrerer Stationaritätsannahmen für zellinterne Komponenten. Interessant wird diese Erweiterung bei langsam veränderlichen zellinternen Größen. Dazu gehören sicherlich nicht das bisher betrachtete zellinterne Intermediat T oder die Energie E, sondern vielmehr neue Größen wie:

- Enzymkonzentrationen,
- Reservestoffe,
- DNA/RNA-Konzentrationen und
- Ribosomenzahl.

Die verschiedenen Modellansätze lassen sich nicht mehr in Kürze zusammenfassen. Es soll daher ein Beispiel genügen. Für weitere Modelle sei auf die Literatur verwiesen (z.B. [HR82] [Ber83] [BHLD89] [Reu89]).

5.2.7.2 Regulation einer Enzymkonzentration

Das Backhefe-Modell von Sonnleitner und Käppeli ist nicht in der Lage, das Wachstum von *Saccharomyces cerevisiae* in einer Batch-Kultur zu beschreiben [27]. Es wurde daher von Sweere et.al. [SGB+88] für diese Situation erweitert. Das Simulationsbeispiel in Kapitel 2 beruht auf dem erweiterten Modell, die verwendeten Parameter sind in [SK86] und [SGB+88] nachzulesen.

Die Diauxie bei der Batch-Kultivierung (vgl. Abb. 2.1) kann jetzt erst adäquat erklärt werden: Nach Aufbrauchen der Glucose steht den Zellen nur noch das vorher produzierte Äthanol zur Verfügung. In dieser Situation spielt die bisher immer vernachlässigte Gluconeogenese (siehe 5.2.4.1) die entscheidende Rolle, denn die Synthese der Grundbausteine für Polysaccharide muß nun in der Zelle selbst erfolgen. Die Enzyme der Gluconeogenese wurden während der Glucose-Wachstumsphase nicht benötigt und müssen folglich neu synthetisiert werden.

Um diesen Effekt zu modellieren, muß zu den bisherigen Beschreibungsgrößen X, S und A eine neue zellinterne Größe Z hinzugenommen werden. Z faßt die Enzyme der Gluconeogenese zusammen und hat eine fiktive Einheit. Die Dynamik von Z wird durch einen linearen Regler — als dem einfachsten denkbaren Modell — beschrieben:

$$\dot{Z} = \frac{1}{\tau} \cdot (Z_{soll} - Z)$$

Der Regler mit der Zeitkonstanten τ regelt die Enzymkonzentration Z auf ihren Sollwert Z_{soll} ein. Die Gleichung ist in der Lage, sowohl die Enzymsynthese ($Z < Z_{soll}$) als auch den Enzymabbau ($Z > Z_{soll}$) zu beschreiben. Der Sollwert wird nun durch

$$Z_{soll} = \frac{A}{A + K_A} \cdot \frac{I_S}{I_S + S}$$

vorgegeben. Dieser Term entspricht bis auf die Skalierungskonstante η_A genau der Äthanol-Aufnahmerate η im Modell von Sonnleitner und Käppeli. Im neuen Modell ist daher:

$$\tilde{\eta} = \eta_A \cdot Z$$

Bei "eingeregelter" Enzymkonzentration $Z \approx Z_{soll}$ geht das neue Modell in das alte über.

5.2.7.3 Die Problematik strukturierter Modelle

Die Problematik des Modells für die Enzymsynthese tritt deutlich zutage, wenn man sie im Kontext eines Reaktionssystems durch die beiden Reaktionen

$$\rightarrow Z \quad (aufbauendeReaktion) \quad \text{und} \quad Z \rightarrow \quad (abbauendeReaktion)$$

beschreibt [28]. Diese Reaktionen sind in keiner Weise an die anderen Reaktionen des Hefemodells gekoppelt. Somit kann keine direkt damit zusammenhängende Größe ohne weiteres von außen gemessen werden [29] ! D.h. die Modellierung der Synthese- und Abbaureaktionen geschieht in gewisser Weise blind. Die Annahme eines Reaktionsmechanismus kann zwar auf dem Wege einer Parameteranpassung (vgl. 10.2) statistisch untermauert werden, jedoch gibt es hierfür kein direktes Indiz. Auf die Problematik der Parameteranpassung in Bezug auf die Validierung von zeitabhängigen Modellen wird in Abschnitt 10.3 noch ausführlich eingegangen.

Formal drückt sich diese Tatsache so aus, daß die stöchiometrische Matrix N im allgemeinen Stoffwechselmodell (5.2.6.1) nicht mehr konstant ist, sondern vielmehr von Z abhängt:

$$\dot{X} = N(X, S, Z) \cdot X_1 \cdot r$$

[27] Es wurde für eine Fed-Batch-Kultur (vgl. 5.3.3.1) entwickelt, in der ständig schwankende Glucose- und Äthanol-Konzentrationen vorliegen.

[28] Die beiden zugehörigen Reaktionsraten sind im Regler-Modell zu einer einzigen zusammengefaßt worden, die auch negativ werden kann

[29] Die On-Line–Bestimmung intrazellulärer Enzymkonzentrationen ist ein anspruchsvolles Problem für die Meßtechnik [ANR+86].

Für den Fall einer konstanten Konzentration Z (eingeregelter Sollwert) geht das Modell in das unstrukturierte Modell aus 5.2.4.5 über, auf das alle linearen Bilanzierungs-Methoden aus Kapitel 6.6 angewendet werden können.

Diese Situation ist formal identisch mit der bei den in 5.2.6.5 diskutierten Variable-Yield-Modellen. Die Abschnitt 5.2.6.5 gemachten Bemerkungen zur Anwendbarkeit von Datenanalysetechniken übertragen sich daher sinngemäß auf strukturierte Modelle. Es sollte daher erst dann zu strukturierten Modellen übergegangen werden, wenn unstrukturierte Modelle in ihren Möglichkeiten erschöpft sind. Daß dies für den praktischen Einsatz nicht unbedingt erforderlich sein muß, zeigen die Phasen-Modelle in 10.3 und 22.3, die zusätzliche On-Line-Information aus dem System aufnehmen und z.B. auf die exakte Beschreibung einer Lag-Phase verzichten.

5.3 Modellierung von Bioreaktoren

Nachdem die Grundlagen zur Modellierung des biochemischen und biologischen Subsystems gelegt sind, fehlt als letztes die Beschreibung des physikalisch-technischen Systems Bioreaktor. Hier bekommt man es mit räumlich verteilten Größen zu tun. Es soll im folgenden begründet werden, warum man in einem Rahmen, in dem eine modellgestützte Datenanalyse sinnvoll ist, von einer bestimmten Modellstruktur ausgehen sollte:

5.3.1 Das allgemeine Reaktormodell

5.3.1.1 Zustandsvariablen

Ganz allgemein sind *Zustandsvariablen* für ein zu modellierendes System ein Satz von zeitabhängigen Variablen

$$X_1(t), \ldots, X_n(t),$$

die zu jedem Zeitpunkt alle Informationen enthalten, welche die zukünftige Dynamik des Systems bestimmen. Vielfach sind solche Zustandsvariablen räumlich verteilte Größen, d.h. sie hängen zusätzlich von einer Raumkoordinate ξ ab. Die Zustandsvariablen werden zu einem Zustandsvektor:

$$X(t,\xi) = (X_1(t,\xi), \ldots, X_n(t,\xi)),$$

zusammengefaßt.

Einer Modellierung liegt stets zugrunde, daß man eine Reihe von Variablen auswählt und annimmt, daß Zustandsvariablen vorliegen. Im Falle eines biologischen Systems kann mit einer kleinen Anzahl von Zustandsvariablen sicherlich nur eine Approximation an das wirkliche Systemverhalten erreicht werden. Als Zustandsvariablen wählt man — mit Ausnahme der thermodynamischen Grundgrößen Druck, Temperatur und Volumen — fast ausschließlich Konzentrationsgrößen der einzelnen beteiligten Stoffe, wie z.B.

- Glucosekonzentration,
- K^+-Konzentration,
- Biomassekonzentration,
- Konzentration des reproduktiven Apparats (DNA, RNA usw.) in der Zelle,
- Intrazelluläre und extrazelluläre Enzymkonzentrationen.

Es geht im folgenden darum, zu beschreiben, wie die einzelnen Stoffkonzentrationen in den verschiedenen Bereichen eines Reaktors zustande kommen.

5.3.1.2 Konvektion, Diffusion und Reaktion

Änderungen der räumlichen Verteilung aller Komponenten in einem Bioreaktor kommen — wie auch in einem chemischen Reaktor — ausschließlich zustande durch

Konvektion, d.h. durch Strömung im Reaktor sowie durch Zuflüsse und Abflüsse.

Diffusion, d.h. durch die Stoffverteilung über den Raum aufgrund der Brownschen Bewegung.

Reaktion, d.h. durch chemische Stofftransformation, aber auch durch Phasenübergänge, wie z.B. zwischen Gasphase und Gelöst-Gasphase.

Zusammen mit dem allgemeinen Stoffwechselmodell für das biologische System werden diese Vorgänge beschrieben durch das partielle Differentialgleichungssystem (vgl. [Wan88] und 5.2.6.1):

$$\begin{array}{rcccccl} \dot{X}_i = & \operatorname{div}(X_i \cdot u) & + & D_i \operatorname{div}(\operatorname{grad} X_i) & + & N \cdot X_1 \cdot r(X) & ,i = 1,\ldots,n-s \\ & \text{(Konvektion)} & + & \text{(Diffusion)} & + & \text{(Reaktion)} & \end{array}$$

mit

X	$=$	$X(t,\xi)$	:	Stoffkonzentrationen, speziell
X_1	$=$	$X_1(t,\xi)$		die Biomassekonzentration,
u	$=$	$u(t,\xi)$	:	Strömungsgeschwindigkeit,
D_i			:	Diffusionskonstanten,
N			:	stöchiometrische Matrix,
r	$=$	$r(X(t,\xi))$	:	spezifische Reaktionsraten im biologischen Subsystem.

5.3.1.3 Praktische Anwendbarkeit

Ein solch allgemeines Modell ist für den praktischen Einsatz ungeeignet. Dies liegt zum einen an der Zeitkomplexität und numerischen Problematik der Lösung partieller Differentialgleichungen, zum anderen aber auch an der Unmöglichkeit, ein reales strömendes System mit seinem chaotischen Verhalten (Turbulenz) exakt zu beschreiben [Lev72].

Benötigt werden brauchbare Approximationen an das reale System. Bei der Auswahl einer solchen Approximation muß darauf geachtet werden, daß die zur Verfügung stehende Meßtechnik eine ausreichende Menge an Stützdaten für die Modellierung bereitstellt: Ein komplexes Modell mit vielen Parametern hat wenig Aussichten auf Validierung, wenn nur eine Größe am Ausgang des Reaktors meßbar ist!

5.3.2 Kompartimentierung

5.3.2.1 Einteilung in Kompartimente und Homogenitätsannahmen

Insbesondere in Hinblick darauf, daß Meßfühler i.A. nur in diskreten Abständen über den Reaktor verteilt angeordnet werden können [30], erweist es sich als sinnvoll, den Reaktor in Kompartimente zu unterteilen. Ein Kompartimente kann ein räumlich zusammenhängender Bereich des Reaktors sein, jedoch durchaus auch die Zusammenfassung disjunkter Bereiche mit gleichen Eigenschaften. Einige Beispiele:

- Ein Strömungsrohrreaktor kann in aufeinanderfolgende zylindrische Abschnitte eingeteilt werden (*Diskretisierung* des Raums, vgl. Abb. 5.8a). Oft wählt man genau so viele Einteilungen, wie Meßfühler vorhanden sind [LMT85].
- Bei einem inhomogen gerührten Reaktor können die Zonen guter und schlechter Durchmischung zu je einem Kompartiment zusammengefaßt werden (vgl Abb. 5.8b) [SMZ+88].
- Bei einem Reaktor, der auf Trägern immobilisierte Biomasse enthält, lassen sich das freie Flüssigkeitsvolumen und der Träger mit seinem Bewuchs unterscheiden (vgl. Abb. 5.8c). Der Bewuchs des Trägers könnte wiederum in mehrere übereinanderliegende Schichtungen kompartimentiert werden.

[30] Es gibt Ausnahmen bei speziellen Meßtechniken, z.B. bei der Untersuchung der Blasenverteilung [WBLL89].

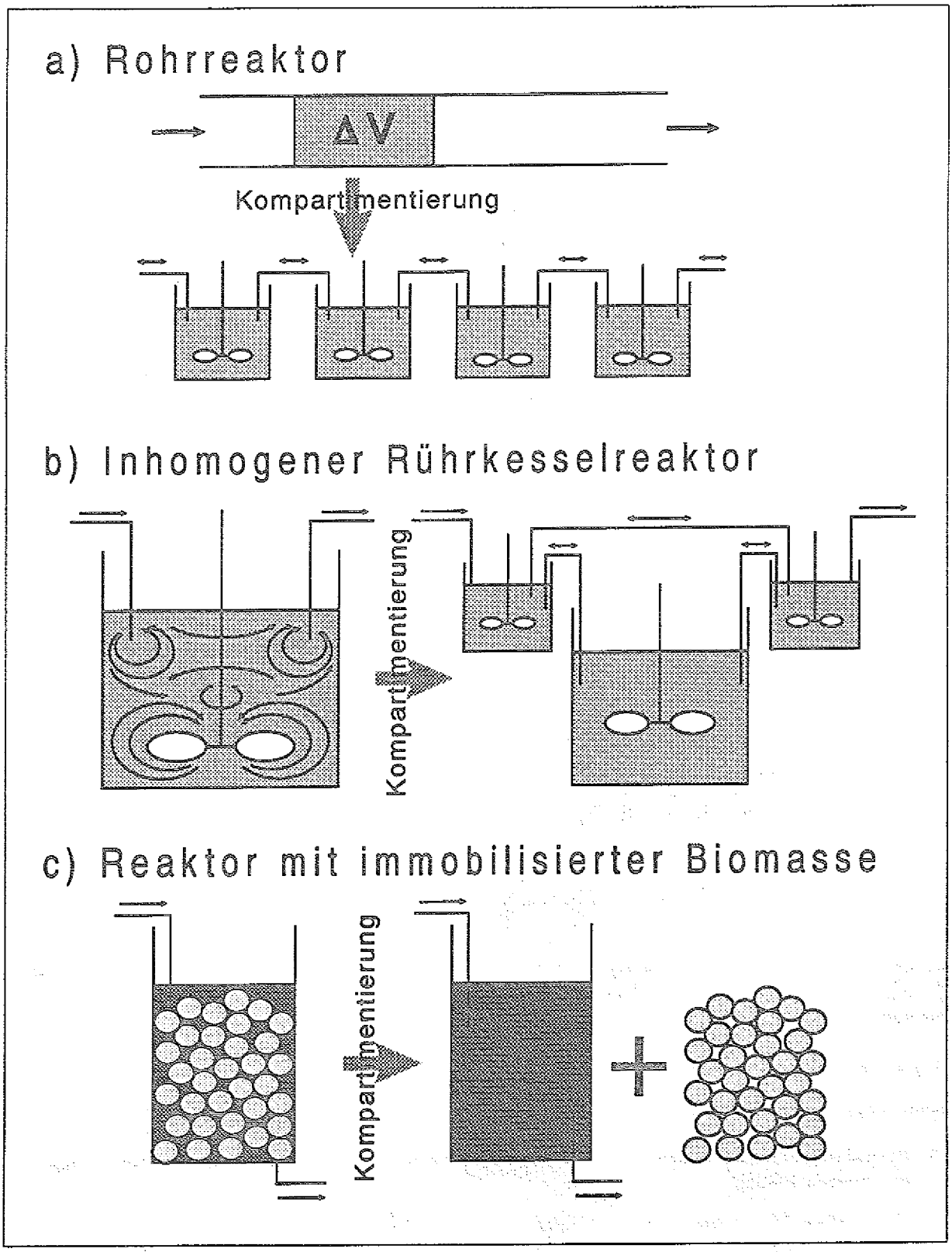


Abbildung 5.8: Kompartimentierung von Bioreaktoren

- Altersverteilungsmodelle unterscheiden Zellen in verschiedenen Altersstadien (gemessen ab der Zellteilung) voneinander [Ran86].

Für jedes Kompartiment wird nun die *Homogenitätsannahme* getroffen, daß innerhalb des Kompartiments alle Stoffe homogen verteilt vorliegen. Gradienten in den Stoffkonzentrationen finden sich daher nur zwischen je zwei Kompartimenten. Der Stoffaustausch zwischen den Segmenten kann dann durch einen Flußterm, der je zwei Abschnitte koppelt, beschrieben werden. Der Diffusionsanteil läßt sich in dieser diskretisierten Sichtweise als weiterer Flußterm auffassen.

5.3.2.2 Modellierung eines einzelnen Kompartiments

Das allgemeine Reaktormodell vereinfacht sich aufgrund der Homogenitätsanahme. Der Diffusionsterm und der Konvektionsterm reduziert sich auf die Flüsse über die Systemgrenzen hinweg:

$$\dot{X} = N \cdot X_1 \cdot r(X) + \frac{\vec{F}_{in} - \vec{F}_{out}}{V} \quad \begin{array}{lll} \text{mit} & \text{dem Zuflußvektor} & \vec{F}_{in}, \\ & \text{dem Abflußvektor} & \vec{F}_{out} \\ \text{und} & \text{dem Volumen} & V. \end{array}$$

Die Skalierung mit dem Volumen V ist erforderlich, weil die Gleichung mit Konzentrationsgrößen formuliert ist. In der speziellen Situation, wo alle Komponenten in einer flüssigen Phase vorliegen und der Stofftransport allein durch die Volumenströme

$$F_{in} \ , \ F_{out}$$

bewirkt wird, gilt:

$$\vec{F}_{in} = F_{in} \cdot X_{in} \quad \text{und} \quad \vec{F}_{out} = F_{out} \cdot X$$

mit dem Vektor X_{in} der Stoffkonzentrationen im Zufluß. Das Modell hat dann die Form:

$$\dot{X} = N \cdot X_1 \cdot r(X) + \frac{F_{in} \cdot X_{in} - F_{out} \cdot X}{V}$$

5.3.2.3 Modellierung des Gesamtsystems

Das Modell für das Gesamtsystem läßt sich nun aus den Modellen der Kompartimente zusammenfügen. Dabei müssen die Flußvektoren $\vec{F}_{in}, \vec{F}_{out}$ entsprechend aufgeschlüsselt werden:

$$\dot{X}^i = N \cdot X_1^i \cdot r(X^i) + \sum_{j=0}^{n} \frac{\vec{F}_{in}^{ij} - \vec{F}_{out}^{ij}}{V}, \ i = 1, \ldots, K$$

Hier bezeichnen die oberen Indizes der Zustandsvektoren die einzelnen Abschnitte $1, 2, \ldots, K$ und die $\vec{F}_{in}^{ij}$, $\vec{F}_{out}^{ij}$ die Flüsse zwischen den Kompartimenten i und j. Das nullte Kompartiment ist die äußere Umgebung des Reaktors.

5.3.2.4 Zusammenfassung

Zusammengefaßt gilt:

1. Bioreaktoren müssen für den Zweck des On-Line-Einsatzes mit gewöhnlichen Differentialgleichungen beschrieben werden.
2. Man zerlege den Reaktor dazu in annähernd homogen durchmischte Segmente.
3. Die Homogenitätsbedingungen sind zu prüfen und gegebenenfalls durch konstruktive Maßnahmen zu verbessern. Andernfalls muß die Diskretisierung (und damit auch die Meßtechnik) verfeinert werden.

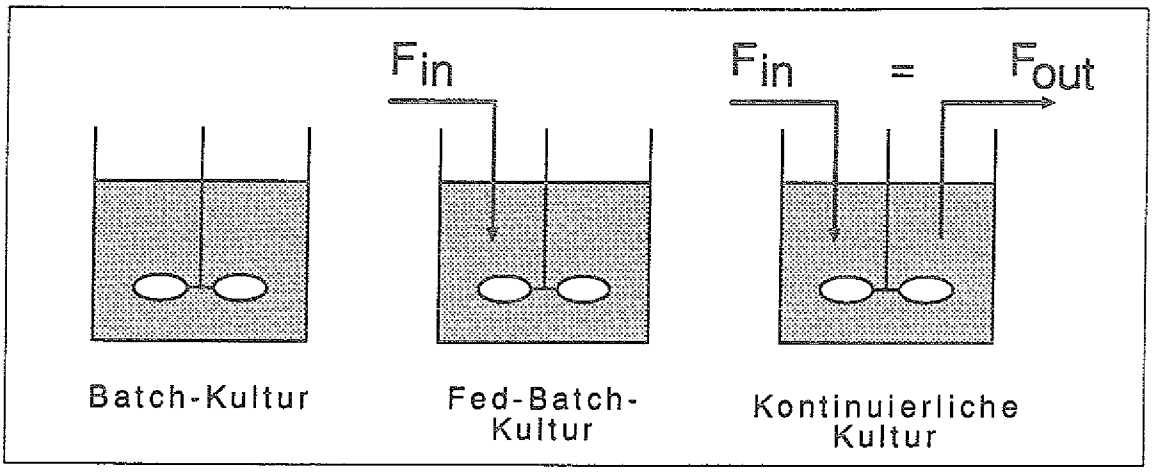


Abbildung 5.9: Betriebsweisen im Rührkesselreaktor

4. Jedes Segment sollte gleich instrumentiert werden, um überall gleiche Informationsdichte zu gewährleisten.

Unter diesen Annahmen kann man sich auf die Modellierung eines einzelnen homogen gerührten Kessels beschränken. Davon soll im Rahmen dieser Arbeit ausgegangen werden [31].

5.3.3 Beispiele

5.3.3.1 Betriebsweisen des Rührkessels

Der homogen gerührte Rührkesselreaktor erweist sich als "Keimzelle" der Reaktormodellierung. Im Fermenterlabor bietet er oft ideale Bedingungen zum Studium des biologischen Systems. In diesem Fall liegt nur ein Kompartiment vor; alle Komponenten sind in der flüssigen Phase gelöst und können nur durch einen Zulauf und einen Abfluß die Systemgrenzen überschreiten [32]. Das allgemeine Modell für diesen Fall wurde bereits in Abschnitt 5.3.2.2 vorgestellt. Man kennt drei typische Betriebsweisen des Rührkessels (vgl. Abb. 5.9):

Batch-Kultur: $F_{in} = 0$ und $F_{out} = 0$
Hier wird der Reaktor zu Anfang gefüllt und dann als geschlossenes System belassen:

$$\dot{X} = N \cdot X_1 \cdot r(X)$$

Kontinuierliche Kultur: $F_{in} = F_{out} > 0$
In dieser Situation besteht ein kontinuierlicher Durchfluß, der bei konstantem Volumen eine Nährstoffzufuhr und einen Abtransport der gelösten Stoffe bewirkt. Mit der *Durchflußrate*

$$D = \frac{F_{in}}{V} = \frac{F_{out}}{V}$$

(ausgetauschter Volumenanteil pro Zeiteinheit) folgt

$$\dot{X} = N \cdot X_1 \cdot r(X) + (X_{in} - X) \cdot D.$$

[31] Auch die regelungstechnische Literatur für verteilte Reaktormodelle geht letzlich von einem Kompartimentierungs- bzw. Diskretisierungsansatz aus [Ray81] [K˙h77].

[32] Die gasförmigen Komponenten, die über die Belüftung an ganz anderer Stelle das System verlassen, sollen zunächst außer acht gelassen werden.

Fed-Batch-Kultur [KD78] [YS84]: $F_{in} > 0,\ F_{out} = 0$
Die Nährstoffzufuhr findet über eine Zufütterung

$$F = F_{in}$$

statt, wobei das Volumen des Systems (mangels Abfluß) zunimmt:

$$\dot{V} = F$$

Das Modell ist (bis auf die Nicht-Konstanz von $\frac{F}{V}$) dem der kontinuierlichen Kultur sehr ähnlich:

$$\dot{X} = N \cdot X_1 \cdot r(X) + (X_{in} - X) \cdot \frac{F}{V}$$

5.3.3.2 Strömungsrohrreaktor

Das Stömungsrohr kann man nach einer Kompartimentierung in gleich große zylindrische Abschnitte als eine Kaskade von kontinuierlich betriebenen Rührkesseln mit gleicher Durchflußrate ansehen. Die Flüsse zwischen den Kompartimenten sind:

$$\vec{F}^{ij}_{in} = \vec{F}^{ij}_{out} = \begin{cases} \alpha \cdot X^i & \text{, falls } j = i+1 \\ \beta \cdot X^i & \text{, falls } j = i-1 \\ 0 & \text{, sonst.} \end{cases}$$

mit geeigneten Konstanten α und β, welche die Effekte der Konvektion und Diffusion (*Rückvermischung*) subsummieren. Kann die Diffusion gegenüber der Konvektion vernachlässigt werden, so ist $\beta = 0$.

5.3.4 Gasübergang

Ein Spezialfall bei der Modellierung von Bioreaktoren ist der Gasübergang in und aus dem Medium, da der Gasaustausch über eine von der Substratzufuhr unabhängige Be- und Entlüftung erfolgt.

5.3.4.1 Der kL-Wert

Im Prinzip entsteht durch die Belüftung im Reaktor ein Zwei-Phasen-System aus einer Gasphase und einer flüssigen Phase. Der Gasaustausch zwischen beiden Phasen erfolgt über die Grenzfläche a. Ein einfaches Modell beschreibt die Sauerstoffkonzentration im Medium (ohne zusätzlichen Verbraucher) durch die Gleichung [Del87]:

$$\dot{O} = \mathrm{k}_L \cdot a \cdot (O^* - O) \quad \text{mit} \quad \begin{array}{lll} O^* & - & \text{Sättigungskonzentration des Sauerstoffs im Medium} \\ a & - & \text{Grenzfläche} \\ \mathrm{k}_L & - & \text{Flüssig-Film-Koeffizient} \end{array}$$

Man beachte, daß das Modell sowohl den Gasübergang in das Medium beschreibt ($O < O^*$) als auch das Ausgasen ($O > O^*$).

5.3.4.2 Abhängigkeit von verschiedenen Faktoren

Auf den Gasübergang wirken Gasstrom, Rührerdrehzahl, Medienzusammensetzung und Reaktorgeometrie ein:

- Reaktorgeometrie, Volumen V und Gasstrom F_O durch das Medium wirkt sich einerseits auf die Blasengröße und -verteilung (a, k_L), andererseits aber auch auf die Druckverhältnisse (O^*) aus.
- Der Rührer mit Drehzahl r beeinflußt in erster Linie die Größe und Verteilung der Gasblasen, also a, aber auch die Aufenthaltsdauer (k_L).

- Eine Änderung der Medienzusammensetzung X verändert zunächst das Sättigungsverhalten (O^*) und den Faktor k_L, über die Viskosität aber auch mittelbar die Fläche a.

Die Abhängigkeiten

$$\begin{array}{rcll} k_L & = & k_L & (X, V, F_0, r) \\ a & = & a & (X, V, F_O, r) \\ O_* & = & O_* & (X, V, F_O) \end{array}$$

sind nur empirisch beschreibbar, da das Turbulenzverhalten eines bestimmten Reaktors nur mit großen Aufwand zu beschreiben wäre [BO86]. Da k_L und a empirisch nicht voneinander getrennt werden können, faßt man sie zum *Sauerstoff-Transport-Koeffizienten* (oder kurz k_La-Wert)

$$k_La\,(X, F_O, r) = k_L \cdot a$$

zusammen. In günstigen Fällen hat man im Arbeitsbereich des Modells nur die Abhängigkeiten [VS86] [BO86]:

$$k_La = k_La\,(r) \quad \text{und} \quad O^* = \text{const}$$

5.3.4.3 Quasistationaritätsannahme

Im Vergleich zum Wachstum der Organismen, hat der Gasübergang viel kleinere Zeitkonstanten. Es liegt daher nahe, den Gasübergang als quasi-stationär anzunehmen. Für das Sauerstoffmodell

$$\dot{O} = k_La \cdot (O^* - O) - \omega \cdot X$$

mit der spezifischen Sauerstoffaufnahmerate ω der Organismen, ergibt sich z.B. die quasi-stationäre Konzentration

$$O_\infty = O^* - \frac{\omega \cdot X}{k_La}.$$

5.4 Zustandsraummodelle

5.4.1 System- und Meßgleichung

Vielfach muß ein Systemmodell in einer besonderen Gestalt vorliegen, um in ein Datenanalyseverfahren Eingang zu finden. Das wichtigste Schema ist das eines *Zustandsraummodells*. Es besteht aus einer Systemgleichung und einer Meßgleichung (vgl. auch Abb. 2.4):

5.4.1.1 Systemgleichung

Die *Systemgleichungen* bilden ein Differentialgleichungssystem, welches die Dynamik des Systems festlegt:

$$\dot{x} = f\,(t, \alpha, x, u)$$

Hierbei ist:

$$\begin{array}{rcll} x & = & (x_1, \ldots, x_n) & \text{der Vektor der Zustandsvariablen} \\ \alpha & = & (\alpha_1, \ldots, \alpha_p) & \text{der Vektor der Systemparameter} \\ u & = & (u_1, \ldots, u_k) & \text{der Vektor der äußeren Einflußgrößen} \end{array}$$

5.4.1.2 Meßgleichung

Der Vorgang der Messung wird eigens modelliert durch die *Meßgleichungen*:

$$y = g(t, \beta, x)$$

Es ist: $y = (y_1, \ldots, y_m)$ der Vektor der Meßgrößen
$\beta = (\beta_1, \ldots, \beta_q)$ der Vektor der Eichparameter

Im Gegensatz zu den Systemparametern α ist der Vektor β der Eichparameter, welcher die physikalischen Eigenschaften der Meßgeräte beschreibt, meist recht gut bekannt.

Die Struktur des allgemeinen Zustandsraummodells

$$\begin{aligned} \dot{x} &= f(t, \alpha, x, u) \quad \text{(Systemgleichung)} \\ y &= g(t, \beta, x) \quad \text{(Meßgleichung)} \end{aligned}$$

ist bereits in Abbildung 2.4a illustriert worden.

5.4.2 Beispiel: Monod-Modell für die kontinuierliche Kultur

5.4.2.1 Systemgleichung

Ein kontinuierlicher Bioreaktor mit Monod'scher Wachstumskinetik ohne Erhaltungsstoffwechsel hat das Modell:

$$\begin{aligned} \dot{X} &= \mu(S) \cdot X - D \cdot X \\ \dot{S} &= \sigma(S) \cdot X - D \cdot (S - S_{in}) \end{aligned} \quad \text{mit} \quad \begin{aligned} \sigma(S) &= \sigma_S \tfrac{S}{S+K_S} \\ \mu(S) &= Y_{X/S} \cdot \sigma(S) \end{aligned}$$

Hierin ist

x	$= (X, S)$	der Zustandsvektor	mit Dimension 2
u	$= (D, S_{in})$	der Steuergrößenvektor	mit Dimension 2
α	$= (\sigma_S, K_S, Y_{X/S})$	der Systemparametervektor	mit Dimension 3

und

$$f\left(t, \begin{pmatrix} \sigma_S \\ K_S \\ Y_{S/X} \end{pmatrix}, \begin{pmatrix} X \\ S \end{pmatrix}, \begin{pmatrix} D \\ S_{in} \end{pmatrix}\right) = \begin{pmatrix} \mu(S) \cdot X & -D \cdot X \\ \sigma(S) \cdot X & -D \cdot (S - S_{in}) \end{pmatrix}.$$

5.4.2.2 Meßgleichung

Geht man von einer Messung der Biomasse über eine Trübungssonde aus, so findet man als Eichkurve oft eine Sättigungskurve vor [Wie87] (OD = Optische Dichte):

$$\mathrm{OD} = \mathrm{OD}_X \cdot \frac{X}{X + K_X}$$

In diesem Fall ist:

y	$= (\mathrm{OD})$	der Meßvektor	mit Dimension 1
β	$= (\mathrm{OD}_X, K_X)$	der Meßparametervektor	mit Dimension 2

und

$$g\left(t, \begin{pmatrix} OD_X \\ K_X \end{pmatrix}, \begin{pmatrix} X \\ S \end{pmatrix}, \begin{pmatrix} D \\ S_{in} \end{pmatrix}\right) = \left(\mathrm{OD}_X \cdot \tfrac{X}{X+K_X} \right).$$

Kapitel 6

Statistische Modelle

Die Beschreibung der statistischen Modelle beschränkt sich auf solche, die bisher im Rahmen der biotechnischen Datenanalyse Anwendung gefunden haben. Das gesamte Methodenspektrum der Statistik wäre zu umfangreich, um in Kürze geschlossen abgehandelt werden zu können. Eine vereinheitlichende Sichtweise wie bei den deterministischen Modellen scheint bei den statistischen Modellen nicht erforderlich zu sein, da die Implementierung statistischer Verfahren in ein Datenanalysesystem in doppelter Hinsicht unkomplizierter ist:

- die eingehenden Daten sind (nach der in 4.1.1.3 getroffenen Konvention) nicht zeitabhängig und
- die zugrunde liegenden Modelle haben eine feste Struktur bzgl. der Parameter.

Aus diesem Grunde entfällt eine aufwendige Behandlung des Kontrolldatenflusses (Zeitablaufsteuerung, iterative Aufrufe) wie bei den (dynamischen) deterministischen und stochastischen Modellen weitestgehend; ein statistisches Verfahren kann als einfaches Eingangs/Ausgangs-Schema mit numerischen Eingangsdaten und numerischen sowie kategorialen Ausgangsdaten angesehen werden. Wie solche Verfahren zu implementieren sind, ist bei Standard-Statistik-Paketen wie SAS, SPSS und BMDP [KW85] vorgeführt worden. Es ist zu erwarten, daß neue statistische Verfahren bei Bedarf in ein biotechnisches Datenanalysesystem aufgenommen werden können, ohne das Grundkonzept zu sprengen.

Besondere Berücksichtigung findet die realistische Modellierung der Rauschanteile biotechnischer Messungen (siehe 6.3). Obwohl bekannt ist, daß die Störgrößen bei biotechnischen Prozessen in der Regel nicht normalverteilt sind, blieb diese Tatsache bisher in der Simulations-Literatur unberücksichtigt! Insbesondere läßt sich die Robustheit der eingesetzen Verfahren nur mit einer realistischen Beschreibung des Rauschens testen.

Ausgehend vom allgemeinen Stoffwechselmodell aus Abschnitt 5.2.6.1 können die Bilanzgleichungen des mikrobiellen Wachstums als lineare Modelle der Statistik interpretiert werden. In dieser Sichtweise erscheint die Bilanzierung entkoppelt von der Stoffwechselmodellierung (siehe 6.6). Die Konsequenzen für die Datenanalyse bestehen in einer klaren Trennung der zugehörigen Analysemethoden, die in der Literatur bisher nicht in dieser Deutlichkeit hervorgetretren ist.

Der Umgang mit Zufallsvariablen fällt dem Praktiker aufgrund der relativ abstrakten Formalismen der Wahrscheinlichkeitstheorie oft schwer. Hier kann ein extensiver Einsatz von Visualisierungstechniken hilfreich sein. Auf solche Techniken wird in den Abschnitten 6.1 und 6.2 eingegangen.

6.1 Grundbegriffe

Vorausgesetzt werden die wahrscheinlichkeitstheoretischen Grundbegriffe

- Zufallsvariable

X

und Zufallsvektor

$$X = (X_1, \ldots, X_n),$$

- (n-dimensionale) Verteilungsfunktion

$$F_X(\xi), \quad \xi = (\xi_1, \ldots, \xi_n)^T \epsilon \, \Re^n$$

und (n-dimensionale) Wahrscheinlichkeitsdichte

$$f_X(\xi), \quad \xi = (\xi_1, \ldots, \xi_n)^T \epsilon \, \Re^n$$

einer Zufallsvariablen,

- insbesondere die Gleichverteilungsdichte

$$f_{[a,b]}(\xi) = \frac{1}{\mathrm{vol}\,([a,b])} \cdot \begin{cases} 1 & \text{, falls} \quad \xi \epsilon [a,b] \\ 0 & \text{, sonst} \end{cases}$$

mit einem n-dimensionalen Intervall $[a,b] \subset \Re^n$ des Volumens $\mathrm{vol}\,([a,b])$

und die Normalverteilungsdichte

$$f_{\mu,\Sigma}(x) = \frac{1}{\sqrt{(2\pi)^n \cdot \det \Sigma}} \cdot e^{-\frac{1}{2}(x-\mu)^T \Sigma^{-1} (x-\mu)}$$

mit $x, \mu \quad \epsilon \, \Re^n$ Vektoren
$\Sigma \quad \epsilon \, \Re^{n \times n}$, eine symmetrische, positiv definite Matrix $(\Sigma = \Sigma^T, \Sigma \geq 0)$.

Speziell in Dimension 1:

$$f_{[a,b]}(\xi) = \frac{1}{b-a} \cdot \begin{cases} 1 & \text{, falls} \quad \xi \epsilon [a,b] \\ 0 & \text{, sonst} \end{cases}$$

und

$$f_{\mu,\sigma}(\xi) = \frac{1}{\sqrt{2\pi \cdot \sigma^2}} \cdot e^{-\frac{1}{2}\frac{(\xi-\mu)^2}{\sigma^2}}.$$

Ferner die 1-dimensionale Exponentialverteilungsdichte:

$$f_\lambda(\xi) = \frac{1}{\lambda} e^{-\lambda t}$$

- Erwartungswert

$$\mathrm{E}[X],$$

m-te Momente

$$\mathrm{E}[X^m]$$

und Kovarianzmatrix

$$\mathrm{Cov}[X] = \begin{pmatrix} \mathrm{Var}[X_1] & \mathrm{Cov}[X_1, X_2] & \cdots & \mathrm{Cov}[X_1, X_n] \\ \mathrm{Cov}[X_2, X_1] & \mathrm{Var}[X_2] & \cdots & \mathrm{Cov}[X_2, X_n] \\ \vdots & \ddots & \vdots & \vdots \\ \mathrm{Cov}[X_n, X_1] & \mathrm{Cov}[X_n, X_2] & \cdots & \mathrm{Var}[X_n] \end{pmatrix}$$

eines Zufallsvektors,

- insbesondere die Rechenregeln

$$\begin{aligned} \mathrm{E}[A \cdot X + b] &= A \cdot \mathrm{E}[X] + b && (A, b \text{ nicht von X abhängig}) \\ \mathrm{Cov}[A \cdot X] &= A \cdot \mathrm{Cov}[X] \cdot A^T && (A \text{ nicht von X abhängig}) \\ \mathrm{Cov}[X] &= \mathrm{E}[X\,X^T] - \mathrm{E}[X]\,\mathrm{E}[X]^T && \text{(Steinerscher Satz)} \\ \mathrm{Cov}[X] &= \mathrm{Cov}[X]^T \\ \mathrm{Cov}[X] &\geq 0, \end{aligned}$$

die in einem Lehrbuch nachgelesen werden können (z.B. [Bau78] [Vog70]).

6.1.1 Veranschaulichung von Zufallsvariablen

Besonderer Wert wird auf die Möglichkeiten zur Veranschaulichung zufälliger Ereignisse gelegt, da die Visualisierung im Rahmen des Datenanalyse-Konzepts eine zentrale Rolle spielt: Eine Zufallsvariable X der Dimension ≤ 3 kann man sich durch eine sogenannte *unabhängige Stichprobe* veranschaulichen. Darunter versteht man einen Satz von Werten $x_1, x_2, \ldots$, der bei mehrmaliger unabhängiger Durchführung des durch X beschrieben Experimentes entsteht. Solche Stichproben kann man mit Hilfe entsprechender Zufallszahlen-Generatoren erzeugen und als Punktwolke im Raum darstellen. Die Abbildungen 6.1b und 6.2b visualisieren Gleichverteilungen und Normalverteilungen in Dimension 1 und 2 durch Punktwolken.

6.1.2 Veranschaulichung von Verteilungen

Verteilungsfunktionen sind darstellbar durch ihre Dichtefunktion (sofern eine solche existiert) [1]. Zwei Beispiele (in Dimension 1 und 2) finden sich in den Abbildungen 6.1a und 6.2a:

a) Die Gleichverteilungsdichte ist gekennzeichnet durch ein Plateau der Höhe

$$\frac{1}{\mathrm{vol}\,([a,b])}$$ über dem *Träger* $[a, b]$ der Verteilung und 0 außerhalb.

b) Die Normalverteilungsdichte ist die bekannte Gauß-Glocke. Man findet die Niveaulinien:

$$\begin{aligned} f(\xi) &= \frac{1}{\sqrt{(2\pi)^n \cdot \det \Sigma}} \cdot e^{-\frac{1}{2}(\xi-\mu)^T \Sigma^{-1} (\xi-\mu)} = \text{const} \\ \Leftrightarrow \quad (\xi-\mu)^T \Sigma^{-1} (\xi-\mu) &= -2 \ln \left(\text{const} \cdot \sqrt{(2\pi)^n \cdot \det \Sigma}\right) \end{aligned}$$

Da Σ symmetrisch und positiv definit ist, handelt es sich um eine Ellipsengleichung. Der Mittelpunkt der Ellipse ist μ.

[1] In höheren Dimensionen kann das mit Hilfe einer 3D-Darstellung oder eines Niveauliniendiagramms erfolgen.

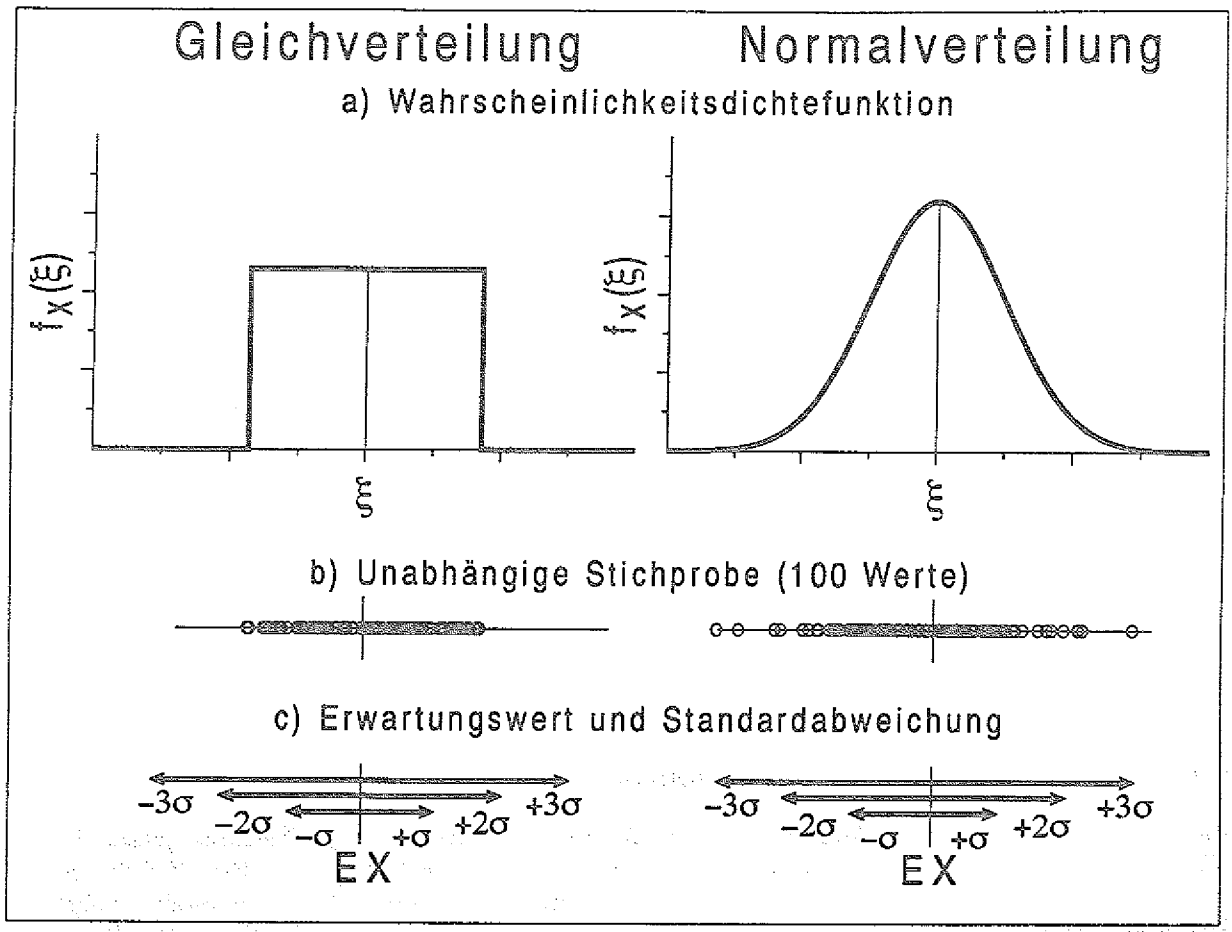


Abbildung 6.1: Visualisierung 1-dimensionaler Wahrscheinlichkeits-Verteilungen

6.1.3 Veranschaulichung von Kovarianzmatrizen

Bekanntlich gilt für eine (μ, Σ)-normalverteilte Zufallsvariable X gerade

$$\mathrm{E}[X] = \mu \quad \text{und} \quad \mathrm{Cov}[X] = \Sigma.$$

Die durch die Gleichungen

$$(\xi - \mu)^T \, \Sigma^{-1} \, (\xi - \mu) = \text{const}$$

beschriebenen Ellipsen enthalten alle Informationen über Form und Lage der zugehörigen Punktwolke. Insbesondere ist μ der Mittelpunkt der Ellipsen und die Matrix Σ bildet ein Streumaß. In der Regel zeichnet man die durch

$$(\xi - \mu)^T \, \Sigma^{-1} \, (\xi - \mu) = m^2, \quad m \in \{1, 2, 3, \ldots\}$$

beschriebenen *$m\Sigma$-Ellipsen* [2]. Im 1-dimensionalen Fall erhält man als Lösung die "Fehlerbalken"

$$\mathrm{E}[X] \pm m \cdot \sigma$$

(σ ist die Standardabweichung). Die Wahrscheinlichkeit

$$p_{n,m} = P((X - \mu)^T \, \Sigma^{-1} \, (X - \mu) \leq m^{-1})$$

[2] Im Rahmen der Test-Theorie auch als *Konfidenz-Ellipsen* bezeichnet [Sch71].

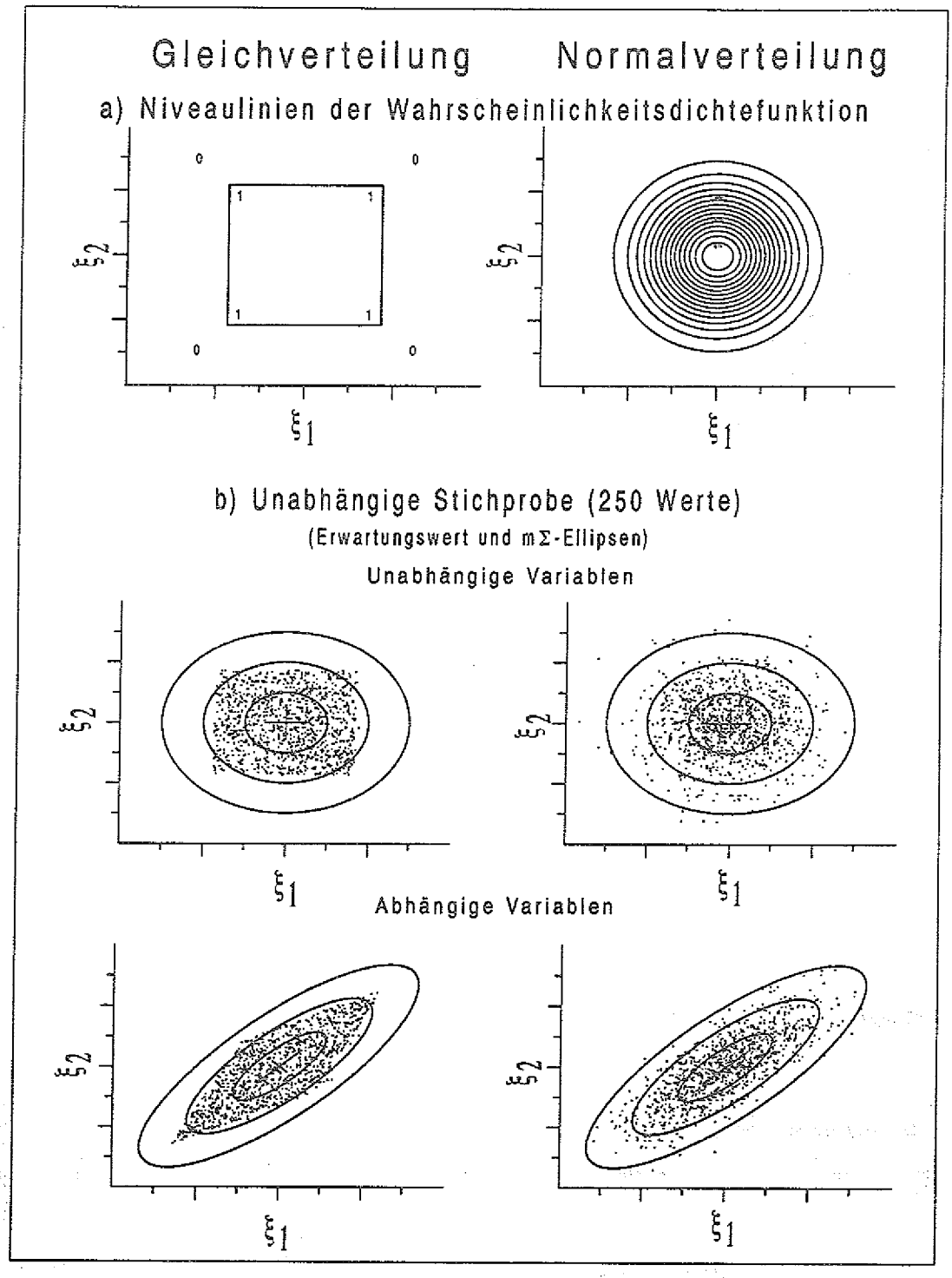


Abbildung 6.2: Visualisierung 2-dimensionaler Wahrscheinlichkeits-Verteilungen

dafür, daß X in der $m\Sigma$-Ellipse liegt, hängt von der Dimension n des Raums ab. Einige Werte (berechnet auf zwei Stellen mittels Monte-Carlo-Integration [PFTV88]) sind:

$p_{n,m}$		m		
		1	2	3
	1	0.68	0.95	0.99
n	2	0.39	0.86	0.99
	3	0.19	0.73	0.97

Man vergleiche die Zahlenwerte mit dem empirischen Befund aus den Punktwolkendarstellungen in Abbildung 6.1b und 6.2b.
Bei nicht normalverteilten Größen ist keine allgemeine Beziehung zwischen der Gestalt der $m\Sigma$-Ellipsen

$$(\xi - \mathrm{E}[X])^T \ \mathrm{Cov}[X]^{-1} \ (\xi - \mathrm{E}[X]) = m^2$$

und der Lage der Verteilung angebbar ([FH84] gibt einige extreme Beispiele). Sie stellen aber im allgemeinen weiterhin ein grobes Maß für die Streuung dar. Die Abbildungen 6.1c und 6.2b zeigen einige $m\Sigma$-Ellipsen für Normal- und Gleich-Verteilungen.

6.1.3.1 Berechnung der Hauptachsen

Die Hauptachsen der $m\Sigma$-Ellipsen können wie folgt aus $\mathrm{Cov}[X]$ berechnet werden [FH84]:

1. Man berechne auf die Länge 1 normierte Eigenvektoren

   $$v_1, \ldots, v_n$$

   zu den Eigenwerten

   $$\sigma_1^2, \ldots, \sigma_n^2$$

   von $\mathrm{Cov}[X]$. Sie stehen paarweise senkrecht aufeinander [3].

2. Die Matrix

   $$V = (v_1, \ldots, v_n)$$

   hat die Eigenschaften:

   $$V^T V = 1 \ \Rightarrow \ V^{-1} = V^T$$

   und

   $$V^T \cdot Cov[X] \cdot V = \begin{pmatrix} \sigma_1^2 & \ldots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \ldots & \sigma_n^2 \end{pmatrix} \stackrel{def}{=} D \ .$$

3. Die Hauptachsen der $m\Sigma$-Ellipse sind dann:

   $$m \cdot \sigma_1 \cdot v_1, \ \ldots, \ m \cdot \sigma_n \cdot v_n$$

6.1.3.2 Berechnung der Wurzel

Man berechnet auf diesem Wege auch die *Wurzel* $\sqrt{\Sigma}$ einer symmetrischen, positiv definiten Matrix $\Sigma = \mathrm{Cov}[X]$

$$\sqrt{\Sigma} = V \cdot \sqrt{D} \ \text{ mit } \ \sqrt{D} = \begin{pmatrix} \sigma_1 & \ldots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \ldots & \sigma_n \end{pmatrix},$$

denn offenbar ist für diese Matrix

[3] Beweise für die Aussagen finden sich z.B. in [Fis75], [Bri83].

$$\sqrt{\Sigma}\sqrt{\Sigma}^T = V\sqrt{D}\sqrt{D}^T V^T = \Sigma.$$

Ein spezieller Fall liegt vor, wenn $\mathrm{Cov}[X]$ bereits diagonal ist. Die Rauschkomponenten X_i und X_j $(i \neq j)$ in je zwei Koordinatenrichtungen sind dann unkorreliert ($\mathrm{Cov}[X_i, X_j] = 0$). In diesem Fall verlaufen die Hauptachsen parallel zu den Koordinatenachsen und in der Diagonalen von $\mathrm{Cov}[X]$ findet man man die Varianzen $\mathrm{Var}[X_i]$, d.h. ein Maß für die Streuung in Richtung X_i. Eine Veranschaulichung der speziellen Lage bietet die Abbildung 6.2b.

6.2 Verteilungen

Ein Großteil der statistischen Verfahren geht von der Annahme einer Normal-Verteilung oder einer daraus abgeleiteten Verteilung (z.B. einer χ^2- oder F-Verteilung) aus. Es muß aber betont werden, daß die Verteilungen praktisch vorkommender Variablen erheblich von der Normalverteilung abweichen können und

- Unsymmetrien,
- Ausreißer oder
- Delta-Distributionen

enthalten können. Solche Verteilungen sind theoretisch schwer zu behandeln. Die Brauchbarkeit statistischer Verfahren, die ursprünglich auf der Annahme einer Normalverteilung beruhen, kann für die praktische Anwendung oft nur auf empirischem Wege geprüft werden (Monte-Carlo-Simulation). Es soll kurz zusammengestellt werden, wie man aus Zufallsvariablen mit gegebener Verteilung neue Zufallsvariablen gewinnen kann, die eine oder mehrere der oben genannten Eigenschaften aufweisen.

6.2.1 Transformation

Eine gegebene Zufallsvariable X mit Dichte $f_X(\xi)$ kann durch eine Abbildung T in eine Zufallsvariable

$$Y = T(X)$$

transformiert werden. Es soll gefordert werden, daß T stetig differenzierbar und invertierbar ist, so daß

$$T', T^{-1} \text{ und } (T')^{-1}$$

existieren. Die Transformationsformel der Wahrscheinlichkeitstheorie besagt dann:

$$f_{T(X)}(T(\xi)) = f_X(\xi) \cdot \det(T'(\xi))^{-1} = f_X(\xi) \cdot \det(T^{-1})'(T(\xi))$$

6.2.1.1 Lineare Transformation

Für die lineare Transformation

$$Y = T(X) = A \cdot X + b$$

gilt insbesondere

$$f_Y(A \cdot \xi + b) = f_X(\xi) \cdot \det A^{-1},$$

woraus mit $\xi = A^{-1}(\zeta - b)$ folgt:

$$f_Y(\zeta) = f_X(A^{-1}(\zeta - b)) \cdot \det A^{-1}$$

Für diese Transformation kann man auch Varianz und Erwartungswert von $T(X)$ einfach ausrechnen (vgl. 6.1):

$$\mathrm{E}[Y] = A \cdot \mathrm{E}[X] + b \quad \text{und} \quad \mathrm{Cov}[Y] = A \cdot \mathrm{Cov}[X] \cdot A^T$$

Speziell durch die Wahl von

$$A = \sqrt{\Sigma \cdot \mathrm{Cov}[X]^{-1}} \quad \text{und} \quad b = \mu - A \cdot \mathrm{E}[X]$$

ergibt sich

$$\mathrm{E}[Y] = \mu \quad \text{und} \quad \mathrm{Cov}[Y] = \Sigma.$$

D.h. auf diesem Wege können Verteilungen mit vorgegebenen Mittelwerten und Kovarianzmatrizen hergestellt werden (Abb. 6.3a und 6.3b zeigt einige Beispiele). Das theoretische Studium von Verteilungen kann sich auf Verteilungen mit Erwartungswert 0 und Kovarianzmatrix 1 beschränken.

6.2.1.2 Polarkoordinatentransformation

Eine weitere wichtige Transformation ist eine verallgemeinerte Polarkoordinatentransformation in 2 Dimensionen:

$$\begin{pmatrix} x \\ y \end{pmatrix} = T \begin{pmatrix} r \\ \phi \end{pmatrix} = \begin{pmatrix} t(r) \cdot \cos 2\pi\phi \\ t(r) \cdot \sin 2\pi\phi \end{pmatrix} \quad \text{mit } r \in [0,1],\ \phi \in [0,1],\ t([0,1]) = [0,\infty]$$

Man findet:

$$\left[\det(T') \begin{pmatrix} r \\ \phi \end{pmatrix}\right]^{-1} = \left[\det \begin{pmatrix} t'(r) \cdot \cos 2\pi\phi & -2\pi\, t(r) \cdot \sin 2\pi\phi \\ t'(r) \cdot \sin 2\pi\phi & 2\pi\, t(r) \cdot \cos 2\pi\phi \end{pmatrix}\right]^{-1} = [-2\pi \cdot t(r) \cdot t'(r)]^{-1}$$

6.2.2 Faltung

Die Summe zweier unabhängiger Zufallsvariablen X und Y hat die Dichte

$$f_{X+Y}(\xi) = f_X * f_Y(\xi) \stackrel{def}{=} \int_{-\infty}^{\infty} f_X(\eta) \cdot f_Y(\xi - \eta)\, d\eta \quad (\textit{Faltungsprodukt}).$$

Erwartungswert und Varianz berechnen sich in dieser Situation aus:

$$\mathrm{E}[X+Y] = \mathrm{E}[X] + \mathrm{E}[Y] \quad \text{und} \quad \mathrm{Var}[X+Y] = \mathrm{Var}[X] + \mathrm{Var}[Y]$$

6.2.2.1 Unsymmetrische Verteilungen

Betrachtet man die üblicherweise in der Wahrscheinlichkeitstheorie herangezogenen Verteilungen (Normalverteilung, Exponentialverteilung, Γ-Verteilung etc.), so stellt man fest, daß sich darunter keine Verteilung befindet, die wie die Normalverteilung auf ganz $\Re$ definiert und positiv, jedoch unsymmetrisch ist. Dagegen gibt es eine Reihe von Verteilungen, die nur auf der positiven Halbachse $\Re^+$ definiert sind. Durch Faltung einer solchen Verteilung mit einer Normalverteilung erhält man unsymmetrische und überall definierte Verteilungen. Abbildung 6.3c zeigt das Ergebnis der Faltung einer Normalverteilung mit einer Exponentialverteilung:

$$f_{X+Y}(\xi) = \frac{1}{2\pi} \int_{-\infty}^{\xi} e^{-\frac{1}{2}\eta^2} e^{-(\xi-\eta)} d\eta$$

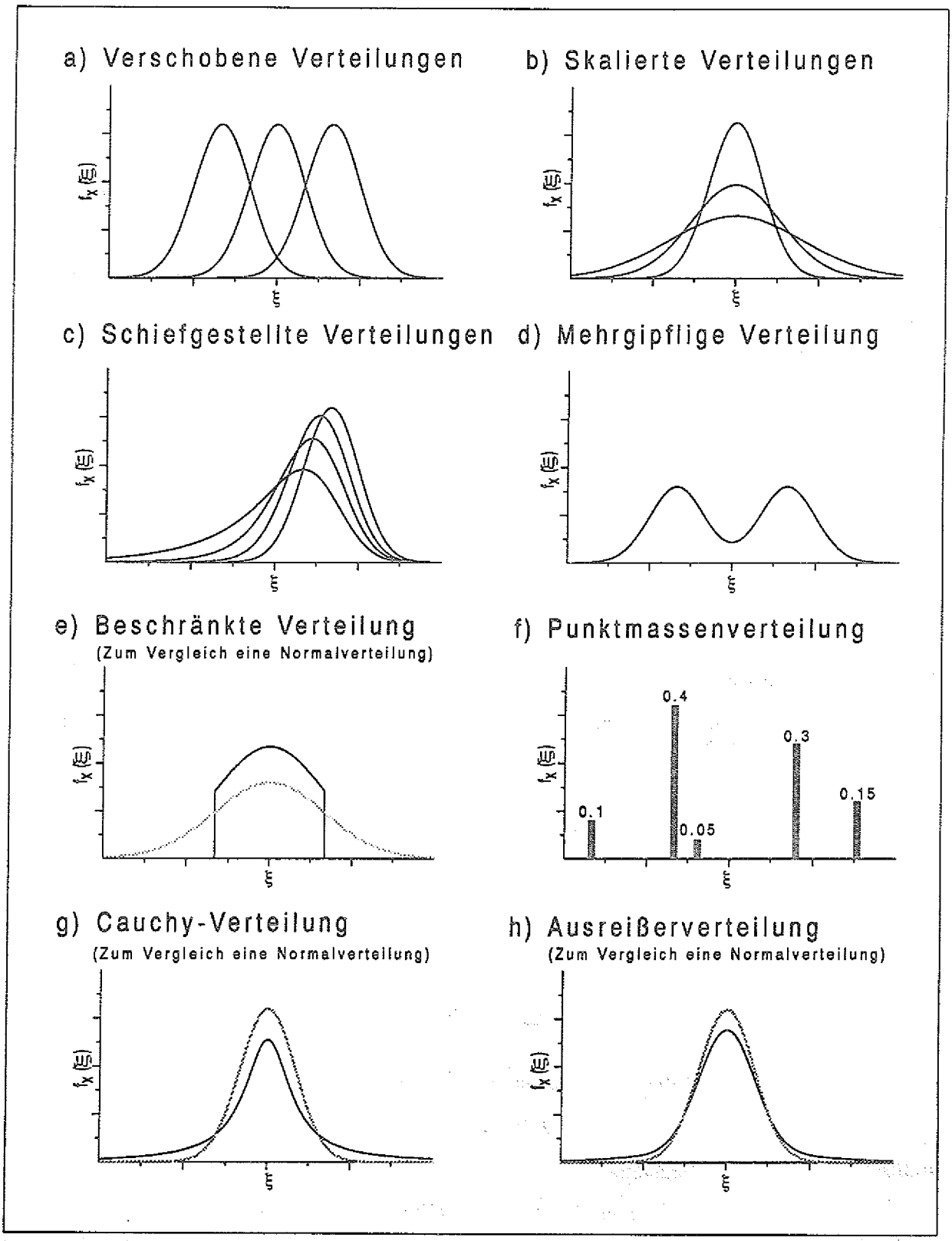


Abbildung 6.3: Einige Beispiele für Verteilungen

6.2.3 Verteilungen mit unendlicher Varianz und beschränkte Verteilungen

6.2.3.1 Verteilungen mit unendlicher Varianz

Eine Wahrscheinlichkeitsdichte mit unendlicher Varianz ist die Cauchy-Verteilung:

$$f(\xi) = \frac{1}{\pi} \cdot \frac{1}{1+\xi^2}$$

(vgl Abb. 6.3g). Ansonsten hat sie viele Eigenschaften (Differenzierbarkeit, Symmetrie, nirgends verschwindend, eingipflig) mit der Normalverteilung gemeinsam. Die Varianzeigenschaft ist als Gegenbeispiel für viele theoretische Resultate nützlich: Die meisten Grenzwertsätze gelten nur für endliche Varianz und werden für die Cauchy-Verteilung konkret falsch [Bau78].

Für die Statistik bedeutet die unendliche Varianz, daß "weiter außen" liegende Werte im Vergleich zur Normalverteilung wesentlich häufiger auftreten ("long tailed distribution"), weshalb man sie auch als *Ausreißer* bezeichnet. Die statistische Behandlung von Verteilungen mit Ausreißern erfordert *robuste* Verfahren, die einen Ausreißer im Vergleich zu anderen Werten geringer wichten. Ausreißerverteilungen sind daher zum Test der Robustheit der eingesetzten Verfahren sehr nützlich.

6.2.3.2 Beschränkte Verteilungen

Das Gegenstück ist eine beschränkte Verteilung: Reale Meßdaten sind aufgrund des Meßbereichs der Instrumente in der Regel bereichs-beschränkt. Man kann jede beliebige Verteilung $f(\xi)$ auf das Intervall $[a,b]$ einschränken, indem man zu

$$f_{[a,b]}(\xi) = \begin{cases} \frac{1}{\int_a^b f(\tau)d\tau} \cdot f(\xi) & \text{falls } \xi \epsilon [a,b] \\ 0 & \text{sonst} \end{cases}$$

übergeht. Abbildung 6.3e zeigt ein Beispiel.

6.2.4 Mischverteilungen

Im Gegensatz zu den stetigen Verteilungen stehen die Punktmassenverteilungen, die einzelnen Punkten $x \epsilon \Re^n$ eine positive Wahrscheinlichkeit zuordnen. Die (im strengen Sinne nicht existierende) Dichte einer solchen Verteilung wird durch eine Dirac-Delta-Funktion

$$\delta_x(\xi) = \begin{cases} \infty & \text{für } \xi = x \\ 0 & \text{für } \xi \neq x \end{cases} \quad \text{mit} \quad \int_{-\infty}^{+\infty} \delta_x(\xi)\, d\xi = 1$$

formalisiert. Will man den Punkten

$$x_1, \ldots, x_m$$

die Wahrscheinlichkeiten

$$\lambda_1, \ldots, \lambda_m \quad \text{mit} \quad \sum_{i=1}^{m} \lambda_i = 1$$

zuordnen, so erreicht man dies durch eine Summe (vgl. auch Abb. 6.3f):

$$f_X(\xi) = \lambda_1 \cdot \delta_{x_1}(\xi) + \ldots + \lambda_m \cdot \delta_{x_m}(\xi).$$

Das *Mischen* mehrerer Zufallsvariablen

$$X_1, \ldots, X_m \quad \text{mit den Wahrscheinlichkeiten} \quad \lambda_1, \ldots, \lambda_m$$

geschieht nun wie folgt: Zunächst wird ein Index i aus $\{1, \ldots, m\}$ mit Wahrscheinlichkeit λ_i zufällig ausgewählt. Der Wert der Mischvariablen X ist dann gleich dem Wert der i-ten Variablen X_i. Dieser Vorgang kann mit Hilfe einer *Verteilervariablen*

$$V \text{ mit } f_V(\xi) = \lambda_1 \cdot \delta_1(\xi) + \ldots + \lambda_m \cdot \delta_m(\xi)$$

formalisiert werden als

$$X = X_V.$$

Die Dichte von X ist

$$f_X(\xi) = \lambda_1 \cdot f_{X_1}(\xi) + \ldots + \lambda_m \cdot f_{X_m}(\xi).$$

Erwartungswert und Varianz berechnet man mit Hilfe des Steinerschen Satzes (siehe 6.1) als

$$\mathrm{E}[X] = \sum_i \lambda_i \, \mathrm{E}[X_i] \quad \text{und} \quad \mathrm{Var}[X] = \sum_i \lambda_i \, \mathrm{E}[X_i^2] \; - \mathrm{E}[X]^2 \; .$$

6.2.4.1 Ausreißerverteilungen

Mit Hilfe einer Mischverteilung aus zwei Verteilungen

$$f_X(\xi) \text{ und } f_Y(\xi) \quad \text{mit} \quad \mathrm{Var}[X] \ll \mathrm{Var}[Y]$$

kann eine *Ausreißerverteilung*

$$f_{XY}(\xi) = (1 - \lambda) \cdot f_X(\xi) + \lambda \cdot f_Y(\xi)$$

konstruiert werden. Hier gibt λ die Ausreißerwahrscheinlichkeit an, und $(1 - \lambda)$ die Wahrscheinlichkeit, einen regulären Wert zu erhalten. Häufig werden für X und Y Normalverteilungen mit gleichem Erwartungswert angenommen. Ein solches Beispiel zeigt Abbildung 6.3h. Eine interessante Konsequenz dieses Ausreißermodells ist das folgende Trennproblem (vgl. auch 2.4.3):

- Ein Ausreißer kann durchaus auch in der Nähe des gemeinsamen Erwartungswertes liegen. Er ist dann nicht von einem regulären Wert unterscheidbar.
- Umgekehrt kann ein regulärer Wert auch "so weit außen" liegen, daß er für einen Ausreißer gehalten werden muß.

6.2.4.2 Mehrgipflige Verteilungen

Mehrgipflige Verteilungen erhält man durch Mischung von Verteilungen mit verschiedenen Erwartungswerten. Es können z.B. Normalverteilungen herangezogen werden. Die Abbildung 6.3d zeigt ein Beispiel. Das entstehende Verteilungsmodell liegt der statistischen Klassifikationstheorie zugrunde (vgl. 6.4).

6.3 Ein Fallbeispiel: Messung der optischen Dichte

Die On-Line-Bestimmung der Biomassekonzentration im Fermenter erfolgt in der Regel mit Hilfe einer optischen Sonde, welche die Trübung der Fermentationsbrühe mißt. Dabei stehen sich, durch einen Spalt getrennt, eine Lichtquelle und eine Photozelle gegenüber (vgl. Abb. 6.4). Diese Messung unterliegt einer zeitweilig sehr starken Störung durch die im Medium vorhandenen Gasblasen. Eine Gasblase, die in den Strahlengang gelangt, kann bei Durchgang der Lichtstrahlen eine sprunghaft verringerte optische Dichte vortäuschen, aber auch eine vergrößerte Dichte durch Reflexion an der Oberfläche. Welcher dieser beiden Effekte auftritt, hängt sehr von der Medienzusammensetzung [4] und damit der Beschaffenheit der Gas-Flüssigkeits-Grenzschicht sowie der Reaktordurchmischung ab. Vielfach treten beide Effekte (Strahlendurchgang und Reflexion) gleichzeitig auf, oft aber nur genau einer der beiden. Die Abbildung 6.5 zeigt einen Beispieldatensatz aus einer Fermentation, bei dem nur Abweichungen nach oben auftreten. Im folgenden Modellierungsbeispiel soll dagegen von Abweichungen nach unten (Strahlendurchgang) ausgegangen werden. Die in Abbildung 6.6 und 6.7 gezeigte Sequenz entspricht der im folgenden schrittweise vollzogenen Modellierung.

[4] Vor allem das verwendete Anti-Schaum-Mittel hat einen starken Einfluß.

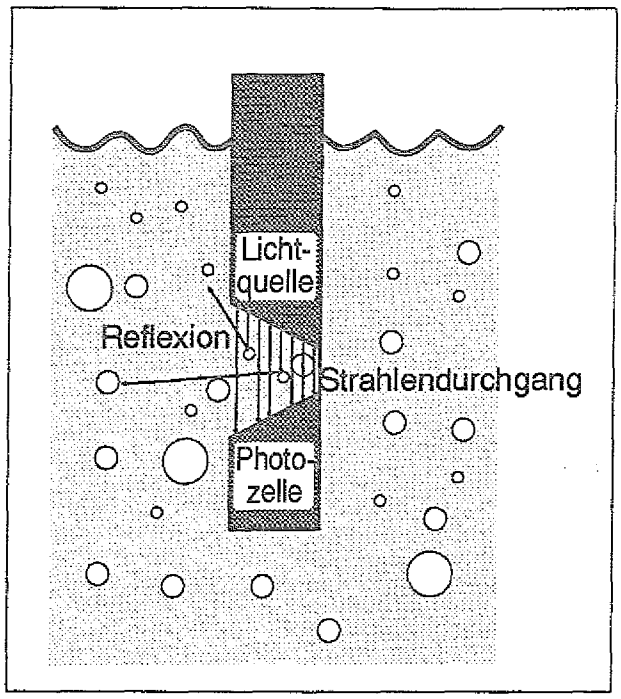


Abbildung 6.4: Aufbau einer Trübungssonde

6.3.1 Unverrauschtes Signal

Um eine Kurve zu erzeugen, die alle Meßbereiche durchläuft und an Anfang und Ende zwei konstante Abschnitte hat, wurde ein logistisches Wachstumsmodell der Gestalt [Roe83]

$$\dot{X} = \mu(X) \cdot X \quad \text{mit} \quad \mu(X) = \mu_{max} \, (1 - \frac{X}{X_{max}})$$

(μ_{max} die maximale Wachstumsrate, X_{max} die Maximalkapazität des Systems) zugrunde gelegt und um eine konstante Lag-Phase zu Beginn ergänzt. Das Signal überstreicht einen Wertebereich von ca. 3 Zehnerpotenzen (vgl. Abb. 6.6a).

6.3.2 Normalverteilter absoluter Meßfehler

In der biotechnischen Literatur wird fast ausschließlich mit normalverteiltem Rauschen gearbeitet (wenn nicht sogar ein gleichverteiltes Rauschen benutzt wird, siehe z.B. [DB86]), d.h. das Modell der Messung ist:

$$Y = X + \sigma \cdot v, \quad v \text{ normalverteilt mit } \mathrm{E}[v] = 0 \text{ und } \mathrm{Var}[v] = 1$$

Bei Verwendung eines Rauschens konstanter Varianz (hier z.B. 10% des Maximalwerts von X) ergibt sich ein vollkommen unrealistisches Bild im Bereich kleiner Meßwerte, wo praktisch ein reiner Rauschprozeß mit Erwartungswert ~ 0 vorliegt (Abb. 6.6b) .

6.3.3 Normalverteilter relativer Meßfehler

Setzt man den Rauschterm mit einer Standardabweichung relativ zum Signal an, d.h.

$$Y = X + \sigma \cdot X \cdot v$$

(σ ist jetzt eine prozentuale Größe), so entsteht das Bild 6.6c. Auch diese Vorstellung ist falsch, da im Bereich kleiner Meßgrößen folgt:

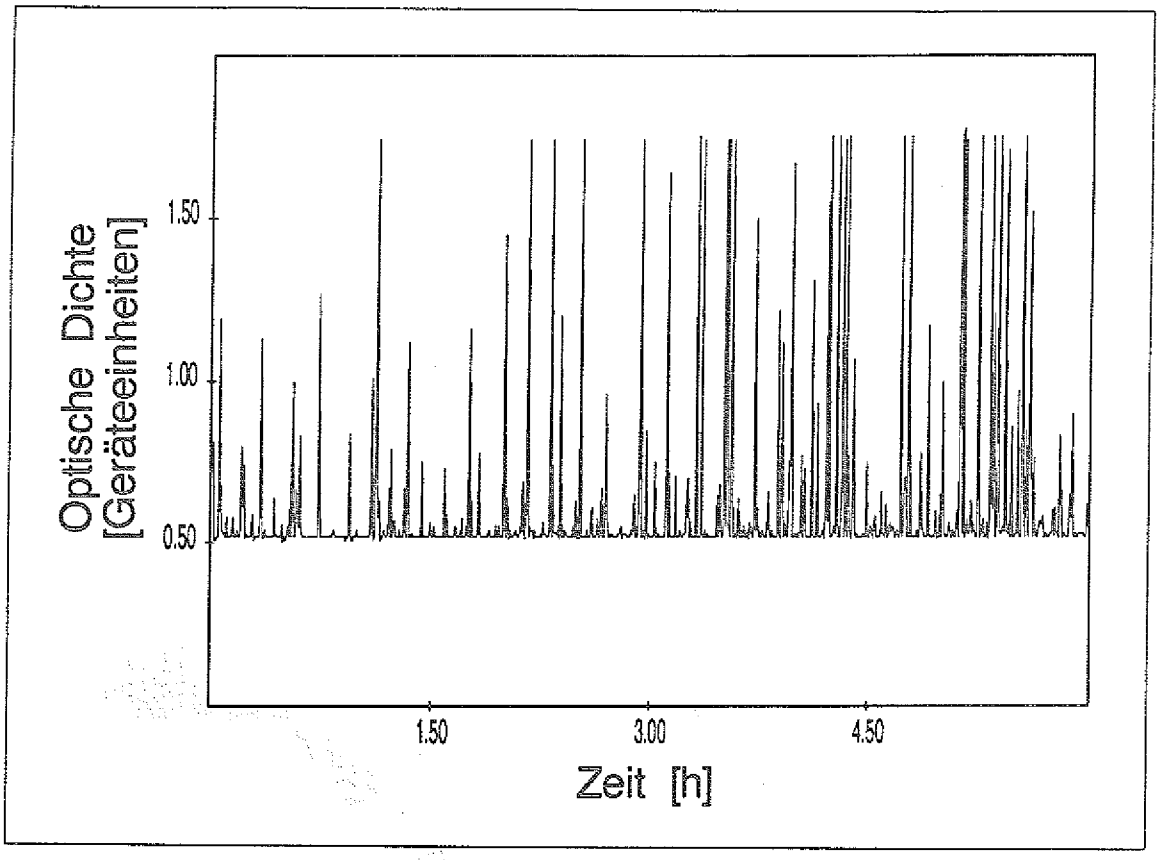


Abbildung 6.5: Typisches Signal einer Trübungssonde

$$X \approx 0 \implies \mathrm{Var}[\sigma \cdot X \cdot v] = \sigma \cdot X \approx 0$$

Diese Gleichung widerspricht der Tatsache, daß Meßgeräte bei verschwindendem Signal ein Restrauschen aufweisen, welches etwa in der Größenordnung der Meßgenauigkeit liegt. Gerade bei einer Trübungsmessung, die Meßwerte mit einer Genauigkeit von höchstens 3 Dezimalstellen liefert, ist das Hintergrundrauschen nicht zu vernachlässigen.

6.3.4 Bereichsabhängiger Meßfehler

Anstelle des konstanten oder linearen Ansatzes wäre eine nichtlinear meßbereichsabhängige Beschreibung der Störung angebracht:

$$Y = X + f(X) \cdot v$$

Eine einfache Funktionenklasse, aus der f ausgewählt werden kann, ist z.B. gegeben durch:

$$f(X) = f'_\infty \cdot X + \frac{f_0^2}{(f'_\infty - f'_0) \cdot X + f_0}$$

Der Funktionsterm ist so konstruiert, daß

$$\begin{aligned} f(0) &= f_0\,, \\ f'(0) &= f'_0 \\ \text{und}\quad f(X) &\sim f'_\infty \cdot X \text{ für } X \to \infty\,. \end{aligned}$$

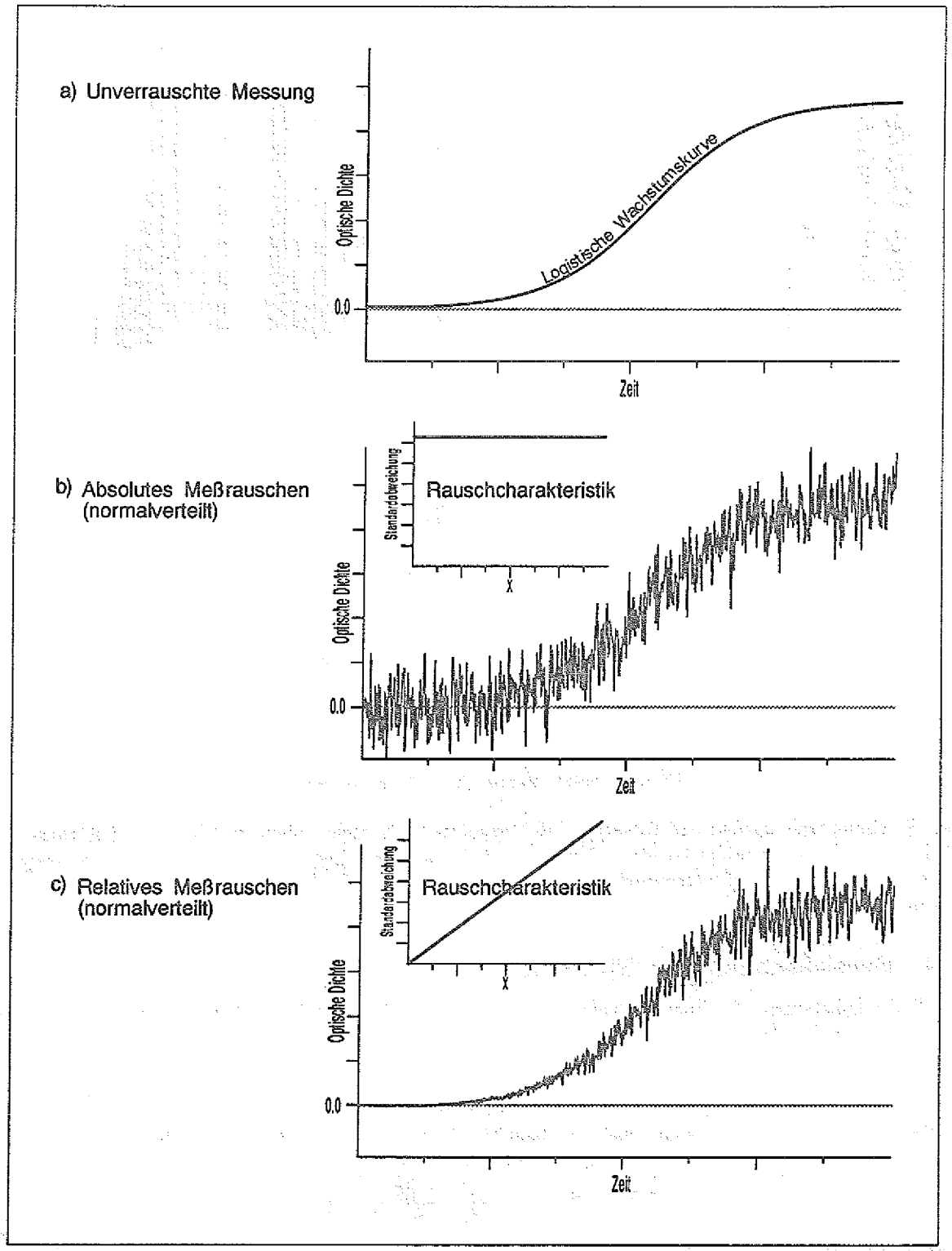


Abbildung 6.6: Modellierung des Meßrauschens bei einer Trübungsmessung (Teil 1)

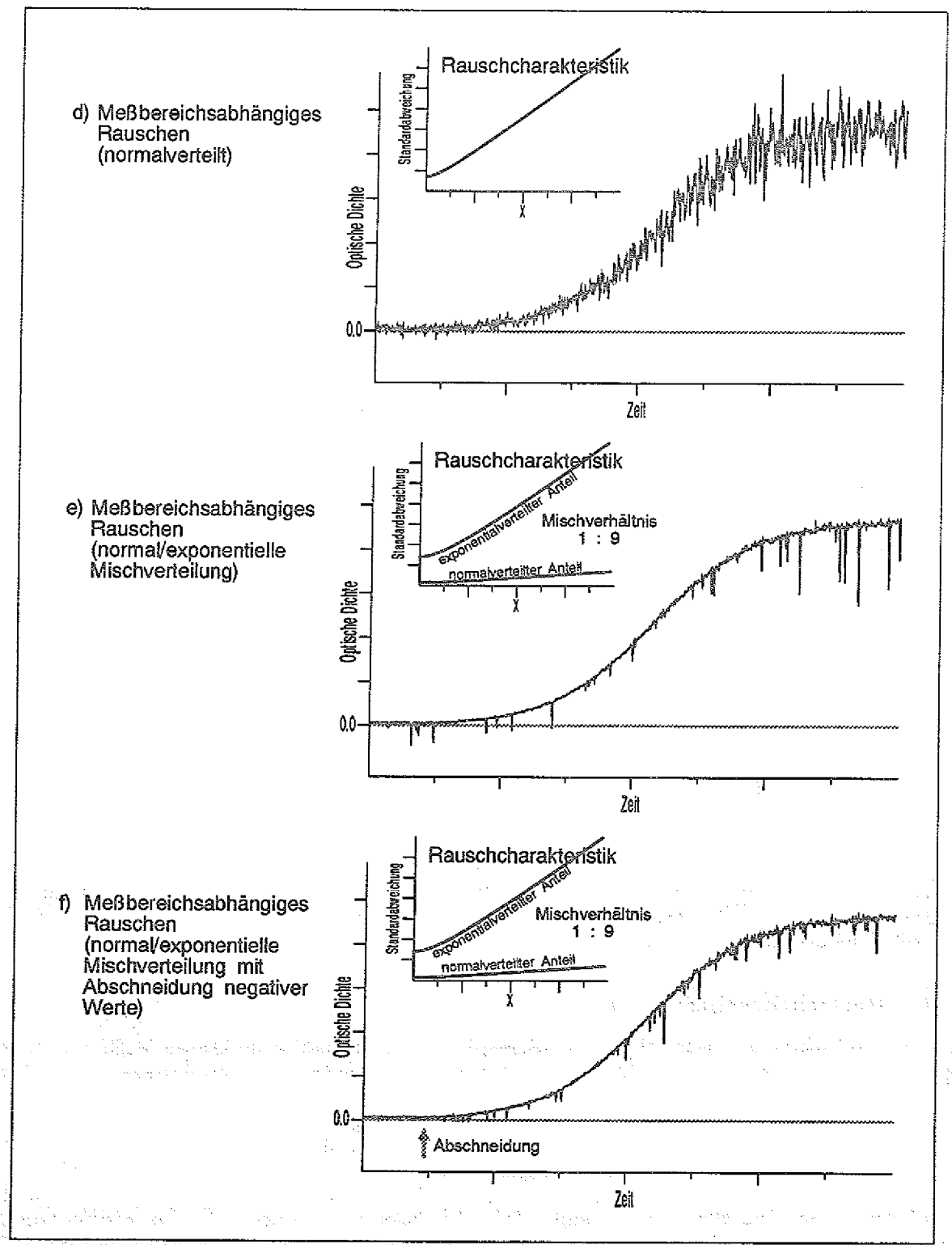


Abbildung 6.7: Modellierung des Meßrauschens bei einer Trübungsmessung (Teil 2)

$f(X)$ ist also asymptotisch linear mit Steigung f'_∞, während $f(0) > 0$ ist. Die untere Grenze der Meßgenauigkeit kann mit f_0 genau eingestellt werden. Eine Skizze von $f(X)$ und das Simulationsergebnis gibt Abbildung 6.7d wieder.

6.3.5 Modellierung der Blasenverteilung

Bis zu diesem Punkt wurde die spezielle Verteilung des Meßfehlers überhaupt nicht berücksichtigt. Zur Beschreibung eignet sich hervorragend ein Mischverteilungsmodell (vgl. 6.2.4)

$$v = v_b$$

Hierbei ist die Verteilervariable b eine Zufallsvariable, die nur die Werte 0 und 1 mit vorgegebenen Wahrscheinlichkeiten annehmen kann:

$$P(b = 0) = 1 - \lambda, \quad P(b = 1) = \lambda.$$

Die Interpretation von b ist:

$$\begin{array}{lll} b = 1 & \Leftrightarrow & \text{"Gasblase im Strahlengang"} \\ b = 0 & \Leftrightarrow & \text{"Keine Gasblase im Strahlengang"} \end{array}$$

Die Variable v_0 muß folglich die durch Gasblasen ungestörte Messung beschreiben. Hierfür kann — mangels weiterer Hypothesen zu den Fehlerquellen — eine Normalverteilung mit sehr kleiner Varianz angenommen werden:

$$v_0 = n \quad (\text{n normalverteilt mit Varianz } \sigma_n^2)$$

Die Variable v_1 muß hingegen in erster Linie die Gasblasenverteilung wiedergeben. Nimmt man als Approximation eine Exponentialverteilung an [5], so erhält man — unter der Annahme, daß das "reguläre" Meßrauschen weiterhin präsent ist — das Modell

$$v_1 = n + e \quad (\text{e exponentialverteilt mit Varianz } \sigma_e^2).$$

Zusammen mit den bisher erreichten Resultaten entsteht das Störungsmodell:

$$Y = X + f(X) \cdot v_b$$

Die Verteilung von v_b hat die drei wählbaren Parameter λ, σ_n und σ_e. Ein visueller Vergleich der Simulation 6.7e mit den realen Prozeßdaten zeigt, daß die Charakteristik des Rauschens bei geeigneter Parameterwahl sehr gut wiedergegeben wird.

6.3.6 Bereichsbeschränkung

Eine letzte Korrektur ist angebracht, da die Blasen-Peaks der Simulation im kleinen Meßbereich noch negativ werden können. Zur Bereichsbeschränkung kann man die unter 6.2.3.2 beschriebene Möglichkeit wählen, oder aber alle Werte < 0 auf den Wert 0 abbilden:

$$\tilde{v} = \begin{cases} v_b & , \text{falls } X + f(X) \cdot v_b \geq 0 \\ 0 & , \text{sonst} \end{cases}$$

Diese Variante gibt dem Wert 0 eine positive Wahrscheinlichkeit, d.h. es entsteht eine Mischverteilung aus einer stetigen Verteilung und einer Punktmassenverteilung. Dies entspricht mit Sicherheit nicht ganz der Realität, dürfte jedoch als Approximation für Simulationszwecke vollauf genügen (Abb. 6.7f).

[5] Die Exponentialverteilung ist die Verweilzeitverteilung der Gasblasen im homogen gerührten Reaktor [Lev72]. Mit wachsender Verweilzeit werden die Blasen kleiner, so daß eine grobe Korrespondenz zwischen Verweilzeitverteilung und Blasengrößenverteilung besteht. Ein detailliertere Modellierung lohnt sich erst bei einem konkreten Objekt.

6.3.7 Verwendung der empirischen Verteilungsfunktion

Für den Fall, daß eine realistische Simulation des Meßrauschens für einen konkreten Prozeß benötigt wird, kann versucht werden, die empirische Verteilungsfunktion des Rauschens in Abhängigkeit von X aufzunehmen. Es besteht dann die Möglichkeit

- mit Hilfe einer Parameteranpassung ein Verteilungsmodell (wie z.B. das soeben beschriebene) an die empirische Verteilungsfunktion anzupassen oder
- unmittelbar Zufallszahlen zu erzeugen, die gemäß der empirischen Verteilungsfunktion verteilt sind (vgl. dazu 10.1.2.4).

6.4 Das statistische Klassifikationsmodell

Die statistische Klassifikation (oder Diskriminanzanalyse) ist ein wichtiges Anwendungsgebiet für Mischverteilungsmodelle. Da in Abschnitt 22.2 ein Beispiel für eine Klassifikation gegeben wird, müssen die Grundlagen (in aller Kürze) erklärt werden:

6.4.1 Aufgabenstellung der Klassifikation

Die Klassifikation geht von Objekten ω aus, welche einer der numerierten *Klassen*

$$\Omega_1, \ldots, \Omega_m$$

angehören. Diese Objekte ω sind nicht direkt zugänglich, vielmehr können lediglich eine Reihe von Messungen

$$X(\omega) = (X_1(\omega), \ldots, X_n(\omega))^T$$

an jedem der Objekte vorgenommen werden. Die Meßergebnisse heißen *Merkmale* des Objekts ω und liegen im sogenannten *Merkmalsraum* $\Re^n$. Aufgabe der Klassifikation ist nun die Bestimmung der Klassenzugehörigkeit von ω aufgrund der gemessenen Merkmale $X_1(\omega), \ldots, X_n(\omega)$.

6.4.1.1 Ein Beispiel

Gegeben seien die Merkmale

$$\begin{aligned} X_1 &= \text{Alter} \\ X_2 &= \text{Größe} \\ X_3 &= \text{Gewicht} \end{aligned}$$

einer Person ω. Es liegt also ein dreidimensionaler Merkmalsraum vor. Aufgrund dieser drei Daten ist eine Einordnung in eine der beiden Klassen

$$\Omega_1 = \{\omega \,|\, \omega \text{ weiblich}\} \quad \text{oder} \quad \Omega_2 = \{\omega \,|\, \omega \text{ männlich}\}$$

vorzunehmen.

6.4.2 Problematik der Klassifikation

Die Problematik der Klassifikation wird an dem Beispiel sehr gut deutlich: Eine eindeutige Zuordnung aufgrund der Merkmale kann nicht vorgenommen werden. Vielmehr kann es (theoretisch) zu jedem Datensatz sowohl Männer als auch Frauen geben, welche diese Merkmale haben. Allerdings sind die Wahrscheinlichkeiten hierfür unterschiedlich groß. Z.B. kann bei den folgenden (extremen) Datensätzen mit großer Sicherheit eine Klasse zugeordnet werden:

$$\begin{array}{llll} (\ 2.20\,\text{m}, & 120\,\text{kg}, & 30\,\text{Jahre}\ ) & \longrightarrow \text{männlich} \\ (\ 1.60\,\text{m}, & 50\,\text{kg}, & 95\,\text{Jahre}\ ) & \longrightarrow \text{weiblich} \\ (\ 1.75\,\text{m}, & 50\,\text{kg}, & 10\,\text{Jahre}\ ) & \longrightarrow \text{weiblich} \end{array}$$

Dagegen geht man bei den Merkmalen

$$(\ 1.70\,\text{m}, \quad 60\,\text{kg}, \quad 25\,\text{Jahre}\ )$$

ein großes Risiko der *Fehlklassifikation* ein. In solchen Fällen wird oft eine zusätzliche *Rückweisungsklasse*

$$\Omega_0 = \{\omega \,|\, \omega \text{ nicht klassifizierbar}\}$$

eingeführt.

6.4.3 Das Mischverteilungsmodell für die Klassifikation

6.4.3.1 Einschränkung auf die Klassen

Das statistische Klassifikationsmodell geht davon aus, daß für jede Klasse Ω_i der auf die Objekte dieser Klasse eingeschränkte Merkmalsvektor

$$X|\Omega_i\,(\omega) \stackrel{def}{=} X\,(\omega\,|\,\omega \in \Omega_i)$$

ein Zufallsvektor ist.

Beispiel: Der Merkmalsvektor $X = (\text{Alter}, \text{Größe}, \text{Gewicht})$ ist eingeschränkt auf die beiden Klassen Frauen und Männer jeweils ein Zufallsvektor, der mit

$$X_{\text{weibl}} \quad \text{bzw.} \quad X_{\text{männl}}$$

bezeichnet werden soll. X_{weibl} und $X_{\text{männl}}$ haben verschiedene Verteilungen, was der Anschauung entspricht, daß Frauen (zumindest in den Altersstufen nach der Pubertät) "im Durchschnitt kleiner und leichter" als Männer sind.

6.4.3.2 Mischung

Aus den Einschränkungen $X|\Omega_i$ des Merkmalsvektors X auf die einzelnen Klassen Ω_i kann nun leicht der gesamte Merkmalsvektor rekonstruiert werden:

$$X(\omega) = \begin{cases} X|\Omega_1(\omega) & , \text{falls } \omega \in \Omega_1 \\ \vdots & \\ X|\Omega_m(\omega) & , \text{falls } \omega \in \Omega_m \end{cases}$$

Dies läßt sich mit Hilfe einer Verteilervariablen V (vgl. 6.2.4) kürzer ausdrücken als:

$$X = X|\Omega_V$$

Zur Festlegung von V muß bekannt sein, wie groß die Wahrscheinlichkeit λ_i dafür ist, daß ein beliebiges Objekt ω in der Klasse Ω_i liegt:

$$\lambda_i = P\,(\omega \in \Omega_i), \qquad \sum_{i=1}^{m} \lambda_i = 1$$

Beispiel: Die Wahrscheinlichkeit dafür, daß eine zufällig ausgewählte Person weiblich oder männlich ist, ist jeweils etwa $\frac{1}{2}$:

$$\lambda_1 = \frac{1}{2}, \quad \lambda_2 = \frac{1}{2}, \quad \lambda_1 + \lambda_2 = 1$$

Die Abbildung 6.8 illustriert diese Situation mit einer Punktwolkendarstellung (man beachte die starke Korrelation zwischen den Klassenverteilungen).

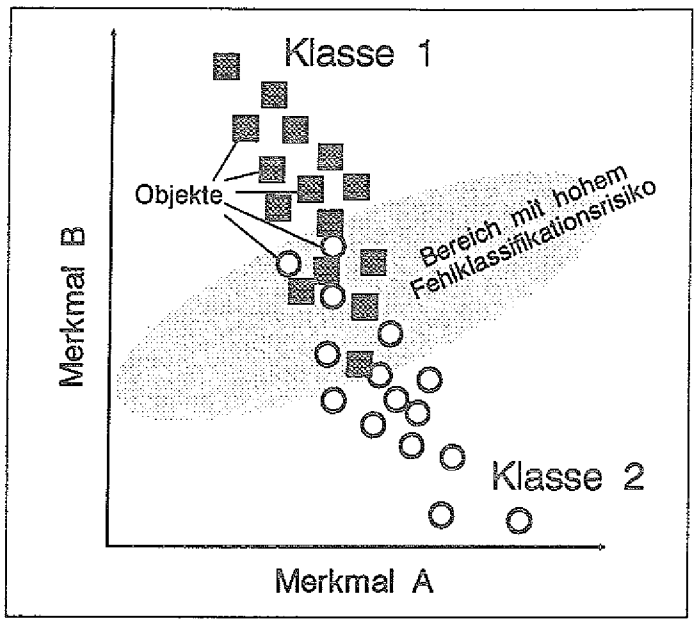


Abbildung 6.8: Visualisierung eines Merkmalsraums

6.5 Lineare Modelle und Regressionsmodelle

6.5.1 Lineare Ansätze

6.5.1.1 Das lineare Modell

Das *allgemeine lineare Modell* hat die Struktur [6]

$$Y = X \cdot \beta + \epsilon$$

mit

dem Vektor der *abhängigen Variablen* $Y = (Y_1 \ \ldots \ Y_n)^T$

dem *Parametervektor* $\beta = (\beta_1 \ \ldots \ \beta_m)^T$

der Matrix der *unabhängigen Variablen* $X = \begin{pmatrix} X_{11} & \ldots & X_{1m} \\ \vdots & \ddots & \vdots \\ X_{n1} & \ldots & X_{nm} \end{pmatrix}$

und der *Störvariablen* $\epsilon = (\epsilon_1 \ \ldots \ \epsilon_m)^T$.

X und β sind deterministische Größen, wobei X als bekannt vorausgesetzt ist [7]. Hingegen ist der Störterm ϵ und folglich auch Y ein Zufallsvektor. Für den Störvektor nimmt man an [8]:

$$\mathrm{E}[\epsilon] = 0 \quad \text{und} \quad \mathrm{Cov}[\epsilon] = \Sigma \quad \text{, mit bekanntem } \Sigma$$

6.5.1.2 Lineare Regressionsmodelle

Häufig will man eine Zufallsvariable y, die z.B. eine fehlerbehaftete Messung beschreibt, als Linearkombination mehrerer Einflußgrößen beschreiben (lineares Regressionsmodell):

[6] Es gibt noch allgemeinere lineare Modelle [FH84], jedoch genügt diese Form für das folgende.
[7] An der Theorie ändert sich wenig, wenn X als zufälliger, aber meßbarer Vektor angesehen wird.
[8] Auch diese Voraussetzung kann abgeschwächt werden.

$$y = \beta_1 \cdot x_1 + \ldots + \beta_m \cdot x_m + \epsilon$$

Führt man dieses Experiment n mal durch, so resultieren n Gleichungen:

$$y_i = \beta_1 \cdot x_{i1} + \ldots + \beta_m \cdot x_{im} + \epsilon_i \ , \quad i = 1, \ldots, n$$

oder kurz

$$Y = X \cdot \beta + \epsilon \qquad \begin{array}{lrcl} \text{mit} & Y & = & (y_i)_{i=1,\ldots,n}, \\ & \beta & = & (\beta_j)_{j=1,\ldots,m}, \\ & X & = & (x_{ij})_{i=1,\ldots,n,\, j=1,\ldots,m} \\ \text{und} & \epsilon & = & (\epsilon_i)_{i=1,\ldots,n}. \end{array}$$

Das lineare Regressionsmodell ist somit ein Spezialfall des allgemeinen linearen Modells. Falls die Experimente unabhängig voneinander durchgeführt werden können, so sind die Störterme unkorreliert, d.h.

$$\mathrm{Cov}\,[\epsilon] = \begin{pmatrix} \sigma_1^2 & \ldots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \ldots & \sigma_n^2 \end{pmatrix}$$

ist eine Diagonalmatrix mit den Varianzen $\sigma_i^2 = \mathrm{Var}\,[\epsilon_i]$. Hat das Rauschen zudem unabhängig von den x_i stets die gleiche Varianz σ^2, so ist speziell

$$\mathrm{Cov}\,[\epsilon] = \sigma^2 \cdot 1 \quad (1 \text{ bezeichnet die Einheitsmatrix}).$$

Für die Einflußgrößen x_i wird oft auch ein Satz von linear unabhängigen Funktionen einer einzelnen Einflußgröße x gewählt:

$$x_1 = f_1(x), \ \ldots, \ x_m = f_m(x)$$

Klassische Beispiele sind:
die Geradenanpassung (vgl. Abb. 6.9a)

$$y = \beta_1 \cdot x + \beta_0 \cdot 1 + \epsilon$$

als Spezialfall der Polynomialen Regression

$$y = \beta_n \cdot x^n + \ldots + \beta_1 \cdot x + \beta_0 \cdot 1 + \epsilon$$

und die Fourier-Reihen-Entwicklung

$$y = \beta_{ns} \cdot \sin(nx) + \beta_{nc} \cdot \cos(nx) + \ldots + \beta_{1s} \cdot \sin(x) + \beta_{1c} \cdot \cos(x) + \beta_0 \cdot 1 + \epsilon \ .$$

6.5.1.3 Fehler in den Variablen

Das lineare Regressionsmodell stellt in vielen Situationen keine adäquate Beschreibung der Versuchssituation dar, da es voraussetzt, daß die unabhängigen Variablen genau vorgebbar oder zumindest exakt meßbar sind. Vor allem bei biologischen Experimenten ist diese Voraussetzung in der Regel verletzt, da die Messung der unabhängigen Variablen selbst einem nicht zu vernachlässigenden Fehler unterliegt [9]. Dies führt (nach Einführung eines weiteren Störterms δ) auf ein lineares Modell mit *Fehlern in den Variablen* (vgl. Abb. 6.9b) [SS78] [GvL79]:

$$Y = (X + \delta) \cdot \beta + \epsilon$$

Solche Modelle sind ungleich schwieriger zu behandeln als normale Regressionsmodelle [Sch71] [Coc68], so daß man sich meist mit der approximativen Beschreibung des Experiments durch ein normales Regressionsmodell begnügt.

[9] Man denke z.B. an enzymatische Meßreihen, bei denen die Startkonzentrationen des Substrats vorzugeben sind.

6.5.1.4 Faktorenanalytisches Modell

Die genaue Beziehung zwischen dem Parametervektor β und der abhängigen Variablen Y ist häufig unbekannt, und es besteht lediglich die Hypothese einer linearen Modellstruktur

$$Y = X \cdot \beta + \epsilon \,,$$

in der sowohl β als auch X unbekannt sind. Man spricht in diesem Fall von einem *Faktorenmodell*. Die Spalten $X_1, X_2, \ldots, X_k$ (auch k ist nicht bekannt) von X werden auch als *Faktoren* bezeichnet, die Einträge von β als *Faktorenladungen*:

$$Y = X_1 \cdot \beta_1 + X_2 \cdot \beta_2 + \ldots + X_k \cdot \beta_k + \epsilon$$

Diese Darstellung zeigt, daß bis auf ein Rauschen alle Werte $X \cdot \beta$ in einem linearen Unterraum des $\Re^n$ liegen. Eine unabhängige Stichprobe erscheint als Punktwolke im Raum, die nahezu eine Hyperebene beschreibt (vgl. Abb. 6.9c). Diese Lage wird präziser beschrieben durch die Gleichung:

$$\mathrm{Cov}\,[Y] = X^T \,\mathrm{Cov}\,[\beta]\, X + \mathrm{Cov}\,[\epsilon] \quad (\text{falls } \beta, \epsilon \text{ unkorreliert sind})$$

Die Aufgabe einer *Faktorenanalyse* ist die Dimensionsschätzung und das Auffinden des linearen Unterraums sowie einer geeigneten Basis dazu aus einer Stichprobe $Y_1, Y_2, \ldots$. Da X und β in dem Produkt $X \cdot \beta$ jeweils nur bis auf einen (Matrix-)Faktor bestimmt werden können, macht man dabei zusätzlich die Normierungsannahme:

$$\mathrm{Cov}\,[\beta] = 1.$$

Es folgt dann das sogenannte *Fundamentaltheorem der Faktorenanalyse* [FH84]:

$$\mathrm{Cov}\,[Y] = X^T X + \mathrm{Cov}\,[\epsilon]$$

6.5.1.5 Gegenüberstellung der linearen Ansätze

Die drei Modellstrukturen

Lineares Modell	$Y = X \cdot \beta + \epsilon$	(ϵ Zufallsvektor,	β unbekannt),
Modell mit Fehlern in den Variablen	$Y = (X + \delta) \cdot \beta + \epsilon$	(ϵ, δ Zufallsvektoren,	β unbekannt),
und das Faktoren-Modell	$Y = X \cdot \beta + \epsilon$	(ϵ, β Zufallsvektoren,	X unbekannt)

sind in Abbildung 6.9 noch einmal schematisch gegenübergestellt.

6.5.2 Nichtlineare Ansätze

6.5.2.1 Nichtlineares Regressionsmodell

Beim nichtlinearen Regressionsmodell wird die lineare Funktion $x_1 \cdot \beta_1 + \ldots + x_m \cdot \beta_m$ ersetzt durch einen beliebigen Funktionsansatz $f(x, \beta)$:

$$y = f(x, \beta) + \epsilon,$$

Erneut sind die unabhängigen Variablen x bekannt und frei wählbar; β ist zu schätzen. Verbunden mit einem solchen *nichtlinearen Regressionsmodell* ist das zugehörige *Parameter-Anpassungsproblem* der Schätzung von β [10].

[10] Bei der gleichzeitigen Anpassung mehrerer Funktionsterme können y und $f(x, \beta)$ auch vektoriell sein. Die Theorie ändert sich durch diese Erweiterung kaum.

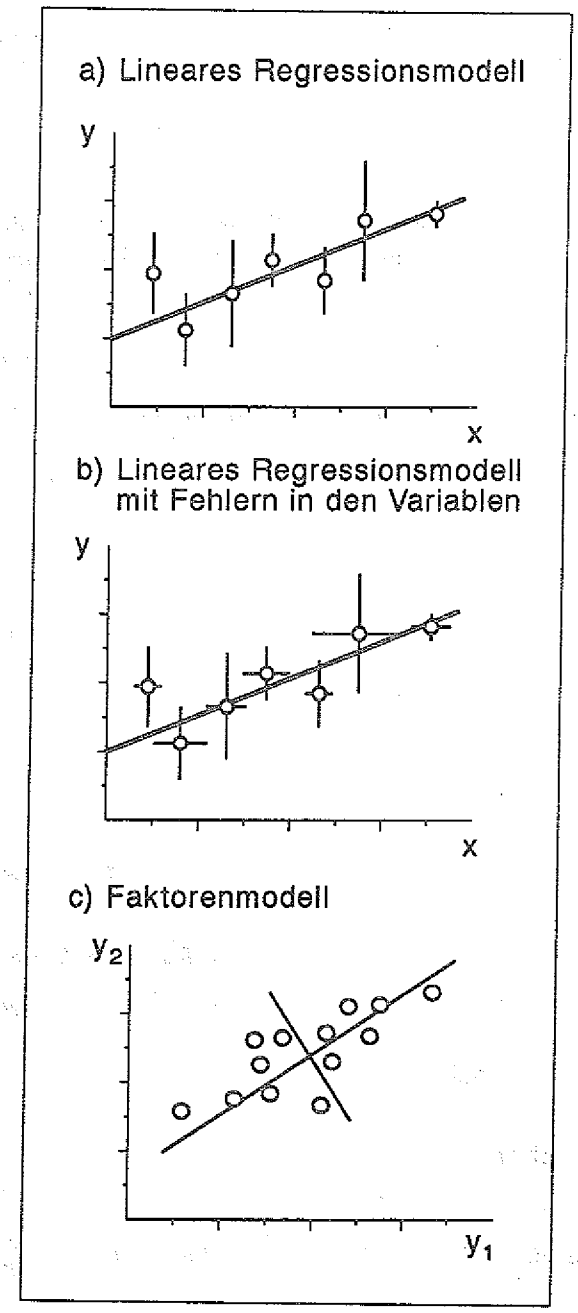


Abbildung 6.9: Verschiedene lineare Modellansätze

Das nichtlineare Regressionsmodell hat insofern eine Sonderstellung zwischen den statistischen und stochastischen Modellen, als die Funktion $f(x,\beta)$ auch durch eine Differentialgleichung gegeben sein kann, so daß die Zeit t explizit enthalten ist (vgl. die Unterscheidung in 4.1.1.3). Im Unterschied zu den später beschriebenen stochastischen Methoden, die ausschließlich auf lokal begrenzten Zeitabschnitten arbeiten, benötigt eine Parameteranpassung stets den gesamten Datensatz. Sie ist also vom Prinzip her eine off-line-Methodik [11]. Aus diesem Grunde ist sie hier als statistische Methode klassifiziert.

6.5.2.2 Nichtlineare Verallgemeinerung des Faktoren-Modells

Bei der Verallgemeinerung des Faktoren-Modells auf den nichtlinearen Fall geht man von Hyperflächen aus, die durch nichtlineare Gleichungen mit Störterm beschrieben werden:

$$f(Y,\beta) = \epsilon$$

Ohne die Annahme einer speziellen Struktur für die Funktion f ist dieses Problem jedoch viel zu allgemein gestellt. Zudem geht die Eindeutigkeit der Lösung selbst bei speziellen Ansätzen verloren [HS89] [Ban88].

6.6 Bilanzierung

Betrachtet wird im folgenden ein verallgemeinertes Reaktionssystem mit n beteiligten Komponenten und m Reaktionsgleichungen (X_1 bezeichnet stets die Biomasse):

$$\begin{aligned} \nu_{11} \cdot X_1 + \cdots + \nu_{1n} \cdot X_n &\xrightarrow{r_1 X_1} \\ &\vdots \\ \nu_{m1} \cdot X_1 + \cdots + \nu_{mn} \cdot X_n &\xrightarrow{r_m X_1} \end{aligned}$$

Bisher wurde von (verallgemeinerten) Reaktionsgleichungen ausgegangen, bei denen die stöchiometrischen Gleichungen für die chemischen Elemente nicht zu gelten brauchen (vgl. 5.1.1.2). Für den Spezialfall, daß derartige Gleichungen anwendbar sind, stößt man auf eine Reihe von linearen Modellansätzen, die statistisch verwertbar sind [12]:

6.6.1 Erhaltungsgrößen

6.6.1.1 Definition einer Erhaltungsgröße

Eine *Erhaltungsgröße* für ein Reaktionssystem

$$\nu_{i1} \cdot X_1 + \cdots + \nu_{in} \cdot X_n \xrightarrow{r_i X_j} \quad , i = 1, \ldots, m$$

ist eine lineare Abbildung

$$e(X) \text{ mit der Eigenschaft } \dot{e}(X(t)) = e(\dot{X}(t)) = 0 \quad \text{wobei } X = (X_1, \ldots, X_n)^T .$$

Die wichtigsten Beispiele für Erhaltungsgrößen sind:

- Die Gesamtmasse der bei der Reaktion beteiligten Komponenten: Liegen alle Komponenten X_j mit einer chemischen Summenformel

$$C_{n_{C,j}} H_{n_{H,j}} O_{n_{O,j}} N_{n_{N,j}} S_{n_{S,j}} P_{n_{P,j}} \cdots$$

[11] Bei den naturgemäß "langsamen" biotechnischen Systemen steht allerdings genug Rechenzeit zur Verfügung, um eine Parameterschätzung auch on-line durchführen zu können [Gol89].

[12] Eine Pionierarbeit auf dem Gebiet der computergestützten Bilanzierung ist [CWW77].

vor, und haben alle X_i die Einheit mol so ist diese Erhaltungsgröße gegeben durch

$$\begin{aligned} m(X_1,\ldots,X_n) \;=\; & M_C \textstyle\sum_j n_{C,j} X_j \;+ M_H \sum_j n_{H,j} X_j \;+ M_O \sum_j n_{O,j} X_j \\ & + M_N \textstyle\sum_j n_{N,j} X_j \;+ M_S \sum_j n_{S,j} X_j \;+ M_P \sum_i n_{P,j} X_i \;+ \ldots , \end{aligned}$$

wobei $M_C, M_H, M_O, M_N, M_S, M_P, \ldots$ die molaren Massen der Elemente bezeichnen.

- Die Gesamtmasse jedes vorkommenden chemischen Elements, z.B.:

$$m_C(X_1,\ldots,X_n) = M_C \sum_j n_{C,j} X_j$$

- Entsprechende Erhaltungsgrößen können auch für Konzentrationen in der Einheit Mol formuliert werden. Die Bezeichnungen $m_C, m_H, m_O, \ldots$ sollen in dieser Situation beibehalten werden, da die Unterscheidung aus dem Kontext heraus klar ist.
- Vielfach wurde auch versucht, die Enthalpie und die Entropie des Systems zu bilanzieren [Bor88] [vK89]. Die Problematik liegt hier darin, daß diese Größen nur in einem kleinen Temperatur- und Druckbereich als linear in den X_j angesehen werden können. Ferner muß für eine statistische Anwendung eine Wärmeflußmessung über den gesamten Fermenter realisiert werden (Kalorimetrie [vSI89]), da im Gegensatz zu den bisher genannten Erhaltungsgrößen zusätzlich ein Energiefluß über die Systemgrenzen hinweg erfolgt.
- Bei der in Abschnitt 5.1.1.2 beschriebenen Enzymreaktion

$$E + S \longleftrightarrow ES \longrightarrow E + P$$

sind die Gesamtmenge des ungebundenen und gebundenen Enzyms

$$m_E(S, E, ES, P) = E + ES$$

und die Gesamtmenge des ungebundenen, gebundenen und bereits umgesetzten Substrats

$$m_S(S, E, ES, P) = S + ES + P$$

Erhaltungsgrößen (vgl. dies mit den Ausführungen in 5.1.2.3).

Im klassischen Kontext der Stöchiometrie chemischer Reaktionen besagt die Tatsache, daß die Stöchiometrie einer Reaktion korrekt ist, nichts anderes, als daß $m_C, m_H, m_O, m_N, m_S, m_P, \ldots$ für alle chemischen Elemente Erhaltungsgrößen der Reaktion sind.

6.6.1.2 Beispiel

Man betrachte das chemische Reaktionssystem (entnommen aus [Roe83])

$$\begin{aligned} CH_4 + 2\,O_2 &\xrightarrow{R_1} CO_2 + 2\,H_2O \\ CH_4 + \tfrac{1}{2}\,O_2 &\xrightarrow{R_2} CH_3OH \\ CH_4 + \tfrac{3}{2}\,O_2 &\xrightarrow{R_3} CO + 2\,H_2O\,. \end{aligned}$$

Bezeichnet man die Konzentrationen mit denselben Bezeichnungen wie die Komponenten selbst, so erhält man den 6-dimensionalen Zustandsvektor

$$X = (\, CH_4,\; O_2,\; CO_2,\; CO,\; H_2O,\; CH_3OH\,)^T$$

und das Differentialgleichungssystem (einheitslos notiert)

$$\dot{X} = \begin{pmatrix} -\frac{1}{1} & -1 & -\frac{1}{1} \\ -2 & -\frac{1}{2} & -\frac{3}{2} \\ 1 & 0 & 0 \\ 0 & 0 & 1 \\ 2 & 0 & 2 \\ 0 & 1 & 0 \end{pmatrix} \cdot \begin{pmatrix} R_1 \\ R_2 \\ R_3 \end{pmatrix}.$$

Auf der anderen Seite existieren für die chemischen Elemente C, H, O die drei Erhaltungsgrößen:

$$\begin{array}{rcl} e_C(X) & = & 1 \cdot CH_4 + 0 \cdot O_2 + 1 \cdot CO_2 + 1 \cdot CO + 0 \cdot H_2O + 1 \cdot CH_3OH \\ e_H(X) & = & 4 \cdot CH_4 + 0 \cdot O_2 + 0 \cdot CO_2 + 0 \cdot CO + 2 \cdot H_2O + 4 \cdot CH_3OH \\ e_O(X) & = & 0 \cdot CH_4 + 2 \cdot O_2 + 2 \cdot CO_2 + 1 \cdot CO + 1 \cdot H_2O + 1 \cdot CH_3OH \ , \end{array}$$

die man auch als einzeilige Matrizen schreiben kann:

$$\begin{array}{rcl} e_C(X) & = & (\ 1,\ 0,\ 1,\ 1,\ 0,\ 1\ ) \cdot X \\ e_H(X) & = & (\ 4,\ 0,\ 0,\ 0,\ 2,\ 4\ ) \cdot X \\ e_O(X) & = & (\ 0,\ 2,\ 2,\ 1,\ 1,\ 1\ ) \cdot X \end{array}.$$

Nach Zusammenfassung dieser Vektoren zu einer Matrix ist die Aussage, daß e_C, e_H, e_O Erhaltungsgrößen sind, äquivalent zu

$$\begin{pmatrix} 1 & 0 & 1 & 1 & 0 & 1 \\ 4 & 0 & 0 & 0 & 2 & 4 \\ 0 & 2 & 2 & 1 & 1 & 1 \end{pmatrix} \cdot \dot{X} = 0,$$

was man sofort nachrechnet.

6.6.2 Bilanzierung von Biomasse

Bei verallgemeinerten Reaktionssystemen, die biologische Komponenten enthalten, stellt sich das Problem der Auffindung geeigneter Erhaltungsgrößen. Bei biotechnischen Systemen werden im allgemeinen nur die vier Elemente

$$C, H, O, N$$

die 90-95% der Gesamtzusammensetzung ausmachen [ESF85], [Roe83] in den Summenformeln berücksichtigt, so daß für eine Bilanzierung nur die vier Erhaltungsgrößen m_C, m_H, m_O, m_N herangezogen werden.

6.6.2.1 Die Elementarzusammensetzung von Biomasse

Bei Anwendung des Konzepts der Erhaltungsgrößen auf Mikroorganismen, muß die Elementarzusammensetzung der Biomasse bzgl. der Elemente C, H, O, N bekannt sein. Es zeigt sich, daß der Anteil dieser Elemente in einem speziellen Organismus über relativ weite Bereiche konstant ist, so daß man (nach Normierung des Koeffizienten von C auf 1.0) von einer "chemischen Formel für Biomasse" sprechen kann:

$$\text{Biomasse} = C_1 H_\beta O_\gamma N_\delta$$

Eine für viele Organismen gültige Faustformel ist

$$C_{1.0} H_{2.0} O_{0.5} N_{0.2}.$$

Genauere Daten sind für verschiedene Organismen und wechselnde Substrate tabelliert worden [Roe83].

6.6.2.2 Beispiel

Das Hefe-Wachstum auf Glucose mit Ammonium als Stickstoffquelle und dem Produkt Äthanol kann beschrieben werden durch die globale Wachtumsgleichung (vgl. 5.2.6.6):

$$\underset{\text{Glucose}}{\nu_S(X)\,C_6H_{12}O_6} + \underset{\text{Sauerstoff}}{\nu_O(X)\,O_2} + \underset{\text{Stickstoffquelle}}{\nu_N(X)\,NH_3}$$

$$\longrightarrow \underset{\text{Biomasse}}{C_{1.0}H_{2.0}O_{0.5}N_{0.2}} + \underset{\text{Äthanol}}{\nu_P(X)\,C_2H_6O} + \underset{\text{Wasser}}{\nu_H(X)\,H_2O} + \underset{CO_2}{\nu_C(X)\,CO_2}$$

Die Gleichung ist eine Gesamtbilanz über den Hefestoffwechsel, d.h. die stöchiometrischen Koeffizienten sind keine Konstanten, sondern vom Systemzustand abhängig (vgl. auch die Diskussion in 5.2.5.3). Anwendung der Bilanzgleichungen für C, H, O, N liefert die Beziehungen:

$$\begin{array}{rrrrrrr}
-6\nu_S & & +1.0 & +2\nu_P & & +\nu_C & =0 \\
-12\nu_S & -3\nu_N & +2.0 & -6\nu_P & +2\nu_H & & =0 \\
+6\nu_S & -2\nu_O & +0.5 & +1\nu_P & +1\nu_H & +2\nu_C & =0 \\
 & -1\nu_N & +0.2 & & & & =0
\end{array}$$

Dise sind 4 Gleichungen für 6 stöchiometrische Koeffizienten. Die Stöchiometrie dieser Reaktion hat somit nur zwei Freiheitsgrade.

6.6.3 Anwendung linearer Modelle auf die Bilanzierung

6.6.3.1 Das allgemeine Stoffwechselmodell als lineares Modell

In Abschnitt 5.2.6.1 wurde die allgemeine Modellstruktur

$$\dot{X} = N \cdot X_1 \cdot r(X) \quad \text{mit} \quad \begin{array}{rcl} \dim X & = & n-s \\ \dim r & = & m-s \end{array},$$

für den mikrobiellen Stoffwechsel hergeleitet, wobei (bezogen auf das zugrundeliegende verallgemeinerte Reaktionssystem)

$$\begin{array}{rcl} n & = & \text{Zahl der Zustandsvariablen,} \\ m & = & \text{Zahl der Reaktionskinetiken und} \\ s & = & \text{Zahl der Stationaritätsannahmen.} \end{array}$$

Es konnte gezeigt werden, daß $\dot{X}$ stets im theoretischen Reaktionsraum (vgl. 5.1.2.3)

$$\operatorname{im} N = \operatorname{span}\{N_1, \ldots, N_{m-s}\}$$

($N_1, \ldots, N_{m-s}$ die Spaltenvektoren von N) des Reaktionssystems liegt. Geht man nun davon aus, daß die Ableitung

$$\begin{array}{ll} \dot{X}(t) & \text{bis auf einen Meßfehler} \\ \epsilon(t) & \text{meßtechnisch zugänglich und die stöchiometrische Matrix} \\ N & \text{bekannt ist,} \end{array}$$

so erweist sich (zu jedem Zeitpunkt t) das Modell

$$\dot{X}(t) = N \cdot X_1 \cdot r(X(t)) + \epsilon(t)$$

als lineares Regressionsmodell zur Schätzung von $r(X(t))$.

An dieser Stelle wird klar, warum bei der Klassifikation der Meßgrößen in Abschnitt 1.2.2.8 eine eigene Unterscheidung eingeführt wurde, zwischen solchen Größen X_i bei denen $\dot{X}_i$ direkt meßbar ist, und solchen, bei denen "nur" X_i gemessen werden kann:

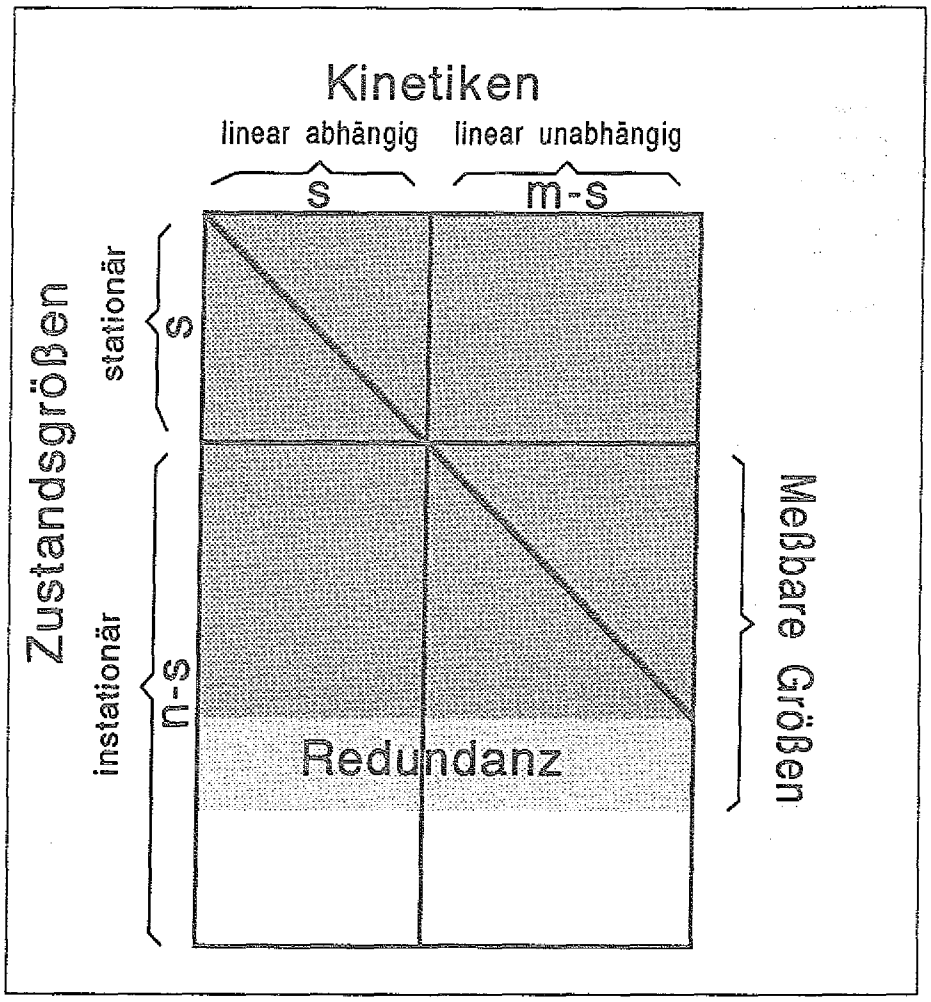


Abbildung 6.10: Dimensionsbeziehungen beim theoretischen Reaktionsraum

Bei direkt meßbarer Ableitung $\dot{X}_i$ liegt im allgemeinen einen kleinerer Meßfehler ϵ vor, als bei einer "lediglich" direkt meßbaren Größe X_i, da in diesem Fall $\dot{X}_i$ nur über eine Differentiation geschätzt werden kann, was bei verrauschten Daten bekanntermaßen mit einem größeren Fehler ϵ verbunden ist.

Aufgrund des vorausgesetzten Maximalrangs von N (siehe 5.2.6.2) hat $r(X)$ und damit auch der theoretische Reaktionsraum die Dimension

$$\dim(\mathrm{im}\, N) = m - s \; .$$

Damit stehen theoretisch n Gleichungen zur Schätzung von $m - s$ Kinetiken zur Verfügung. In der Praxis sind jedoch nicht alle Größen X_j meßtechnisch zugänglich (z.B. ist H_2O niemals meßbar). Daher müssen die entsprechenden Gleichungen aus dem — in der Regel überbestimmten — Gleichungssystem $\dot{X} = N \cdot X_1 \cdot r(X)$ gestrichen werden. Die Zahl der Gleichungen, die — bis zum Erreichen eindeutiger Bestimmtheit des Gleichungssystems — herausgenommen werden können, ist $(n-s)-(m-s) = n-m$. Folglich müssen mindestens $m - s$ Messungen zur Verfügung stehen, um $r(X)$ aufgrund des linearen Modells schätzen zu können. Diese Dimensionsüberlegungen veranschaulicht die Abbildung 6.10.

Folgende Fälle sind somit zu unterscheiden:

a) Zahl der gemessenen Größen $> m - s$:
Die Meßgrößen enthalten dann redundante Information, die mit Hilfe der statistischen Schätztheo-

rie zur Reduktion des Fehlers bei den geschätzten Kinetiken benutzt werden kann (Ausgleichsrechnung). Hierfür gibt es mehrere Anwendungen:

- Es ist möglich, dem gesamten Stoffwechsel des Organismus "von außen zuzusehen" (Stoffwechselüberwachung).
- Fehlerhafte Meßgrößen können erkannt werden (Meßfehlererkennung) [TFRF89].
- Bei a priori bekannten Kinetiken sind Abweichungen vom regulären Verlauf detektierbar (Erkennung kritischer Zustände).

b) Zahl der gemessenen Größen $= m - s$:
In dieser Situation stehen die zu schätzenden Kinetiken in einer eineindeutiger Beziehung zu den gemessenen Größen. Diese drückt sich in Form einer eindeutig bestimmten linearen Gleichung aus. Allerdings übertragen sich die Meßfehler dabei direkt auf die Schätzungen.

c) Zahl der gemessenen Größen $< m - s$:
Obwohl für diesen Fall aufgrund der Gleichungen keine kinetischen Daten schätzbar sind, bestehen doch interessante Anwendungsmöglichkeiten:

i) Es können noch Linearkombinationen der r_i ermittelt werden. Z.B. kann bei der Backhefe das Verhältnis von Gärungs- und Atmungsstoffwechsel mit Hilfe des *respiratorischen Quotienten*

$$\mathrm{RQ} = \frac{\dot{O}_2}{\dot{CO}_2} \iff \dot{O}_2 - \mathrm{RQ} \cdot \dot{CO}_2 = 0$$

($\dot{O}_2$ die Sauerstoffaufnahme, $\dot{CO}_2$ die CO_2-Produktion) überwacht werden.

ii) Eine weitere Perspektive stellt die Beobachter- und Filtertheorie dar, auf die in den Abschnitten 9.2 und 11.2 noch eingegegangen wird. Ein Beobachter wertet nicht nur die Information $\dot{X}(t)$ zu einem festen Zeitpunkt t aus, sondern zusätzlich auch die Information, die im Zeitverlauf von $X(t)$ enthalten ist. Auf diese Weise können — unter gewissen Beobachtbarkeitsbedingungen — auch solche Größen geschätzt werden, die mit Hilfe der Bilanzierung allein nicht zugänglich wären.

Sobald die unabhängigen Kinetiken $r_1, \ldots, r_{m-s}$ des allgemeinen Modells bestimmt sind, können daraus auch die abhängigen Kinetiken $r_{m-s+1}, \ldots, r_{n-s}$ nach dem in Abschnitt 5.2.6.1 beschriebenen Verfahren berechnet werden. Dabei werden allerdings die Schätzfehler mit übertragen!

Beispiel : Bei einem Wachstum mit wachstumsgekoppelter Produktbildung wird der gesamte Stoffwechsel durch die beiden Reaktionsraten σ (Substrataufnahme) und m_S (Erhaltungsstoffwechsel) bestimmt. Hier liegen (vgl. 5.2.2.3 und 5.2.6.3) die

n	$=$	5	Zustandsgrößen	$X, S, P, T, E,$
m	$=$	4	Kinetiken	σ, ρ, m_S, μ und
s	$=$	2	abhängigen Kinetiken	ρ, μ

vor, d.h. der stöchiometrische Reaktionsraum ist $m - s = 2$-dimensional. Es genügt folglich die Messung von 2 am Stoffwechsel beteiligter Komponenten, (z.B. Biomasse, Substrat, Produkt, CO_2, O_2) um alle anderen zu berechnen. Nach Schätzung der beiden Kinetiken σ, m_S kann daraus die Wachstumsrate μ und die Energieproduktion ρ berechnet werden.

6.6.3.2 Die Erhaltungsgleichungen als lineares Modell

Gegeben seien t Erhaltungsgrößen, die man in Form von einzeiligen Matrizen darstellen kann:

$$\begin{aligned} e_1 &= (e_{1,1}, \ldots, e_{1,n-s}) \\ &\vdots \\ e_t &= (e_{t,1}, \ldots, e_{t,n-s}) \end{aligned}$$

Dann gilt für jede dieser Matrizen zu allen Zeiten:

$$e_i \cdot \dot{X}_j(t) = 0$$

Zusammengefaßt zur $t \times (n-s)$-*Erhaltungsmatrix*

$$E = \begin{pmatrix} e_{1,1} & \cdots & e_{1,n-s} \\ \vdots & \ddots & \vdots \\ e_{t,1} & \cdots & e_{t,n-s} \end{pmatrix}$$

folgt:

$$E \cdot \dot{X}(t) = 0$$

Anders gesagt: $\dot{X}(t)$ liegt stets in dem linearen Unterraum

$$\ker E = \{\xi \mid E\xi = 0\} \subset \Re^{n-s}.$$

Dieser Raum soll im folgenden als der *stöchiometrische Reaktionsraum* bezeichnet werden. Die Begriffsbildung ist — im Gegensatz zum Begriff des theoretischen Reaktionsraums (vgl. 5.1.2.3) — abhängig von der Zahl der zur Verfügung stehenden Bilanzgleichungen!

Es soll angenommen werden, daß $t \leq n-s$ gilt und E Maximalrang hat [13] (ggfs. streiche man linear abhängige Erhaltungsleichungen). Dann hat der Unterraum $\ker E$ nach der Dimensionsformel für lineare Abbildungen [Fis75] die Dimension

$$\dim(\ker E) = (n-s) - \operatorname{rang} E = (n-s) - t$$

so daß eine Basis

$$F_1, \ldots, F_{(n-s)-t}$$

von $\ker E$ berechnet werden kann. Bezüglich dieser Basis kann stets geschrieben werden:

$$\dot{X} = \beta_1 \cdot F_1 + \ldots + \beta_{(n-s)-t} \cdot F_{(n-s)-t} \stackrel{def}{=} F \cdot \beta$$

mit der aus den Basisvektoren zusammengesetzten $(n-s)-t \times (n-s)$-Matrix

$$F = (F_1, \ldots, F_{(n-s)-t}) \ .$$

F kann als eine hypothetische stöchiometrische Matrix zu den hypothetischen Reaktionskinetiken $\beta_1(X)$ $\ldots \beta_t(X)$ interpretiert werden. Formal ist diese Situation identisch mit der beim allgemeinen Stoffwechselmodell.

Da $\dot{X}$ nur mit Störtermen meßbar ist, liegt ein lineares Modell vor. Es enthält $n-s$ Gleichungen, zur Schätzung der Parameter $\beta_1, \ldots, \beta_{(n-s)-t}$ (Abb. 6.11, vgl. auch mit Abb. 6.10). Analog zu dem in Abschnitt 6.6.3.1 Gesagten, müssen weiterhin die nicht meßtechnisch zugänglichen Größen $\dot{X}_j$ gestrichen werden. Es ist zu unterschieden:

a) Zahl der gemessenen Größen $> (n-s) - t$:
Es ist redundante Information vorhanden, die zur Fehlerreduktion bei den geschätzten hypothetischen Kinetiken benutzt werden kann.

b) Zahl der gemessenen Größen $= (n-s) - t$:
Die zu schätzenden Kinetiken sind durch die gemessenen Größen eindeutig bestimmt, wobei eine Fehlerreduktion nicht mehr möglich ist.

[13] Diese Annahme trifft bei praktisch anwendbaren Fällen in der Regel zu. Desweiteren wird in Abschnitt 6.6.4.1 gezeigt, daß stets $t < n - m$ gelten muß.

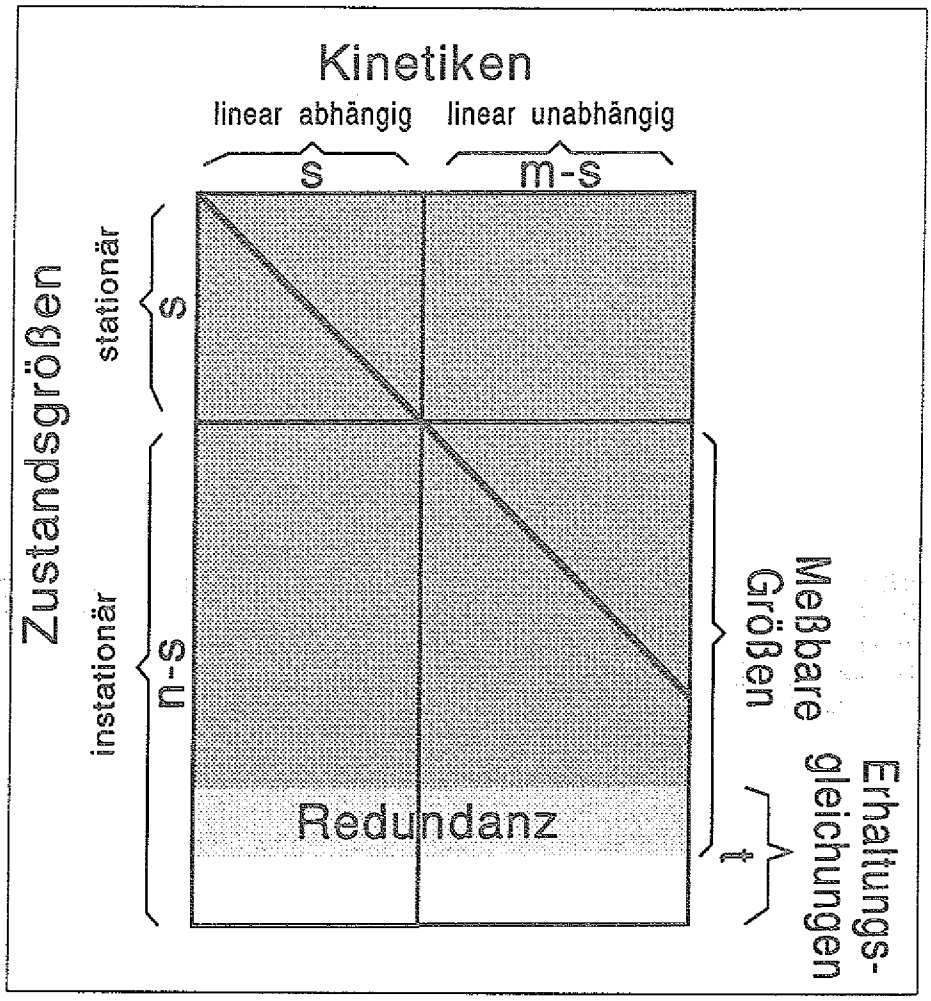


Abbildung 6.11: Dimensionsbeziehungen beim stöchiometrischen Reaktionsraum

c) Zahl der gemessenen Größen $< (n-s)-t$:
In diesem Fall sind möglicherweise noch Filter oder Beobachter einsetzbar.

Beispiel: Bei der in Abschnitt 6.6.2.2 aufgeführten allgemeinen Reaktionsgleichung für das Wachstum auf Glucose mit Produkt Äthanol liegen

$$\begin{array}{rcll} n-s & = & 7 & \text{Komponenten} \qquad X, S, P, O_2, NH_3, CO_2, H_2O \text{ und} \\ t & = & 4 & \text{Erhaltungsgleichungen für die Elemente} \quad C, O, H, N \end{array}$$

vor. Es folgt, daß mindestens $(n-s)-t = 3$ Komponenten gemessen werden müssen, um alle anderen berechnen zu können. Da die Konzentration des freiwerdenden Wassers und der Stickstoffquelle nicht bzw. nur schwer gemessen werden können, bleiben drei von den übrigen Größen übrig, die je nach meßtechnischer Ausrüstung gewählt werden können.

6.6.3.3 Das allgemeine Stoffwechselmodell als Faktoren-Modell

Wenn kein konkretes Modell des Reaktionsmechanismus vorliegt, so besteht trotzdem Grund zu der Annahme, daß ein allgemeines Stoffwechselmodell

$$\dot{X} = N \cdot X_1 \cdot r(X)$$

(unbekannter Dimensionalität) existiert. Dieses Modell kann — da N unbekannt ist — als Faktoren-Modell gelesen werden. Sind genügend viele Messungen zu verschiedenen Zeitpunkten $t_1, t_2, \ldots$ vorhanden, so ist daraus der theoretische Reaktionsraum $\operatorname{im} N$ schätzbar. Der aus der Schätzung resultierende Raum wird als *empirischer Reaktionsraum* bezeichnet [Lia89]. Dieser Begriff ist somit abhängig von der zur Verfügung stehenden Meßinformation. Es ist ferner zu beachten, daß beim faktorenanalytischen Ansatz — im Gegensatz zu den bisher genannten Verfahren — die Messung zu einem Zeitpunkt t nicht genügt!

6.6.4 Die Beziehung zwischen theoretischem, stöchiometrischem und empirischem Reaktionsraum

Auf der Basis des allgemeinen Stoffwechselmodells

$$\dot{X} = N \cdot X_1 \cdot r(X)$$

und einer Erhaltungsmatrix mit

$$E \cdot \dot{X} = 0$$

wurden die Begriffe

des theoretischer Reaktionsraums	$\operatorname{im} N$,
des stöchiometrischen Reaktionsraums	$\ker E$ und
des empirischen Reaktionsraums ($\hat{N}$ eine Schätzung von N)	$\operatorname{im} \hat{N}$

eingeführt. Die Beziehung dieser Räume untereinander soll verdeutlicht werden.

6.6.4.1 Unterraumbeziehungen

Wendet man die Erhaltungs-Matrix E auf das allgemeine Stoffwechselmodell an, so ergibt sich:

$$E \cdot \dot{X} = E \cdot N \cdot X_1 \cdot r(X) = 0$$

Da diese Gleichung für beliebige Kinetiken $r(X)$ Gültigkeit hat, folgt:

$$E \cdot N = 0$$

Diese Gleichung drückt eine Unterraumbeziehung zwischen dem theoretischen Reaktionsraum $\operatorname{im} N$ und dem stöchiometrischen Reaktionsraum $\ker E$ aus:

$$\operatorname{im} N \subset \ker E$$

Insbesondere ist die Basis $N_1, \ldots, N_{m-s}$ von $\operatorname{im} N$, bestehend aus den Spalten von N auch ein System linear unabhängiger Vektoren in $\ker E$. Dieses System läßt sich zu einer Basis $F_1, \ldots, F_{(n-s)-t}$ von $\ker E$ erweitern. Es besteht somit eine Korrespondenz:

$$\begin{array}{lcllcl}
F_1 & \leftrightarrow & N_1 & \beta_1 & \leftrightarrow & r_1 \\
 & \vdots & & & \vdots & \\
F_{m-s} & \leftrightarrow & N_{m-s} \quad \text{und} & \beta_{m-s} & \leftrightarrow & r_{m-s} \\
F_{m-s+1} & \leftrightarrow & ? & \beta_{m-s+1} & \leftrightarrow & 0 \\
 & \vdots & & & \vdots & \\
F_{(n-s)-t} & \leftrightarrow & ? & \beta_{(n-s)-t} & \leftrightarrow & 0
\end{array}$$

Insbesondere muß — unter der Voraussetzung, daß E Maximalrang hat — gelten:

$$(n-s) - t \geq m - s \quad \Rightarrow \quad n - m > t$$

Es folgt, daß die Maximalzahl von linear unabhängigen Erhaltungsgrößen für das System — vorausgesetzt das jeweilige Stoffwechselmodell ist korrekt — gleich $n - m$ ist [14]. Diese Beziehung ist aus der Literatur

[14] Beachte hier die in 5.2.6.2 gestellte Forderung, daß $m < n$ gilt.

über chemische Reaktionssysteme bekannt [Fei77].

In vielen Situationen bewirkt der Reaktionsmechanismus der Reaktion, daß der Raum $\ker E$ nicht ganz ausgeschöpft wird ($m - s < t$). Eine Anwendung des linearen Modells auf der Basis des Reaktionsmodells hat dann den großen Vorteil gegenüber dem linearen Modell auf der Basis der Erhaltungsgleichungen, daß weniger Messungen benötigt werden, um die Parameter zu schätzen.

Beispiel: Die beiden Beispiele aus Abschnitt 6.6.3.1 und 6.6.3.2 für das Wachstum mit gekoppelter Produktbildung illustrieren dies sehr deutlich:

Zur Berechnung der Konzentrationen aller Komponenten sind:

$$\text{aufgrund der Stöchiometrie} \quad (n - s) - t = 3 \text{ Messungen}$$

erforderlich, wogegen

$$\text{auf der Basis eines Stoffwechselmodells} \quad m - s = 2 \text{ Messungen}$$

genügen. Der theoretische Reaktionsraum enthält somit mehr Information als der stöchiometrische Reaktionsraum.

Über die Beziehung des empirischen Reaktionsraums zu den anderen Räumen kann allgemein nichts ausgesagt werden, da er auch vom Verlauf des jeweils zugrundeliegenden Experiments abhängig ist. Falls zusätzlich eine Erhaltungsmatrix E gegeben ist, muß auch der empirische Reaktionsraum in $\ker E$ enthalten sein, was als Zusatzforderung bei seiner Bestimmung benutzt werden kann.

6.6.4.2 Vergleich mit bestehenden Ansätzen

Der methodische Unterschied zwischen der vorgestellten Bilanzierung mittels Erhaltungsgleichungen und dem von Roels vorgeschlagenen Bilanzierungsansatz [Roe83] wird deutlich: Da in der Systematik von Roels von einer globalen Wachstumsgleichung ausgegangen wird, ist der Begriff des theoretischem Reaktionsraums (mangels Annahmen über den Reaktionsmechanismus) nicht definierbar. Es muß daher mit dem stöchiometrischen Reaktionsraum gearbeitet werden, der — wie oben gezeigt — im allgemeinen weniger Information enthält.

Beim Pathway-Modeling treten diese Schwierigkeiten nicht auf, da von vornherein mit einer bekannten Stöchiometrie gerechnet wird. Allerdings wird hier die beschriebene Trennung der Methoden zwischen der Stoffwechselmodellierung und der Bilanzierung nicht klar: Theoretischer und stöchiometrischer Reaktionsraum sind in dieser Situation meist identisch, da genügend viele Erhaltungsgrößen angenommen werden (d.h. $t = n - m$). Dies erschwert eine praktische Anwendung.

Die hier vollzogene begriffliche Trennung zwischen theoretischem und stöchiometrischem Reaktionsraum ist im wesentlichen neu und umfaßt die in der Literatur beschriebenen Ansätze. Dabei wird den praktisch auftretenden Gegebenheiten Rechnung getragen.

6.6.4.3 Modelle mit variabler Stöchiometrie

In den Abschnitten 5.2.6.5 und 5.2.7.3 wurde bereits auf die Problematik von Modellen des Typs

$$\dot{X} = N(X) \cdot X_1 \cdot r(X)$$

mit einer zustandsabhängigen stöchiometrischen Matrix eingegangen. Solche Systeme treten bei Variable-Yield-Modellen und strukturierten Modellen auf.

Es wird unmittelbar klar, daß in dieser Situation der Begriff des theoretischen Reaktionsraums nicht mehr anwendbar ist, da der Raum

$$\operatorname{im} N(X)$$

nun — meist auf unbekannte Weise — zustandsabhängig ist. Ganz analoges gilt für den empirischen Reaktionsraum.

Dagegen bleibt die Beziehung

$$E \cdot \dot{X} = 0 \implies E \cdot N(X) = 0$$

für Erhaltungsgrößen weiterhin richtig. Es bestehen folglich lineare Abhängigkeiten zwischen den zustandsabhängigen stöchiometrischen Koeffizienten in $N(X)$. Ein Beispiel hierfür wurde bereits in Abschnitt 6.6.2.2 gegeben. Das Hauptproblem bei der Anwendung der stöchiometrischen Bilanzierung bleibt allerdings stets die Auffindung geeigneter praktisch auswertbarer Erhaltungsgrößen.

6.6.5 Vorgehensweise im offenen System

In offenen Systemen findet ein Transport einzelner Komponenten über die Systemgrenzen hinaus statt (z.B. im kontinuierlichen oder Fed-Batch-Reaktor). Die Bilanzierung unterscheidet sich in dieser Situation nicht wesentlich von der im abgeschlossenen System. Dies soll am Beispiel der kontinuierlichen Kultur mit nicht gasförmigen Komponenten kurz illustriert werden [15]:
Die allgemeine Gleichung für die kontinuierliche Kultur ist (vgl. 5.3.3.1):

$$\dot{X} = N \cdot X_1 \cdot r(X) + D \cdot (X_{in} - X) \; .$$

Umformung liefert:

$$\dot{X} + D \cdot (X - X_{in}) = N \cdot X_1 \cdot r(X)$$

Auf der linken Seite stehen jetzt nur noch Größen, die bekannt sind (Einlaufkonzentrationen) oder ohnehin zur Bestimmung der Ableitungen $\dot{X}$ benötigt werden (Stoffkonzentrationen im Reaktor). Das Ergebnis ist also erneut ein lineares Modell mit der abhängigen Variablen:

$$\dot{X} + D \cdot (X - X_{in})$$

Im stationären Betrieb vereinfacht sich dieser Ausdruck zu:

$$D \cdot (X_{in} - X)$$

Im Unterschied zu den Voraussetzungen der stationären Identifikation werden hier keine Annahmen über eine spezielle Gestalt der Kinetiken $r(X)$ gemacht.

6.7 Sprung-Modelle

6.7.1 Modellstruktur

Zur Detektion von abrupten Veränderungen in Signalen und dynamischen Systemen wird ein Sprung-Modell benötigt. Ein solches Sprungmodell geht von einem Sprungzeitpunkt t_{step} aus und beschreibt das Signal bzw. die Dynamik des Systems vor und nach dem Sprung durch zwei parametrisierte Funktionen $f_<(\alpha_<, t)$ und $f_\geq(\alpha_\geq, t)$. Das Sprungmodell hat dann die Gestalt

$$f(\alpha_<, \alpha_\geq, t_{step}, t) = \begin{cases} f_<(\alpha_<, t) & \text{falls } t < t_{step} \\ f_\geq(\alpha_\geq, t) & \text{falls } t \geq t_{step} \end{cases} + \epsilon$$

mit einem Rauschterm ϵ (vgl. Abb. 6.12). Die Parameter $\alpha_\geq$ werden dabei als unbekannt angenommen, wogegen $\alpha_<$ oft aus der Vorgeschichte des Signals bekannt ist. Entscheidend ist, daß die Funktionen $f_<$ und $f_\geq$ nicht von t_{step} abhängig sein dürfen und für alle t definiert sind. Dies macht einen Vergleich des Modells ohne Sprung ($t_{step} = \infty$) einerseits und der Modelle mit Sprung ($t_{step} < \infty$) andererseits mit den realen Meßdaten möglich. Mit Hilfe eines Entscheidungskriteriums kann dann zunächst entschieden werden, ob überhaupt ein Sprung vorliegt, bevor Zeitpunkt und Parameter des Sprungs geschätzt werden. Das Sprungmodell ist also mit dem nichtlinearen Regressionsmodell verwandt.

[15] Die Vorgehensweise ist leicht auf andere Situationen übertragbar.

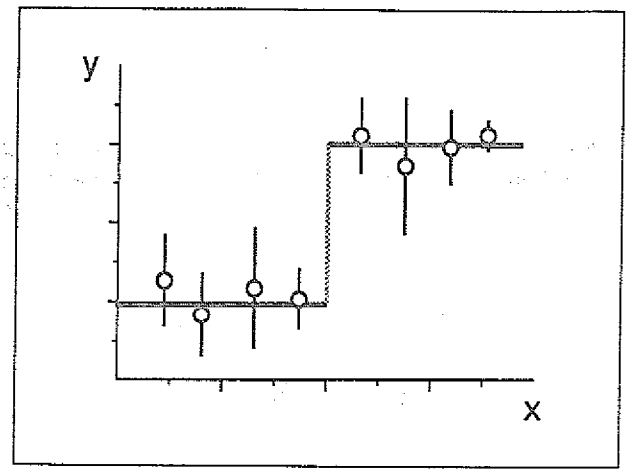


Abbildung 6.12: Sprungmodell

Für die praktische Anwendung ist wichtig, daß der abschnittsweise lineare Ansatz

$$f(\alpha_<, \alpha_\geq, t_{step}, t) = \begin{cases} \alpha_{<,1} \cdot f_{<,1}(t) + \ldots + \alpha_{<,n} \cdot f_{<,n}(t) & (t < t_{step}) \\ \alpha_{\geq,1} \cdot f_{\geq,1}(t) + \ldots + \alpha_{\geq,n} \cdot f_{\geq,n}(t) & (t \geq t_{step}) \end{cases} + \epsilon$$

noch in einer expliziten Form numerisch behandelt werden kann, obwohl $f(\alpha_<, \alpha_\geq, t_{step}, t)$ nichtlinear in t_{step} ist (siehe 10.2.4 und 10.5.3).

6.7.2 Beispiele

6.7.2.1 Ausfall eines Meßgerätes

Der Ausfall eines Meßgerätes kann durch ein sprunghaftes Absinken des Meßsignals (meist auf den Wert 0) gekennzeichnet sein:

$$f(t_{step}, t) = \begin{cases} \alpha_< & (t < t_{step}) \\ 0 & (t \geq t_{step}) \end{cases} + \epsilon$$

Dies ist einer der seltenen Fälle, wo sowohl der Parameter $\alpha_<$ vor dem Sprung, als auch der Parameter $\alpha_\geq = 0$ nach dem Sprung als bekannt vorausgesetzt werden können.

6.7.2.2 Übergang von der Lag-Phase in die exponentielle Wachstumsphase

Viele Mikroorganismen vollziehen in einer Batch-Kultur den Übergang von der Lag-Phase (Wachstum $\mu = 0$) in die exponentielle Phase (Wachstum $\mu = \text{const.}$) nahezu sprunghaft, nachdem die für das Wachstum erforderlichen Enzyme synthetisiert worden sind. Eine approximative Beschreibung des Wachstums ist dann

$$\dot{X} = \mu \cdot X$$

mit dem Sprungmodell

$$\mu(\mu_\geq, t_{step}, t) = \begin{cases} 0 & (t < t_{step}) \\ \mu_\geq & (t \geq t_{step}) \end{cases} + \epsilon.$$

Hier ist der Parameter $\mu_\geq$ nach dem Sprung unbekannt und muß zusätzlich geschätzt werden (siehe Kapitel 10).

Kapitel 7

Stochastische Modelle

Seit der Einführung der Kalman-Filter-Technik in das biotechnische Methodenspektrum [SS84] sind die durch einen Systemrauschterm erweiterten stochastischen Zustandsraummodelle bekannt. Die Konsequenzen einer solchen stochastischen Sichtweise des biologischen Systems werden erst dann deutlich, wenn zu einer entspechenden stochastischen Modellierung und Simulation übergegangen wird. Dieser Schritt wurde bisher nicht vollzogen. Die theoretischen Grundlagen der stochastischen Modellierung sollen deshalb kurz vorgestellt werden (siehe 7.1, 7.2). Die praktische Anwendung für eine realistische Systemsimulation steht im Vordergrund (vgl. 7.3 7.4). Die vorgestellten Visualisierungsmethoden tragen hierbei zum Verständnis bei. Insbesondere ergibt sich auf dem Wege der Simulation ein anschaulicher Zugang zur Theorie des Kalman-Filters (siehe 11.3).

7.1 Stochastische Prozesse

7.1.1 Informelle Definition eines stochastischen Prozesses

Ein *stochastischer Prozeß* ist eine Funktion der Zeit, die zu jedem Zeitpunkt t eine Zufallsvariable oder einen Zufallsvektor

$$X(t)$$

liefert. Ein konkreter Ausgang des Zufallsexperiments, welches durch $X(t)$ beschrieben wird, ist eine einzelne Funktion

$$x(t),$$

eine sogenannte *Realisierung* von $X(t)$. Stochastische Prozesse dienen zur Beschreibung von Zufallsexperimenten, deren Ausgänge Funktionen der Zeit sind.

7.1.2 Veranschaulichung von stochastischen Prozessen

Will man einen stochastischen Prozeß $X(t)$ analog zu einer gewöhnlichen Zufallsvariablen X veranschaulichen, so muß man zwei Aspekte berücksichtigen:

a) Eine *unabhängige Stichprobe* besteht jetzt aus einem Satz von Realisierungen

$$x_1(t), x_2(t), x_3(t), \ldots$$

die alle zusammen in ein Koordinatensystem eingezeichnet werden müssen. Abbildung 7.1 veranschaulicht dies für einen Prozeß in 1 und 2 Dimensionen.

b) Auf der anderen Seite bildet eine solche Folge von Funktionen zu jedem festen Zeitpunkt $t_1, t_2, \ldots$ eine gewöhnliche unabhängige Stichprobe. Deren Verteilung kann dargestellt werden (siehe Abb. 7.1) durch

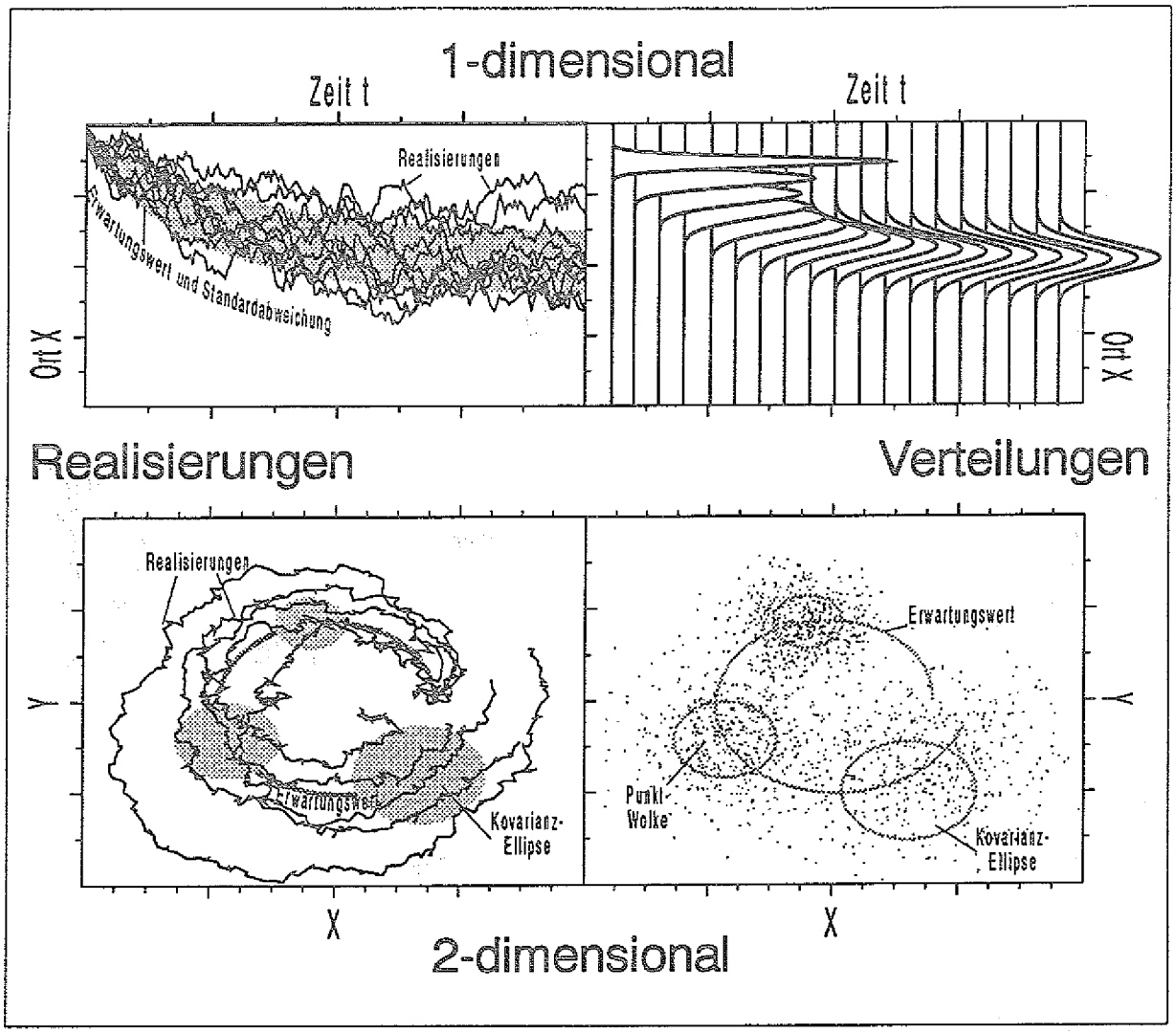


Abbildung 7.1: Visualisierung von stochastischen Prozessen

- die Wahrscheinlichkeitsdichte.
- eine Punktwolke.

Die Punktwolkendarstellung entspricht einer stroboskopartigen "Beleuchtung" der Funktionswerte zu den diskreten Zeiten $t_1, t_2, \ldots$. Dabei geht die Verbindung zwischen den Punkten aufeinanderfolgender Punktwolken verloren.

Mit den beiden Veranschaulichungsmethoden wird alternativ der zeitlichen Aspekt

$$\{X_i(t), t > 0\}_{i=1,2,\ldots}$$

oder den räumlichen Aspekt

$$\{X_i(t), i = 1, 2, \ldots\}_{t=t_1,t_2,\ldots}$$

des Prozesses betont.

7.1.3 Wahrscheinlichkeits-Dichte und Momente stochastischer Prozesse

Bei einem stochastischen Prozeß $X(t)$ sind

$$f(t,\xi),\ \mathrm{E}[X(t)] \text{ und } \mathrm{Cov}[X(t)]$$

Funktionen der Zeit. Sie können durch Kurven, bzw. eine Folge von Kovarianzellipsen oder Punktwolken veranschaulicht werden (vgl. Abb. 7.1).

7.2 Stochastische Differentialgleichungen

7.2.1 Definition von stochastischen Differentialgleichungen

7.2.1.1 Naive Definition

Die naive Definition einer stochastischen Differentialgleichung erweitert eine gewöhnliche Differentialgleichung

$$\dot{x} = a(t,x)$$

um einen "rein zufälligen" normalverteilten Rauschvektor $e(t)$ mit der Normierung

$$\mathrm{E}[e] = 0,\ \mathrm{Cov}[e] = 1$$

und eine möglicherweise zustandsabhängige Skalierungsmatrix

$$\sqrt{b(t,x)} \quad (b(t,x) = b(t,x)^T,\ b(t,x) \geq 0)$$

zu einer Gleichung vom Typ:

$$\dot{x} = a(t,x) + \sqrt{b(t,x)} \cdot e(t)$$

Durch die Verwendung der Wurzel wird erreicht, daß (vgl. 6.1):

$$\mathrm{Cov}[\sqrt{b(t,x)} \cdot e] = \sqrt{b(t,x)} \cdot \mathrm{Cov}[e] \cdot \sqrt{b(t,x)}^T = b(t,x)$$

Der Ausdruck "rein zufällig" besagt, daß $e(t)$ ein *Weißes Rauschen* ist. D.h.

$$e(t+\Delta t) \text{ und } e(t) \text{ sind unabhängig für beliebig kleine } \Delta t$$

Charakteristisch für ein Weißes Rauschen ist somit die Nicht-Vorhersagbarkeit in die Zukunft.

Bemerkung: Die Beschränkung auf normalverteilte Rauschterme $e(t)$ erscheint an dieser Stelle willkürlich. Die Theorie zeigt jedoch, daß die Normalverteilung die einzige Verteilung ist, die zu stetigen Lösungen $x(t)$ führt [Gar85]. Nicht normalverteilte Rauschterme erzeugen Sprünge im Zeitverhalten [1], die für die Modellierung der meisten natürlichen Systeme unerwünscht sind.

7.2.1.2 Der Wiener-Prozeß

Leider führt die naive Definition einer stochastischen Differentialgleichung zu mathematischen Widersprüchen, die sich je nach Zugang verschieden manifestieren:

a) Beim frequenzanalytischen Zugang stellt man fest, daß an einem Weißen Rauschen $e(t)$ alle Frequenzen ν mit derselben Energie beteiligt sind (konstantes Spektrum) [Pri81] [2]. Folglich ist die Gesamtenergie des Rauschens gegeben durch

$$\mathrm{Var}[e(t)] = \infty.$$

[1] In [Gar85] findet sich ein Simulationsbeispiel für ein Cauchy-verteiltes Rauschen.

[2] Übertragen auf das Licht ergäbe sich die Farbe Weiß. Daher der Name "Weißes Rauschen".

b) Bei dem Versuch, den sogenannten *Wiener-Prozeß* $W(t)$ mit

$$\dot{W}(t) = e(t)$$

aus $e(t)$ zu konstruieren, stellt man fest, daß die einzig denkbare Lösung für $W(t)$ überhaupt nicht differenzierbar ist [Gar85]!

Immerhin gelingt es, beim Zugang b), einen Prozeß $W(t)$ zu konstruieren, der folgende Eigenschaften hat [Pri81]:

- $W(t)$ hat stetige, aber nicht differenzierbare Realisierungen
- Die Zuwächse

$$\Delta W(t) = W(t) - W(t - \Delta t)$$

sind für beliebiges Δt normalverteilt mit Erwartungswert

$$\mathrm{E}[\Delta W(t)] = 0$$

und Kovarianzmatrix

$$\mathrm{Cov}[W(t)] = \Delta t.$$

- Je zwei Zuwächse

$$\Delta W(t_1) \quad \text{und} \quad \Delta W(t_2) \qquad (t_1 \neq t_2)$$

sind voneinander unabhängig.

Obwohl das Weiße Rauschen nicht existiert, gibt es somit doch einen Prozeß $W(t)$, der fast alle Eigenschaften einer Lösung von

$$\dot{W} = e \quad \Leftrightarrow \quad W(t) = \int_0^t e(\tau)\, d\tau$$

hat — mit Ausnahme der Differenzierbarkeit! Es gilt die (naive) Merkregel:

$$dW = e \cdot dt$$

7.2.1.3 Exakte Definition

Die Erfahrung mit dem Wiener-Prozeß legt nahe, die Lösung der stochastischen Differentialgleichung

$$\dot{X} = a(t, X) + \sqrt{b(t, X)} \cdot e(t)$$

zu *definieren* als die Lösung der Integralgleichung

$$X(t) - X(0) = \int_0^t a(\tau, X(\tau))\, d\tau + \int_0^t \sqrt{b(\tau, X(\tau))}\, dW(\tau).$$

Im mehrdimensionalen Fall ist dW entsprechend als ein Vektor von $n = \dim X$ voneinander unabhängigen Wiener-Prozessen zu lesen.

Eine alternative Formulierung der stochastischen Differentialgleichung, die — je nach Kontext — große Vorzüge hat, ist die differentielle Schreibweise:

$$dX = a(t, X)\, dt + \sqrt{b(t, X)}\, dW$$

mit dem Wiener-Prozeß $W(t)$. Im folgenden sollen beide Schreibweisen nebeneinander verwendet werden.

Bemerkung: Bei näherer Betrachtung stellt sich heraus, daß die Integraldefinition immer noch einer Präzisierung bedarf. Im Gegensatz zur Integration gewöhnlicher Funktionen $x(t)$, für die im wesentlichen nur ein Integralbegriff existiert [3], ist ein Integralbegriff für stochastische Funktionen $x(t)$ nicht eindeutig definierbar!

Bei der exakten Definition einer stochastischen Differentialgleichung ist es zur Konvention geworden, das sogenannte *Ito-Integral* zugrunde zu legen. Es beruht im Prinzip auf der obigen differentiellen Formulierung und einem geeigneten Grenzübergang $dt \to 0$. Die Existenz und Eindeutigkeit der Lösungen kann auf der Grundlage dieser Definition (unter gewissen Lipschitz- und Beschränktheitsbedingungen an $a(t,X)$ und $b(t,X)$) nachgewiesen werden kann [Arn74].

Die exakte Definition des Ito-Integrals sowie seine Beziehung zum *Stratonovich-Integral* kann z.B. in [Arn74] und [Gar85] nachgelesen werden.

7.2.1.4 Der Ito-Kalkül

Das Rechnen mit Ito-Integralen, die den Wiener-Prozeß enthalten, hat seine ganz eigenen Rechenregeln, was sich vor allem bei der Berechnung von Erwartungsvektoren und Kovarianzmatrizen für $X(t)$ bemerkbar macht. Die wichtigsten Rechenregeln des Ito-Kalküls sind (unter gewissen Bedingungen an die Funktionen $G(\tau), H(\tau)$ [4] [Gar85]:

$$\begin{aligned}
\mathrm{E}\left[\int_0^t G(\tau)\,dW(\tau)\right] &= 0 \\
\mathrm{E}\left[\int_0^t G(\tau)\,d\tau\right] &= \int_0^t \mathrm{E}\left[G(\tau)\right]d\tau \\
\mathrm{E}\left[\int_0^t G(\tau)\,dW(\tau)\cdot\int_0^t H^T(\tau)\,d\tau\right] &= 0 \\
\mathrm{E}\left[\int_0^t G(\tau)\,dW(\tau)\cdot\int_0^t H^T(\tau)\,dW^T(\tau)\right] &= \int_0^t \mathrm{E}\left[G(\tau)H^T(\tau)\right]d\tau
\end{aligned}$$

Mit einiger Vorsicht kann man die Merkregeln:

$$\mathrm{E}\left[dW(\tau)\right]=0,\quad \mathrm{E}\left[dW(\tau)\,d\tau\right]=0,\quad \mathrm{E}\left[dW^2(\tau)\right]=d\tau \quad \text{und} \quad \mathrm{E}\left[dW^n(\tau)\right]=0 \text{ für } n>2$$

formulieren.

7.2.2 Dichtefunktionen, Erwartungsvektoren und Kovarianzmatrizen bei stochastischen Differentialgleichungen

7.2.2.1 Die Fokker-Planck-Gleichung

Zu jedem Zeitpunkt t hat der Zufallsvektor $X(t)$ mit

$$\dot{X}(t) = a(t,X) + \sqrt{b(t,X)}\,e(t)$$

eine Wahrscheinlichkeitsdichte

$$f(t,\xi),\quad \xi \in \Re^n.$$

Wird eine Anfangsverteilung $f(0,x)$ für den Prozeß $X(t)$ vorgegeben, so kann die zeitliche Entwicklung der Dichtefunktion durch eine partielle Differentialgleichung beschrieben werden. Dies ist die *Fokker-Planck-Gleichung* [Gar85]:

$$\frac{\partial f(t,X)}{\partial t} = -\sum_i \frac{\partial}{\partial x_i}[a_i(t,x)\cdot f(t,x)] + \frac{1}{2}\sum_{i,j}\frac{\partial^2}{\partial x_i \partial x_j}[b_{ij}(t,x)\cdot f(t,x)]$$

Sie setzt sich aus einer

[3] Die Integralbegriffe von Rieman, Riemann-Stieltjes und Lebesque führen bei "nicht-pathologischen" Funktionen alle zum selben Ergebnis. Die Realisierungen eines Wiener-Prozesses, die stetig, aber nirgends differenzierbar sind, gehören allerdings zu den "pathologischen Funktionen"!

[4] Die Funktionen dürfen *nicht antizipierend* sein. D.h. sie dürfen nicht "die Zukunft vorwegnehmen" [Arn74].

Drift-Komponente, die dem deterministischen Anteil $a(t,x)$ der Gleichung entspricht, und einer

Diffusions-Komponente, die dem stochastischen Anteil $b(t,x)$ entspricht,

zusammen. Mit Hilfe der Fokker-Planck-Gleichung hat man alle Informationen zur Vorhersage des zukünftigen Systemverhaltens in der Hand. Die Gleichung kann leider aufgrund der enormen Zeitkomplexität des numerischen Problems nicht on-line gelöst werden. D.h. sie ist für den Einsatz bei der Prozeßkontrolle unbrauchbar.

7.2.2.2 Die Matrix-Riccati-Differentialgleichung

Es gibt eine Situation, in der die Fokker-Planck-Gleichung explizit gelöst werden kann, nämlich dann wenn:

1. $f(0,\xi)$ normalverteilt ist und
2. die Gleichung linear ist, was bedeutet, daß

$$\begin{array}{rrcl} & a(t,X) & = & A(t)\cdot X \quad \text{mit einer höchstens zeitabhängigen Matrix } A(t) \\ \text{und} & b(t,X) & = & B(t) \quad \text{mit einer höchstens zeitabhängigen symmetrischen, positiv semi-definiten Matrix } B(t). \end{array}$$

In diesem Fall kann gezeigt werden, daß $f(t,\xi)$ zu allen Zeiten t normalverteilt bleibt. Es genügt daher, $X(t)$ durch Erwartungsvektor und Kovarianzmatrix zu charakterisieren. Zunächst leitet man die beiden Integralgleichungen

$$\begin{array}{rcl} \mathrm{E}[X(t)-X(0)] & = & \mathrm{E}\left[\int_0^t A(\tau)\cdot X(\tau)\,d\tau\right] + \mathrm{E}\left[\int_0^t \sqrt{B(\tau)}\,dW(\tau)\right] \\ \mathrm{Cov}[X(t)] & = & \mathrm{Cov}\left[\int_0^t A(\tau)\cdot X(\tau)\,d\tau \cdot \int_0^t X^T(\tau)\cdot A^T(\tau)\,d\tau\right] \\ & + & \mathrm{Cov}\left[\int_0^t A(\tau)\cdot X(\tau)\,d\tau \cdot \int_0^t X^T(\tau)\cdot \sqrt{B(\tau)}\,dW(\tau)^T\right] \\ & + & \mathrm{Cov}\left[\int_0^t \sqrt{B(\tau)}\cdot X(\tau)\,dW(\tau) \cdot \int_0^t X^T(\tau)\cdot A(\tau)\,d\tau\right] \\ & + & \mathrm{Cov}\left[\int_0^t \sqrt{B(\tau)}\cdot X(\tau)\,dW(\tau) \cdot \int_0^t X^T(\tau)\cdot \sqrt{B(\tau)}\,dW(\tau)^T\right] \end{array}$$

ab und daraus mit Hilfe des des Ito-Kalküls (vgl. 7.2.1.4) und Differentiation:

$$\begin{array}{rcl} \frac{d}{dt}\mathrm{E}[X(t)] & = & A(t)\cdot \mathrm{E}[X(t)] \\ \frac{d}{dt}\mathrm{Cov}[X(t)] & = & A(t)\cdot \mathrm{Cov}[X(t)] + \mathrm{Cov}[X(t)]\cdot A(t)^T + B(t) \end{array}$$

Dies ist die *Matrix-Riccati-Differentialgleichung* zu einer linearen stochastischen Differentialgleichung [5].

Bemerkung: Die Ungültigkeit der Matrix-Riccati-Differentialgleichung für den Fall eines zustandsabhängigen Meßrauschens $b(t,X)$ kann sehr gut an dem elementaren Beispiel

$$\dot{X} = X\cdot e$$

illustriert werden. Die Abbildung 7.2 zeigt einige Realisierungen dieses Prozesses, der sich fundamental von einem linearen Prozeß (vgl. Abb. 7.3, 7.4, 7.5) unterscheidet. Der Prozeß hat unsymmetrische Verteilungen $f_x(t,\xi)$!

[5] Im Zusammenhang mit dem kontinuierlichen Kalman-Filter und dem Minimalvarianz-Regler kennt man eine weitere (nichtlineare) Form der Gleichung (vgl. [BC85], [CC86]).

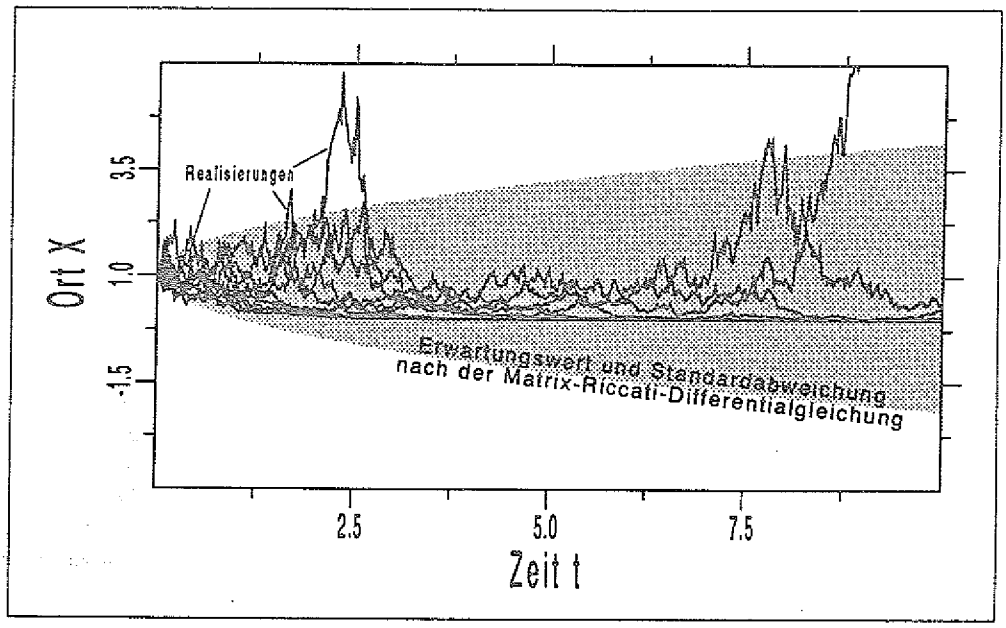


Abbildung 7.2: Lösung einer stochastischen Differentialgleichung mit zustandsabhängigem Rauschterm.

7.2.2.3 Der nichtlineare Fall

Nähert man eine nichtlineare stochastische Differentialgleichung

$$\dot{X} = a(t,X) + \sqrt{b(t,X)}\,e$$

durch ihre Linearisierung (vgl. auch 9.2.3.1)

$$\dot{X} \approx a(t,X^*) + \frac{\partial}{\partial X}a(t,X)_{|X=X^*} \cdot (X - X^*) + \sqrt{b(t,X^*)}\,e$$

um einen Arbeitspunkt X^* an, so kann man für einen kleinen Zeitraum Δt das Systemverhalten approximativ voraussagen, wenn gilt:

> $f(t,\xi)$ ist in der Umgebung von X^* durch eine Normalverteilung approximierbar, deren Erwartungsvektor in der Nähe von X^* liegt.

Eine formale Präzisierung dieser Aussage findet sich in [Gar85]. Auf derartigen Approximationen beruht die nichtlineare Filtertheorie. Analoge Annahmen für die Approximation müssen für den Fall eines zustandsabhängigen Systemrauschens gemacht werden.

7.2.3 Eigenschaften der Matrix-Riccati-Differentialgleichung

Einige allgemeine Eigenschaften der linearen Matrix-Riccati-Differentialgleichung

$$\dot{P} = A(t) \cdot P + P \cdot A^T(t) + B(t)$$

sollen kurz zusammengefaßt werden. Sie sind nützlich zur Prüfung numerischer Programme anhand von Zahlenbeispielen.

7.2.3.1 Linearität

Die Gleichung ist linear in P. In der Dimension 2 soll dies kurz vorgerechnet werden. Hier schreibt man:

$$
\begin{aligned}
&\begin{pmatrix} \dot{P}_{11} & \dot{P}_{12} \\ \dot{P}_{21} & \dot{P}_{22} \end{pmatrix} \\
= &\begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} \cdot \begin{pmatrix} P_{11} & P_{12} \\ P_{21} & P_{22} \end{pmatrix} + \begin{pmatrix} P_{11} & P_{12} \\ P_{21} & P_{22} \end{pmatrix} \cdot \begin{pmatrix} A_{11} & A_{21} \\ A_{12} & A_{22} \end{pmatrix} + \begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix} \\
= &\begin{pmatrix} 2A_{11}P_{11} + A_{12}P_{12} + A_{12}P_{21} + B_{11} & A_{11}P_{12} + A_{12}P_{22} + A_{21}P_{11} + A_{22}P_{12} + B_{12} \\ A_{11}P_{21} + A_{12}P_{22} + A_{21}P_{11} + A_{22}P_{21} + B_{21} & A_{21}P_{12} + A_{21}P_{21} + 2A_{22}P_{22} + B_{22} \end{pmatrix}
\end{aligned}
$$

Oder:

$$
\begin{pmatrix} \dot{P}_{11} \\ \dot{P}_{12} \\ \dot{P}_{21} \\ \dot{P}_{22} \end{pmatrix} = \begin{pmatrix} 2A_{11} & A_{12} & A_{12} & 0 \\ A_{21} & A_{11} + A_{22} & 0 & A_{12} \\ A_{21} & 0 & A_{11} + A_{22} & A_{12} \\ 0 & A_{21} & A_{21} & 2A_{22} \end{pmatrix} \begin{pmatrix} P_{11} \\ P_{12} \\ P_{21} \\ P_{22} \end{pmatrix} + \begin{pmatrix} B_{11} \\ B_{12} \\ B_{21} \\ B_{22} \end{pmatrix}
$$

Diese Gestalt der Matrix-Riccati-Differentialgleichung liegt im Prinzip der numerischen Lösung zugrunde, da $n \times n$-Matrizen rechnerintern als n^2-Vektoren in aufeinanderfolgenden Speicherzellen abgelegt werden. Aus diesem Grunde muß auch nicht eigens ein Differentialgleichungs-Löser für Matrix-Gleichungen programmiert werden. Der Rechenaufwand wächst mit n^2.

7.2.3.2 Symmetrie

Da die rechte Seite der Riccati-Gleichung aufgrund der Symmetrie von P, B eine symmetrische Matrix darstellt —

$$(A\,P + P\,A^T + B)^T = P\,A^T + A\,P + B \quad \text{falls} \quad P = P^T \text{ und } B = B^T$$

— ist $P(t)$ zu allen Zeiten symmetrisch, wenn nur der Startwert $P(0)$ symmetrisch ist. Die Komplexität der Riccati-Gleichung reduziert sich dann unter Ausnutzung der Symmetriebeziehungen

$$P_{ij} = P_{ji} \quad \text{und} \quad B_{ij} = B_{ji}$$

auf $O\left(\frac{1}{2}n(n+1)\right)$. Z.B. im Fall $n = 2$:

$$
\begin{pmatrix} \dot{P}_{11} \\ \dot{P}_{12} \\ \dot{P}_{22} \end{pmatrix} = \begin{pmatrix} 2A_{11} & 2A_{12} & 0 \\ A_{21} & A_{11} + A_{22} & A_{12} \\ 0 & A_{21} & 2A_{22} \end{pmatrix} \begin{pmatrix} P_{11} \\ P_{12} \\ P_{22} \end{pmatrix} + \begin{pmatrix} B_{11} \\ B_{12} \\ B_{22} \end{pmatrix}
$$

Für die numerische Lösung ist diese Darstellung nicht unbedingt sinnvoll (vgl. 11.3.5, [CC86],[Sch75]).

7.2.3.3 Lösung der homogenen Gleichung

Die Lösung der homogenen Gleichung

$$\dot{P} = A(t)P + PA^T(t)$$

kann aus den Lösungen des Systems

$$\dot{x} = A(t)x$$

gewonnen werden. Dazu konstruiere man die *Fundamentalmatrix* [Wal76]

$$X = (x^1, \ldots, x^n)$$

aus n linear unabhängigen Lösungen x^i von $\dot{x} = A(t)x$. Für diese gilt:

$$\dot{X} = A(t) \cdot X$$

Es folgt:

$$(X\,\dot{X}^T) = \dot{X}\,X^T + X\,\dot{X}^T = A \cdot (X\,X^T) + (X\,X_T) \cdot A^T$$

Die homogene Gleichung ist also ausgehend von $P(0)$ mit $P(0) = P(0)^T$, $P(0) \geq 0$ wie folgt zu lösen (vgl. [BC85]):

Man setze	$X(0)$	$=$	$\sqrt{P(0)}$
und löse mit diesem Startwert die Matrix-Gleichung	$\dot{X}$	$=$	$A(t) \cdot X$.
Dann löst	$P(t)$	$=$	$X(t)\,X(t)^T$
die homogene Gleichung	$\dot{P}$	$=$	$A(t)P + PA^T(t)$.

Für die inhomogene Gleichung kann man dann die Methode der Variation der Konstanten ansetzen [Wal76].

7.2.3.4 Stabilität

Da eine inhomogene zeitinvariante lineare Differentialgleichung genau dann stabil ist, wenn das homogene System stabil ist, folgt aus der obigen expliziten Lösung des homogenen Systems sofort:

$$\dot{x} = A \cdot x \text{ stabil} \iff \dot{P} = AP + PA^T + B \text{ stabil}$$

Die asymptotische Kovarianzmatrix berechnet man aus

$$AP_\infty + P_\infty A^T = -B\ .$$

Bereits bei zeitabhängigen linearen Systemen kann kann das Stabilitätsproblem nicht mehr allgemein gelöst werden (vgl. [BC85]).

7.3 Einige Beispiele

Für die Arbeit mit stochastischen Prozessen ist die Kenntnis einiger grundlegender Beispiele, die immer wieder als Grundbausteine für die Konstruktion stochastischer Modelle gebraucht werden, unumgänglich. Bei diesen Beispielen soll zusätzlich die Matrix-Riccati-Differentialgleichung explizit gelöst werden, um ihre Wirkungsweise und Struktur zu verdeutlichen.

7.3.1 Die Brownsche Bewegung

Die *Brownsche Bewegung* ist ein rein diffusiver Prozess ohne Driftanteil. In n Dimensionen wird sie beschrieben durch

$$\dot{X} = \sqrt{B} \cdot e,$$

wobei in der Regel $B = \sigma \cdot 1$ (1 die Einheitsmatrix) angenommen wird (d.h. Diffusion unabhängig in allen Raumrichtungen und mit gleicher Streuung). Für den Fall $B = 1$ liegt exakt der n-dimensionale Wiener-Prozeß vor. Die Abbildung 7.3 zeigt einige typische Realisierungen des Brownschen Prozesses in 1 und 2 Dimensionen.

7.3.1.1 Berechnung des Erwartungsvektors

Die beiden folgenden Ergebnisse sind in Abbildung 7.3 veranschaulicht.

$$\frac{d}{dt}\,\mathrm{E}[X] = 0 \quad \Rightarrow \quad \mathrm{E}[X(t)] = \mathrm{E}[X(0)] = \text{const}$$

Der Schwerpunkt einer Teilchenwolke ändert sich folglich nicht (Drift = 0).

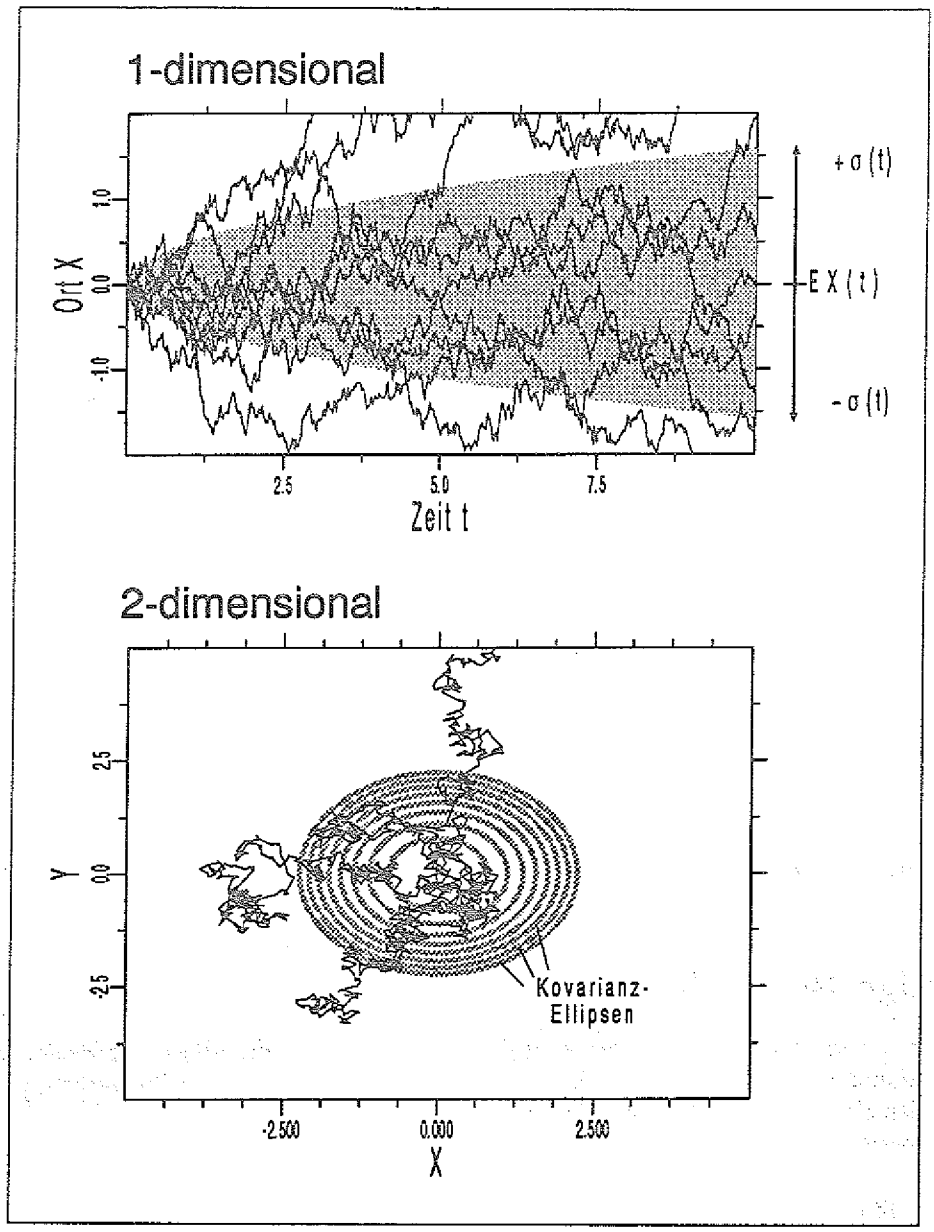


Abbildung 7.3: Brownsche Bewegung in 1 und 2 Dimensionen

7.3.1.2 Berechnung der Kovarianzmatrix

$$\frac{d}{dt}\mathrm{Cov}[X] = B\ B \cdot t + \mathrm{Cov}[X(0)]$$

Für den Fall, daß alle Teilchen der Teilchenwolke zu Anfang in einem Punkt konzentriert sind (also $\mathrm{Cov}[X(0)] = 0$), erfolgt die Verteilung über den Raum mit linear anwachsender Kovarianzmatrix. Da die Kovarianzellipse über die Wurzel mit dieser verknüpft ist, wächst diese nur mit $\sqrt{t}$ an. Die Wurzelbeziehung für die Brownsche Bewegung ist als Ficksches Gesetz bekannt [ALS77].

7.3.2 Linearer Prozeß 1. Ordnung

Der lineare Prozeß 1. Ordnung [6] ist der 1-dimensionale Prozeß

$$\dot{X} = a \cdot X + \sigma \cdot e \text{ mit } a \neq 0.$$

Einige Realisierungen sind in Abbildung 7.4 veranschaulicht.

7.3.2.1 Berechnung des Erwartungswertes

$$\frac{d}{dt} \mathrm{E}[X] = a \cdot \mathrm{E}[X] \quad \Rightarrow \quad \mathrm{E}[X(t)] = \mathrm{E}[X(0)] \cdot e^{at}$$

Das Verhalten ist aus der linearen Theorie gewöhnlicher Differentialgleichungen bekannt:

$a < 0$ Exponentielle Dämpfung, also stabiles Verhalten:
$\mathrm{E}[X(t)] \to 0$
$a > 0$ Exponentielles Anwachsen, also instabiles Verhalten:
$\mathrm{E}[X(t)] = 0$ falls $\mathrm{E}[X(0)] = 0$
$\mathrm{E}[X(t)] \to \pm\infty$ falls $\mathrm{E}[X(0)] \neq 0$

Für $a < 0$ beschreibt die Gleichung das Verhalten eines Stoßdämpfers, der versucht, Stoßeinwirkungen entgegenzuwirken und eine Ruhelage wiederherzustellen.

7.3.2.2 Berechnung der Varianz

$$\frac{d}{dt} \mathrm{Var}[X] = 2a \cdot \mathrm{Var}[X] + \sigma^2 \quad \Rightarrow \quad \mathrm{Var}[X(t)] = \mathrm{Var}[X(0)] \cdot e^{2at} - \frac{\sigma^2}{2a}(1 - e^{2at})$$

Es folgt für

$a < 0$ ein stabiles Verhalten: $\mathrm{Var}[X(t)] \to -\frac{\sigma^2}{2a}$
$a > 0$ ein instabiles Verhalten: $\mathrm{Var}[X(t)] \to \pm\infty$.

Die Konsequenz für den "Stoßdämpfer" ($a < 0$) ist, daß sich eine konstante mittlere Auslenkung von der Ruhelage einstellt. Es gilt:

Für die Stärke der eintreffenden Stöße:
$\sigma \nearrow \Rightarrow \mathrm{Var}[X(\infty)] \nearrow$ ("Feldweg")
$\sigma \searrow \Rightarrow \mathrm{Var}[X(\infty)] \searrow$ ("Autobahn")

und für die Stärke der Dämpfung:
$a \nearrow \Rightarrow \mathrm{Var}[X(\infty)] \searrow$ ("Harte Federung")
$a \searrow \Rightarrow \mathrm{Var}[X(\infty)] \nearrow$ ("Weiche Federung")

7.3.2.3 Bemerkungen zur Korrelationszeit

Zwei lineare Prozesse 1. Ordnung mit

$$a_1, a_2 < 0 \text{ und } \frac{\sigma_1^2}{2a_1} = \frac{\sigma_2^2}{2a_2}$$

verhalten sich bzgl. der Varianz asymptotisch gleich. Der Unterschied liegt in der Konvergenzgeschwindigkeit

$$\mathrm{Var}[X(t)] \to \mathrm{Var}[X(\infty)],$$

[6] Je nach Umfeld auch als Autoregressiver Prozeß 1.Ordnung (AR(1)-Prozeß) [Pri81] oder als Ornstein-Uhlenbeck-Prozeß [Gar85] bezeichnet.

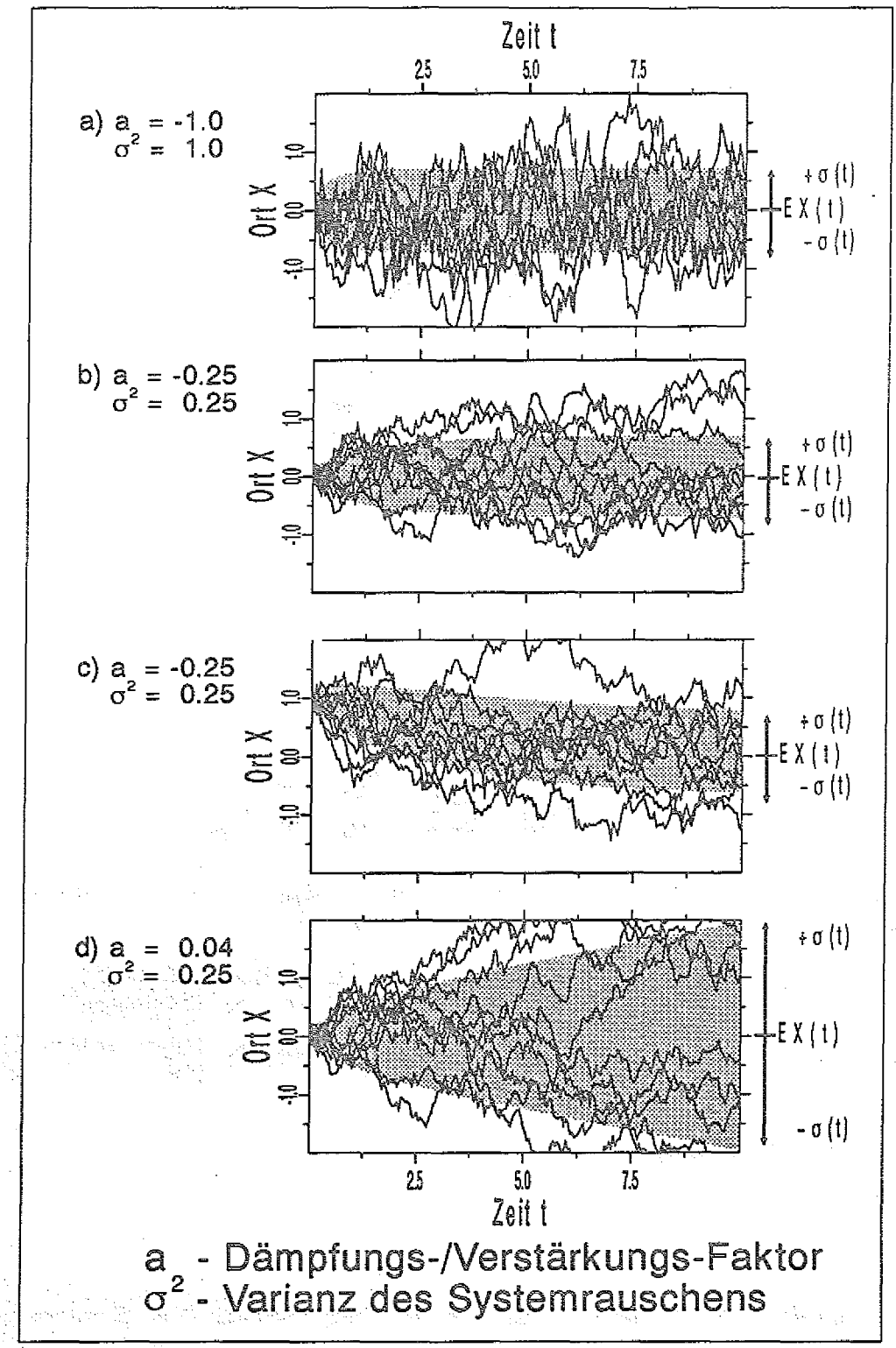


Abbildung 7.4: Linearer Prozeß 1. Ordnung

die mit e^{a_1} bzw. e^{a_2} geht. Der Prozeß mit hoher Dämpfung a hat dabei eine stärkere Fluktuation; die Schwingungen sind hochfrequenter, wie in Abbildung 7.4 zu erkennen ist. Anders ausgedrückt: Mit steigender Dämpfung (aber konstantem Quotienten $\frac{\sigma^2}{2a}$) wird die "Korrelationszeit" Δt, in der etwa

$$X(t) \approx X(t + \Delta t)$$

gilt, immer kürzer. Man könnte sagen:

> "Der Prozeß mit starker Dämpfung nimmt schneller Information auf, verliert aber auch schneller sein Gedächtnis als ein schwach gedämpfter Prozeß."

Formal korrekt drückt man dies mit Hilfe der sogenannten Autokorrelationsfunktion

$$\rho(\Delta t) = \mathrm{Cov}\,[X(t), X(t + \Delta t)]$$

aus [Pri81].

7.3.3 Linearer Prozeß 2.Ordnung

Die Gleichungen für das allgemeine zeitinvariante Problem

$$\dot{x} = A \cdot x + \sqrt{B} \cdot e$$

in 2 Dimensionen haben bereits 12 zu berücksichtigende Parameter (4 Parameter für die Matrix A, 3 für die symmetrische Matrix B, 2 Startwerte für x und 3 Startwerte für $\mathrm{Cov}\,[x]$, so daß selbst für spezielle Systeme die Lösungsterme ganze Seiten füllen können! Solche Beispiele sind in [BH85] oder [Sch75] explizit durchgerechnet. Hier soll nur der leicht zu behandelnde Spezialfall

$$\begin{pmatrix} \dot{x} \\ \dot{y} \end{pmatrix} = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} \sigma & 0 \\ 0 & \sigma \end{pmatrix} \cdot \begin{pmatrix} e_x \\ e_y \end{pmatrix}$$

untersucht werden.

7.3.3.1 Berechnung des Erwartungsvektors

Das deterministische System ist in Form der Schwingungsgleichung

$$\ddot{x} = -x,$$

die man mit $y = \dot{x}$ sofort zu

$$\dot{x} = y \quad \text{und} \quad \dot{y} = \ddot{x} = -x$$

umformt, wohlbekannt. Mit dem speziellen Startvektor

$$(x(0), y(0))^T = (1, 0)^T$$

ist die Lösung der Erwartungsvektor-Gleichung gegeben durch:

$$\mathrm{E}\begin{bmatrix} x(t) \\ y(t) \end{bmatrix} = \begin{pmatrix} \sin t \\ \cos t \end{pmatrix}.$$

Im (x, y)-Zustandsraum präsentiert sich diese Lösung als kreisförmige Bewegung um den Ursprung.

7.3.3.2 Berechnung der Kovarianzmatrix

Unter Ausnutzung der Symmetrie von

$$\mathrm{Cov}\begin{bmatrix} x(t) \\ y(t) \end{bmatrix} = P = \begin{pmatrix} P_{11} & P_{12} \\ P_{21} & P_{22} \end{pmatrix}$$

ergibt sich die Matrix-Riccati-Differentialgleichung in der Gestalt

$$\begin{pmatrix} \dot{P}_{11} \\ \dot{P}_{12} \\ \dot{P}_{22} \end{pmatrix} = \begin{pmatrix} 0 & -1 & 0 \\ -1 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix} \cdot \begin{pmatrix} P_{11} \\ P_{12} \\ P_{22} \end{pmatrix} + \begin{pmatrix} \sigma^2 \\ 0 \\ \sigma^2 \end{pmatrix} .$$

Es soll vom Startwert

$$\mathrm{Cov}\,[P(0)] = 0 \iff P_{11}(0) = 0,\ P_{12}(0) = 0,\ P_{22}(0) = 0$$

ausgegangen werden. Das folgende intuitive Vorgehen führt dann auf einen Lösungsansatz: Das deterministische System führt eine kreisförmige Bewegung aus, die folglich die Eigenschaft hat, Abstände und Flächen zu erhalten. Die Kreisbewegung trägt somit zum Auseinanderdriften einer Punktwolke im Zustandsraum nicht bei. Hierfür sorgt einzig der Diffusionsterm B. Da B wiederum als Diagonalmatrix mit gleichen Einträgen in der Diagonalen gewählt wurde, sind die zugehörigen Kovarianzellipsen kreisförmig. D.h. die diffusive Bewegung erfolgt richtungsunabhängig, also insbesondere invariant unter einer Kreisbewegung. Dies begründet die Hypothese, daß sich die Kovarianzmatrix nicht von der bei einer Brownschen Bewegung unterscheidet:

$$P(t) = B \cdot t = \begin{pmatrix} \sigma^2 & 0 \\ 0 & \sigma^2 \end{pmatrix} \cdot t$$

Eine Probe zeigt, daß die Hypothese korrekt ist. Das Ergebnis in der Veranschaulichung durch Punktwolken und 1Σ-Ellipsen zeigt die Abbildung 7.5 [7].

7.3.3.3 Weitere Beispiele

Sobald man eine flächen- oder winkelverändernde Matrix A oder eine Diffusionsmatrix $B \neq \sigma^2 \cdot 1$ einsetzt, wird die Rechnung ungleich schwieriger. Die Abbildung 7.5 zeigt einige Simulationen für den Fall einer spiralförmigen Bewegung im Zustandsraum ($\alpha < 0$: Bewegung nach innen, $\alpha > 0$: Bewegung nach außen)

$$A = \begin{pmatrix} \alpha & -1 \\ 1 & \alpha \end{pmatrix} \quad \text{sowie} \quad B = \begin{pmatrix} \sigma_x^2 & 0 \\ 0 & \sigma_y^2 \end{pmatrix}.$$

Für den Fall $\alpha < 0$ liegt ein stabiles System vor und die asymptotische Kovarianzmatrix hat die Gleichung (vgl. 7.2.3.2):

$$AP_\infty + P_\infty A^T = B \iff \begin{pmatrix} 2\alpha & -2 & \\ 1 & 2\alpha & -1 \\ & 2 & 2\alpha \end{pmatrix} \begin{pmatrix} P_{11} \\ P_{12} \\ P_{22} \end{pmatrix} = \begin{pmatrix} -\sigma_x^2 \\ 0 \\ -\sigma_y^2 \end{pmatrix}$$

mit der Lösung

$$P = \frac{1}{4\alpha(\alpha^2+1)} \cdot \begin{pmatrix} (2\alpha^2+1)\sigma_x^2 + \sigma_y^2 & \alpha(\sigma_y^2 - \sigma_x^2) \\ \alpha(\sigma_y^2 - \sigma_x^2) & (2\alpha^2+1)\sigma_y^2 + \sigma_x^2 \end{pmatrix}.$$

Die Eigenwerte sind:

$$\lambda_{1/2} = \pm\alpha\sqrt{\alpha^2+1} \cdot (\sigma_x^2 - \sigma_y^2) - (\alpha^2+1)(\sigma_x^2 + \sigma_y^2),$$

d.h. die asymptotischen Kovarianzellipsen verlieren ihre Kreisform für $\sigma_x \neq \sigma_y$ und erhalten eine schiefe Lage im Koordinatensystem. In Abbildung 7.5 ist deutlich die Konvergenz der 1Σ-Ellipsen zu erkennen. Das rechnerische Resultat kann gut als Test für die numerische Programmierung herangezogen werden.

[7] In der Abbildung wird die Kreisförmigkeit der Ellipsen leider durch die verschieden starke Achsenstreckung in x- und y-Richtung wieder zerstört ...

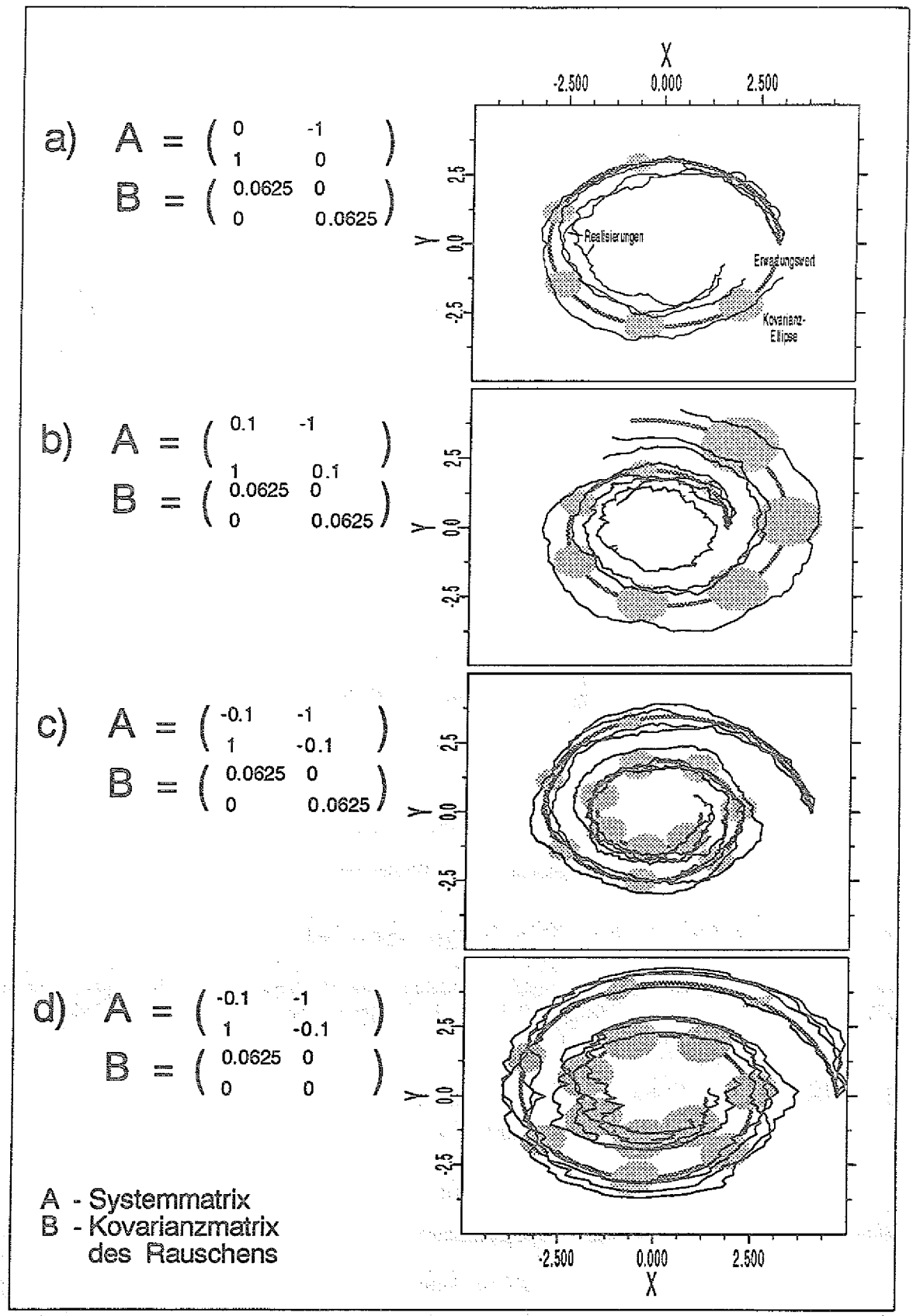


Abbildung 7.5: Linearer Prozeß 2. Ordnung

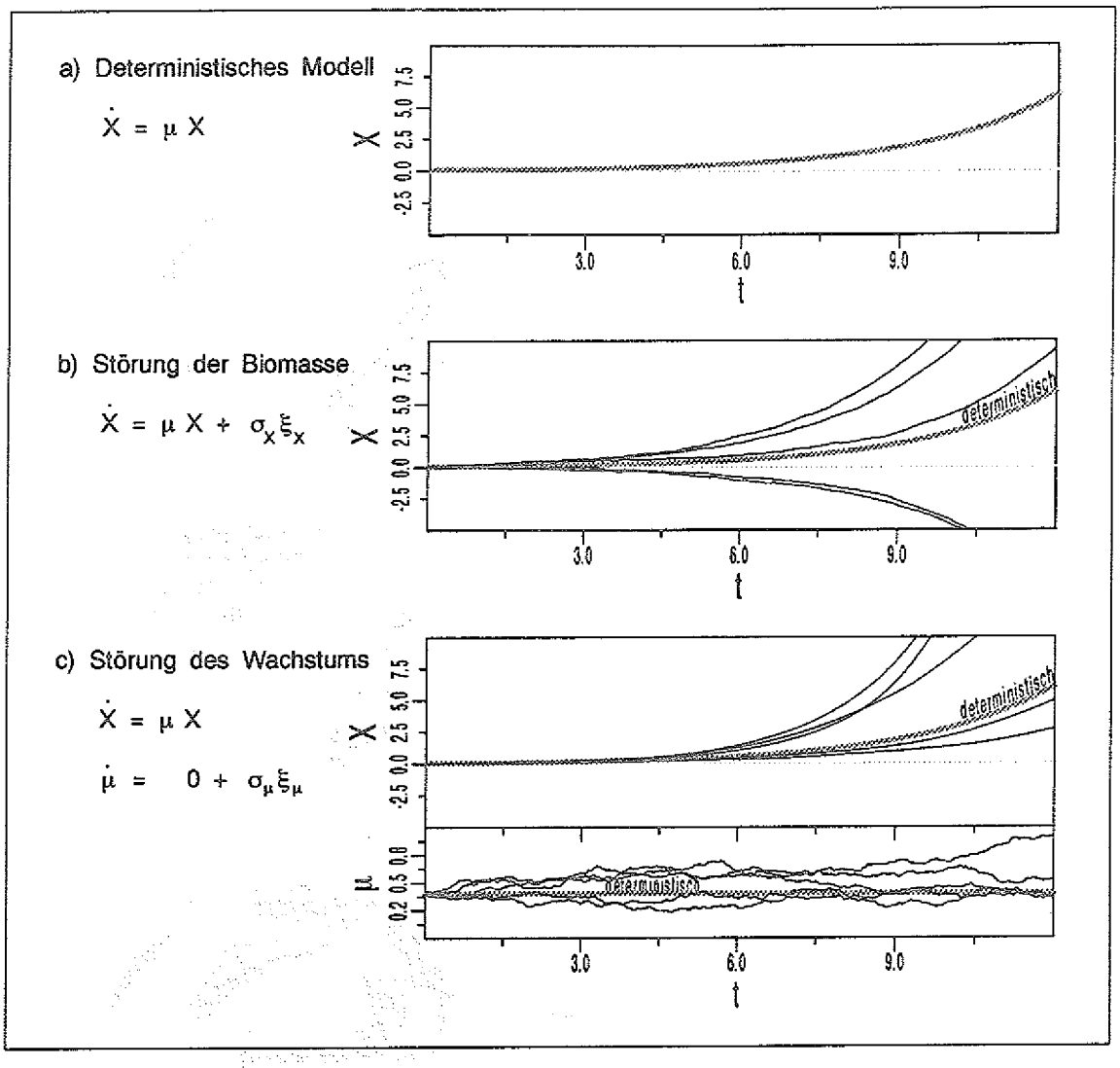


Abbildung 7.6: Stochastisches Wachstumsmodell (Teil 1)

7.4 Ein stochastisches Wachstumsmodell

Als ein Standardbeispiel aus dem biotechnischen Umfeld soll das stochastische Modell für das exponentielle Wachstum herangezogen werden. Das Modell wird schrittweise entwickelt, das Resultat jedes Schritts ist (zusammen mit den verwendeten Parametern) in der Abbildungssequenz 7.6-7.7 wiedergegeben.

7.4.1 Deterministisches Modell

Das deterministische Modell für das exponentielle Wachstum ist gegeben durch

$$\dot{X} = \mu \cdot X.$$

Es führt auf die bekannte exponentielle Wachstumskurve

$$X(t) = X(0) \cdot e^{\mu \cdot t}.$$

In einem realen System unterliegt das Wachstum Schwankungen durch die verschiedensten Störeinflüsse. Es bietet sich daher an, ein stochastisches Modell anzusetzen.

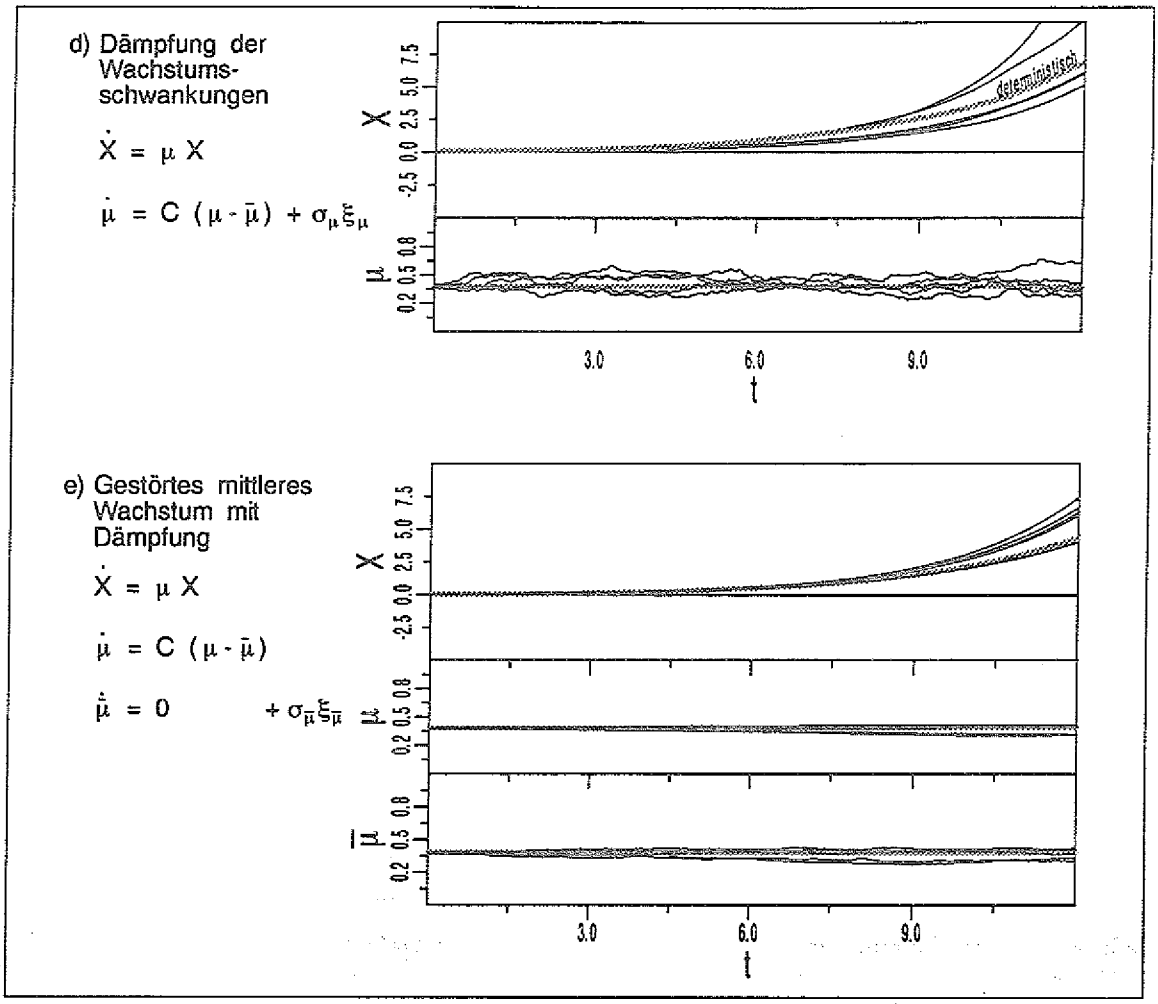


Abbildung 7.7: Stochastisches Wachstumsmodell (Teil 2)

7.4.2 Störung der Biomasse

Die naheliegende Erweiterung des Modells wäre

$$\dot{X} = \mu \cdot X + \sigma_X \cdot e_X .$$

Das Verhalten dieses linearen Modells 1.Ordnung ist aus Abschnitt 7.3.2 bekannt. Mit dem Startwert $\mathrm{Var}[X(0)] = 0$ gilt:

$$\mathrm{E}[X(t)] = X(0) \cdot e^{\mu \cdot t} \quad \text{und} \quad \mathrm{Var}[X(t)] = \frac{\sigma_X^2}{2\mu} \cdot e^{2\mu \cdot t}$$

Die Konsequenz ist eine exponentiell anwachsende Abweichung vom Mittel, wobei die Biomasse in vielen Fällen negativ wird!

Die anschauliche Ursache für die Unbrauchbarkeit des Modells ist folgende: Es beschreibt eine Störung der Biomasse, aber keineswegs eine Störung des Wachstums. Physikalisch gesehen besagt das Modell, daß auf stochastische Weise ständig Biomasse aus dem System entnommen und wieder hinzugefügt wird!

7.4.3 Störung des Wachstums

Um eine Störung des Wachstums zu beschreiben, geht man am einfachsten von dem Differentialgleichungs-Modell

$$\dot{\mu} = 0$$

für konstantes Wachstum aus und stört diese Gleichung. Es entsteht das System

$$\begin{aligned} \dot{X} &= \mu \cdot X + 0 \\ \dot{\mu} &= 0 + \sigma_\mu \cdot e_\mu \ . \end{aligned}$$

Wie die Abbildung zeigt, stellt auch dieses Modell nicht zufrieden: Die Wachstumsrate wird nun als eine Brownsche Bewegung beschrieben, so daß (für $\mathrm{Var}[\mu(0)] = 0$)

$$\mathrm{E}[\mu(t)] = \mu(0) \ \text{ und } \ \mathrm{Var}[\mu(t)] = \sigma_\mu^2 \cdot t$$

und sich somit nach einiger Zeit auch ein negatives Wachstum einstellen kann. Das Modell sagt also noch zu große Wachstumsschwankungen voraus.

7.4.4 Dämpfung der Wachtumsschwankungen

Die Annahme eines "nahezu konstanten" Wachstums, drückt man besser durch einen Prozeß mit asymptotisch konstanter Varianz aus. Hier hilft das schon bekannte "Stoßdämpfermodell" aus 7.3.2:

$$\begin{aligned} \dot{X} &= \mu \cdot X + 0 \\ \dot{\mu} &= a \cdot (\mu - \bar{\mu}) + \sigma_\mu \cdot e_\mu \ . \end{aligned}$$

mit $a < 0$ und der mittleren Wachstumsrate $\bar{\mu}$. Man errechnet

$$\mathrm{E}[\mu(t)] = \bar{\mu} \ \text{ und } \ \mathrm{Var}[\mu(t)] \to \frac{\sigma_\mu^2}{2a}.$$

Die Biomassekurve kommt nun dem tatsächlich zu erwartenden Verhalten schon sehr nahe. Trotzdem ist das Verhalten μ für ein realistisches Modell zu unruhig. Eine weitere Glättung wird erreicht durch Zwischenschalten einer zusätzliche Filterebene, wodurch $\bar{\mu}$ selbst wieder zu einem stochastischen Prozeß wird:

$$\begin{aligned} \dot{X} &= \mu \cdot X + 0 \\ \dot{\mu} &= a \cdot (\mu - \bar{\mu}) + 0 \\ \dot{\bar{\mu}} &= 0 + \sigma_{\bar{\mu}} \cdot e_{\bar{\mu}} \ . \end{aligned}$$

Die zweite Gleichung beschreibt jetzt eine Glättung des Prozesses $\bar{\mu}(t)$ durch Integration. Das entstandene Modell liegt dem erweiterten Kalman-Filter zugrunde, welches Stepanopoulos und San in ihrem grundlegenden Artikel [SS84] zur Wachstumsraten-Schätzung einsetzen [8]. Es fällt auf, daß in der letzten Gleichung die Konstanz von $\bar{\mu}$ wieder verloren gegangen ist. Man könnte dies durch Hinzunahme einer vierten Gleichung erneut auffangen. Die Vorgehnsweise kann iteriert werden. Allerdings wird dabei durch die wiederholten Glättungsoperationen das Verhalten von $\mu(t)$ zunehmend träger.

7.4.5 Bedeutung der stochastischen Modellierung

Die diskutierte Sequenz unterstreicht die Rolle der stochastischen Simulation: Die Bedeutung der auftretenden Gleichungen und Parameter kann so am schnellsten erfaßt werden. Insbesondere verschafft die Simulation ein intuitives Verständnis der stochastischen Vorgänge.

Bei einem Filterentwurf ist dieses Verständnis von zentraler Bedeutung. Genau wie bei einer deterministischen Methode muß auch hierzu ein korrektes Systemmodell vorliegen! Grob kann gesagt werden, daß ein Kalman-Filter die Zustandsgrößen eines Systems genau so schätzen wird, wie es das Modell voraussagt. Für die genannten Modelle heißt das:

[8] Stephanopoulos und San setzen bei einer kontinuierlichen Kultur für die Konstante a die Durchflußrate D ein. Diese Setzung ist dadurch motiviert, daß D die Zeitkonstante des Wachstums entscheidend bestimmt.

- Der erste Ansatz liefert eine erratische Biomasseschätzung,
- der zweite eine unruhige Wachstumsraten-Schätzung und
- die verfeinerten Modelle schließlich immer glattere Verläufe.

Abgesehen vom ersten Ansatz muß entschieden werden, welches Modell für die geplante Anwendung das brauchbarste ist. Die Nachteile der verfeinerten Modelle sind hierbei die größere Zahl von Parametern und die Trägheit der Schätzungen in bezug auf das Folgeverhalten bei schnellen Änderungen der Wachstumsrate (vgl. auch 2.5) [9].

7.5 Stochastische Zustandsraummodelle

7.5.1 Meß- und Systemrauschen

Das stochastische Zustandsraummodell ist das Analogon des deterministischen Zustandsraummodells in der stochastischen Kategorie:

$$\begin{aligned} \dot{x} &= f(t,\alpha,x,u) + \sqrt{r(t,\alpha,x)}\cdot e_x \quad \text{(Systemgleichung)} \\ y &= g(t,\beta,x) + \sqrt{s(t,\beta,x)}\cdot e_y \quad \text{(Meßgleichung)} \end{aligned}$$

Dabei wird davon ausgegangen, daß auf das System und auf die Messung je ein Störsignal einwirkt. e_x und e_y sind voneinander unabhängige Weiße Rauschterme mit

$$\mathrm{E}[e_x] = \mathrm{E}[e_y] = 0 \;\text{ und }\; \mathrm{Cov}[e_x] = \mathrm{Cov}[e_y] = 1 \ .$$

Das Modell wurde bereits in Abbildung 2.4 schematisiert. Es sei darauf hingewiesen, daß das Meßrauschen e_y — im Gegensatz zum normalverteilten Systemrauschen e_x (vgl. 7.2.1.1) — eine beliebige Verteilung haben kann. Die theoretischen Resultate beruhen allerdings in der Regel auf einer Normalverteilungsannahme.

Nach dem was in 7.2.1.2 über die Nichtexistenz des Weißen Rauschens gesagt wurde, muß eine Interpretation der Gleichungen angegeben werden: Die Systemgleichung ist zunächst als stochastische Differentialgleichung im Ito'schen Sinne zu verstehen. Die Meßgleichung — welche den auftretenden Meßfehler beschreiben soll — gewinnt erst in Hinblick auf die Anwendung Sinn: Geht man davon aus, daß eine Abtastung stets nur in diskreten Zeitabständen erfolgen kann, so empfiehlt sich die *semi-diskrete* Notation

$$\begin{aligned} \dot{x} &= f(t,\alpha,x,u) + \sqrt{r(t,\alpha,x)}\cdot e_x \\ y(t_i) &= g(t_i,\beta,x(t_i)) + \sqrt{s(t_i,\beta,x(t_i))}\cdot e_y(t_i) \end{aligned} \qquad i = 0,1,2,\ldots \ .$$

Dabei stellen die t_i die Abtastzeitpunkte dar. In der semi-diskreten Schreibweise ist e_y ein diskretes Weißes Rauschen, das als Folge unabhängiger identisch verteilter Zufallsvariablen durchaus existent ist [10]. Wenn im folgenden von einem stochastisch Zustandsraummodell die Rede ist, so sei damit stets die semi-diskrete Form gemeint.

In der Filtertheorie tritt die Meßgleichungen oft auch in einer rein kontinuierlichen Bedeutung auf, die allein keinen Sinn macht. Allerdings kommen solche kontinuierlichen Gleichungen nur im Kontext der Filterung und Regelung vor. Es tritt hierbei stets eine Rückkopplung der Messung y auf den Zustandsvektor x auf. In dieser Situation läßt sich ein normalverteiltes e_y als weiterer Störterm in der Ito-Gleichung interpretieren (vgl. dazu 11.2.1.2).

[9] Es sollte noch vermerkt werden, daß sich ein Entwurf, bei dem μ negativ werden kann, nicht nachteilig auf das zugehörige Filter auswirkt. Es orientiert sich an den gegebenen Meßdaten und im realen System ist μ stets beschränkt.

[10] Um der Forderung nach Unabhängigkeit von $e_y(t_i)$ und $e_y(t_{i+1})$ praktisch nachzukommen, müssen die Abtastintervalle $\Delta t_i = t_{i+1} - t_i$ groß genug gewählt werden. Es gibt jedoch auch diverse Erweiterungen der Theorie für korrelierte e_y sowie für wechselseitige Korrelationen zwischen e_x und e_y [CC86].

7.5.2 Die Matrix-Riccati-Differentialgleichung bei einem stochastischen Zustandsraummodell

Für das lineare Modell

$$\begin{aligned} \dot{x} &= A \cdot x + B \cdot u &+ \sqrt{R} \cdot e_x \\ y &= C \cdot x &+ \sqrt{S} \cdot e_y \end{aligned}$$

folgen aus den Rechenregeln für Erwartungsvektoren und Kovarianzmatrizen sowie den Unabhängigkeitsforderungen an e_x und e_y die Gleichungen:

$$\begin{aligned} \tfrac{d}{dt} \mathrm{E}[x] &= A \cdot \mathrm{E}[x] + B \cdot u \\ \mathrm{E}[y] &= C \cdot \mathrm{E}[x] \end{aligned}$$

und

$$\begin{aligned} \tfrac{d}{dt} \mathrm{Cov}[x] &= A \cdot \mathrm{Cov}[x] + \mathrm{Cov}[x] \cdot A^T &+ R \\ \mathrm{Cov}[y] &= C^T \cdot \mathrm{Cov}[x] \cdot C &+ S \,. \end{aligned}$$

Sie gelten auch für ein nicht-normalverteiltes Meßrauschen.

7.5.3 Biotechnisches Beispiel

Im Gegensatz zum Systemrauschen, wo stets von einem (hypothetischen) normalverteilten Weißen Rauschen ausgegangen werden muß (vgl. 7.2.1.1), kann im semi-diskreten Modell ohne weiteres eine beliebige Verteilung angenommen werden. Kombiniert man z.B. das soeben vorgestellte Modell für das stochastische Wachstum (7.4) mit dem Modell für die Trübungsmessung (6.3), so entsteht ein stochastisches Zustandsraummodell, dessen explizite Formulierung sich hier erübrigt.

Kapitel 8

Logische und heuristische Modelle

Logische und heuristische Modelle sind nicht Thema dieser Arbeit. Vielmehr wurde versucht, alle die Komponenten des biotechnischen Methodenspektrums, die nicht heuristischer Natur sind, in ein einheitliches Gesamtkonzept zu fassen und damit einer kreativen Verwendung durch den Experimentator zugänglich zu machen. Heuristische und logische Ansätze entstehen dann im Verlauf einer Datenanalyse-Sequenz ganz von selbst, wenn versucht wird, den intuitiven Vorgang der Informationsgewinnung zu formalisieren und schließlich zu automatisieren. Ein heuristisches Modell zur Prozeßüberwachung kann durchaus als eine Simulation des menschlichen Vorgehens bei der Datenanalyse verstanden werden! Es sollen daher lediglich einige der wichtigsten Ansätze angeschnitten werden, um eine Vorstellung davon zu vermitteln, welche Formalismen zu den logischen und heuristischen Modellen gerechnet werden können.

8.1 Aussagenlogische Ausdrücke

Aussagenlogische Ausdrücke stellen die älteste und einfachste Methode zur Formalisierung logischer Zusammenhänge dar. Ein solcher Ausdruck ist eine Verknüpfung von Wahrheitswerten und logischen Variablen durch die logischen Operatoren $\neg$, $\wedge$, $\vee$, $\rightarrow$ etc.. Die logischen Variablen können nur die Wahrheitswerte "wahr" und "falsch" annehmen. Alle Variablenwerte sind zum Zeitpunkt der Auswertung des Ausdrucks bekannt.
Beispiel: Folgende Ausdrücke können zur Fehlerdiagnose dienen:

$$\begin{pmatrix} \texttt{Zulaufpumpe_defekt} \\ \wedge\ \texttt{Ablaufpumpe_funktionsfähig} \\ \wedge\ \texttt{Durchfluß_positiv} \end{pmatrix} \rightarrow \texttt{Volumen_sinkt}$$

$$\begin{pmatrix} (\ \texttt{Rührer_defekt} \vee \texttt{Belüftung_defekt}\ ) \\ \wedge\ \texttt{Sauerstoffsonde_funktionsfähig} \end{pmatrix} \rightarrow \texttt{Sauerstoffkonzentration_sinkt}$$

$$\begin{pmatrix} \texttt{Trübungssonde_funktionsfähig} \\ \wedge\ \texttt{Durchfluß_Null} \end{pmatrix} \rightarrow \begin{pmatrix} \texttt{Wachstum_positiv} \\ \updownarrow \\ \texttt{Trübung_nimmt_zu} \end{pmatrix}$$

Man beachte, daß alle Ausdrücke gemäß der Forderung aus 4.2.3 frei von numerischen Ausdrücken sind. Die eigentliche Berechnung der Wahrheitswerte findet an anderer Stelle statt.

8.2 Regelsysteme

Aussagenlogische Ausdrücke erlauben nur die Formulierung statischer Zusammenhänge und haben damit einen recht eingeschränkten Anwendungsbereich [BN77]. Um die Formulierung allgemeinerer Aussagen zu ermöglichen, muß der Aussagenkalkül um sogenannte *Prädikate* erweitert werden, also Ausdrücke, deren Wahrheitswert von Variablen abhängt. Damit erhöht sich die Komplexität logischer Modelle wesentlich,

denn nun werden rekursive Strukturen möglich. Das klassische Beispiel ist die Definition der Fakultät durch:

$$x = n! \Leftrightarrow \begin{cases} x = n \cdot (n-1)! & \text{, falls } n > 0 \\ x = 1 & \text{, falls } n = 0 \end{cases}$$

Prädikatenlogische Ausdrücke werden oft in Form von *Klauseln* [BN77] notiert, welche die Gestalt:

$$\text{Konklusion} \Leftarrow \text{Prämisse}_1 \wedge \text{Prämisse}_2 \wedge \ldots \wedge \text{Prämisse}_n$$

haben. "Oder"-Ausdrücke entstehen durch Aufschreiben mehrerer Klauseln mit derselben Konklusion. So z.B. für die Fakultät:

$$\begin{aligned} x = n! \Leftarrow\ & n > 0 \\ \wedge\ & x = n \cdot y \\ \wedge\ & y = (n-1)! \end{aligned}$$

$$\begin{aligned} x = n! \Leftarrow\ & n = 0 \\ \wedge\ & x = 1 \end{aligned}$$

Die rekursive Struktur tritt deutlich zutage: Um die Aussage $x = n!$ zu verifizieren, muß implizit eine neue Variable y eingeführt werden, die wiederum rekursiv neue Variablen erzeugt. Auf diese Weise können Zusammenhänge mit großer Komplexität formalisiert werden [1].

Ein Satz von Klauseln für einen bestimmten Problembereich bildet ein sogenanntes *Regelsystem*. Die oben notierten aussagenlogischen Ausdrücke können z.B. — nach einer entsprechenden Umformung — zum Bestandteil eines Regelsystems zur Störungserkennung werden (vgl. [LG87]). Regelsysteme sind die Grundlage vieler Expertensysteme [HRWL83].

Mit Hilfe von Regelsystemen und verwandten Strukturen können qualitative Aussagen über ein System formuliert werden, über die quantitative Modelle nicht vorliegen (vgl. dazu [MSS89] [For81] [Aik80] [BAB+89]).

8.3 Algorithmische Darstellung

Kontrollstrategien sind auch in Form von Programmtexten in einer Programmiersprache oder einer Meta-Sprache formulierbar. Man spricht von *algorithmischer Wissensrepräsentation*. Die genannten Beispiele lassen sich z.B. folgendermaßen in ein Programm integrieren:

```
if (      Zulaufpumpe_defekt
     and  Ablaufpumpe_funktionsfähig
     and  Durchfluß_positiv      )
then
     Volumen_sinkt := true ;
if (      ( Rührer_defekt or Belüftung_defekt )
     and      Sauerstoffsonde_funktionsfähig       )
then
     Sauerstoffkonzentration_sinkt := true ;
```

Der entscheidende Unterschied zu den Regelsystemen liegt darin, daß die Reihenfolge der Abarbeitung vom Programmierer vorzugeben ist. Ein starrer Datenfluß hat Vorteile, wenn der Weg, der zum Ziel führt, vorher bekannt ist, und Nachteile, wenn von vornherein nicht klar ist, wie man die Fakten zu kombinieren hat, um eine bestimmte Aussage treffen zu können.

An dem oben genannten Beispiel wird das deutlich: Die algorithmische Formulierung der logischen Zusammenhänge ist nur dann anwendbar, wenn man an Prognosen ("Volumen wird sinken!", "Sauerstoffkonzentration wird sinken!") interessiert ist. Zur Fehlerdiagnose ("Wo liegt die Ursache einer sinkenden Sauerstoffkonzentration?") müßten die aussagenlogischen Ausdrücke erst umgeformt und das Programm anschließend neu geschrieben werden!

[1] Dies gilt vor allem dann, wenn zeitliche Zusammenhänge hinzu kommen (vgl. dazu [All81]).

8.4 Syntaktische Klassifikation

Ein sehr elegantes heuristisches Verfahren zur Mustererkennung ist die syntaktische Klassifikation [Nie83]. Da für sie später noch ein Einsatzbeispiel vorgestellt wird, soll der Ansatz kurz beschrieben werden: Die *syntaktische Klassifikation* geht im Gegensatz zur statistischen und heuristischen Klassifikation, die numerische Daten als Eingaben haben, von Symbolfolgen aus. Es liegen also kategoriale Eingangsdaten in der Form

A B A D C B C ...

vor. Aufgabe der syntaktischen Klassifikation ist die Zuordnung von Segmenten einer solchen Symbolfolge (sogenannten *Wörtern*) zu einer syntaktischen Klasse. Das bekannteste Beispiel für einen syntaktischen Klassifikator ist ein Scanner für eine Programmiersprache, der ein Programm in die syntaktischen Grundelemente (Bezeichner, Zahlen, Operatoren, reservierte Wörter etc.) zerlegt (vgl. 18.4).

Die Definition der syntaktischen Klassen erfolgt im allgemeinen mit Hilfe einer Grammatik. Grammatiken werden in 18.3.1 noch ausführlicher besprochen. Hier soll ein einfaches Beispiel genügen [2]: Verfolgt man ein Signal $x(t)$ über die Zeit und schätzt in diskreten Zeitabständen die Ableitung $\dot{x}(t)$, so kann die folgende grobe Symbolzuordnung vorgenommem werden:

$\nearrow$	bedeutet	$\dot{x}(t) \gg 0$
$\rightarrow$	bedeutet	$\dot{x}(t) \approx 0$
$\searrow$	bedeutet	$\dot{x}(t) \ll 0$

Aus dem Signal $x(t)$ entsteht damit eine Symbolfolge, wie z.B.

$$\nearrow\nearrow\rightarrow\nearrow\rightarrow\searrow\rightarrow\searrow\searrow\rightarrow\nearrow \ ,$$

Nun könnte eine Klassifikation von Zeitabschnitten im Signalverlauf folgendermaßen aussehen:

"Plateau"	:	$\rightarrow\rightarrow\rightarrow$
"Aufwärtstrend"	:	$\nearrow\nearrow\nearrow$ oder $\nearrow\rightarrow\nearrow\nearrow$ oder $\nearrow\nearrow\rightarrow\nearrow$
"Abwärtstrend"	:	$\searrow\searrow\searrow$ oder $\searrow\rightarrow\searrow\searrow$ oder $\searrow\searrow\rightarrow\searrow$
"Maximum"	:	$\nearrow\rightarrow\searrow$ oder $\nearrow\rightarrow\rightarrow\searrow$
"Minimum"	:	$\searrow\rightarrow\nearrow$ oder $\searrow\rightarrow\rightarrow\nearrow$
"Aufwärts-Stufe"	:	$\nearrow\rightarrow\rightarrow$ oder $\nearrow\nearrow\rightarrow\rightarrow$
usw....		

Die Anordnung

Signal	*Filtern*	Ableitung	*Symbolzuordnung*	Symbolfolge	*Klassifikation*	Diagnose
$x(t)$	$\longrightarrow$	$\dot{x}(t)$	$\longrightarrow$	$\ldots\nearrow\rightarrow\rightarrow\ldots$	$\longrightarrow$	"Aufwärts-Stufe"

bietet ein weiteres Beispiel für die vorgeschlagene 3-Ebenen-Architektur von Prozeßüberwachungssystemen (vgl. 4.3.1).

8.5 Weitere Ansätze

Einige weitere Techniken, die möglicherweise bei der Bioprozeßüberwachung zum Einsatz kommen können, sind in der Literatur beschrieben:

- Netzwerkmodelle, wie z.B. die *Petri-Netze* für die Modellierung paralleler kommunizierender Strukturen [Rei82].
- *Fuzzy Models* auf der Basis einer unscharfen Logik (*Fuzzy Logic*), die anstelle der Wahrheitswerte numerische "Möglichkeitswerte" verwendet [DP80] [BG89].

[2] Weitere Beispiele, vor allem aus dem Bereich der Erkennung gesprochener Sprache, finden sich in [Nie83].

- Lernfähige *Neronale Netze* zur Klassifikation numerischer Daten, die auf einem heuristischen, durch die Struktur von Nervenzellen-Netzen motivierten Ansatz beruhen [CERS88].

Dabei bleibt je nach Formulierung der Modelle offen, ob man sie eher auf der logisch/heuristischen Ebene ansiedelt oder als weitere Form einer Schnittstelle zwischen numerischen und logisch/kategorialen Daten ansieht.

Teil III

Methoden zur modellgestützten Datenanalyse

Modelle und Methoden sind strikt voneinander getrennt worden, um die Korrespondenz zwischen den Methoden der Datenanalyse, Prozeßsteuerung und -überwachung einerseits und den zugrunde liegenden Modellen andererseits zu betonen. Die Modelle beschreiben, wie die gemessenen Daten zustande kommen. Die zu verwendenden Methoden leiten sich in der Regel unmittelbar aus den Modellen ab.

Insbesondere wird durch eine modellmäßige Beschreibung in mathematischer oder formaler Gestalt meist auch eine Simulation des modellierten Systems möglich [3] . Derartige Simulationen erweisen sich als ein fundamentaler Bestandteil der Datenanalyse:

- Sie vermitteln ein Gefühl für die Größenordnung und Bedeutung der auftretenden Effekte in Abhängigkeit von den Systemparametern.
- Speziell die stochastische Simulation vermittelt eine Vorstellung der Auswirkungen von Störgrößen, die auf analytischem Wege meist nicht zu erlangen ist.
- Durch einen qualitativen Vergleich simulierter Daten mit vorgegebenen Meßdaten ist ein frühzeitiges Aussondern unbrauchbarer Modelle möglich.
- Jede eingesetzte Datenanalysetechnik sollte zunächst an simulierten Daten getestet werden. Dabei können die zu erwartenden Effekte im Vorfeld eines Experiments studiert werden. Auf der anderen Seite sollten Methoden, die schon bei simulierten Daten Instabilitäten — z.B. bei Modellfehlern — aufweisen, gar nicht erst am realen Prozeß zum Einsatz kommen.
- Der Entwurf von Reglern, Steuerungen und Prozeßüberwachungs-Komponenten kommt ohne Simulation nicht aus. Ein Test solcher Bausteine im geschlossenen Regelkreis würde ein Vielfaches der Zeit benötigen und kann im Fall von Instabilitäten sogar das biologische System schädigen.

Die besondere Bedeutung der Simulation im Datenanalyse-Konzept legt nahe, die Typen der verwendeten Modelle möglichst genau zu spezifizieren, um dann — noch vor der Programmierung irgendwelcher Datenanalysebausteine — zuerst die Werkzeuge zur Simulation bereitzustellen.

[3] Dies gilt nicht nur für die in Gleichungsform notierten Modelle, sondern durchaus auch für heuristische und logische. Einige Ansätze zur *qualitativen Simulation* finden sich in [Kui86].

Kapitel 9

Deterministische Methoden

Die deterministischen Methoden gehören zum regelungstechnischen Grundwissen. Es soll daher genügen, lediglich die Aspekte aufzuzeigen, die speziell für den biotechnischen Einsatz wichtig oder zum Verständnis der nachfolgenden Filtertheorie grundlegend sind. Insbesondere wurde auf eine Behandlung von Reglern verzichtet. Es muß aber noch einmal betont werden, daß (vgl. die Ausführungen in 3.1.3.1) die Theorie der Regler vollkommen analog zur Theorie der Beobachter zu entwickeln ist. Die meisten Ergebnisse für Beobachter lassen unter Ausnutzung des Dualitätsprinzips [BC85] auf Regler übertragen. Damit liegen z.B. die klassischen PID-Regler (als Spezialfall linearer Regler) oder auch parameteradaptive Regler in unmittelbarer Reichweite der Methodik!

9.1 Deterministische Simulation

Die deterministische Simulation eines durch ein Zustandsraummodell

$$\begin{aligned} \dot{x} &= f(t, \alpha, x, u) \\ y &= g(t, \beta, x) \end{aligned}$$

beschriebenen Systems mit Hilfe von Differentialgleichungs-Lösern gehört heute zum Standard [HNW87]. In der Regel leistet ein Runge-Kutta-Verfahren 4. oder höherer Ordnung (ggfs. mit einer Schrittweitensteuerung) beste Dienste. Für weitere Verfahren — insbesondere bei steifen Differentialgleichungen — sei auf die Literatur verwiesen [WA86] [HNW87] [Gea71], [BD81]. Bzgl. der praktischen Verwendbarkeit der Methoden sind einige Bemerkungen erforderlich:

9.1.0.1 Steife Systeme

Steifheiten entstehen dann, wenn die Zeitkonstanten verschiedener dynamischer Vorgänge im System um mehrere Größenordnungen differieren. Das typische biotechnische Beispiel für ein steifes System ist ein Wachstums-Modell mit Sauerstoffterm:

$$\begin{aligned} \dot{X} &= \mu \cdot X \\ \dot{S} &= -\sigma \cdot X \\ \dot{O} &= -\omega \cdot X + \mathrm{k_L a}(O^* - O) \end{aligned} \qquad \text{mit} \quad \begin{aligned} \sigma &= \sigma(S, O) \\ \mu &= Y_{X/S}(\sigma - m_S) \end{aligned}$$

Bei diesem System hat die Sauerstoffaufnahme eine Zeitkonstante $\ln 2/\mathrm{k_L a}$ in der Größenordnung einiger Minuten, wogegen sich die Zeitkonstante $\ln 2/\mu$ des Wachstums in der Größenordnung mehrerer Stunden bewegt [1]. Um die entstehende Steifheit zu integrieren, muß ein Runge-Kutta-Verfahren mit einer Schrittweite aufgesetzt werden, die eine korrekte Lösung der Sauerstoff-Gleichung gewährleistet. Abgesehen von numerischen Problemen ergeben sich extrem lange Simulationszeiten, die bei einem interaktiven System äußerst unerwünscht sind.

[1] Zwei typische Werte bei *Saccharomyces cerevisiae* sind $\mathrm{k_L a} = 592\ \mathrm{h^{-1}}$ und $\mu = 0.3\ \mathrm{h^{-1}}$ [SGB+88].

9.1.0.2 Beseitigung von Steifheiten durch Stationaritätsannahmen

Bevor nun ein spezielles Verfahren für steife Differentialgleichungen eingesetzt wird, sollte darüber nachgedacht werden, ob dies für den praktischen Einsatz sinnvoll ist. Die Zeitkonstante des Sauerstoffübergangs bewegt sich nämlich nur wenig über der Zeitkonstanten der Reaktordurchmischung. Ganz abgesehen davon können für den Transport des Sauerstoffs in die Zellen in diesen zeitlichen Größenordnungen keine Quasistationaritäts-Annahmen mehr aufrecht erhalten werden kann [2]. Eine realistische Simulation großer Zeitabschnitte mit dem obigen Modell ist daher von vorn herein nicht zu erwarten!
Bei speziellen Fragestellungen bieten sich Stationaritätsannahmen an:

- Soll das Modell zum Entwurf einer Sauerstoff-Regelung herangezogen werden, so genügt die Betrachtung kleiner Zeitabschnitte, so daß ohne weiteres

$$X, S = \text{const}$$

  angenommen werden kann.

- Geht es andererseits um das Verhalten des Wachstums in einem größeren Zeitabschnitt, so ist die Quasistationaritäts-Annahme

$$\dot{O} = 0 \Rightarrow O = O^* - \frac{\omega \cdot X}{k_L a}$$

  (vgl. 5.1.3.2) angebracht. Im Prinzip werden durch diese Annahme die schnellen Fluktuationen der Sauerstoffkonzentration über die Zeit gemittelt [3].

9.1.0.3 Steifheiten bei stochastischen Systemen

Das Problem steifer Differentialgleichungen sollte beim Entwurf von Filtern nicht auftreten: Die Grundlage eines Filters ist die unscharfe Beschreibung des zugrundeliegenden Systems durch eine stochastische Differentialgleichung (vgl. 7.5). Nun ist es ein Merkmal steifer Differentialgleichungen, daß kleinste Veränderungen des Systemzustands zum Zeitpunkt t große Veränderungen zu einem späteren Zeitpunkt $t + \Delta t$ nach sich ziehen. Dieser Effekt steht einer unscharfen Modellierung prinzipiell entgegen!

Ein Filter ist somit nicht in der Lage, eine schnelle Änderung in der Dynamik des Systems adäquat nachzuvollziehen (Trägheit des Filters, vgl. 2.5). Die Anwendung von Filtern bei Systemen mit schnell veränderlicher Dynamik ist aus diesem Grunde nicht ratsam.

9.1.1 Schrittweitensteuerung

Die Runge-Kutta-Verfahren lassen sich leicht mit einer Schrittweitensteuerung ausstatten, die dafür sorgt, daß die Schrittweite des Differentialgleichungs-Lösers der aktuellen Systemdynamik angepaßt wird [PFTV88]. Damit lassen sich auch leichte Steifheiten ohne weiteres überwinden.

Trotzdem hat die Schrittweitensteuerung einen entscheidenden Nachteil: Bei einer On-Line-Simulation, die auf dem Bildschirm verfolgt wird, verzerrt sich die Zeitskala, d.h. der Echtzeit-Eindruck geht verloren. Vor allem bei Phasenraumportraits, in denen die Zeitinformation nur noch implizit über die Zeichengeschwindigkeit der Kurven vermittelt wird, widerspricht dies dem Visualisierungsprinzip für die Datenanalyse (vgl. 1.3.2.2). Aus diesem Grunde sollte jedes On-Line-Simulationsprogramm die Möglichkeit zum Abschalten der Schrittweitensteuerung anbieten.

[2] [BO86] schlüsselt den Sauerstofftransport in die Zelle in eine Reihe von Schritten auf, deren Zeitkonstanten in dieser Situation alle nicht vernächlässigbar sind.
[3] Eine kritische Betrachtung zu dieser Vorgehensweise findet sich in [Gel72].

9.2 Beobachter

9.2.1 Aufbau eines Beobachters

Ein reales System sei beschreibbar durch ein Zustandsraummodell der Gestalt:

$$\begin{aligned} \dot{x} &= f(t, \alpha, x, u) \\ y &= g(t, \beta, x) \end{aligned}$$

Oft ist die Zahl der Meßgrößen kleiner als die Zahl der Systemzustände, d.h.:

$$\dim y < \dim x$$

Gerade in der Biotechnologie tritt diese Situation häufig auf (vgl. auch 6.6.3.1). Z.B. sind nur wenige Meßgrößen direkt mit der Biomassekonzentration X oder der Substratkonzentration S korreliert und meist reicht die Instrumentierung nur aus, um eine der beiden Größen on-line zu bestimmen. Es ist dann nicht möglich, mit Hilfe der Meßgleichung $y = g(t, \beta, x)$ unmittelbar alle Systemzustandsgrößen (im Beispiel X und S) auszurechnen [4].

Ein Zustandsbeobachter versucht dieses Problem durch Verfolgung der Meßgrößen $y(t)$ über die Zeit zu lösen. Er nimmt somit mehr Information aus dem System auf, als die Meßapparatur, die stets nur einen einzigen Zeitpunkt t berücksichtigt. Ganz allgemein wird von einem Zustandsraummodell

$$\begin{aligned} \dot{\hat{x}} &= \hat{f}(t, \hat{\alpha}, \hat{x}, u) \\ \hat{y} &= \hat{g}(t, \hat{\beta}, \hat{x}) \end{aligned}$$

des realen Systems ausgegangen. Es können dabei durchaus *Modellfehler* auftreten, d.h. das reale System wird durch das Modell nicht korrekt oder nicht vollständig beschrieben. Bei biologischen Systemen kann man im allgemeinen nur von einer Approximation

$$\hat{f} \sim f, \quad \hat{g} \sim g, \quad \hat{\alpha} \sim \alpha, \quad \hat{\beta} \sim \beta$$

ausgehen. Wichtig ist jedoch, daß die *Schätzer* $\hat{x}$ und $\hat{y}$ in Dimension und Bedeutung den Zustandsvariablen x und y entsprechen, also insbesondere

$$\dim \hat{x} = \dim x \quad \text{und} \quad \dim \hat{y} = \dim y$$

gilt. Der Unterschied von Modell und Realität zeigt sich im Parallelvergleich, der on-line vollzogen werden kann (vgl. Abb. 9.1): Die beiden Systeme erzeugen dann verschiede Werte $y(t)$ und $\hat{y}(t)$. Ohne weitere Kenntnis des realen Systems stellt also die Differenz

$$y(t) - \hat{y}(t)$$

die einzige Information über den Unterschied zwischen Realität und Modell dar. Der Grundgedanke bei der Konstruktion eines Beobachters ist nun, diese Differenz on-line zur Korrektur der aktuellen Schätzung $\hat{x}$ einzusetzen. Das Ziel ist dabei, eine Konvergenz

$$x(t) - \hat{x}(t) \rightarrow 0$$

[4] Man vergleiche diese Situation mit den Kriterien zur Schätzbarkeit der kinetischen Parameter aus dem Systemzustand zum Zeitpunkt t (siehe 6.6.3.2). Die Zustandsbeobachtung bietet eine Möglichkeit, durch Schätzung weiterer Zustandsgrößen die Kriterien auch dann noch zu erfüllen, wenn die entsprechenden linearen Gleichungssysteme unterbestimmt sind! Die Zustandsschätzung könnte von daher als das "Missing Link" bei der Schätzung von Stoffwechselparametern bezeichnet werden.

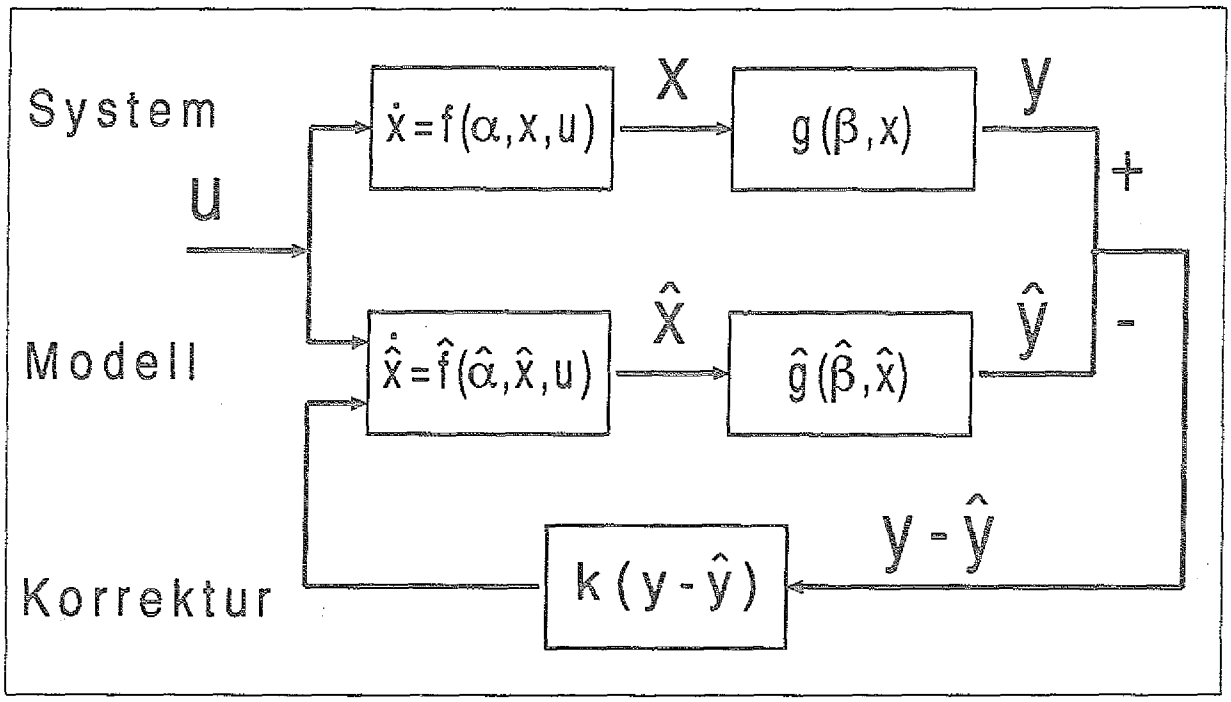


Abbildung 9.1: Allgemeines Schema eines Zustandsbeobachters

zu erreichen. Diese Rückkopplung geschieht über eine Korrekturfunktion

$$k(y - \hat{y}).$$

Zusammen mit dem korrekten Systemmodell ergeben sich die Beobachtergleichungen, die in Abbildung 9.1 schematisiert wiedergegeben sind:

Für das reale System:
$$\begin{aligned} \dot{x} &= f(t, \alpha, x, u) \\ y &= g(t, \beta, x) \end{aligned}$$

Für den Beobachter:
$$\begin{aligned} \dot{\hat{x}} &= \hat{f}(t, \hat{\alpha}, \hat{x}, u) + k(y - \hat{y}) \\ \hat{y} &= \hat{g}(t, \hat{\beta}, \hat{x}) \end{aligned}$$

In dieser Allgemeinheit kann über die Konvergenz $x(t) - \hat{x}(t) \rightarrow 0$ mit theoretischen Mitteln wenig ausgesagt werden. Dazu müssen Spezialfälle betrachtet werden.

9.2.2 Lineare zeitinvariante Theorie

Im linearen zeitinvarianten Fall sind f und g zeitunabhängig und linear in x, u, d.h. man kann den Beobachter mit Hilfe von Matrizen folgendermaßen formulieren:

Reales System:
$$\begin{aligned} \dot{x} &= Ax + Bu \\ y &= Cx \end{aligned}$$

Beobachter-Gleichungen:
$$\begin{aligned} \dot{\hat{x}} &= \hat{A}\hat{x} + \hat{B}u + K(y - \hat{y}) \\ \hat{y} &= \hat{C}\hat{x} \end{aligned}$$

K heißt die *Verstärkungsmatrix* des Beobachters.

9.2.2.1 Der Konvergenzsatz bei korrektem Modell

Für den Fall, daß Modell und Realität übereinstimmen, d.h.

$$A = \hat{A}, \quad B = \hat{B}, \quad C = \hat{C},$$

berechnet man

$$\begin{aligned} \frac{d}{dt}(x - \hat{x}) &= Ax - A\hat{x} + Bu - Bu - K(y - \hat{y}) \\ &= A(x - \hat{x}) - K(Cx - C\hat{x}) \\ &= (A - KC)(x - \hat{x}) \ . \end{aligned}$$

Es folgt der Stabilitätssatz:

$$x - \hat{x} \to 0 \ \ \forall_{x(0)} \iff A - KC \ \text{ist eine } \textit{stabile Matrix}^5 \ .$$

Die Verstärkungsmatrix K muß demnach so gewählt werden, daß alle Eigenwerte von $A - KC$ einen negativen Realteil haben. Ist eine solche Wahl nicht möglich, so spricht man von einem *nicht (linear) beobachtbaren* System. Ein Beobachterentwurf beinhaltet nun die möglichst geschickte Wahl von K. Hierzu stellt die Beobachtbarkeitstheorie diverse Hilfsmittel zur Verfügung, die z.B. in [BC85] zusammengestellt sind.

9.2.2.2 Eindimensionaler Fall

Im eindimensionalen Fall sind $A = (a)$, $B = (b)$, $C = (c)$ und $K = (k)$ einfache Zahlen. Für den Fall $c > 0$ ($c < 0$ analog) folgt sofort

$$x - \hat{x} \to 0 \ \ \forall_x(0) \iff a - kc < 0 \iff k > \frac{a}{c}.$$

Die Beobachtbarkeit des eindimensionalen Systems ist also gegeben, solange nur $c \neq 0$ ist. Im pathologischen Fall $c = 0$, der eine "Messung" ohne jede Systeminformation beschreibt, ist nur ein stabiles System ($a < 0$) beobachtbar, welches aus sich heraus gegen 0 konvergiert.

Die Konvergenz entspricht der von $e^{(a-kc)t}$, d.h. sie wird um so besser, je kleiner $a - kc$ wird. Für $k \to \infty$ kann man theoretisch eine beliebig schnelle Konvergenz erreichen. Erst in der stochastischen Theorie wird klar werden, daß für $k \to \infty$ das Meßrauschen immer mehr verstärkt und auf den Schätzer $\hat{x}$ übertragen wird. Die Abbildung 9.2 zeigt einen Simulationslauf mit verschieden starkem aufgeschalteten Meßrauschen.

9.2.2.3 Ein zweidimensionales Beispiel

Beobachtet werden soll die kreisförmige Bewegung (vgl. 7.3.3.1):

$$\ddot{x} = -x \iff \begin{aligned} \dot{x}_1 &= -x_2 \\ \dot{x}_2 &= x_1 \end{aligned}$$

aufgrund der Messung

$$y = x_1 \ .$$

Die linearen Beobachtergleichungen für dieses Problem sind

$$\begin{aligned} \dot{\hat{x}}_1 &= -\hat{x}_2 + k_1 \cdot (y - \hat{y}) \\ \dot{\hat{x}}_2 &= \hat{x}_1 + k_2 \cdot (y - \hat{y}) \ . \end{aligned}$$

[5] Eine Matrix heißt stabil, wenn alle ihre Eigenwerte in der linken komplexen Halbebene liegen.

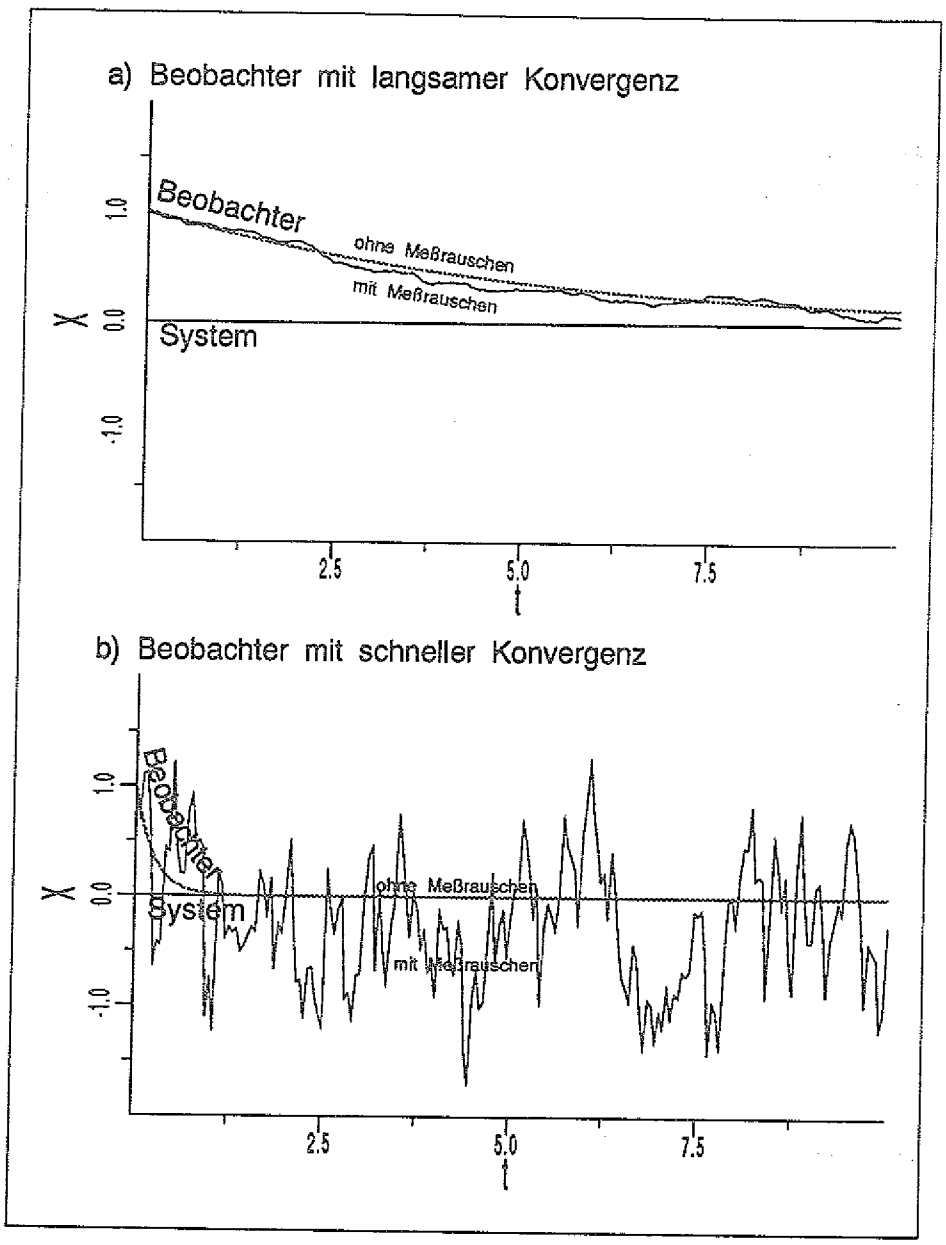


Abbildung 9.2: 1-dimensionaler Beobachter

Es folgt:

$$\frac{d}{dt}\begin{pmatrix} x_1-\hat{x}_1 \\ x_2-\hat{x}_2 \end{pmatrix} = \begin{pmatrix} -k_1 & -1 \\ 1-k_2 & 0 \end{pmatrix} \cdot \begin{pmatrix} x_1-\hat{x}_1 \\ x_2-\hat{x}_2 \end{pmatrix}$$

Die Eigenwerte der Matrix berechnet man mit

$$\lambda = -\frac{1}{2}\cdot k_1 \pm \sqrt{4(k_2-1)+k_1^2}.$$

Betrachtet werden soll nur der Spezialfall, daß beide Eigenwerte gleich sind, d.h.

$$4(k_2-1)+k_1^2 = 0 \quad \Rightarrow \quad k_2 = -\frac{1}{4}\cdot k_1^2 + 1.$$

Die Konvergenzbedingung ist dann gerade

$$k_1 > 0 .$$

Je größer man k_1 wählt, um so schneller erfolgt die Konvergenz (zumindest theoretisch). Es treten dabei ganz analoge Effekte wie im eindimensionalen Fall auf [WSS88]: Kleine k_1-Werte bewirken ein zu träges Folgeverhalten, wogegen große k_1-Werte das Rauschen verstärken. Ein Optimum für die Wahl von k_1 ist "in der Mitte" zu suchen. Die stochastische Theorie wird hierfür ein quantitatives Kriterium bereitstellen.

9.2.2.4 Auswirkungen von Modellfehlern

Im allgemeinen reagieren Beobachter auf Modellfehler sehr sensibel. Im eindimensionalen linearen Fall lassen sich bereits alle Effekte aufzeigen. Läßt man die Annahme eines korrekten Modells fallen, so kann man schreiben:

$$\begin{aligned}
\tfrac{d}{dt}(x-\hat{x}) &= ax + bu - \hat{a}\hat{x} - \hat{b}u - k(cx - \hat{c}\hat{x}) \\
&= \hat{a}x + (a-\hat{a})x - \hat{a}\hat{x} \\
&\quad + \hat{b}u + (b-\hat{b})u - \hat{b}u \\
&\quad - k(\hat{c}x + (c-\hat{c})x - \hat{c}\hat{x}) \\
&= ((a-\hat{a}) - k(c-\hat{c}))\cdot x \\
&\quad + (\hat{a} - k\hat{c})\cdot(x-\hat{x}) \\
&\quad + (b-\hat{b})\cdot u
\end{aligned}$$

oder in Matrixform:

$$\frac{d}{dt}\begin{pmatrix} x \\ x-\hat{x} \end{pmatrix} = \begin{pmatrix} a & 0 \\ (a-\hat{a}) - k(c-\hat{c}) & \hat{a} - k\hat{c} \end{pmatrix} \cdot \begin{pmatrix} x \\ x-\hat{x} \end{pmatrix} + \begin{pmatrix} b \\ b-\hat{b} \end{pmatrix} \cdot u$$

Untersucht werden sollen zwei Eigenschaften des Systems:

Stabilität :

Das homogene Problem ($b = \hat{b} = 0$) hat die Lösungen

$$\begin{aligned}
x(t) &= x(0)\cdot e^{at}, \\
(x-\hat{x})(t) &= \frac{(a-\hat{a}) - k(c-\hat{c})}{a - (\hat{a} - k\hat{c})} \cdot x(0) \cdot (e^{at} - e^{(\hat{a}-k\hat{c})t}) + (x-\hat{x})(0)\cdot e^{(\hat{a}-k\hat{c})t} .
\end{aligned}$$

Falls ein Modellfehler vorliegt, d.h. $(a-\hat{a}) - k(c-\hat{c}) \neq 0$, so ist der Beobachter instabil, sofern das System selbst instabil ist ($a > 0$). Die Beobachtung instabiler Systeme ist somit äußerst problematisch!

Im stabilen Fall ($a < 0$) ist der Beobachter stabil, wenn die bekannte Bedingung

$$\hat{a} - k\hat{c} < 0 \iff k > \frac{\hat{a}}{\hat{c}} \quad (\text{im Falle } \hat{c} > 0)$$

gilt.

Diese Ergebnisse lassen sich bei konstantem u auf das inhomogene System übertragen. Die Untersuchung des Systems für beliebiges $u = u(t)$ gestaltet sich bereits sehr schwierig (vgl. [BC85]).

Asymptotischer Beobachtungsfehler :

Für den Spezialfall $u(t) = \text{const.}$ soll Beispielhaft soll — die Stabilität des Beobachters vorausgesetzt — der asymptotische Beobachtungsfehler ausgerechnet werden. Aus den stationären Gleichungen errechnet man:

$$
\begin{aligned}
x(\infty) &= -\frac{b}{a} \cdot u \\
(x - \hat{x})(\infty) &= \frac{1}{\hat{a} - k\hat{c}} \left[((a - \hat{a}) - k(c - \hat{c})) \cdot \frac{b}{a} - (b - \hat{b}) \right] \cdot u
\end{aligned}
$$

Für $k \to \infty$ gilt insbesondere:

$$
(x - \hat{x})(\infty) \to \frac{c - \hat{c}}{\hat{c}} \cdot \frac{b}{a} \cdot u
$$

D.h. für $c \neq \hat{c}$ resultiert ein Beobachtungsfehler, der selbst bei einem ungestörten System nicht beliebig weit zu verringern ist [6].

9.2.3 Nichtlineare Beobachter

Die Bezeichnung "nichtlinear" kann sich sowohl auf die Modellfunktionen f, g als auch auf die Art der Rückkopplung k beziehen. Allgemeine Konvergenzresultate für nichtlineare Beobachter sind aufgrund der Komplexität dieser Systeme kaum zu erwarten. Für einige spezielle Systemtypen liegen Ergebnisse vor, die meist auf der Annahme eines korrekten Modells beruhen [DB86] [Sch82c] [Wil77] [KH82]. Im Falle eines Modellfehlers ist eine analytische Behandlung — wie oben schon im linearen Fall zu sehen — kaum noch möglich.

9.2.3.1 Linearisierung

Eine Technik zur Beobachtung nichtlinearer Systeme, die bei nicht zu großen Abweichungen $(x - \hat{x})$ Erfolg verspricht, ist die Linearisierung des Systemmodells. Dabei wird das nichtlineare Modell

$$
\begin{aligned}
\dot{x} &= f(t, \alpha, x, u) \\
y &= g(t, \beta, x)
\end{aligned}
$$

durch seine Linearisierung

$$
\begin{aligned}
\dot{x} &\approx \frac{\partial}{\partial x} f(t, \alpha, x, u)_{|x=x^*, u=u^*} \cdot (x - x^*) \\
&\quad + \frac{\partial}{\partial u} f(t, \alpha, x, u)_{|x=x^*, u=u^*} \cdot (u - u^*) + f(t, \alpha, x^*, u^*) \\
y &\approx \frac{\partial}{\partial x} g(t, \beta, x)_{|x=x^*} \cdot (x - x^*) + g(t, \alpha, x^*)
\end{aligned}
$$

an der Stelle (x^*, u^*) approximiert. Auf dieses System kann dann die lineare Theorie angewendet werden.

9.2.3.2 Verschiedene Linearisierungsansätze

Für die Wahl von x^* und u^* sind mehrere Strategien denkbar:

Linearisierung um einen festen Betriebspunkt (x^*, u^*) :

In diesem Fall wird angenommen, daß sich sowohl der Systemzustand $x(t)$ als auch die Kontrollvariable $u(t)$ nie allzu weit von x^*, u^* entfernen werden. Am ehesten ist dies zu erwarten, wenn (x^*, u^*) ein stabiler stationärer Punkt des realen Systems ist, dessen Lage man a priori gut schätzen kann. Für einen solchen stationären Punkt verschwinden die konstanten Terme in der Approximation.

[6] Es sollte angemerkt werden, daß der Meßparameter c i.a. am besten bekannt ist (vgl. 5.4).

Linearisierung um eine Trajektorie $(x^*(t), u^*(t))$:
Der Ansatz kann verfolgt werden, wenn die Systementwicklung $x^*(t)$ zur Kontrolle $u^*(t)$ a priori schätzbar ist. Er setzt voraus, daß die Differenz von $(x(t), u(t))$ zur gewählten *Referenztrajektorie* stets klein bleiben wird. Bei instationärer Fahrweise des Systems kann eine solche Bedingung nur für einen beschränkten Zeitraum aufrecht erhalten werden (z.B. bei einer Batch-Kultur).

Linearisierung um eine On-Line-Schätzung $\hat{x}(t)$ und eine Trajektorie $u^*(t, \hat{x}(t))$:
Es wird eine On-Line-Schätzung des Systemzustands als Linearisierungspunkt gewählt. In der Regel wird dies die Schätzung sein, die der Beobachter selbst liefert. Dies setzt voraus, daß zum Zeitpunkt $t = 0$ bereits eine gute Schätzung des Anfangszustands besteht. Andernfalls besteht die Gefahr, daß sich System und Beobachter "verlieren".

9.2.3.3 Konstruktion eines Beobachters auf der Basis der Linearisierung

Der auf der Basis der Linearisierung konstruierte Beobachter hat eine gemischt linear/nichtlineare Gestalt. Als Systemmodell wird die nichtlineare Gleichung belassen, wogegen die Rückkopplung linear mit einer Verstärkungsmatrix $K = K(x^*(t), u^*(t))$ erfolgt:

$$\begin{aligned} \dot{\hat{x}} &= \hat{f}(t, \hat{\alpha}, \hat{x}, u) + K(y - \hat{y}) \\ \hat{y} &= \hat{g}(t, \hat{\beta}, \hat{x}) \end{aligned}$$

Die Berechnung von K erfolgt auf der Basis des linearisierten Systems mit den Mitteln der linearen Beobachtbarkeitstheorie (vgl. 9.2.2.1). In der Regel werden dazu sogenannte *Polvorgaben* gemacht, nach denen die Adaptation von $K(t)$ erfolgt [BC85]. Dabei sind die verschiedenen Linearisierungsansätze zu unterscheiden:

Linearisierung um einen festen Betriebspunkt :
Die Linearisierung bleibt dann fest, so daß ein festes K für den gesamten Zeitraum berechnet werden kann.

Linearisierung um eine Trajektorie :
Hier verändert sich die Linearisierung mit dem Systemzustand, so daß sich zunächst ein zeitabhängiges $K = K(t)$ ergibt. $K(t)$ kann dabei im voraus bestimmt werden, da die Referenztrajektorie vorgegeben ist. In günstigen Fällen gibt es ein konstantes K, welches zu allen Zeitpunkten die Konvergenzbedingung des linearen System erfüllt.

Linearisierung um eine On-Line-Schätzung :
In dieser Situation muß K on-line aus der jeweils gültigen Linearisierung gewonnen werden. Dazu sind ggfs. On-Line-Eigenwertberechnungen erforderlich, welche hohe Anforderungen an die verwendete Numerik stellen. Ein Ansatz aus dieser Klasse ist in [Zei77] beschrieben.

9.2.3.4 Ein Beispiel

Das Monod-Modell

$$\begin{aligned} \dot{X} &= Y_{X/S} \cdot \sigma(S) \cdot X + D \cdot (-X) \\ \dot{S} &= \sigma(S) \cdot X + D \cdot (S_{in} - S) \end{aligned}$$

für die kontinuierliche Kultur linearisiert sich mit der Kontrollvariablen D zu

$$\begin{aligned} \frac{d}{dt} \begin{pmatrix} X - X^* \\ S - S^* \end{pmatrix} \approx & \begin{pmatrix} Y_{X/S} \cdot \sigma(S^*) \cdot X^* + D^* \cdot (-X^*) \\ \sigma(S^*) \cdot X^* + D^* \cdot (S_{in} - S^*) \end{pmatrix} \\ & + \begin{pmatrix} -D^* + Y_{X/S} \cdot \sigma(S^*) & Y_{X/S} \cdot \frac{\partial}{\partial S} \sigma(S^*) \cdot X \\ \sigma(S^*) & -D^* + \frac{\partial}{\partial S} \sigma(S^*) \cdot X \end{pmatrix} \cdot \begin{pmatrix} X - X^* \\ S - S^* \end{pmatrix} \\ & + \begin{pmatrix} -X^* \\ S_{in} - S^* \end{pmatrix} \cdot (D - D^*) \end{aligned}$$

Speziell im stationären Fall (beachte: $\dot{X} = 0 \Rightarrow Y_{X/S} \cdot \sigma(S) = D$):

$$\frac{d}{dt}\begin{pmatrix} X - X^* \\ S - S^* \end{pmatrix} = \begin{pmatrix} 0 & Y_{X/S} \cdot \frac{\partial}{\partial S}\sigma(S^*) \cdot X \\ \sigma(S^*) & -D^* + \frac{\partial}{\partial S}\sigma(S^*) \cdot X \end{pmatrix} \cdot \begin{pmatrix} X - X^* \\ S - S^* \end{pmatrix} + \begin{pmatrix} -X^* \\ -S^* \end{pmatrix} \cdot (D - D^*)$$

Eine Untersuchung der Linearisierungsgüte mit Hilfe von Simulationen findet sich [Sch89] [SW89].

9.2.4 Adaptive Beobachter

Dem Problem des Modellfehlers kann dadurch begegnet werden, daß der Beobachter zusammen mit der Schätzung $\hat{x}$ des Systemzustands zugleich auch einige der Modellparameter

$$\hat{\alpha} = (\alpha_1, \ldots, \alpha_r)^T$$

anpaßt ($\hat{\beta}$ ist i.a. gut bekannt).

9.2.4.1 Umwandlung einzelner Parameter zu Zustandsvariablen

Dazu wird der kritischen Parameter in eine konstante Zustandsgröße des Systems umgewandelt:

$$\dot{\alpha}_i = 0$$

Dieser Ansatz erlaubt oft — zusammen mit einer Linearisierung der entstehenden, meist nichtlinearen Systeme — die On-Line-Anpassung des Modells. Dabei muß stets berücksichtigt werden, daß ohne Informationsverlust nicht beliebig viele Parameter zu Zustandsgrößen gemachen werden können. Es gilt das folgende *Unschärfeprinzip*:

> "Aufgrund einer begrenzten Menge an Information können nicht beliebig viele Parameter beliebig genau geschätzt werden [7]."

Je mehr Parameter man also zu schätzen versucht, um so unschärfer wird die Schätzung des einzelnen Parameters. Für einen nichtlinearen Beobachter geht dies oft mit dem Verlust der Konvergenz einher!

9.2.4.2 Ein eindimensionales Beispiel

Falls in dem eindimensionalen Modell

$$\begin{aligned} \dot{x} &= a \cdot x + u \\ y &= x \end{aligned}$$

der Parameter a nicht genau bekannt ist, so kann man die Gleichung

$$\dot{a} = 0$$

hinzufügen. Ein mit Hilfe einer Linearisierung erstellter Beobachter hat die Gestalt:

$$\begin{aligned} \dot{\hat{x}} &= \hat{a} \cdot \hat{x} + u + k_x \cdot (y - \hat{y}) \\ \dot{\hat{a}} &= 0 + k_y \cdot (y - \hat{y}) \end{aligned}$$

Die Koeffizienten k_x und k_y sind aufgrund der Linearisierung

$$\begin{aligned} \tfrac{d}{dt}(x - x^*) &\approx a^* \cdot x^* + a^* \cdot (x - x^*) + x^* \cdot (a - a^*) + u \\ \tfrac{d}{dt}(a - a^*) &\approx 0 \end{aligned}$$

zu berechnen. D.h. alle Eigenwerte der Matrix

$$\begin{pmatrix} a^* & x^* \\ 0 & 0 \end{pmatrix} - \begin{pmatrix} k_x \\ k_y \end{pmatrix} \cdot (1, 0) = \begin{pmatrix} a^* - k_x & x^* \\ -k_y & 0 \end{pmatrix}$$

müssen in der linken komplexen Halbebene liegen [8].

[7] Das Unschärfeprinzip ist in der statistischen Theorie und in einer verwandten Situation der Signaltheorie auch quantitativ formulierbar (vgl. [Pri81]).

[8] Für dieses Beispiel ließen sich k_x und k_y auch konkret ausrechnen und es könnte ein expliziter Konvergenznachweis geführt werden. Für das Verständnis des Prinzips gebügen die Ausführungen jedoch vollauf.

Kapitel 10

Statistische Methoden

Im Gegensatz zur deterministischen Simulationstechnik sind die verschiedenen Methoden der Simulation stochastischer Vorgänge nicht geläufig. Unter Biotechnologen ist die Beschreibung von Störungen kaum über die Benutzung eines Zufallszahlengenerators für gleichverteilte Zufallszahlen (wie er in den meisten Programmiersprachen implementiert ist) hinausgekommen. Auf der anderen Seite beruhen aber fast alle statistischen und stochastischen Verfahren auf expliziten Verteilungsannahmen. Zum Test solcher Methoden ist es ratsam, ein Spektrum von Simulationstechniken zur Verfügung zu haben (siehe 10.1).

Die derzeit am häufigsten eingesetzte statistische Auswertungsmethode ist die nichtlineare Parameteranpassung bei Differentialgleichungs-Modellen. Sie erfährt eine kritische Analyse im Kontext einer mehr stochastischen Sichtweise biologischer Systeme (siehe 10.3). An Beispielen kann gezeigt werden, daß dieses Verfahren nur mit Vorbehalten angewendet werden sollte. Dadurch erfährt die Technik der stationären Identifikation eine starke Aufwertung insbesondere dann, wenn sie durch explorative Techniken zur Datenanalyse unterstützt wird (siehe 10.4).

10.1 Zufallszahlengeneratoren

Eine Bemerkung vorweg: "Zufallszahlen" gibt es nicht! Zufallsvariablen sind mathematische Modelle von realen Vorgängen, bei denen nicht alle Einflußfaktoren bekannt sind. Die Prüfung, ob eine gegebene Größe X eine Zufallsvariable mit gegebenen Eigenschaften ist, würde die simultane Durchführung von überabzählbar vielen Kopien desselben Experiments erfordern! Eine Approximation an diese Vorgehensweise ist die häufige unabhängige Wiederholung eines Experiments, wobei auch die Unabhängigkeit eine nicht streng beweisbare Hypothese ist.

Aus diesem Grunde sollte man korrekter von *Pseudo-Zufallszahlengeneratoren* sprechen. Von einem solchen Generator muß verlangt werden, daß er möglichst viele Eigenschaften einer Zufallsvariablen mit gegebener Verteilung reproduzieren kann. Hierzu können diverse statistische Größen, wie Erwartungswert, Varianz, Verteilung, Unabhängigkeit aufeinanderfolgender Zahlen, Autokorrelation oder das Frequenzspektrum an einer sehr großen Stichprobe geprüft werden [Knu73]. Ein *guter* Zufallszahlengenerator unterscheidet sich in allen diesen Kriterien nicht signifikant von den theoretisch zu erwartenden Ergebnissen.

10.1.1 Kongruenzverfahren zur Erzeugung gleichverteilter Zufallszahlen

Zufallszahlen werden heute standardmäßig mit Hilfe eines sogenannten *Kongruenzverfahrens* erzeugt [PFTV88]. Dies ist ein iteratives Schema des Typs:

$$X_{j+1} = (a \cdot X_j + b) \bmod c \quad \text{mit} \quad X_j, a, b, c \in \{0, 1, 2, 3, \ldots\}$$

Folgendes ist zu bemerken:

- Die Parameter des Generators sind a, b, c und der Startwert X_0 (engl. *Seed*). Wählt man X_0 stets gleich, so entsteht bei jedem Durchlauf dieselbe Folge von Zufallszahlen, andernfalls hat sich eine Startwertwahl in Abhängigkeit von der Uhrzeit verbreitet.

- Die Zahlenwerte sind offenbar nicht im geringsten "zufällig", sondern genau determiniert.
- Der Generator kann nur Zahlen zwischen 0 und $c-1$ erzeugen, so daß nach spätestens c Iterationen ein Zyklus auftritt.
- Ein möglichst "unregelmäßiges" Verhalten des Generators erreicht man durch Wahl von teilerfremden Zahlen, nach Möglichkeit Primzahlen (vgl. [Knu73]).

Übliche Werte für c liegen zwischen 10^4 und 10^8, so daß $4-8$-stellige Zufallszahlen entstehen. Die Zahlen

$$\frac{X_0}{c}, \frac{X_1}{c}, \frac{X_2}{c}, \ldots$$

stellen dann eine Approximation an eine wiederholte unabhängige Durchführung eines Experiments mit einer $[0,1]$-verteilte Zufallsvariablen dar. Einige Zahlenwerte für (im statistischen Sinne) gute Kongruenzgeneratoren finden sich in [PFTV88] [Knu73], einige Gütekriterien in [Knu73]. In [PFTV88] ist ferner beschrieben, wie man einen gegebenen Zufallszahlengenerator durch ein Mischverfahren verbessern kann.

10.1.2 Erzeugung von Zufallszahlen mit beliebiger Verteilung

Ist einmal ein Zufallszahlengenerator für eine $[0,1]$-gleichverteilte Zufallsvariable X konstruiert worden, so läßt sich daraus leicht ein Generator für Zufallszahlen mit einer beliebigen vorgegebenen Verteilung

$$f_Y(\xi)$$

gewinnen. Die verschiedenen Techniken sind im Prinzip bereits im Modell-Kapitel (6.2) vorweggenommen worden und können kurz zusammengefaßt werden:

10.1.2.1 Eindimensionale Transformation

Im eindimensionalen Fall kann man die Transformationsformel für Wahrscheinlichkeitsdichten schreiben als:

$$f_{T(X)}(\xi) = f_X(T^{-1}(\xi)) \cdot \left[T'(T^{-1})(\xi)\right]^{-1} = f_X(T^{-1}(\xi)) \cdot (T^{-1})'(\xi)$$

Mit den Vorgaben

$$Y = T(X) \quad \text{und} \quad f_X(\xi) = 1, \;\; \xi \epsilon [0,1]$$

folgt unmittelbar

$$f_Y(\xi) = (T^{-1})'(\xi) \implies T^{-1}(\xi) = \int_{-\infty}^{\xi} f_Y(\zeta) d\zeta \; .$$

woraus sich — für den Fall, daß die Integralfunktion analytisch lösbar und invertierbar ist — $T(\xi)$ berechnen läßt.

1.Beispiel: Für die Erzeugung einer Exponentialverteilung $f_Y(\xi) = \lambda \cdot e^{-\lambda\xi}$ $(\xi > 0)$ ergibt sich:

$$T^{-1}(\xi) = \int_0^{\xi} \lambda \cdot e^{-\lambda\zeta} d\zeta = 1 - e^{-\lambda\xi} \;\; \Rightarrow \;\; T(\xi) = -\frac{1}{\lambda} \ln(1-y)$$

2.Beispiel: Für die Erzeugung einer Cauchy-Verteilung $f_Y(\xi) = \frac{1}{\pi} \cdot \frac{1}{1+\xi^2}$ gilt analog:

$$T^{-1}(\xi) = \frac{1}{\pi} \arctan\xi + \frac{1}{2} \;\; \Rightarrow \;\; T(\xi) = \tan(\pi \cdot (\xi - \frac{1}{2}))$$

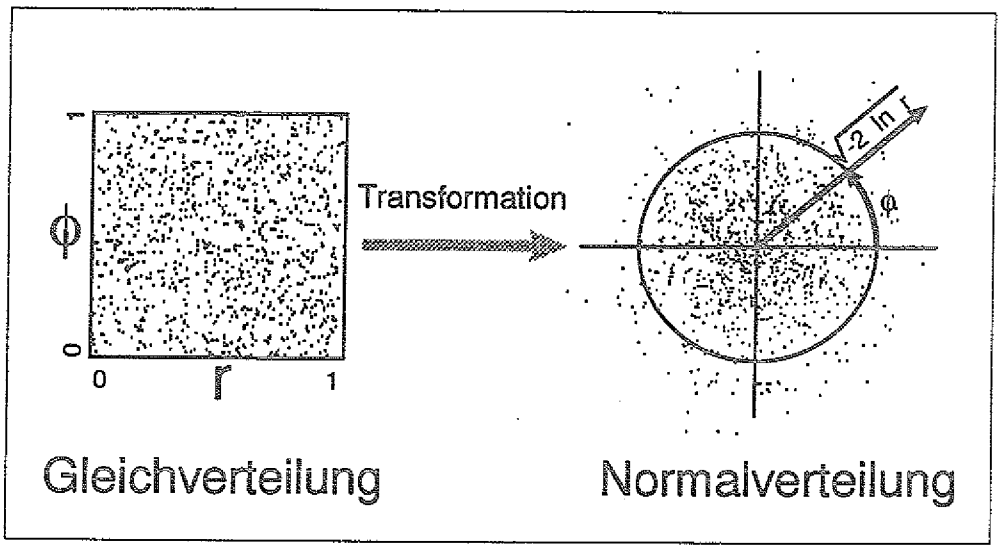


Abbildung 10.1: Box-Muller-Transformation

10.1.2.2 Die Box-Muller-Transformation

Bei der Normalverteilung scheitert dieser Ansatz daran, daß sich das Integral nicht analytisch lösen läßt. Hier verwendet man die schon in Abschnitt 6.2.1.2 vorgestellte verallgemeinerte Polarkoordinatentransformation

$$\begin{pmatrix} x \\ y \end{pmatrix} = T\begin{pmatrix} r \\ \phi \end{pmatrix} = \begin{pmatrix} t(r) \cdot \cos 2\pi\phi \\ t(r) \cdot \sin 2\pi\phi \end{pmatrix}$$

mit der speziellen Funktion

$$t(r) = \sqrt{-2 \ln r} \quad \Rightarrow \quad \det\,(T^{-1})'(T\begin{pmatrix} r \\ \phi \end{pmatrix}) = -\frac{1}{2\pi \cdot t(r) \cdot t\prime(r)} = \frac{r}{2\pi},$$

die dann als *Box-Muller-Transformation* bezeichnet wird. Für eine zweidimensionale Standard-Normalverteilung $f_Y(\xi)$ und eine $[0,1] \times [0,1]$-Gleichverteilung $f_X(\xi)$ folgt mit Hilfe der Transformationsformel (vgl. 6.2.1):

$$\begin{aligned} f_Y(T\begin{pmatrix} r \\ \phi \end{pmatrix}) &= f_Y\begin{pmatrix} t(r) \cdot \cos 2\pi\phi \\ t(r) \cdot \sin 2\pi\phi \end{pmatrix} \\ &= \frac{1}{2\pi} \cdot e^{-\frac{1}{2}t(r)^2(\cos^2 2\pi\phi + \sin^2 2\pi\phi)} \\ &= \frac{1}{2\pi} \cdot r \\ &= f_X\begin{pmatrix} r \\ \phi \end{pmatrix} \cdot \det(T^{-1})'(T\begin{pmatrix} r \\ \phi \end{pmatrix}) \\ &= f_{T(X)}(T\begin{pmatrix} r \\ \phi \end{pmatrix}). \end{aligned}$$

Folglich ist $f_{T(X)}$ eine Normalverteilungsdichte. Die Box-Muller-Transformation erlaubt somit, die Erzeugung je zweier normalverteilter Zufallszahlen aus zwei gleichverteilten Zufallszahlen. In [PFTV88] ist ein effizientes Verfahren zur Realisierung dieser Transformation beschrieben. Eine anschauliche Vorstellung von der Wirkungsweise der Transformation gibt die Abbildung 10.1.

10.1.2.3 Skalierung

Meist erzeugt man mit Hilfe einer Transformation T zunächst eine Zufallsvariable mit Erwartungswert 0 und Varianz 1. Diese kann dann mit Hilfe einer linearen Transformation auf beliebige Erwartungswerte und Varianzen skaliert werden (vgl. 6.2.1.1).

10.1.2.4 Rejektion

Die Rejektionsmethode ist ein effizientes Verfahren zur Erzeugung von Zufallszahlen mit einer ganz beliebigen parametrisch vorgegebenen Wahrscheinlichkeitdichte $f_Y(\xi)$. Das Verfahren ist in [PFTV88] beschrieben. Es benötigt lediglich eine Majorante

$$g(\xi) \geq f_Y(\xi),$$

die selbst nicht unbedingt eine Wahrscheinlichkeitsdichte sein muß, aber eine analytische Behandlung nach dem Transformationsverfahren zuläßt. Das Verfahren funktioniert um so effizienter, je besser $g(\xi)$ die Dichte $f_Y(\xi)$ approximiert. Im Grenzfall $g(\xi) = f_Y(\xi)$ geht das Rejektionsverfahren in die Transformationsmethode über.

Die Rejektionsmethode erlaubt insbesondere die Behandlung von empirisch vorgegebenen Verteilungen: Die Dichtefunktion einer realen Zufallsvariablen ist dazu anhand einer Stichprobe zu schätzen und mit einer geeigneten parametrisierten Funktion zu glätten. Dazu können z.B. glättende Spline-Funktionen verwendet werden. Auf die resultierende geglättete Dichtefunktion kann dann das Rejektionsverfahren angewendet werden.

Ein Problem bei dieser Vorgehensweise ist das Verhalten der Dichte im Bereich $\xi \to \pm\infty$. Hier liegt im allgemeinen keine ausreichende Zahl von Stichproben vor, um eine empirisch gestützte Glättungsfunktion einzusetzen. Stattdessen sollte man aus der Systemkenntnis heraus eine geeignete Funktion (z.B. Normalverteilung, Exponentialverteilung oder Cauchy-Verteilung) für die Randbereiche verwenden.

10.1.3 Kombination von Verteilungen

Auch die Techniken, die für die Kombination gegebener Zufallsvariablen

$$X_1, \ldots, X_m$$

zu einer neuen zufälligen Größe

$$X$$

anwendbar sind, sind in 6.2.4 bereits vorweggenommen:

10.1.3.1 Faltung

Die Faltung zweier Wahrscheinlichkeitsdichten erreicht man durch die Addition der zugehörigen Zufallsvariablen:

$$Y = X_1 + \ldots + X_m$$

10.1.3.2 Beschränkte Verteilungen

Die Einschränkung einer Zufallsvariable auf ein Intervall $[a, b]$ läßt sich auf algorithmischem Wege realisieren:

"Erzeuge eine Zufallszahl x, so lange bis $x \in [a, b]$"

Da die Grenzen $[a, b]$ in praktisch relevanten Fällen meist so gewählt sind, daß nur selten ein Wert außerhalb von $[a, b]$ liegt, ist diese Technik kaum mit einer Zeiteinbuße verbunden.

10.1.3.3 Mischverteilungen

Die Erzeugung einer Punktmassenverteilung

$$f_Y(\xi) = \lambda_1 \cdot \delta_{x_1}(\xi) + \ldots + \lambda_m \cdot \delta_{x_m}(\xi) \quad \text{mit} \quad \sum_i \lambda_i = 1$$

wird mit Hilfe einer Fallunterscheidung auf der Basis einer $[0,1]$-gleichverteilten Zufallsvariablen programmiert:

$$Y = \begin{cases} x_1 & \text{, falls } 0 \leq X < \lambda_1 \\ x_2 & \text{, falls } \lambda_1 \leq X < \lambda_1 + \lambda_2 \\ \vdots & \\ x_m & \text{, falls } \lambda_1 + \ldots + \lambda_{m-1} \leq X < \lambda_1 + \ldots + \lambda_m = 1 \,. \end{cases}$$

Insbesondere erhält man auf diese Weise eine Verteilervariable V mit

$$f_V(\xi) = \lambda_1 \cdot \delta_1(\xi) + \ldots + \lambda_m \cdot \delta_m(\xi)$$

und daraus die Mischung Y von $Y_1, \ldots, Y_m$ als:

$$Y = \begin{cases} Y_1 & \text{, falls } V = 1 \\ Y_2 & \text{, falls } V = 2 \\ \vdots & \\ Y_m & \text{, falls } V = m \end{cases}$$

Bei einer effizienten Implementierung muß nur jeweils eine der Variablen Y_i ausgewertet werden.

10.1.4 Ein Baukasten zur Erzeugung von Zufallszahlen

Die vorgestellten Techniken erlauben es, aus gegebenen Grundbausteinen ganz beliebige Zufallszahlengeneratoren zusammenzubauen. Diese Grundbausteine sind:

1. Ein Generator für $[0,1]$-gleichverteilte Zufallszahlen.
2. Transformationsverfahren zur Erzeugung von Zufallszahlen mit vorgegebener Verteilung unter Zuhilfenahme eines bereits vorhandenen Zufallszahlengenerators:
   a) Eindimensionale Transformation,
   b) Mehrdimensionale Transformation,
   c) Rejektion,
   d) Skalierung.
3. Kombination von Zufallszahlengeneratoren:
   a) Faltung,
   b) Beschränkung,
   c) Mischung.

10.2 Regression

Das allgemeine lineare Regressionsproblem besteht darin, den Parametervektor β in einem Regressionsmodell

$$y = X \cdot \beta + \epsilon$$

aufgrund der Matrix X der unabhängigen und des Vektors y der abhängigen Variablen zu schätzen. Ein *Schätzer* für das unbekannte, aber feste β hat dann die Gestalt

$$\hat{\beta} = \hat{\beta}(X, y).$$

Da die Schätztheorie als einer der Hauptzweige der mathematischen Statistik sehr umfangreich ist, sollen hier nur die Grundlagen zusammengefaßt werden, die zum Verständnis der Beispiele unmittelbar benötigt werden.

10.2.1 Gütekriterien für Schätzer

10.2.1.1 Definition von "besten" Schätzern

Da $\hat{\beta} = \hat{\beta}(X, y)$ über y implizit auch von ϵ abhängt, ist $\hat{\beta}$ selbst wieder eine Zufallsvariable und kann unterschiedliche Werte bei verschiedenen Identifikationsexperimenten annehmen. Ein "guter" Schätzer $\hat{\beta}$ ist ein solcher, der den Parameter β "im Mittel am besten" schätzt. Diese informelle Definition hat keine eindeutige mathematische Formalisierung und damit auch keine eindeutige Lösung. Vielmehr muß ein spezielles Gütekriterium

$$\kappa(\beta, \hat{\beta})$$

vorgeben werden, dessen Minimierung den *besten* Schätzer liefert:

$$\kappa(\beta, \hat{\beta}) \rightarrow \min$$

Verschiedene Gütekriterien können somit verschiedene "beste" Schätzer $\hat{\beta}$ liefern. Um das Optimierungsproblem rechnerisch handhabbar zu machen, beschränkt man sich meist auf eine gewisse Klasse S von Schätzern:

$$\hat{\beta} = \hat{\beta}(X, y) \in S$$

10.2.1.2 Beispiele für Minimierungskriterien

a) Minimierung der mittleren quadratischen Abweichung:

$$\kappa(\beta, \hat{\beta}) = \mathrm{E}[\beta - \hat{\beta}]^2$$

b) In Verallgemeinerung zu a) kann ein beliebiges Abstandsmaß (genauer eine Metrik [Bri83]) minimiert werden:

$$\kappa(\beta, \hat{\beta}) = \mathrm{E}[f(\beta - \hat{\beta})]$$

Insbesondere kennt man die verallgemeinerten Normen

$$f(x) = x^T Q\, x \quad \text{mit} \quad Q = Q^T, Q \geq 0\,.$$

c) Ein anderer Ansatz ist der, nicht die Differenz $(\beta - \hat{\beta})$ zu betrachten, sondern vielmehr die Vorhersagegüte des Modells, also $y - X\,\hat{\beta}$. Man gelangt mit

$$\kappa(\beta, \hat{\beta}) = \mathrm{E}\left[(y - X\,\hat{\beta})^T\, Q\, (y - X\,\hat{\beta})\right] \qquad (Q = Q^T, Q \geq 0)$$

zu den *Kleinste Quadrate*-Schätztern (kurz KQ-Schätzer). Die Namensgebung beruht darauf, daß für den speziellen Fall $Q = 1$ und $y = (y_i)_{i=1,\ldots,n}$, $X = ((X_{ij}))_{i=1,\ldots,n,\, j=1,\ldots,m}$ gilt:

$$(y - X\,\beta)^T\, Q\, (y - X\,\beta) = \sum_{i=1}^{n} \left[y_i - \sum_{j=1}^{m} X_{ij}\,\beta_j\right]^2$$

d) Auch das in c) verwendete Abstandsmaß läßt sich mit einer beliebigen Abstandsfunktion verallgemeinern. Zu nennen sind hier vor allem Distanzmaße zu robusten Schätzung bei vorliegenden Ausreißern, insbesondere die sogenannte *Huber-Metrik* [Hub81].

10.2.1.3 Beispiele für spezielle Klassen von Schätzern

a) Für die spezielle Klasse

$$S = \{\, \hat{\beta} \,|\, \mathrm{E}[\hat{\beta}] = \beta \,\}$$

der *unverzerrten Schätzer*, die den Parameter β "im Mittel richtig" schätzen, ist das Kriterium der minimalen mittleren quadratischen Abweichung gleich dem *Kovarianzkriterium*:

$$\kappa(\beta, \hat{\beta}) = \mathrm{Cov}\,[\beta - \hat{\beta}]$$

b) Eine besonders einfache Klasse von Schätzern bilden die *linearen* Schätzer des Typs

$$S = \{\hat{\beta} \,|\, \hat{\beta}(X, y) = A(X) \cdot y\},$$

mit einer Matrix A, die von X abhängen darf.

10.2.2 Lineare Regression

Für das lineare Regressionsproblem können explizite Lösungen angeben werden, wenn man sich auf unverzerrte lineare Schätzer beschränkt.

10.2.2.1 Lineare Regression bzgl. des Kovarianzkriteriums

Für dieses Kriterium gilt das *Gauß-Markov-Theorem*:

Der beste lineare unverzerrte Schätzer (BLUE)

$$\hat{\beta}(X, y)\ \epsilon\ S = \{\hat{\beta} \,|\, \hat{\beta}(X, y) = A(X) \cdot y \wedge \mathrm{E}[\hat{\beta}] = \beta\,\}$$

für das linerare Regressionsproblem

$$y = X \cdot \beta + \epsilon \quad \text{mit} \quad \mathrm{Cov}\,[\epsilon] = \Sigma$$

bzgl. des Kriteriums

$$\kappa(\beta, \hat{\beta}) = \mathrm{Cov}\,[\beta - \hat{\beta}] \rightarrow \min$$

ist gegeben durch

$$\hat{\beta} = A(X) \cdot y \stackrel{def}{=} (X^T \Sigma^{-1} X)^{-1} X^T \Sigma^{-1} \cdot y \ .$$

Das Minimierungskriterium — ausgewertet für den BLUE — ergibt

$$\mathrm{Cov}\,[\beta - \hat{\beta}] = X^T \Sigma^{-1} X^{-1}.$$

Einige Verfahren zur numerisch stabilen Berechnung des BLUE finden sich in [FH84]

10.2.2.2 Lineare Regression bzgl. des Kleinste-Quadrate-Kriteriums

Für die kleinsten Quadrate erhält man im linearen Fall interessanterweise dasselbe Resultat:

Der beste lineare unverzerrte KQ-Schätzer bzgl. des Kriteriums

$$\kappa(\beta, \hat{\beta}) = \mathrm{E}\,[y - X \cdot \hat{\beta}]^T \Sigma^{-1} (y - X \cdot \hat{\beta}) \rightarrow \min$$

kann berechnet werden aus

$$\hat{\beta}(X, y) = \min_{\bar{\beta}} \ (y - X \cdot \bar{\beta})^T \Sigma^{-1} (y - X \cdot \bar{\beta}).$$

Er ist identisch mit dem BLUE.

Der Beweis beruht unmittelbar auf der Ableitung:

$$\frac{\partial}{\partial \beta} [(y - X \cdot \beta)^T \Sigma^{-1} (y - X \cdot \beta)] = 2\, X^T \Sigma^{-1} X\, \beta - 2\, X^T \Sigma^{-1} y \stackrel{!}{=} 0$$

Wichtig ist, daß der zweite Taylor-Koeffizient des Kriteriums $\kappa(\beta, \hat{\beta})$, also

$$\frac{1}{2} \frac{\partial^2}{\partial \beta^2} [(y - X \cdot \beta)^T \Sigma^{-1} (y - X \cdot \beta)] = X^T \Sigma^{-1} X \ ,$$

gerade die Kovarianzmatrix $\mathrm{Cov}\,[\beta - \hat{\beta}]$ des KQ-Schätzers $\hat{\beta}$ ist.

10.2.3 Nichtlineare Regression

Der genannte Satz über den KQ-Schätzer beim linearen Modell liefert zugleich ein Rezept zur numerischen Berechnung des besten Schätzers. Bei der Verallgemeinerung auf das nichtlinearen Regressionsproblems

$$y = f\,(x, \beta) + \epsilon$$

wird daher ein Kleinste-Quadrate-Ansatz bevorzugt. Es muß angemerkt werden, daß der Satz in der nichtlinearen Situation nur noch approximativ gilt, d.h. die Minimierung des Kriteriums

$$\kappa\,(\beta, \hat{\beta}) = \sum_{i=1}^{n} \frac{(y_i - f\,(x_i, \hat{\beta}))^2}{\sigma_i^2}$$

läßt nur einen suboptimalen Schätzer erwarten.

10.2.3.1 Linearisierung

Liegt eine Lösung des Minimierungsproblems vor, so kann $\kappa(\beta, \hat{\beta})$ im Punkte $\hat{\beta}$ durch eine Taylor-Entwicklung zweiten Grades bzgl. $\hat{\beta}$ approximiert werden:

$$\kappa(\beta, \hat{\beta} + \Delta\beta) \approx \kappa(\beta, \hat{\beta}) + \frac{1}{2} \frac{\partial^2}{\partial \hat{\beta}^2} \kappa(\beta, \hat{\beta}) \cdot \Delta\hat{\beta}^2$$

Diese Näherung entspricht einem linearen Regressionsproblem. Daraus resultiert eine Näherung der Kovarianzmatrix des Schätzers:

$$\mathrm{Cov}[\beta - \hat{\beta}] \approx \frac{1}{2} \frac{\partial^2}{\partial \hat{\beta}^2} \kappa(\beta, \hat{\beta})$$

10.2.3.2 Optimierverfahren

Die nichtlineare Parameteranpassung ist ein spezielles Optimierungsproblem des Typs

$$f(\beta) \rightarrow \min, \quad \beta \in M \subset \Re^n$$

und damit Gegenstand eines ganzen Forschungsbereiches. Dementsprechend gibt es eine Vielzahl von nichtlinearen Optimierungsverfahren, die hier nicht diskutiert werden sollen (ein Standardwerk der Optimierung ist z.B. [Fle87]). Das bekannteste und immer weiter verbesserte Verfahren für die Parameteranpassung ist der *Levenberg-Marquardt-Algorithmus* [Mor78], der auf der speziellen Struktur des Parameteranpassungsproblems aufsetzt.

10.2.3.3 Problematik der nichtlinearen Optimierung

Bei der Parameteranpassung (so wie bei der Optimierung allgemein) treten zwei Probleme auf, die in den Lehrbüchern meist nicht oder nur kurz abgehandelt werden, aber die eigentliche Schwierigkeit beim Entwurf praktisch anwendbarer Algorithmen ausmachen:

Bereichsgrenzen für die Parameter :
: Die Formulierung der Standard-Optimierverfahren erfolgt in der Regel für den gesamten Raum $\Re^m$. Die Parameter in Modellen für natürliche Systeme haben dagegen Nebenbedingungen für den physikalisch sinnvollen Bereich. Meist werden diese Nebenbedingungen durch Intervallgrenzen

$$u_i \leq \beta_i \leq o_i \ , \ i = 1, \ldots, m$$

oder allgemeiner durch einen Satz von Gleichungen in der Form

$$f_j(\beta_1, \ldots, \beta_m) \geq 0 \ , \ j = 1, \ldots, s$$

vorgegeben. Damit können beliebige Gebiete $M \subset \Re^m$ als zulässige Bereiche definiert werden.

Solche Nebenbedingungen können zur Folge haben, daß das gesuchte Minimum auf dem Rand des Bereichs M liegt. In diesem Fall schießen die üblichen Verfahren ohne eine Modifikation über das Gebiet M hinaus. Eine einfache Einschränkung des Suchbereichs genügt zur Behebung dieses Fehlers nicht, da ein so modifiziertes Verfahren am Rand von M in eine endlose Iteration gerät. Einige Ansätze zur Lösung dieses Problems finden sich in [BS86a] [BS86b] [Pow78].

Globale Konvergenz :
: Das große Problem aller Optimierungsalgorithmen ist die Garantie der Konvergenz gegen ein Minimum von $f(\beta)$ mit einer gewissen Konvergenzgeschwindigkeit. Im allgemeinen kann nur bewiesen werden [SB73],

- daß ein Verfahren gegen ein lokales (aber nicht unbedingt globales) Minimum von f konvergiert

- daß es eine gewisse Konvergenzgeschwindigkeit hat, wenn der Startwert β_0 des Verfahrens bereits in der Nähe des Minimums liegt.

Derartige theoretische Resultate sagen über die praktische Anwendbarkeit eines Verfahrens nicht viel aus. Speziell für das Parameteranpassungsproblem gibt es derzeit einige heuristische Ansätze, die eine stark verbesserte globale Konvergenz erwarten lassen [Boc83], diese jedoch auch nicht garantieren können. Dazu soll in Abschnitt 10.3 noch ein Beispiel gegeben werden.

10.2.4 Parameterschätzung beim Sprung-Modell

10.2.4.1 Diskrete Form des Modells

Betrachtet wird das Sprungmodell aus 6.7 in der einfachsten Situation, d.h. mit konstanten Funktionen

$$f_< (\alpha_<, t) = \alpha_< \quad \text{und} \quad f_\geq (\alpha_\geq, t) = \alpha_\geq$$

vor und nach dem Sprungzeitpunkt t_{step}. Da in der Praxis stets diskrete Meßdaten vorliegen, ist das einfache Sprungmodell besser in der diskreten Form

$$f(s, \Delta\alpha, i) = \begin{cases} \alpha_< & ,1 \leq i \leq s-1 \\ \alpha_< + \Delta\alpha & ,s \leq i \leq n \end{cases} + \epsilon_i \quad \text{mit} \quad \mathrm{E}[\epsilon_i] = 0, \mathrm{Var}[\epsilon_i] = \sigma^2$$

mit der Sprunghöhe $\Delta\alpha$ und der Sprungstelle s handhabbar. Dabei wird $\alpha_<$ (z.B. aufgrund einer Schätzung durch ein Filter) als bekannt angenommen. Die Zeitpunkte $t_1, \ldots, t_n$ müssen nicht äquidistant liegen; wichtig ist nur, daß die Rauschterme ϵ_i als unabhängig mit gleicher Varianz angenommen werden können.

10.2.4.2 KQ-Schätzung

Es sei nun ein Satz von Meßwerten

$$y_1, \ldots, y_n$$

vorgegeben. Dann erlaubt ein KQ-Ansatz die Konstruktion zweier Schätzer

$$\begin{array}{rll} & \hat{s} & \text{für den Sprungzeitpunkt} \\ \text{und} & \hat{\Delta\alpha} & \text{für die Sprunghöhe.} \end{array}$$

Das Kleinste-Quadrate-Kriterium ist

$$\kappa(s, \Delta\alpha) = \sum_{i=1}^{s-1} (y_i - \alpha_<)^2 + \sum_{i=s}^{n} (y_i - \alpha_< - \Delta\alpha)^2.$$

10.2.4.3 Konstruktion der KQ-Schätzer

Gesucht ist das Minimum

$$(\hat{s}, \hat{\Delta\alpha})$$

von $\kappa(s, \Delta\alpha)$. Minimierung über $\Delta\alpha$ durch Ableiten und Nullsetzen liefert nach kurzer Rechnung:

$$\hat{\Delta\alpha}(s) = \frac{1}{n-s+1} \sum_{i=s}^{n} (y_i - \alpha_<)$$

Anschließendes Minimieren von

$$\begin{aligned}\kappa(s,\hat{\Delta\alpha}(s)) &= \sum_{i=1}^{s-1}(y_i-\alpha_<^2) \\ &\quad +\sum_{i=s}^{n}(y_i-\alpha_<^2) \quad -2\hat{\Delta\alpha}(s)\cdot\sum_{i=s}^{n}(y_i-\alpha_<) \qquad +(n-s+1)\cdot\hat{\Delta\alpha}(s)^2 \\ &= \sum_{i=1}^{n}(y_i-\alpha_<^2) \quad -2\hat{\Delta\alpha}(s)\cdot\sum_{i=s}^{n}(y_i-\alpha_<-\tfrac{1}{2}\hat{\Delta\alpha}(s)) \\ &= \text{const.} \quad -2\hat{\Delta\alpha}(s)\cdot\sum_{i=s}^{n}(y_i-\alpha_<-\tfrac{1}{2}\hat{\Delta\alpha}(s))\end{aligned}$$

über s ergibt das globale Minimum

$$\hat{s} = \arg\max_{1\le s\le n} \hat{\Delta\alpha}(s)\cdot\sum_{i=s}^{n}(y_i-\alpha_<-\frac{1}{2}\hat{\Delta\alpha}(s))$$

[1] und schließlich den globalen Sprunghöhenschätzer

$$\hat{\Delta\alpha} = \hat{\Delta\alpha}(\hat{s})\ .$$

Die Schätzer hängen nicht von der Varianz σ^2 des Rauschens ab. Dieser Parameter spielt erst bei dem später beschriebenen Test auf Signifikanz des Sprungs eine Rolle (siehe 10.5.3). Ferner sei bemerkt, daß die obigen Rechnungen auf allgemeine lineare Regressionsansätze für $f_<$ und $f_\ge$ erweitert werden können [WJ76].

10.3 Kritische Bemerkungen zur Anwendung der Parameteranpassung

Obwohl die Parameteranpassung mit Hilfe der Methode der kleinsten Quadrate weithin verwendet wird, ist bzgl. der Aussagekraft der Ergebnisse einige Skepsis angebracht. Dies gilt vor allem dann, wenn sie auf dynamische Systeme angewendet wird, die zeitlichen Schwankungen unterliegen. Einige Beispiele sollen dies verdeutlichen:

10.3.1 "Vertikale" und "horizontale" Minimierungskriterien

10.3.1.1 Parameteranpassung bei einer Sinus-Funktion

Der KQ-Ansatz geht von einem "vertikalen" Fehlermodell aus, d.h. bei der Anpassung einer Funktion

$$f(\beta,t)$$

an einen Datensatz

$$(t_1,y_1),(t_2,y_2),\ldots,(t_n,y_n)$$

bestimmen die vertikalen Abstände

$$|f(\beta,t_i)-y_i|$$

das Minimierungskriterium. Welche Konsequenzen dieses Vorgehen haben kann, zeigt das Beispiel der Anpassung einer Sinusfunktion

$$f(A,\omega,t) = A\cdot\sin(\omega t)$$

bzgl. Amplitude A und Winkelgeschwindigkeit ω an einen idealen Datensatz

$$y_i = \sin(2\pi\frac{i}{N}),\quad i=1,\ldots,N\cdot M$$

[1] Bedingt durch das negative Vorzeichen ist ein Maximum statt eines Minimums gesucht.

(M Schwingungen mit je N Abtastpunkten). Die Abbildung 10.2 zeigt das Niveauliniendiagramm des Minimierungsfunktionals für $N = 8$ und $M = 4$. Offenbar ist eines der üblichen Minimierungsverfahren, die im wesenlichen auf der Verfolgung von Gradienten beruhen, kaum in der Lage, das globale Minimum zu finden! Nur wenn der Startwert sehr nahe bei $(A,\omega) = (1,1)$ liegt, kommt es zur Konvergenz; andernfalls findet das Verfahren mit großer Wahrscheinlichkeit nur das "große" Nebenminimum A bei $(A,\omega) = (-0.2606, 0.6339)$ oder eines der anderen Nebenminima. Das "Anpassungsergebnis" für die verschiedenen Nebenminima gibt ebenfalls Abbildung 10.2 wieder.

10.3.1.2 Ursache der schlechten Konvergenz

Zur Auffindung der Ursache für das schlechte Ergebnis genügt die Auftragung der Funktion $f(A,\omega,t)$ für Startwerte, die nicht zur Konvergenz führen, im Vergleich zum gesuchten Ergebnis $f(1,1,t)$ (Abb. 10.3a). Es wird sofort deutlich, daß die vertikalen Abstände nur auf der ersten aufsteigenden Flanke der Sinusfunktion einen Beitrag zur Minimierung leisten. Mit fortschreitender Zeit verschieben sich die Phasen von Meßpunkten und Anpassungsfunktion, so daß das "vertikale" Maß wertlos wird. Insbesondere gibt es Bereiche der Gegenläufigkeit, in denen die Ableitungen von $f(A,\omega,t)$ und $f(1,1,t)$ verschiedene Vorzeichen haben. In diesen Bereichen beinflussen die vertikalen Abstände den Minimierungsradienten genau in der falschen Richtung!

Diese Überlegung zeigt, daß ein "horizontales" Abstandsmaß besser geeignet wäre, bei dem jeder Meßpunkt mit dem entsprechenden gleichphasigen Funktionswert verglichen wird (vgl. auch Abb. 10.3b). Dazu ist eine Streckung der Zeitachse um den Faktor $\frac{1}{\omega}$ erforderlich:

$$|f(A,\omega,\frac{1}{\omega}t_i) - y_i|$$

10.3.1.3 Ein lokaler Ansatz zur globalen Stabilisierung

Das "horizontale" Abstandsmaß ist rein theoretischer Natur, da hierzu der wahre Parameter $\omega = 1$ bekannt sein müßte. Trotzdem kann das Prinzip aufgegriffen werden. Die Beobachtung, daß das KQ-Kriterium nur so lange nützlich ist, wie Anpassungsfunktion und Meßwerte gleichphasig verlaufen, führt auf den Ansatz, die Funktion $f(A,\omega,t)$ mit einer Phasenverschiebung ϕ zu versehen und abschnittsweise die Funktion

$$f(A,\omega,\phi,t) = A \cdot \sin(\omega t + \phi)$$

anzupassen. Dies illustriert die Abbildung 10.4.

Eine solche Phasenverschiebung läßt sich z.B. durch den folgenden Ad-hoc-Ansatz erreichen, der keinen Anspruch auf Eleganz oder Verallgemeinerbarkeit erhebt: ω wird für alle Punkte global vorgegeben. An der Stelle t_i berechne man A_i und ϕ_i so, daß gilt

$$\begin{array}{lllll} \text{i)} & sin(t_i) & = f(A,\omega,\phi,t_i) & = y_i \\ \text{ii)} & cos(t_i) & = f'(A,\omega,\phi,t_i) & \approx \frac{y_{i+1}-y_i}{t_{i+1}-t_i} \end{array} .$$

In das Kriterium geht nun die Differenz zum nächsten Meßwert an der Stelle t_{i+1} ein (vgl. Abb. 10.4):

$$|f(A_i,\omega,\phi_i,t_{i+1}) - y_{i+1}|.$$

Damit der globale Parameter A überhaupt im Minimierungskriterium enthalten ist, wird zusätzlich den Term

$$|A_i - A|.$$

addiert. Das Niveauliniendiagramm für ideale Werte y_i zeigt die Abbildung 10.4. Es bereitet offenbar keine Schwierigkeit, das Minimum zu finden!

Eine Anmerkung ist noch erforderlich: Da der Ad-hoc-Ansatz nur eine Approximation der Ableitung durch den Differenzenquotienten [2] verwendet, ist das gefundene Minimum mit einem Fehler behaftet. Aus diesem Grunde sollte es anschließend als Startwert für die oben beschriebene gewöhnliche Parameteranpassung verwendet werden, um das exakte Minimum zu iterieren.

[2] Die Problematik dieser Approximation bei verrauschten Daten soll hier außer acht gelassen werden.

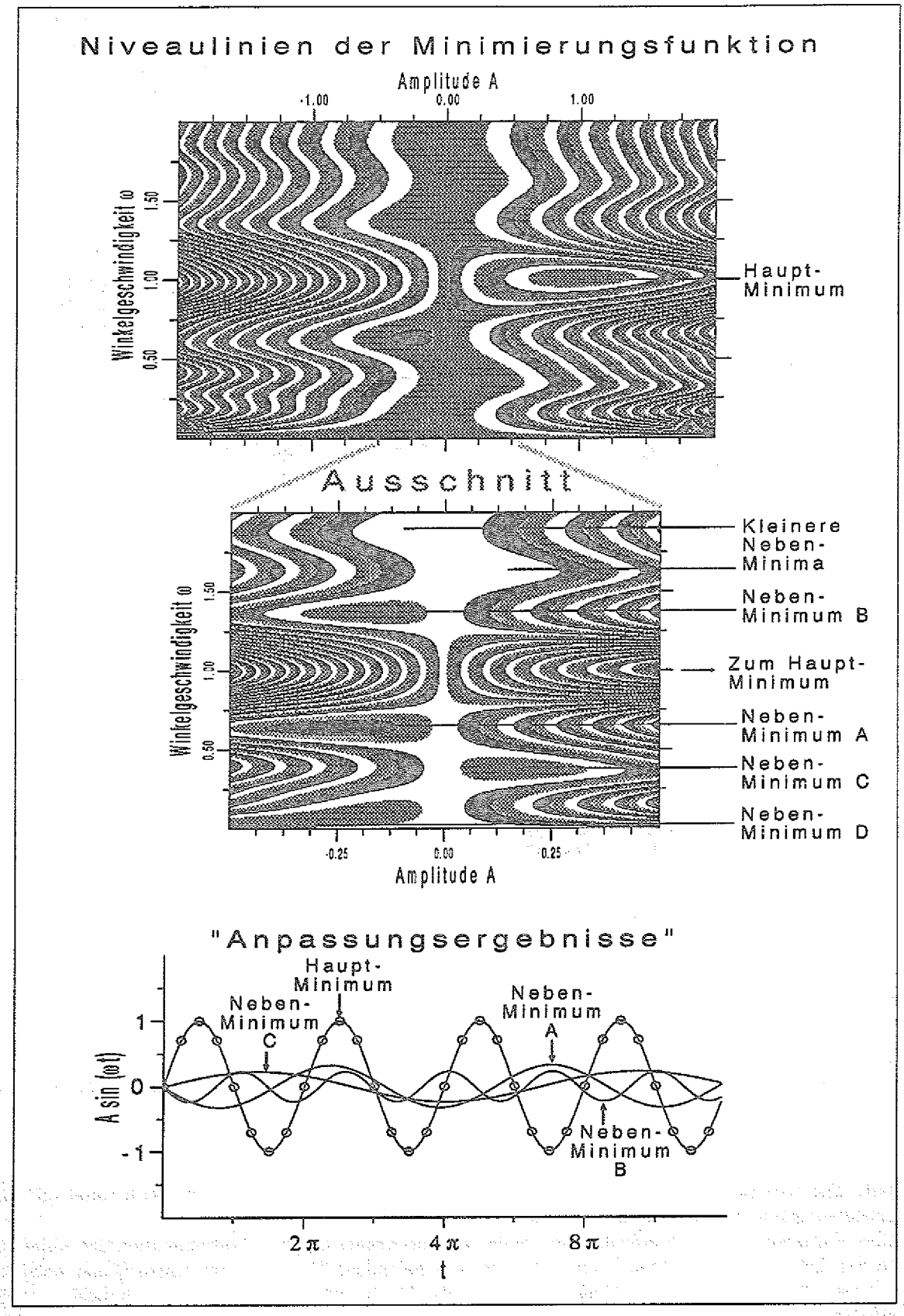


Abbildung 10.2: Niveauliniendiagramm für die Anpassung einer Sinusfunktion

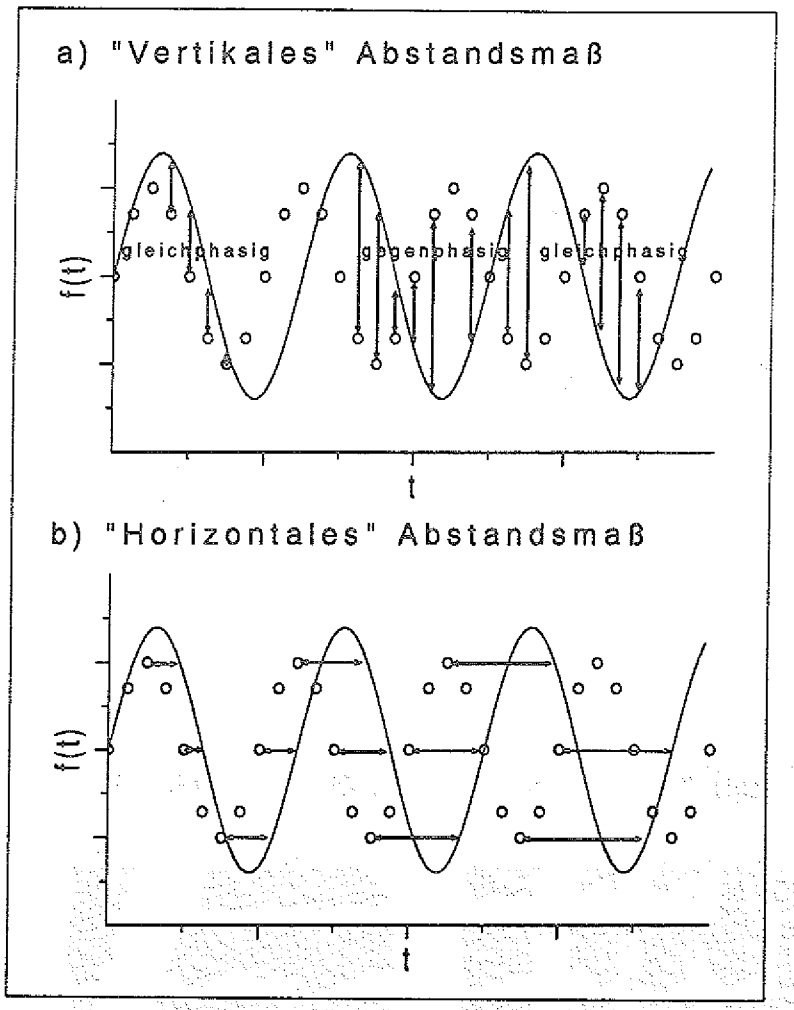


Abbildung 10.3: "Vertikale" und "horizontale" Minimierungskriterien

10.3.1.4 Verallgemeinerung der Ergebnisse

Die obige Sinusfunktion kann alternativ durch die Differentialgleichung

$$\ddot{x} = -\omega x, \quad x(0) = 0, \; \dot{x}(0) = A$$

beschrieben werden. Die festgestellten Effekte sind im Prinzip auf beliebige Differentialgleichungs-Anpassungsprobleme übertragbar; allerdings treten sie bei periodischen Systemen am deutlichsten hervor. Der Grundgedanke des oben angewendeteten lokalen Ansatzes zur globalen Stabilisierung läßt sich übertragen. Darauf beruhen das leistungsfähige Verfahren von Bock [Boc83] und der in der Modellbank Biotechnologie verwendete Algorithmus [Gol89].
Interessant ist nun eine ganz andere Fragestellung:

Was passiert, wenn die Dynamik des Systems stochastischen Schwankungen unterliegt?

Z.B. könnte der Parameter ω im obigen System stochastisch schwanken (d.h. $\omega = \omega(t)$). Die Konsequenz wäre eine Schwingung mit wechselnder Frequenz! Das Minimierungsproblem in der "vertikalen" Formulierung hat jetzt überhaupt keinen Sinn mehr, wogegen ein "horizontaler" lokaler Ansatz immerhin noch eine "durchschnittliche" Winkelgeschwindigkeit $\bar{\omega}$ schätzen kann [3].

[3] Unter diesem Gesichtspunkt werden in [JJM86] "Äpfel mit Birnen verglichen".

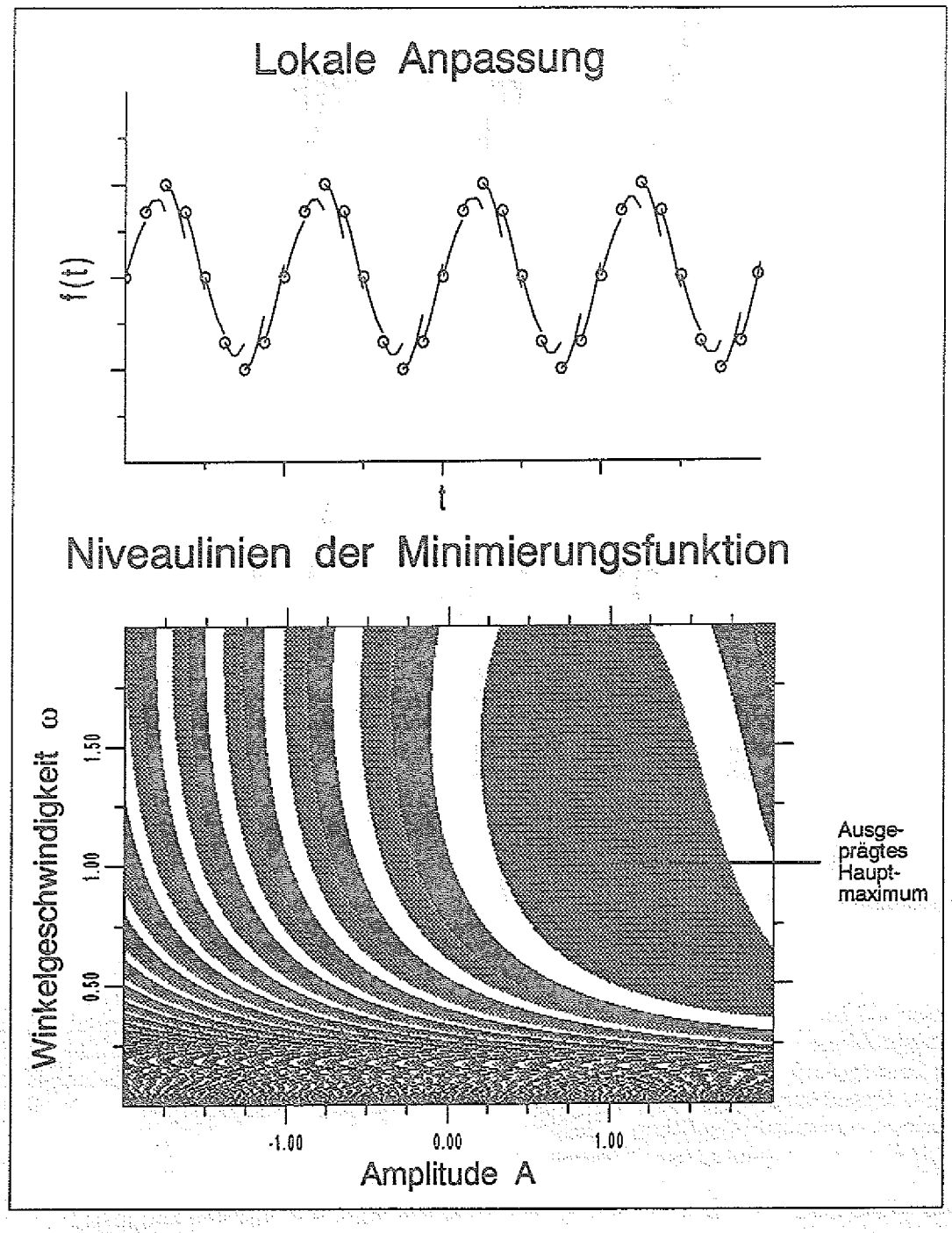


Abbildung 10.4: Global stabilisierte Parameteranpassung

10.3.2 Biotechnologische Relevanz

Die KQ-Parameteranpassung geht von einem rein deterministischen Systemmodell aus, womit insbesondere auch die Vorstellung der Vorhersagbarkeit über größere Zeiträume und die damit verbundenen Anwendungen der Kontrolltheorie verbunden sind. Es muß gefragt werden, ob diese Annahme bei biologischen Systemen haltbar ist.

Diese Frage läßt sich sicherlich nicht allgemein beantworten und würde einiger Fallstudien bedürfen, jedoch liegt ein Schwerpunkt dieser Arbeit darin, die Alternative einer stochastischen Systembeschreibung herauszustellen. Schwankungen in der Systemdynamik können mit Hilfe von stochastischen Modellen ohne weiteres aufgefangen werden, da die Unschärfe der Parameter eine der wesentlichen Eigenschaften stochastischer Modelle ist. Die Festlegung auf eine stochastische Sichtweise biotechnischer Systeme steht allerdings einer langfristigen Vorhersage des Systemverhaltens prinzipiell entgegen! Dies erfordert ein Umdenken bzgl. der Methodik.

10.3.2.1 Parameteranpassung bei Lag-Phasen und Wachstumsstörungen

Ein besonderes Licht auf das Problem wirft die Anpassung von Wachstumsmodellen bei Vorliegen von Lag-Phasen oder Wachstumsstörungen. Dies sei an dem elementaren Wachstumsmodell

$$\dot{X} = \mu \cdot X, \quad X(0) = X_0$$

erläutert. Durch Übergang zu einem zeitabhängigen μ kann sowohl eine anfängliche Lag-Phase, als auch eine zwischenzeitliche Wachstumsstörung beschrieben werden in der Form

$$\mu(t) = \begin{cases} \mu_0 & \text{, falls } t < t_1 \\ \mu_1 & \text{, falls } t_1 \leq t < t_2 \\ \mu_0 & \text{, falls } t_2 \leq t \end{cases}$$

mit $0 = \mu_0 < \mu_1$ und $0 < t_1 < t_2 = \infty$ für eine Lag-Phase

bzw. $0 \leq \mu_1 < \mu_0$ und $0 < t_1 < t_2$ für eine Störung.

Diese Situation reproduziert genau das oben skizzierte Problem: Da sich die Dynamik des Systems verändert, kommt es zu einem zeitverzögerten Wachstum. Die Parameteranpassung mit dem einfachen exponentiellen Wachstumsmodell kann diese Verzögerung aufgrund des "vertikalen" Fehlermodells auf keine Weise berücksichtigen. Die Abbildung 22.10 zeigt ein Anpassungsergebnis in einer ganz ähnlichen Situation.

Natürlich besteht der Einwand, daß das einfache exponentielle Modell aufgrund des Modellfehlers nicht in der Lage ist, eine Lag-Phase zu beschreiben und deshalb um einen Verzögerungsparameter t_1 zu erweitern ist. Es stellt sich dann die Frage, welche praktische Bedeutung der so gewonnene Parameter t_1 hat: Die Dauer einer Lag-Phase hängt bei Mikroorganismen stark von den Vorbedingungen ab und kann im allgemeinen nicht vorhergesagt werden. Ein Versuch einer Modellierung findet sich in [PHB78]. D.h. der Parameter t_1 ist zwar *a posteriori* schätzbar, hat aber für die *a priori*-Vorhersage des Prozeßverlaufs keine Relevanz! Ein Verfahren, daß — von einer konstanten Lag-Phase ausgehend — eine Vorhersage des exponentiellen Verlaufs macht, wird gegen Ende der exponentiellen Phase unter Umständen eine Vorhersagedifferenz von mehreren Zehnerpotenzen aufweisen!

10.3.2.2 Das Phasenkonzept für die Datenanalyse und Prozeßüberwachung

In [Ber83] wird eine Parameteranpassung für ein Modell mit einem Verzögerungsparameter durchgeführt. Hier kommt interessanterweise die folgende Anpassungsstrategie zur Anwendung [4]:

1. Anpassung des Verzögerungsparameters durch ("horizontales") Ausmessen der Lag-Phase.
2. ("Vertikale") Parameteranpassung für die exponentielle Phase.

[4] Ein weiteres interessantes Beispiel, die Bestimmung des $k_L a$-Wertes mit der sogenannten dynamischen Methode wird in Abschnitt 22.3 vorgestellt.

Diese hybride Vorgehensweise scheint auch für den On-Line-Einsatz in vielen biotechnischen Situationen die geeignete Methode zu sein (vgl. auch [Hal89]):

- Nicht oder nur schlecht vorhersagbare Verzögerungs-Parameter sind mit geeigneten lokal arbeitenden Verfahren (wie z.B. dem in 2.5 beschriebenen Sprungdetektor für das Ende einer Lag-Phase) zu bestimmen.
- Abschnitte des Wachstums, die weitgehend deterministisch vorhersagbar sind — also insbesondere keinen Störungen unterliegen — , werden mit klassischen Parameteranpassungstechniken behandelt.
- Störungen, d.h. Abweichungen vom idealen Verlauf, müssen durch Verfahren zur Störungserkennung (z.B. durch die Auswertung von redundanter Information bei der Bilanzierung, vgl. 6.6.3.1) erkannt werden.

Erneut ergeben sich weitreichende Konsequenzen für die Struktur eines Prozeßüberwachungssystems: Es muß umfangreiche Möglichkeiten zur Phasenablaufsteuerung in Verbindung mit dem parallelen Betrieb mehrerer kommunizierender Datenanalysebaustein bereitstellen! Einige Schritte in Richtung auf eine deratige Flexibilität sind in dieser Arbeit unternommen worden (vgl. Kapitel 4 und 20).

10.4 Stationäre Identifikation

10.4.1 Der stationäre Betrieb der kontinuierlichen Kultur

10.4.1.1 Die stationären Gleichungen

Ein klassischer Ansatz zur Bestimmung der Stoffwechselparameter eines Mikroorganismus ist die *stationäre Identifikation* in einer kontinuierlichen Kultur. Mit Beschränkung auf die Bilanzgleichungen für nicht gasförmige Komponenten (vgl. 5.3.3.1) und der Annahme, daß alle diese Komponenten durch denselben Zufluß zugeführt werden, erhält man aus dem allgemeinen Stoffwechselmodell die Beziehung [5]

$$\dot{X} = N \cdot X_1 \cdot r(X) + D \cdot (X_{in} - X)$$

mit

dem Vektor der Konzentrationen	X	insbesondere
der Biomasse	X_1	,
den Zulaufkonzentrationen	X_{in}	,
der stöchiometrischen Matrix	N	,
dem Kinetik-Vektor	$r(X)$	und
der Durchflußrate	D	.

In vielen Fällen stellen sich bei vorgegebenem $D = D^*$ stabile stationäre Konzentrationen

$$X^* = X^*(D^*)$$

der Komponenten ein. Die Abhängigkeit $X^*(D^*)$ wird dabei durch die stationäre Gleichung

$$\dot{X} = 0 \iff N \cdot X_1 \cdot r(X^*) = D \cdot (X^* - X_{in})$$

ausgedrückt. Die Bedingungen für die Existenz solcher stationären Zustände wurden vielfach untersucht [AL84] und sollen hier nicht eigens behandelt werden. Bei Vorliegen einer Instabilitität ist es meist möglich, den stationären Zustand durch Regelung von D von außen zu stabilisieren (Turbidostat, Nutristat, Bistat [AL84] [MKE89] [EKB72] [SCAGGM89]). Nach Eintreten der Stabilisierung wird dabei D (bis auf geringe Regelschwankungen) auf einen festen Wert D^* eingeregelt.

[5] Die Einschränkung auf nicht gasförmige Komponenten erfolgt nur aus Gründen der Einfachheit der Formeln. Sie kann ohne weiteres entfallen.

Durch Variation von D^* kann dieses Experiment nun dazu verwendet werden, die Abhängigkeit $X^*(D^*)$ auszumessen. Sofern das obige stationäre Gleichungssystem ausreichend bestimmt ist (vgl. 6.6.3.1), legt es die Reaktionsraten $r = r(X^*)$ eindeutig fest. Eine Auftragung der Reaktionsraten $r(X^*)$ über X^* führt dann zu einem klassischen Parameteranpassungsproblem für die Stoffwechselkinetiken $r(X)$.

Bemerkung: Wegen der Unschärfe von D^* (Turbulenz im Reaktor, Regelschwankungen) handelt es sich streng genommen um ein Problem mit Fehlern in den Variablen! Dies ist eine Tatsache, die bisher selten bedacht wurde. Einige Methoden zur Regression bei Fehlern in den Variablen finden sich in [Sch71] [GvL79].

Die Technik der stationären Identifikation steht in einem gewissen Gegensatz zur dynamischen Parameteranpassung, denn die Zeit ist in den stationären Werten nicht mehr enthalten. Insbesondere können instationäre Übergangszustände nicht berücksichtigt werden. Eine neue Methode zur instationären Identifikation, die nicht auf der problematischen nichtlinearen Parameteranpassung beruht (vgl. 10.3), wird in Abschnitt 22.1 vorgestellt.

10.4.1.2 Praktische Durchführung der Methode

Die oben beschriebene stationäre Gleichung ist in dieser Form nicht einsetzbar, da in der Praxis meist weder die Kinetiken $r(X)$ noch die stöchiometrische Matrix N bekannt sind [6]. Bei speziellen Modellansätzen ist N jedoch meist durch wenige Parameter bestimmt, so daß versucht werden kann, die Parameter des gesamten Produkts

$$N \cdot r(X^*) = \frac{D^*}{X_1} \cdot (X^* - X_{in})$$

anzupassen. Als abhängige Größen stehen dann allerdings nicht mehr direkt die Kinetiken $r(X)$ zur Verfügung, sondern nur entsprechende Linearkombinationen von diesen.

10.4.2 Beispiel: Parameterbestimmung bei Modellen mit Erhaltungsstoffwechselterm

10.4.2.1 Messung der Linearkombinationen

Das Konzept läßt sich anhand des Parameteranpassungproblems bei Modellen mit Erhaltungsterm verdeutlichen. Die stationäre Gleichung lautet dann (vgl. 5.2.6.3):

$$\begin{pmatrix} Y_{X/S} & -Y_{X/S} \\ 1 & 0 \end{pmatrix} \cdot \begin{pmatrix} \sigma(S^*) \\ m_s(S^*) \end{pmatrix} = \frac{D^*}{X^*} \cdot \begin{pmatrix} X^* & \\ S^* & -S_{in} \end{pmatrix}$$

Gemessen werden können somit nur die spezifische Wachstrumsrate

$$\mu(S^*) = Y_{X/S} \cdot (\sigma(S^*) - m_s(S^*)) = D^*$$

als Linearkombination von $\sigma(S^*)$ und $m_s(S^*)$ sowie direkt die spezifische Substrataufnahmerate

$$\sigma(S^*) = \frac{D^*}{X^*} \cdot (S^* - S_{in}) \,.$$

[6] An dieser Stelle setzt die faktorenanalytische Methode zur Bestimmung des empirischen Reaktionsraums an (vgl. 6.6.3.3).

10.4.2.2 Auftragung der gemessenen Werte

Nach Durchführung des Experiments liegt eine Reihe von Datensätzen

$$(S_i^*,\ \sigma(S_i^*),\ \mu(S_i^*)),\quad i = 1, \ldots, n$$

vor. Durch geschickte Auftragung in geeignete Koordinatensysteme können daraus der Ertragskoeffizient $Y_{X/S}$ und die Parameter der Kinetiken σ und m_S geschätzt werden. Die Abbildungen 10.5 und 10.6 zeigen die typischen Kurvenverläufe

für die denkbaren Auftragungen
$\sigma(S^*)$ über S^*,
$\mu(S^*)$ über S^*,
$\mu(S^*)$ über $\sigma(S^*)$
sowie den — praktisch nicht meßbaren — Verlauf von
$m_S(S^*)$ über S^*,

und zwar für die vier möglichen Kombinationen aus

1. einer Substrataufnahmekinetik gemäß

$$\text{a)}\quad \sigma(S) = \sigma_S \frac{S}{S+K_S} \qquad \text{(Monod-Modell)}$$
$$\text{oder b)}\quad \sigma(S) = \sigma_s \frac{S}{\frac{S^2}{I_S}S+K_S} \qquad \text{(Inhibierungs-Modell)}$$

2. und einem Erhaltungsterm gemäß

$$\text{a)}\quad m_S(S) = B \qquad \text{(konstant)}$$
$$\text{oder b)}\quad m_S(S) = A \cdot S + B \qquad \text{(linear anwachsend).}$$

Bei allen Auftragungen muß bedacht werden, daß eine Messung im Bereich negativer Wachstumsraten in der kontinuierlichen Kultur nicht möglich ist, was unmittelbar aus der stationären Gleichung $\mu(S^*) = D$ folgt. Aber auch in der Nähe des Null-Wachstums reicht das einfache Modell nicht aus, da bei Substratkonzentrationen nahe 0 ganz neue Stoffwechselwege "ins Spiel" kommen. Z.B. können die Zellen einen Reservestoffwechsel haben, der zusätzliche Energie aus zellinternen Speicherstoffen bereitstellt. Die Modellierung solcher Vorgänge ist äußerst problematisch [TRJ88] [TRJ89]. Die Gültigkeit der kinetischen Ansätze erstreckt sich somit nur über die in den Abbildungen schwarz durchgezogenen Kurvenabschnitte.

10.4.2.3 Informationsgehalt der Auftragungen

Die unterschiedlichen Auftragungen differieren in ihrem Informationsgehalt. Hinsichtlich der zu schätzenden Parameter liegt die folgende Strategie nahe:

1. Bestimmung der Parameter σ_S, K_S und bei einer — an den Kurvenverläufen leicht zu erkennenden — Inhibierung auch von I_S aus der Auftragung von $\sigma(S^*)$ über S^*.

2. Bestimmung der Parameter $Y_{X/S}$, A und B aus der Auftragung von $\sigma(S)$ über $\mu(S)$. Im Falle $A = 0$ ist dies eine einfache lineare Regression

$$\mu(S) = Y_{X/S} \cdot \sigma(S) - (Y_{X/S} \cdot B)\ .$$

Im Falle $A \neq 0$ ist man gezwungen, die Beziehung von μ und σ explizit zu machen:

$$\mu(S) = Y_{X/S} \cdot (\sigma(S) - m_S(\sigma^{-1}(\sigma(S))),$$

was im Falle einer Inhibierungskinetik für σ zu einer Inversen σ^{-1} mit zwei Ästen führt [7].

[7] Auf die explizite Behandlung dieses Problemfalls kann hier nicht eingegangen werden.

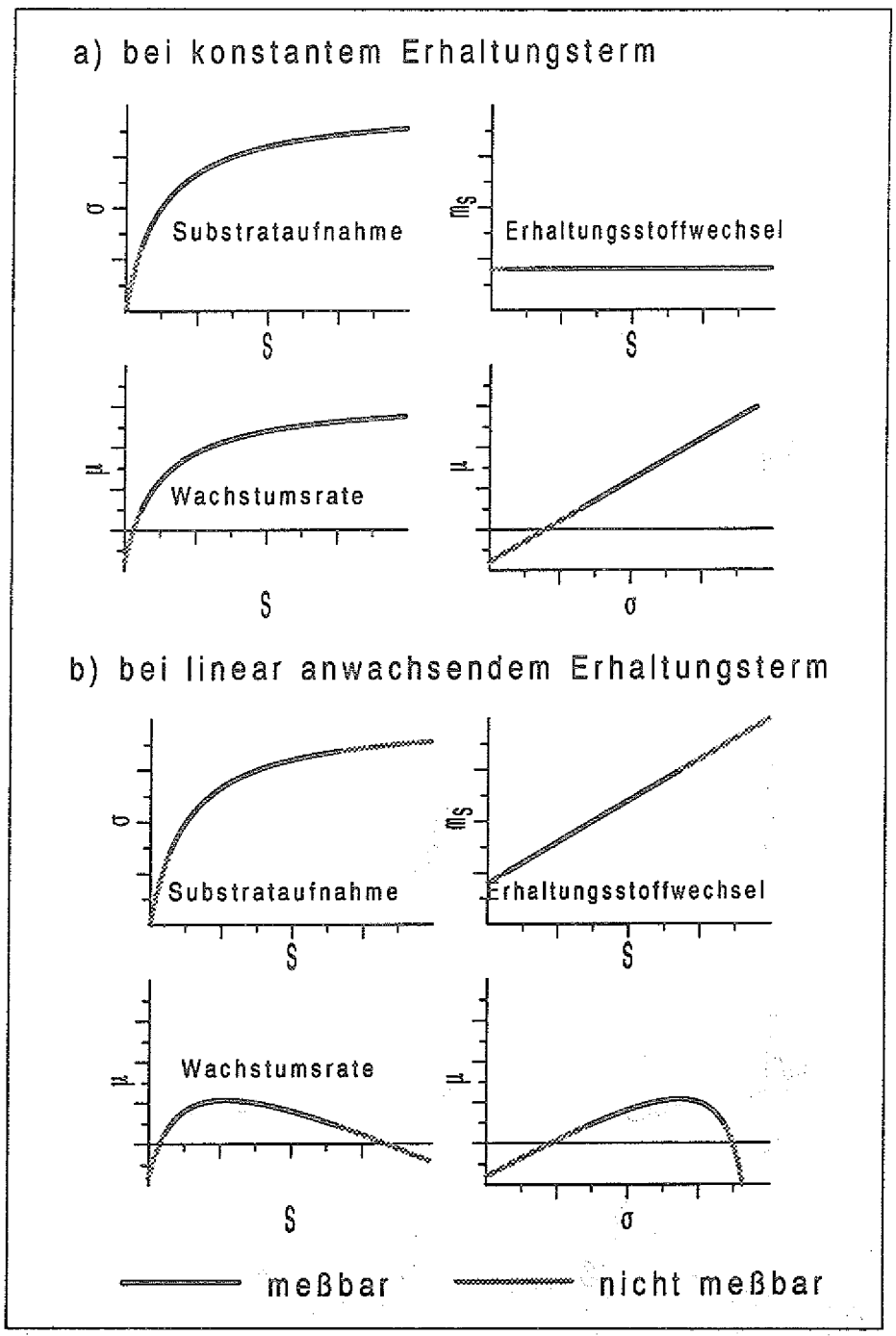


Abbildung 10.5: Stationäre Identifikation beim Monod-Modell

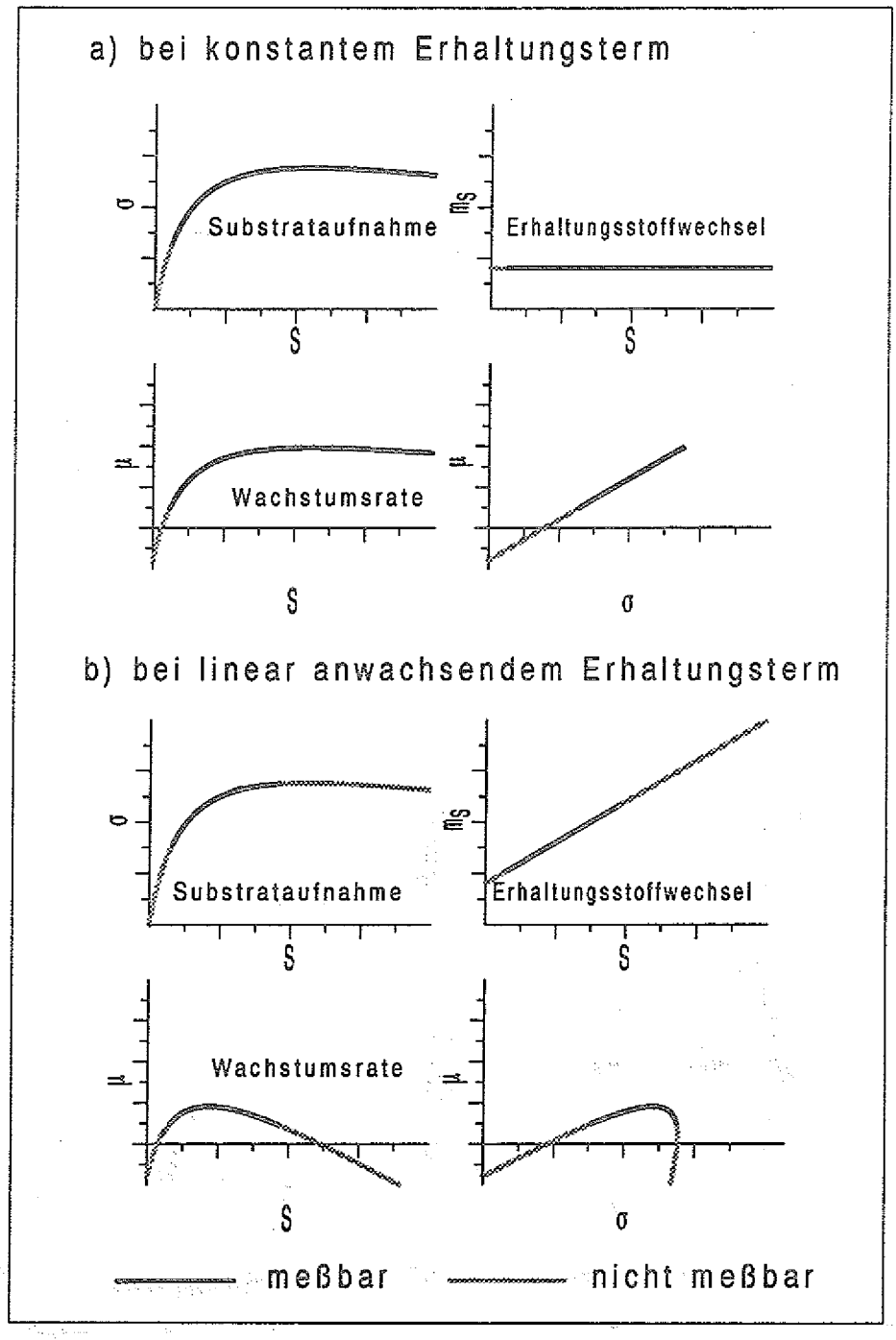


Abbildung 10.6: Stationäre Identifikation beim Inhibierungs-Modell

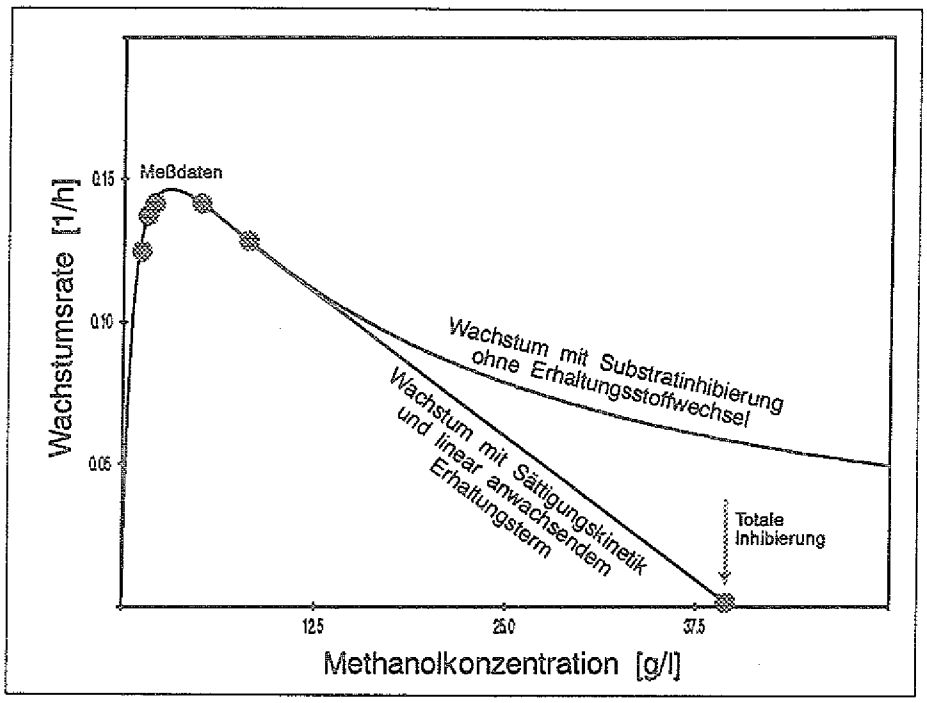


Abbildung 10.7: Wachstumskinetik von *Candida boidinii*

Interessant ist, daß die Auftragung von $\mu(S)$ über S für die Bestimmung der Parameter am schlechtesten geeignet ist, da sie die Bestimmung des Produkts $\mu(0) = -Y_{X/S} \cdot B$ nur mit großen Fehlergrenzen zuläßt! Dagegen entkoppeln die beiden anderen Auftragungen die Substrataufnahme und den Erthaltungsstoffwechsel. Trotzdem wird die Auftragung von μ über S^* üblicherweise als erstes durchgeführt!

Diese Resultate können direkt an den graphischen Auftragungen abgelesen werden, was ein weiterer Beleg dafür ist, daß die extensive Anwendung von Visualisierungstechniken und eine explorative Vorgehensweise den Informationsgewinnungsprozeß stark beschleunigen können. Eine explizite Sensitivitätsanalyse kann nun herangezogen werden, um die Resultate auch quantitativ zu formulieren. Man vergleiche dazu die Diskussion zur Explorativen und konfirmatorischen Datenanalyse in 3.4.

10.4.2.4 Ein Literaturbeispiel

Die kontinuierliche Fermentation der methylotrophen Hefe *Candida boidinii* wurde schon in Abschnitt 1.2 als Beispiel herangezogen. Quantitative Ergebnisse für das Wachstumsverhalten der Hefe auf Methanol liegen kaum vor [RGRW75] [SHRD78]. In [RGRW75] findet man die in Abbildung 10.7 wiedergegebenen Daten für die Wachstumsrate $\mu = \mu(S)$. Zusätzlich zu diesen Daten ist bekannt, daß *Candida boidinii* bei einer Methanolkonzentration von mehr als ca. 4% nicht mehr wächst [PP76]. Dagegen ist auch bei sehr kleinen Substratkonzentrationen noch ein positives Wachstum zu verzeichnen. Für die Substrataufnahmerate liegen derzeit keine publizierten Werte vor (!).

Bereits aufgrund der wenigen Datenpunkte kann gesagt werden:

- Der konstante Term B des Erhaltungsstoffwechsels ist verschwindend klein aufgrund des positiven Wachstums bei kleinen Substratkonzentrationen.
- Ein Modell mit konstantem Erhaltungsterm ist nicht in der Lage, das Wachstumsverhalten adäquat zu beschreiben, wie die Abbildungen 10.5 und 10.6) zeigen [8].

Ein ansteigender Erhaltungsstoffwechsel ist auch aus biologischer Sicht naheliegend: Methanol schädigt die Zellmembran der meisten Organismen derart stark, daß es als Sterilisationsmittel verwendet werden

[8] Interesseanterweise wird in [RGRW75] eine Standard-Inhibierungskinetik an die vier gemessenen Datenpunkte angepaßt!

kann. Die methylotrophen Organismen haben dagegen Widerstandsmechanismen entwickelt. Der Energieaufwand zur Aufrechterhaltung des Membranpotentials wird mit zunehmender Methanolkonzentration anwachsen [LEY84].

Das durch die Annahme eines linear anwachsenden Erhaltungsstoffwechsels begründete Modell entspricht exakt der in [CLT76] rein empirisch gefundenen Wachstumskinetik für methylotrophe Organismen. Dies ist ein weiterer Beleg für die Effektivität einer deduktiven Vorgehensweise bei der Modellierung.

10.5 Statistische Tests

Statistische Tests liegen im Prinzip allen Verfahren zur Erkennung von

- Phasenübergängen [Wil76],
- Störungen [Him78],
- Meßfehlern [WS83] [Jef81] oder
- kritischen Zuständen [GS81] [CS89]

zugrunde. Sie sind aus diesem Grunde auch für die Datenanalyse von besonderer Wichtigkeit. Die Kenntnis der prinzipiellen Arbeitsweise eines statistischen Tests, insbesondere der Begriffe

- Hypothese H_0 und Alternativhypothese H_1,
- Signifikanzniveau α,
- Teststatistik $t(X)$ ($X = (X_1, \ldots, X_m)$ eine Stichprobe)

wird vorausgesetzt. Sie sind z.B. in [Sac84] nachzuschlagen.

10.5.1 Likelihood-Quotienten-Test

Die *Likelihood-Funktion* für eine gewonnene Stichprobe

$$X = (X_1, \ldots, X_m)$$

ist die Wahrscheinlichkeitsdichtefunktion der Verteilung des Zufallsvektors X, ausgewertet an der Stelle, die durch die Stichprobe gegeben ist. Sie hängt im allgemeinen zusätzlich von den Parametern der Verteilung ab:

$$L(\Theta; X) = f_X(\Theta; \xi_1, \ldots, \xi_n)_{|\xi_1 = X_1, \ldots, \xi_n = X_n}$$

Sie ist (mit Vorsicht) ein Maß dafür, wie wahrscheinlich der gemessene Ausgang $(X_1, \ldots, X_n)$ unter der Annahme des Parameters Θ ist.

Beispiel: Bei einer Normalverteilungsdichte (Gauß-Glocke)

$$f_{\mu,\sigma}(\xi) = \frac{1}{\sqrt{2\pi \cdot \sigma^2}} \cdot e^{-\frac{1}{2}\frac{(\xi-\mu)^2}{\sigma^2}}$$

sind Ausgänge x mit kleinem Abstand $|x-\mu|$ vom Mittelwert wahrscheinlich, wogegen (im Verhältnis zu σ) große Abweichungen $|x-\mu|$ sehr selten auftreten.

Bestehen nun zwei Hypothesen

$$H_0 : \Theta = \Theta_0, \quad H_1 : \Theta = \Theta_1,$$

für die Parameter einer Verteilung, so kann die Wahrscheinlichkeit des Ausgangs $X = (X_1, \ldots, X_n)$ unter der Hypothese H_0 mit seiner Wahrscheinlichkeit unter H_1 durch Quotientenbildung verglichen werden:

$$\lambda = \frac{L(\Theta_1; X)}{L(\Theta_0; X)}$$

Für den Fall, daß die Verteilung mit dem Parameter Θ_1 den Ausgang des Experiments besser beschreibt, wird λ groß. Dies führt zum *Likelihood-Quotienten-Test*

$$H_0 \text{ ablehnen , falls } \lambda > \lambda^*_\alpha,$$

wobei λ^*_α durch die Wahrscheinlichkeit $P(\lambda < \lambda^*_\alpha) = \alpha$ (α das Signifikanzniveau) bestimmt ist. In der Regel benutzt man den Likelihood-Quotienten-Test in der logarithmierten Form:

$$\ln \lambda = \ln L(\Theta_1; X) - \ln L(\Theta_0; X) > \ln \lambda^*_\alpha$$

10.5.2 Ausreißertest

Der Likelihood-Quotienten-Test läßt sich zur Konstruktion eines einfachen Ausreißer-Tests verwenden: Eine Meßgröße $X(t)$ sei durch ein Ausreißer-Modell (vgl. 6.2.4.1) beschrieben, das von

(μ, σ_0) - normalverteilten Meßwerten mit Wahrscheinlichkeit $1 - \lambda$ und

(μ, σ_1) - normalverteilten Ausreißern mit Wahrscheinlichkeit λ

ausgeht [9]. Man beachte, daß beide Verteilungen denselben Erwartungswert μ haben, d.h. man geht von einer verschieden starken Störung desselben Meßwerts und nicht von einer Verzerrung aus. Der Parameter μ sei (z.B. über eine Filterung des Signals) bekannt. Dann ist der logarithmierte Likelihood-Quotient gerade:

$$\ln \lambda = \frac{1}{2} \cdot [\ln(\sigma_0) - \ln(\sigma_1) + \frac{(x-\mu)^2}{\sigma_0^2} - \frac{(x-\mu)^2}{\sigma_1^2}]$$

Nach Umformung wird daraus der Test

$$(x-\mu)^2 > \frac{\sigma_0^2 \cdot \sigma_1^2}{\sigma_1^2 - \sigma_0^2} \cdot (2 \ln \lambda^*_\alpha - \ln(\sigma_0) + \ln(\sigma_1)),$$

der die allgemeine Gestalt

$$|x - \mu| > \text{const}$$

hat. Dies bestätigt nachträglich die heuristische Vorgehensweise, einen Meßwert als Ausreißer anzusehen, wenn er um einen gewissen Betrag vom geschätzten Erwartungswert abweicht. Das Verfahren bleibt selbst dann anwendbar, wenn σ_0 und σ_1 nicht bekannt sind. In dieser Situation kann allerdings kein Signifikanzniveau bestimmt werden und λ^*_α ist heuristisch vorzugeben.

[9] Der Test berücksichtigt in dieser einfachen Form nicht die Gewichtung mit λ. Geht man zu einem *gewichteten Likelihood-Quotienten-Test* über (vgl. auch 10.7), so ändert sich qualitativ nichts, quantitativ wird sich der Test jedoch wesentlich verbessern.

10.5.2.1 Fehler 1. und 2. Art

Der Ausreißertest illustriert gut die Problematik eines statistischen Tests: Aufgrund des in Abschnitt 6.2.4.1 skizzierten Trennproblems für das Ausreißermodell können bei Anwendung des Tests zwei Irrtümer auftreten (vgl. auch Abb. 2.11):

- Entscheidung für einen Ausreißer, obwohl ein regulärer Meßwert vorliegt (Fehler 1. Art). Die Wahrscheinlichkeit für diesen Irrtum wird durch das Signifikanzniveau α vorgegeben.
- Entscheidung für einen regulären Wert, obwohl ein Ausreißer vorliegt (Fehler 2. Art). Die Wahrscheinlichkeit β für diesen Irrtum kann beim Ausreißertest nicht vorgegeben werden. Sie resultiert aus der Wahl von α.

Beim praktischen Einsatz des Ausreißer-Tests wird in der Regel ein niedriges Signifikanzniveau gewählt, da ein Fehler 1. Art hier — im Gegensatz zur Anwendung von Tests im medizinischen Bereich — meist keine negativen Folgen hat.

10.5.3 Sprungdetektion

Das Problem der Sprungdetektion läßt sich ebenfalls auf einen Likelihood-Quotienten-Test zurückführen. In Abschnitt 6.7 ist ein kontinuierliches Sprungmodell formuliert, das in diskreter Form lautet:

$$f(\Delta\alpha, s, i) = \begin{cases} \alpha_< & ,1 \le i \le s-1 \\ \alpha_< + \Delta\alpha & ,s \le i \le n \end{cases} + \epsilon_i \quad \text{mit} \quad \mathrm{E}[\epsilon_i] = 0, \mathrm{Var}[\epsilon_i] = \sigma^2$$

Der entsprechende KQ-Schätzer

$$\hat{s} \text{ und } \hat{\Delta\alpha}$$

für Sprungzeitpunkt und Sprunghöhe wurde in Abschnitt 10.2.4 abgeleitet. Die Frage, ob der erkannte Sprung signifikant ist, führt auf das Testproblem

$$\begin{aligned} H_0: &\quad \text{Kein Sprung erfolgt} \quad (s > n)\,, \\ H_1: &\quad \text{Sprung erfolgt} \quad (s = \hat{S}, \Delta\alpha = \hat{\Delta\alpha})\,. \end{aligned}$$

Nimmt man die Störterme ϵ_i als unabhängig normalverteilt mit den Wahrscheinlichkeitsdichtefunktionen $f_{(\alpha_<,\sigma^2)}(\xi)$ bzw. $f_{(\alpha_<+\hat{\Delta\alpha},\sigma^2)}(\xi)$ an, so sind die Likelihood-Funktionen gegeben durch

$$L_0(y_1, \ldots, y_n) = \prod_{i=1}^{n} f_{(\alpha_<,\sigma^2)}(y_i) \quad \text{unter der Hypothese } H_0$$

und

$$L_1(y_1, \ldots, y_n) = \prod_{i=1}^{\hat{s}-1} f_{(\alpha_<,\sigma^2)}(y_i) \cdot \prod_{i=\hat{s}}^{n} f_{(\alpha_<+\hat{\Delta\alpha},\sigma^2)}(y_i) \quad \text{unter der Hypothese } H_1.$$

Der logarithmierte Likelihood-Quotient lautet (nach Kürzen):

$$\begin{aligned} \ln\lambda &= \ln L_1(y_1, \ldots, y_n) - \ln L_0(y_1, \ldots, y_n) \\ &= -\frac{1}{2\sigma^2}\sum_{i=\hat{s}}^{n}(y_i - \alpha_< - \hat{\Delta\alpha})^2 - \frac{1}{2\sigma^2}\sum_{i=\hat{s}}^{n}(y_i - \alpha_<)^2 \\ &= \frac{1}{\sigma^2}\cdot\hat{\Delta\alpha}\cdot\sum_{i=\hat{s}}^{n}(y_i - \alpha_< + \frac{1}{2}\hat{\Delta\alpha}) \end{aligned}$$

Die Gleichungen des auf diesen Detektor aufgebauten Tests sind in 2.5.2 zusamengefaßt worden. Es sei darauf hingewiesen, daß der Test durch geeignetes Umformen der Maximumsbildung für $\hat{s}$ (vgl. 10.2.4)

in eine iterative Gestalt gebracht werden kann, so daß ein neu eintreffender Meßwert nicht die komplette Neuberechnung des Maximums zur Folge hat. In dieser iterativen Gestalt spricht man je nach Formulierung von einem Kumulative-Summen-Test oder einem Page-Hinkley-Test [BB86].

In der Praxis wählt man ein Zeitfenster in die Vergangenheit, in dem das Signal $y(t)$ als annähernd konstant angesehen werden kann, und zeichnet die Teststatistik

$$\lambda = \lambda(t)$$

über der Zeit auf. Sobald $\lambda(t)$ die Signifikanzgrenze λ_α^* überschreitet, wird der "Sprung-Alarm" ausgelöst. Bei zu niedrig angesetztem Signifikanzniveau kommt es zu Fehldetektionen (Fehler 1.Art), wogegen ein zu hoch angesetztes Niveau lediglich zu einer verzögerten Detektion führt (Fehler 2.Art).

10.6 Faktorenanalyse

Die Problematik der Faktorenanalyse auf der Basis des Modells

$$Y = X \cdot \beta + \epsilon, \quad \text{mit} \quad \mathrm{Cov}[\beta] = 1$$

ist mit dem Fundamentaltheorem (vgl. 6.5.1.4)

$$\mathrm{Cov}[Y] = X\,X^T + \mathrm{Cov}[\epsilon]$$

bereits umrissen: In der Regel liegt aufgrund einer gegebenen Stichprobe eine Schätzung Σ der Kovarianzmatrix $\mathrm{Cov}[Y]$ vor. Dann bestehen einige Probleme der Faktorenanalyse darin [æbe71],

- die Dimension von β zu schätzen,
- die Matrix X der *Hauptkomponenten* zu schätzen,
- die *Faktorenladungen* β zu schätzen.

10.6.1 Hauptkomponentenanalyse

Die Idee, die der Hauptkomponentenmethode zur Faktorenalyse zugrunde liegt, kann kurz skizziert werden: Für den Fall, daß $\epsilon = 0$ gilt, ist die Matrix X im Prinzip durch die Wurzel $\sqrt{\Sigma}$ gegeben, die man mit Hilfe der bereits beschriebenen Hauptachsentransformation (vgl. 6.1.3.1) berechnet. Dabei ergeben sich nach entsprechender Anordnung die Hauptachsen

$$v_1, v_2, \ldots, v_n \quad \text{der Länge} \quad \sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_n$$

des Kovarianzellipsoids zu Σ. Da die Beobachtungen $Y = (Y_1, \ldots, Y_n)$ nach Möglichkeit durch weniger als n Faktoren beschrieben werden sollen, ist zu erwarten, daß alle Beobachtungen Y in einem niederdimensionalen Unterraum des $\Re^n$ liegen. Der Kovarianzellipsoid muß daher entartet sein, d.h. einige Achsen haben die Länge $\sigma_i = 0$. Folglich ist nach der Hauptachsentransformation die Dimension $\dim\beta$ gleich der Zahl der $\sigma_i = 0$. Die Matrix X wird dann aus den nicht entarteten Hauptachsen v_i gebildet.

Bei einem (kleinen) Rauschens ϵ kommt in der Kovarianzgleichung der zusätzliche Term $\mathrm{Cov}[\epsilon]$ hinzu. Er wird dazu führen, daß die kleinen Eigenwerte nicht mehr exakt gleich 0 sind:

$$\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_m \gg 0 \quad \text{und} \quad \sigma_{m+1}, \sigma_{m+2}, \ldots, \sigma_n \approx 0$$

Man schätzt daraus — nach einer geeigneten Präzisierung der Relation $\gg 0$ — die Dimension $\dim\beta = m$ und bildet den Schätzer für X wie oben. Dieser Ansatz impliziert einen Signifikanztest für die Hypothese $H_0 : \sigma_i = 0$. Die Details sind in der Literatur nachzulesen [FH84] [æbe71].

10.6.2 Nichtlineare Hauptkomponentenanalyse

Einige interessante Anwendungen für nichtlineare Verallgemeinerungen bei der Mustererkennung sowie entsprechende numerische Techniken sind in [HS89] [Ban88] beschrieben.

10.7 Klassifikation

Das der statistischen Klassifikation zugrundeliegende Mischverteilungsmodell

$$X = X|\Omega_V$$

mit der Verteilervariablen V wurde schon in Abschnitt 6.4 vorgestellt. Die zugehörige Dichtefunktion hat die Gestalt

$$f_X(\xi) = \sum_{i=1}^{n} \lambda_i \cdot f_{X|\Omega_i}(\xi) \quad \text{mit} \quad \lambda_i = P(\omega \epsilon \Omega_i)$$

für die zu klassifizierenden Objekte ω aus der zugrundeliegenden Grundgesamtheit Ω.

10.7.1 Das Klassifikationsproblem

Auch das Klassifikationsproblem wurde in 6.4 bereits informell beschrieben. Es kann jetzt exakt definiert werden als das Problem, einen optimalen Schätzer

$$\hat{k} = \hat{k}(X)$$

zu finden, der aufgrund der Merkmale $X(\omega)$ eines Objekts ω dessen Zugehörigkeit zur Klasse $\Omega_{\hat{k}}$ schätzt.

Das Problem hat genau wie bei der Parameterschätzung (vgl. 10.2) keine eindeutige Lösung. Vielmehr muß zunächst

- ein Gütemaß für $\hat{k}$ und
- eine Klasse von Funktionen, aus denen $\hat{k}$ gewählt werden kann,

festgelegt werden, um Optimalität definieren zu können. Als ein mögliches Gütemaß sei hier nur die Gesamt-Wahrscheinlichkeit

$$\sum_{i=1}^{n} P(\hat{k}(X|\Omega_i) \neq i)$$

einer Fehlklassifikation genannt, die zu minimieren wäre [10].

10.7.2 Einige Arten von Klassifikatoren

Vorgestellt werden nur einige prinzipielle Ansätze. Eine ausführliche Darstellung findet sich z.B. in [Nie83] [FH84].

10.7.2.1 Likelihood-Klassifikator

Ganz allgemein ensteht ein Klassifikator, wenn man die *gewichteten Likelihood-Funktionen*

$$L_i = \lambda_i \cdot f_{X|\Omega_i}$$

zu Hilfe nimmt. $L_i(X)$ gibt an, wie "ähnlich" die Merkmale im Vektor X den Eigenschaften der Klasse Ω_i sind. Der Klassifikator wählt dann die "ähnlichste" Klasse aus:

$$\hat{k}(X) = \arg\min_i L_i(X)$$

Unter gewissen Bedingungen kann man für die Likelihood-Klassifikatoren Optimalitätseigenschaften nachweisen (siehe z.B. [FH84]).

[10] $P(\hat{k}(X)|\Omega_i \neq i)$ ist hier die Wahrscheinlichkeit, daß ein Objekt ω_i nicht der Klasse Ω_i zugeordnet wird.

10.7.2.2 Abstandsklassifikator

Für den Fall, daß alle Einschränkungen $X|\Omega_k$ normalverteilt sind, kann die gewichtete Likelihood-Funktione explizit formuliert und das Optimierungsproblem gelöst werden. Erneut ist eine Logarithmierung der Likelihood-Funktione sinnvoll. Man stößt (unter Weglassung der stets gleichen 2π-Terme) auf die *Diskriminanz-Funktionen*

$$d_i(X) = \ln \lambda_i - \frac{1}{2} \ln \det \Sigma_i - \frac{1}{2} (X - \mu_i)^T \Sigma^{-1} (X - \mu_i) \ ,$$

wobei μ_i und Σ_i den Erwartungswert und die Kovarianzmatrix von $X|\Omega_i$ bezeichnen.

$d_i(X)$ ist nichts anderes als ein verallgemeinertes Abstandsmaß, das die Distanz von X zu dem jeweiligen Klassenschwerpunkt μ_i berechnet. Der Klassifikator wählt dann diejenige Klasse i mit dem (im verallgemeinerten Sinne) kürzesten Abstand von μ_i zu X aus. Dieses Vorgehen ist in der Abbildung 10.8a verdeutlicht.

Für den Spezialfall, daß alle Kovarianzen und alle Gewichte gleich sind, d.h.

$$\Sigma_1 = \ldots = \Sigma_n = \Sigma \quad \text{und} \quad \lambda_1 = \ldots = \lambda_n = \frac{1}{n} \ ,$$

läßt sich zeigen, daß die Abstandsmaße äquivalent zu dem auch geometrisch interpretierbaren *Mahalanobis-Abstand*

$$||X - \mu_i||_\Sigma^2 = (X - \mu_k)^T \Sigma^{-1} (X - \mu_k)$$

sind. Im Falle $\Sigma = 1$ ist dies das klassische euklidische Abstandsmaß.

10.7.2.3 Gebietsklassifikation

Alle Merkmalsvektoren X, die derselben Klasse Ω_i zugeordnet werden, bilden im Merkmalsraum ein Gebiet

$$G_i = \{X \mid \hat{k}(X) = i\} \ .$$

Die Gebiete sind disjunkt und überdecken gemeinsam den gesamten Merkmalsraum. Eine alternative Darstellung des Klassifikators $\hat{k}$ ist somit gegeben durch

$$\hat{k}(X) = i \iff X \in G_i \ .$$

X wird der Klasse Ω_i zugeordnet, in deren Gebiet es liegt. Dies verdeutlicht die Abbildung 10.8b.

Für den Fall normalverteilter $X|\Omega_i$ können die Gebiete erneut explizit ausgerechnet werden. Liegen Mahalanobis-Abstände vor, so sind die einzelnen Gebiete durch gerade Linien bzw. Flächen begrenzt. Sie haben also die Gestalt von Vielecken bzw. Vielflachen. Speziell beim euklidischen Abstand sind die Begrenzungslinien bzw. -flächen Teilstücke der Mittelsenkrechten aller Verbindungslinien zwischen den Schwerpunkten μ_i. In der allgemeinen Situation ergeben sich hingegen sehr komplizierte Verläufe der Randlinien, die sich dann aus Stücken von Kegelschnitten (Parabeln, Hyperbeln, Ellipsen und deren höherdimensionale Analoga) zusammensetzen.

10.7.3 Heuristische Klassifikation

In der Praxis ist meist weder eine Normalverteilungsannahme noch eine Annahme identischer Verteilungen haltbar. Des weiteren sind die Verteilungen nicht immer genau bekannt. In dieser Situation motivieren die exakten Ergebnisse für den normalverteilten Spezialfall die Verwendung krummlinig begrenzter Gebiete zur Konstruktion eines Klassifikators. Solche Begrenzungen können durchaus auch aufgrund heuristischer Kriterien gezogen werden (vgl. Abb. 10.8c). Dabei ist zwar keine Optimalität des Klassifikators zu erwarten, jedoch genügen in vielen Situationen bereits suboptimale Lösungen. Ein solches Beispiel wird in Abschnitt 22.2 vorgestellt.

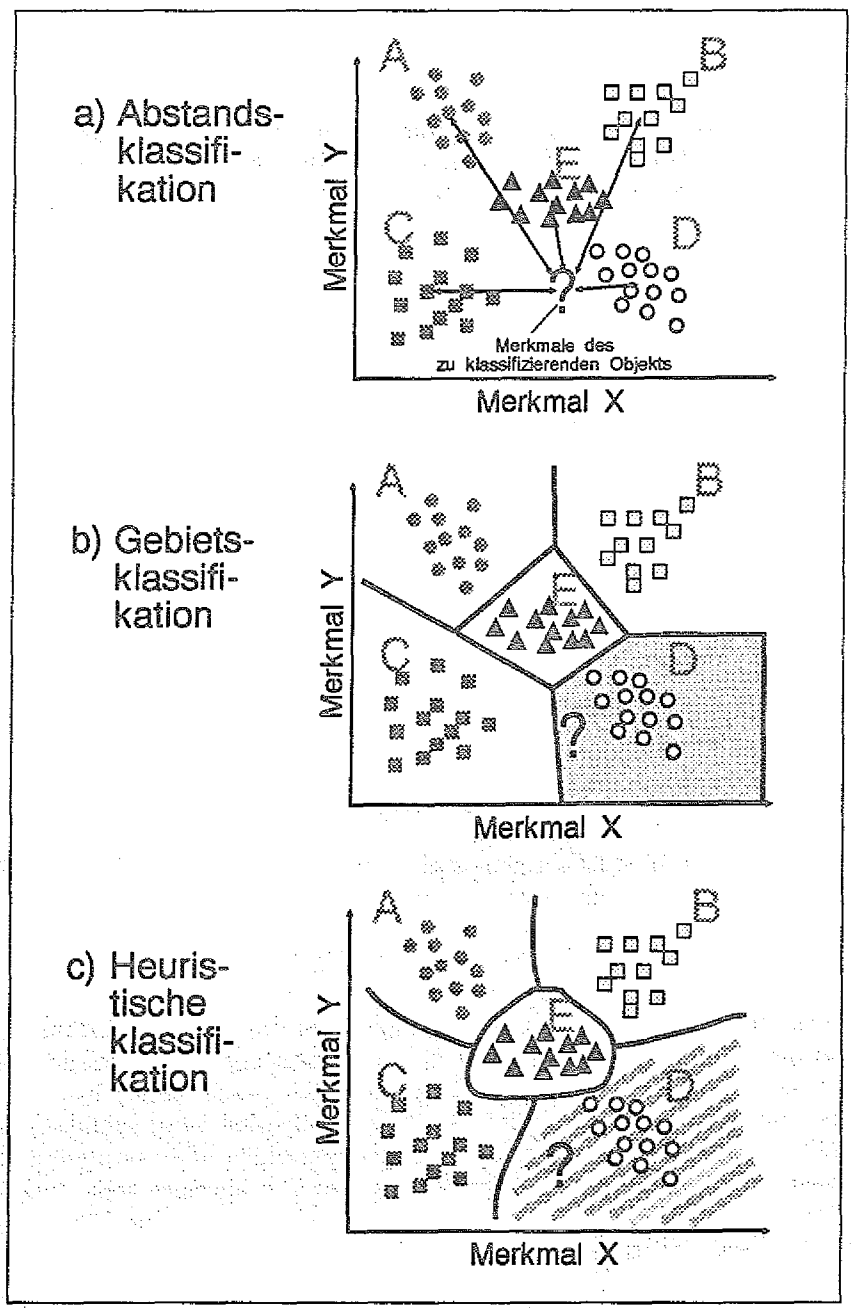


Abbildung 10.8: Verschiedene Methoden der Klassifikation

10.7.6 Lernstichproben

In der Regel sind die Verteilungen $f_{X|\Omega_i}$ a priori nicht bekannt, und es besteht zunächst das zusätzliche Problem der Schätzung dieser Verteilungen bzw. ihrer Parameter. Zu diesem Zweck wird eine sogenannte *Lernstichprobe* herangezogen. Eine solche Stichprobe besteht aus einer Reihe von Objekten

$$\omega_1, \omega_2, \omega_3, \ldots ,$$

deren Klassenzugehörigkeit a priori bekannt ist.

Beispiel: Bei dem bereits in 10.7 erwähnten Beispiel der Geschlechtsbestimmung aufgrund der Merkmale Größe, Gewicht und Alter könnten die Daten von je 100 zufällig ausgewählten Männern und Frauen als Lernstichprobe herangezogen werden.

Die Konstruktion eines Klassifikators aufgrund einer Lernstichprobe bezeichnet man als *Lernphase*. Angelernt werden können sowohl Abstandsklassifikatoren und Gebietsklassifikatoren als auch heuristische Klassifikatoren. Ein Beispiel wird in Abschnitt 22.2 gegeben. Die Details würden hier zu weit führen und werden z.B. in [FH84] und [Nie83] diskutiert.

10.8 Nichtparametrische Glättungsverfahren

Eine wichtige Klasse statistischer Verfahren, die im Rahmen dieser Arbeit nicht zur Anwendung kommen, jedoch in einem Datenanalysesystem nicht fehlen dürfen, sind die nichtparametrischen Verfahren zur Glättung und Differentiation von Meßsignalen. Hierzu gehören

- lokale gewichtete lineare Regressionsschätzer,
- Fensterschätzer [GM84] [GMM84],
- glättende Splines [EMR90],
- Approximationsverfahren [WS79] und
- Fourier-Transformation und -Rücktransformation [PFTV88].

Sie zeichnen sich durch verhältnismäßig wenige Stellparameter aus, ermöglichen eine Grobanalyse von Datensätzen und unterstützen die Visualisierung. Die meisten dieser Verfahren sind auch für den On-Line-Einsatz erweiterbar.

Kapitel 11

Stochastische Methoden

Ist die Erzeugung von Zufallszahlen zumindest noch vom Prinzip her bekannt, so beginnt bei der Simulation stochastischer dynamischer Systeme für die meisten regelungstechnisch interessierten Biotechnologen Neuland. Es soll versucht werden, die wichtigsten Ansätze aus diesem — sehr theoretischen — Gebiet zusammenzustellen und bzgl. ihrer praktischen Anwendbarkeit auszuwerten (siehe 11.1). Erst auf dieser Grundlage sind die schon seit langem verwendeten stochastischen Zustandsraummodellmodelle auch numerisch simulierbar!

Eine weitere Datenanalysemethode, die unmittelbar auf das allgemeine Stoffwechselmodell angewendet werden kann, ist die Kalman-Filter-Technik (siehe 11.3). Sie erweist sich als ein Komplement zu den linearen Verfahren der Statistik in der Situation, wo über die Momentan-Meßdaten keine ausreichende Systeminformation zu erhalten ist (vgl. 6.6.3.1).

Erneut sei auf die Konzepte der stochastischen Kontrolle aufmerksam gemacht (vgl. 3.1.3.1), die hier zwar nicht behandelt werden, sich jedoch vollkommen analog zur Filtertheorie entwickeln lassen. Insbesondere korrespondieren der lineare stochastische Regler und das lineare Filter sowie der Minimal-Varianz-Regler und das Kalman-Filter.

11.1 Stochastische Simulation

Die numerische Lösung stochastischer Differentialgleichungen unterscheidet sich wesentlich von der Lösung gewöhnlicher Differentialgleichungen. Dieser Unterschied soll kurz charakterisiert werden:

11.1.1 Das stochastische Euler-Verfahren

11.1.1.1 Naiver Lösungsansatz

Ein naiver Ansatz zur Lösung der stochastischen Differentialgleichung

$$\frac{dx}{dt} = a(t,x) + \sqrt{b(t,x)} \cdot e(t)$$

($e(t)$ ein normalverteiltes Weißes Rauschen mit $\mathrm{E}[e(t)] = 0$, $\mathrm{Var}[e(t)] = 1$) ist die Übertragung des Euler-Verfahrens aus der deterministischen Theorie:

$$dx = a(t,x)\,dt + \sqrt{b(t,x)} \cdot e(t)\,dt\ .$$

(nach "Multiplikation mit dt"). Dies entspricht der Iteration:

$$x(t+dt) = x(t) + a(t,x)\,dt + \sqrt{b(t,x)} \cdot e(t)\,dt$$

mit einer kleinen Schrittweite dt und standard-normalverteilten Zufallszahlen $e(t)$. Speziell mit $a = 0$ und $b = 1$ ergibt sich eine Approximation an den Wiener-Prozeß mit

$$W(t+dt) = W(t) + e(t)\,dt \implies W(n \cdot dt) = dt \cdot \sum_{i=1}^{n} e(i \cdot dt) + W(0)\ .$$

Offenbar hat die so gewonnene Approximation normalverteilte und unabhängige Zuwächse

$$dW(t) = W(t+dt) - W(t) = e(t).$$

Bei der Berechnung der Varianz von $W(1)$ zum Startwert $\mathrm{Var}\,[W(0)] = 0$ und der Schrittweite

$$dt = \frac{1}{n} \quad \Leftrightarrow \quad n = \frac{1}{dt}$$

resultiert daraus jedoch ein Widerspruch:

$$\begin{aligned}
\mathrm{Var}\,[X(1)] &= \mathrm{Var}\,[dt \cdot \sum_{i=1}^{n} e(i \cdot dt)] \\
&= dt^2 \cdot \sum_{i=1}^{n} \mathrm{Var}\,[e(i \cdot dt)] \\
&= dt^2 \cdot n \\
&= dt
\end{aligned}$$

Es stellt sich heraus, daß der gewählte Ansatz nicht richtig sein kann, weil die Varianz von $X(1)$ von der Schrittweite abhängt und insbesondere $\mathrm{Var}\,[X(1)] \to 0$ für $dt \to 0$ gilt.

11.1.1.2 Korrekte Formulierung des Verfahrens

Der Ito-Kalkül verlangt, daß die (naiv interpretierte) Gleichung

$$\mathrm{Var}\,[dW] = dt$$

gilt (vgl. 7.2.1.4), d.h. die Zuwächse $dW(t)$ müssen eine Varianz erhalten, die proportional zur Schrittweite ist. Eine Zufallsvariable, welche diese Bedingung erfüllt, ist

$$dW(t) = e(t) \cdot \sqrt{dt}\ .$$

Das korrekte stochastische Euler-Verfahren ist nun gegeben durch:

$$x(t+dt) = x(t) + a(t,x) \cdot dt + \sqrt{b(t,x)} \cdot e(t) \cdot \sqrt{dt}$$

Speziell für die Approximation des Wiener-Prozesses

$$W(t+dt) = W(t) + e(t) \cdot \sqrt{dt} \implies W(n \cdot dt) = \sqrt{dt} \cdot \sum_{i=1}^{n} e(i \cdot dt) + W(0)$$

berechnet man jetzt $(\mathrm{Var}\,[X(0)] = 0)$

$$\mathrm{Var}\,[X(1)] = \mathrm{Var}\,[\sqrt{dt} \cdot \sum_{i=1}^{n} e(i \cdot dt)] = dt \cdot n = 1$$

unabhängig von $n = \frac{1}{dt}$. Die Abbildung 11.1 gibt einige Approximationen der zweidimensionalen Brownschen Bewegung für verschiedene Schrittweiten dt wieder. Es zeigt sich, daß für $dt \to 0$ der Anteil der kleinen Frequenzen wächst, wobei das Diffusionsverhalten im Großen gleich beibt.

Im folgenden wird das soeben beschriebene Verfahren zur numerischen Approximation des Wiener-Prozesses vorausgesetzt, so daß das Euler-Verfahren in der prägnanteren Form

$$x(t+dt) = x(t) + a(t,x)\,dt + \sqrt{b(t,x)}\,dW(t)$$

geschrieben werden kann. Dies entspricht formal genau der differentiellen Formulierung der stochastischen Differentialgleichung (vgl. 7.2.1.3).

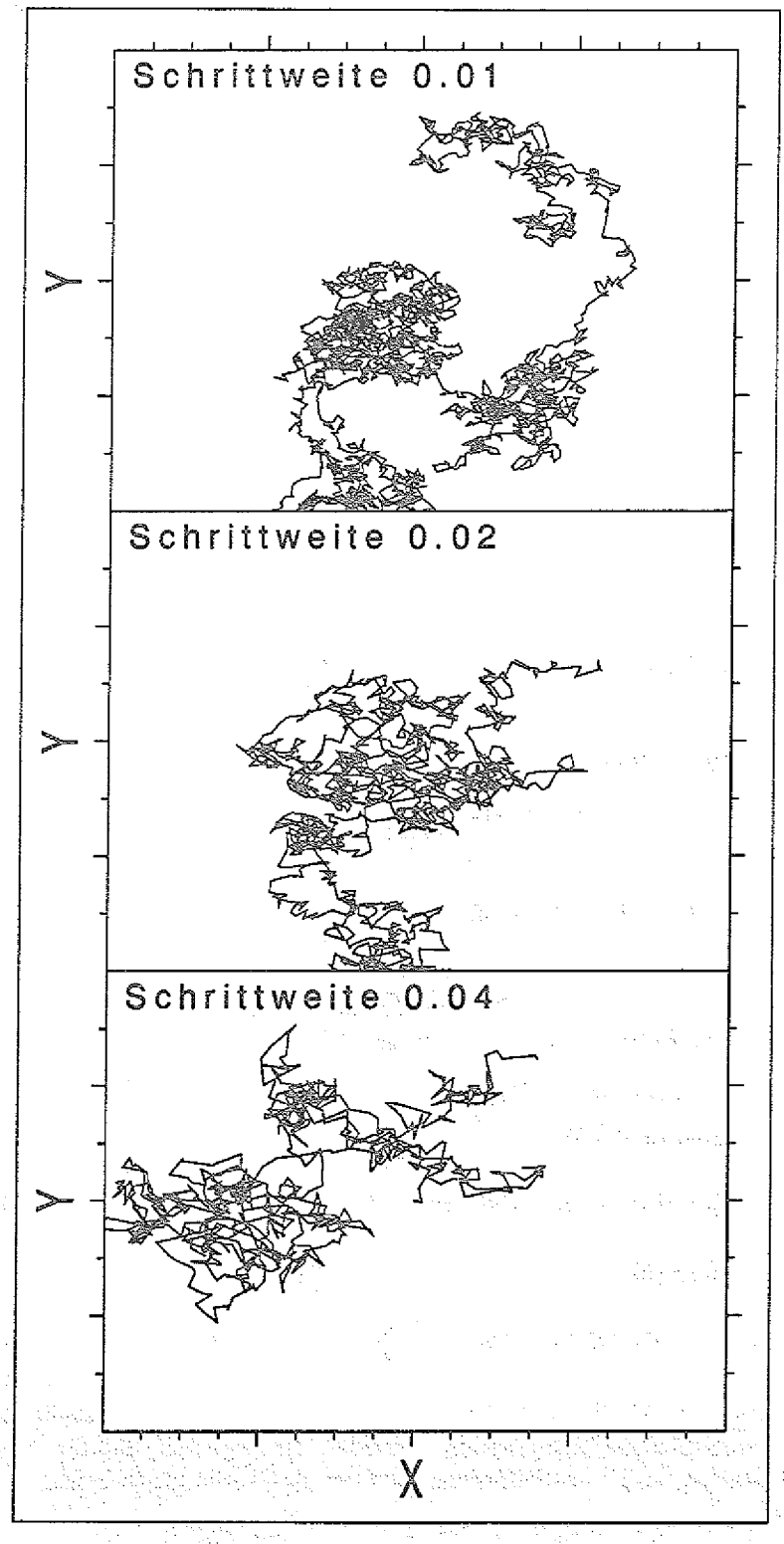


Abbildung 11.1: Stochastisches Euler-Verfahren für den zweidimensionalen Brownschen Prozeß

11.1.3 Verfahren höherer Ordnung

11.1.3.1 Übertragung der Verfahren in die stochastische Kategorie

Die Konvergenz des Euler-Verfahrens kann bewiesen werden. Es hat wie im deterministischen Fall die Konvergenzordnung 1 [1]. Dies motiviert die Übertragung von Verfahren höherer Ordnung in die stochastische Kategorie. Dazu gehört zunächst das Euler-Cauchy-Verfahren (deterministisch 2. Ordnung)

$$x(t+dt) = x(t) + \frac{1}{2}[a(t,x(t)) + a(t,\bar{x}(t))] \cdot dt + \frac{1}{2}[\sqrt{b(t,x(t))} + \sqrt{b(t,\bar{x}(t))}] \cdot dW(t)$$

mit

$$\bar{x}(t) = x(t) + a(t,x(t)) \cdot dt + \sqrt{b(t,x(t))} \cdot dW(t) \quad \text{(Euler-Schritt)}$$

und weiter das Runge-Kutta-Verfahren (deterministisch 4. Ordnung) und seine verschiedenen Varianten mit noch besserer Konvergenz.

11.1.3.2 Konvergenz der Verfahren

Leider konvergieren die Verfahren höherer Ordnung bei stochastischen Differentialgleichungen nicht gegen die richtige Lösung, wie schon ein einfaches Beispiel zeigt: Betrachtet wird die Gleichung

$$dx = x\,dW, \quad x_0 \neq 0\ ,$$

die nur einen zustandsabhängigen Diffusionsterm enthält und bereits in Abschnitt 7.2.2.2 diskutiert wurde (siehe auch Abb. 7.2). Für das einfache Euler-Verfahren berechnet man jetzt

$$\begin{aligned}
x(t+dt) &= x(t) + x(t)\,dW(t) \\
\Rightarrow\ x(n\cdot dt) &= x(0)\cdot\prod_{i=1}^{n}[1+dW(i\cdot dt)] \\
\Rightarrow \mathrm{E}[x(n\cdot dt)] &= x(0)\cdot\prod_{i=1}^{n}[1+\mathrm{E}[dW(i\cdot dt)]] \\
\Rightarrow\ \mathrm{E}[x(t)] &= x(0) \ \text{für alle } t\ ,
\end{aligned}$$

da die Faktoren paarweise unabhängig sind. Analog für das Euler-Cauchy-Verfahren [2]:

$$\begin{aligned}
x(n\cdot dt) &= x(0)\cdot\prod_{i=1}^{n}[1+dW(i\cdot dt)+\frac{1}{2}dW^2(i\cdot dt)] \\
\Rightarrow \mathrm{E}[x(n\cdot dt)] &= x(0)\cdot(1+\frac{1}{2}\cdot dt)^n \\
\Rightarrow\ \mathrm{E}[x(t)] &= x(0)\cdot e^{\frac{1}{2}\cdot t}\ \forall_t \quad \text{(Nach Grenzübergang)}\ .
\end{aligned}$$

Dieselbe Beziehung ergibt sich für das Runge-Kutta-Verfahren.
Ganz allgemein kann (unter gewissen Beschränktheitsbedingungen für die Ableitungen) gezeigt werden, daß ein Runge-Kutta-Schema höherer Ordnung, angewendet auf die Gleichung

$$dx = a(t,x)\,dt + \sqrt{b(t,x)}\,dW$$

eine Lösung der stochastischen Differentialgleichung

[1] Der Begriff "Konvergenzordnung" muß dazu in einem entsprechenden stochastischen Sinne definiert werden.
[2] Die Faktoren entsprechen jeweils den ersten Termen in der Taylor-Entwicklung der Exponentialfunktion. Das gilt auch für das Runge-Kutta-Verfahren.

$$dx = [a(t,x) + \lambda \cdot \frac{\partial\sqrt{b(t,x)}}{\partial x} \cdot \sqrt{b(t,x)}]\,dt + \sqrt{b(t,x)}\,dW\ ,$$

approximiert, wobei λ eine Konstante ist, die aus den Koeffizienten des Verfahrens ausgerechnet werden kann. Insbesondere gilt

$\lambda = 0$ für das Euler-Verfahren,

$\lambda = \frac{1}{2}$ für das Euler-Cauchy-Verfahren,

$\lambda = \frac{1}{2}$ für das Runge-Kutta-Verfahren.

Das oben durchgerechnete Euler-Cauchy-Schema löst also nicht die Gleichung $dx = x\,dW$, sondern vielmehr die Gleichung:

$$dx = \frac{1}{2}\,x\,dt + x\,dW$$

Zur Probe: Das Euler-Schema — auf die letztgenannte Gleichung angewendet — liefert

$$\begin{aligned} x(t+dt) &= x(t) + \frac{1}{2}\,x(t)\,dt + x(t)\,dW \\ \Rightarrow\quad x(n\cdot dt) &= x(0)\cdot\prod_{i=1}^{n}[1 + \frac{1}{2}\cdot dt + dW(i\cdot dt)] \\ \Rightarrow\quad \mathrm{E}[x(t)] &= x(0)\cdot e^{\frac{1}{2}t}\ . \end{aligned}$$

11.1.3.3 Konvergenzordnung der Verfahren

Auch die Konvergenzordnung ist nicht dieselbe wie im deterministischen Fall. Die Zusammenhänge sind hier recht kompliziert: Man bilde die Funktion

$$S(t,x) \stackrel{def}{=} \sqrt{b}\cdot\frac{\partial a}{\partial x} - \frac{\partial\sqrt{b}}{\partial t} - a\cdot\frac{\partial\sqrt{b}}{\partial x} - \frac{1}{2}\cdot b\cdot\frac{\partial^2\sqrt{b}}{\partial x^2}\ .$$

Dann gilt (vgl. [Rue82]):

Im Falle $S(t,x) \neq 0$
kann mit einem allgemeinen Runge-Kutta-Schema stets nur die Konvergenzordnung 2 erreicht werden. Diese Ordnung wird vom Euler-Cauchy-Verfahren erzielt [3].

Im Falle $S(t,x) = 0$ ist immerhin die Ordnung 3 erreichbar. Sie wird vom (klassischen) Runge-Kutta-Schema erzielt.

11.1.3.4 Zusammenfassung

Ein allgemeines Runge-Kutta-Schema kann eingesetzt werden,

- wenn die stochastische Differentialgleichung vorher mit dem Term

$$-\lambda\cdot\frac{\partial\sqrt{b(t,x)}}{\partial x}\cdot\sqrt{b(t,x)}$$

korrigiert und das Verfahren dann auf die veränderte Gleichung angewendet wird.

[3] In engem Zusammenhang mit dem Euler-Cauchy-Schema stehen das sogenannte Heun-Schema [Blu72] und das Milshtein-Schema [Mil78].

- wenn der Diffusionsteil nicht zustandsabhängig ist, so daß gilt:

$$\frac{\partial\sqrt{b(t,x)}}{\partial x} = 0.$$

  Insbesondere gilt dies für lineare Gleichungen.

Dabei wird

- durch das Euler-Verfahren die Konvergenzordnung 1 erzielt,
- unter allgemeinen Bedingungen ($S \neq 0$) mit dem Euler-Cauchy-Verfahren die bestmögliche Konvergenzordnung 2 und
- in Spezialfällen ($S = 0$) (zu denen die linearen Systeme gehören) mit dem Runge-Kutta-Verfahren die bestmögliche Ordnung 3.

Fazit: Die Anwendung von Verfahren mit (deterministischer) Konvergenzordnung größer als 4 ist wenig sinnvoll.

11.1.4 Hybride Verfahren

Die bisher vorgestellten Verfahren legen nahe, daß Drift- und Diffusionsanteil einer stochastischen Differentialgleichung auch getrennt bearbeitet werden können. So ist es z.B. denkbar, auf den deterministischen Anteil ein Runge-Kutta-Verfahren der Ordnung 4 anzuwenden, wogegen der stochastische Anteil mit einem Euler-Verfahren behandelt wird. Solche Vorgehensweisen können dann sinnvoll sein, wenn das deterministische System eine schnelle Dynamik hat, die mit Verfahren höherer Ordnung besser approximiert wird [PT85] (vgl. dazu aber auch 9.1.0.3).

11.2 Filter

Filter sind das stochastische Analogon zu den Zustandsbeobachtern. Das allgemeine Filter hat somit bis auf das System- und Meßrauschen im stochastischen Zustandsraummodell dieselbe Gestalt wie ein Beobachter (vgl. das Beobachterschema in Abb. 9.1):

$$\text{Gleichungen des realen Systems:} \quad \begin{aligned} \dot{x} &= f(t,\alpha,x,u,e_x) \\ y &= g(t,\beta,x,e_y) \end{aligned}$$

$$\text{Filtergleichungen:} \quad \begin{aligned} \dot{\hat{x}} &= \hat{f}(t,\hat{\alpha},\hat{x},u) + k(y-\hat{y}) \\ \hat{y} &= \hat{g}(t,\hat{\beta},\hat{x}) \end{aligned}$$

(mit voneinander unabhängigen Weißen Rauschtermen e_x, e_y). Ein Filter muß im Vergleich zum Beobachter zusätzlich noch die Rauschterme kompensieren: Der Systemzustand muß aus dem Rauschen "herausgefiltert" werden.

11.2.1 Lineare zeitinvariante Theorie

11.2.1.1 Die Gleichungen des linearen Filters

Ebenso wie bei den Beobachtern ist nur das lineare zeitinvariante System einer allgemeinen analytischen Behandlung zugänglich. Das lineare Filter mit korrektem Modell hat die Struktur:

$$\text{Reales System:} \quad \begin{aligned} \dot{x} &= Ax + Bu + \sqrt{R}\cdot e_x \quad , \operatorname{Cov}[e_x] = 1 \\ y &= Cx + \sqrt{S}\cdot e_y \quad , \operatorname{Cov}[e_y] = 1 \end{aligned}$$

$$\text{Filter-Gleichungen:} \quad \begin{aligned} \dot{\hat{x}} &= A\hat{x} + Bu + K(y-\hat{y}) \\ \hat{y} &= C\hat{x} \end{aligned}$$

11.2.1.2 Der Konvergenzsatz

Aus den Filtergleichungen leitet man unmittelbar ab [4]:

$$\begin{aligned}\frac{d}{dt}(x-\hat{x}) &= Ax + Bu + \sqrt{R}\cdot e_x \\ &- A\hat{x} - Bu - K(Cx + \sqrt{S}\cdot e_y - C\hat{x}) \\ &= (A-KC)(x-\hat{x}) + \sqrt{R}\cdot e_x - K\sqrt{S}\cdot e_y\end{aligned}$$

Bis auf die beiden stochastischen Terme ist diese Gleichung aus der deterministischen Theorie bekannt. Erwartungswertbildung führt nun zu

$$\frac{d}{dt}\,\mathrm{E}[x-\hat{x}] = (A-KC)\cdot \mathrm{E}[x-\hat{x}]$$

und [5]

$$\frac{d}{dt}\,\mathrm{Cov}[x-\hat{x}] = (A-KC)\cdot \mathrm{Cov}[x-\hat{x}] + \mathrm{Cov}[x-\hat{x}]\cdot(A-KC)^T + K\,S\,K^T + R\,.$$

Der Erwartungswert konvergiert somit unter denselben Bedingungen wie beim Beobachter (vgl. 9.2.2.1):

$$\mathrm{E}[x-\hat{x}] \to 0 \iff A-KC \text{ stabile Matrix}$$

Wie schon in 7.2.3.4 gezeigt, ist diese Bedingung wiederum äquivalent mit der Stabilität der Matrix-Riccati-Differentialgleichung, so daß zusammengefaßt werden kann:

> Falls $A-KC$ Eigenwerte mit einem Imaginärteil > 0 hat, so divergiert sowohl $\mathrm{E}[x-\hat{x}](t)$ als auch $\mathrm{Cov}[x-\hat{x}](t)$.
>
> Falls alle Eigenwerte von $A-KC$ einen negativen Imaginärteil haben, so konvergiert $\hat{x}(t)$ im Erwartungswert gegen $x(t)$
>
> $$\mathrm{E}[x-\hat{x}] \to 0\,,$$
>
> und die Kovarianzmatrix $P = \mathrm{Cov}[x-\hat{x}](t)$ konvergiert gegen P_∞ mit
>
> $$(A-KC)\cdot P_\infty + P_\infty\cdot(A-KC)^T + K\,S\,K^T + R = 0.$$

11.2.1.3 Konstruktion eines optimalen Filters

Der Konvergenzsatz zeigt, daß unter Einfluß eines Rauschens der Systemzustand auch asymptotisch nicht beliebig genau geschätzt werden kann. Das Problem der optimalen Filterung besteht nun darin, die Verstärkungsmatrix K so zu wählen, daß Konvergenz eintritt und außerdem $P_\infty = P_\infty(K)$ "möglichst klein" wird:

$$K \text{ optimal} \;\Leftrightarrow\; (A-KC) \text{ stabil} \;\wedge\; P_\infty(K) = \min_{\bar{K}} P_\infty(\bar{K})$$

Dazu muß zunächst definiert werden, was mit der Beziehung

$$P_\infty(K) < P_\infty(\bar{K})$$

gemeint sein soll. Es gibt für symmetrische positiv semidefinite Matrizen eine naheliegende Definition, nämlich (siehe auch Abb. 11.2)

---

[4] Die Störung der Meßgröße hat streng genommen erst von dem Moment an eine Bedeutung, wo die Rückkopplung des Filters geschlossen ist (vgl. 7.5.1). Sie tritt dann als weiterer Rauschterm in einer stochastischen Differentialgleichung auf.

[5] Die Matrix-Riccati-Differentialgleichung für solche Systeme mit zwei voneinander unabhängigen stochastischen Termen erhält man analog zu den Gleichungen mit einem Störterm.

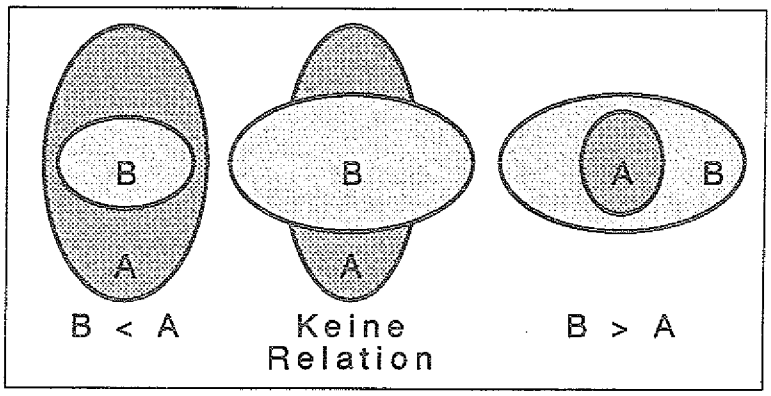


Abbildung 11.2: Eine Vergleichsrelation zwischen Kovarianzmatrizen

$$P < \bar{P} \overset{def}{\iff} 1\Sigma\text{-Ellipse von } P \subset 1\Sigma\text{-Ellipse von } \bar{P}.$$

Mit anderen Worten: "Die Zufallsvariable x mit $P = \mathrm{Cov}[x]$ streut in alle Richtungen weniger stark als $\bar{x}$ mit $\bar{P} = \mathrm{Cov}[\bar{x}]$."
Man beachte, daß diese Relation nicht total ist, d.h. es gibt symmetrische positiv semidefinite Matrizen, die in keiner Relation zueinander stehen, z.B.:

$$P = \begin{pmatrix} 1 & 0 \\ 0 & 2 \end{pmatrix} \text{ und } \bar{P} = \begin{pmatrix} 2 & 0 \\ 0 & 1 \end{pmatrix}.$$

Obwohl die $<$-Relation zwischen Matrizen weniger angenehme Eigenschaften hat, als die $<$-Relation für reelle Zahlen, kann doch gezeigt werden, daß Optimierungsprobleme vom oben beschriebenen Typ eine eindeutige Lösung haben. Diese Lösung kann effektiv berechnet werden [Vog80]. Bezogen auf das Filterproblem, gibt es somit eine optimale Verstärkungsmatrix K, welche $P_\infty(\bar{K})$ minimiert. Dieses K definiert das *Optimalfilter*. Wie diese Lösung numerisch zu bestimmen ist, soll hier nicht weiter untersucht werden, zumal das noch zu besprechende Kalman-Filter gerade so konstruiert ist, daß eine Konvergenz gegen das Optimalfilter eintritt.

11.2.1.4 Eindimensionales Beispiel

Am Beispiel des eindimensionalen Systems

$$\begin{array}{rclll} \dot{x} & = & ax + bu & + \sigma_x \cdot e_x & \qquad \dot{\hat{x}} = a\hat{x} + bu + k(y - \hat{y}) \\ y & = & cx & + \sigma_y \cdot e_y & \qquad \hat{y} = c\hat{x} \end{array}$$

soll das Optimalfilter berechnet werden. Betrachtet wird nur der Fall $c > 0$. Das Filter konvergiert dann für:

$$a - kc < 0 \iff k > \frac{a}{c}$$

Die asymptotische Varianz $p_\infty = \mathrm{Var}[x - \hat{x}](\infty)$ genügt der Gleichung:

$$(a - kc) \cdot p_\infty + p_\infty \cdot (a - kc) + k\sigma_y^2 k + \sigma_x^2 = 0 \iff p_\infty = -\frac{1}{2} \cdot \frac{k^2\sigma_y^2 + \sigma_x^2}{a - kc}$$

Abbildung 11.3 gibt einen typischen Kurvenverlauf für $p_\infty = p_\infty(k)$ im Bereich $[\frac{a}{c}, \infty]$ wider. Eine Kurvendiskussion zeigt, daß $p_\infty(k)$ exakt die Verhaltensweisen widerspiegelt, die beim linearen eindimensionalen Beobachter nur empirisch konstatiert werden konnten:

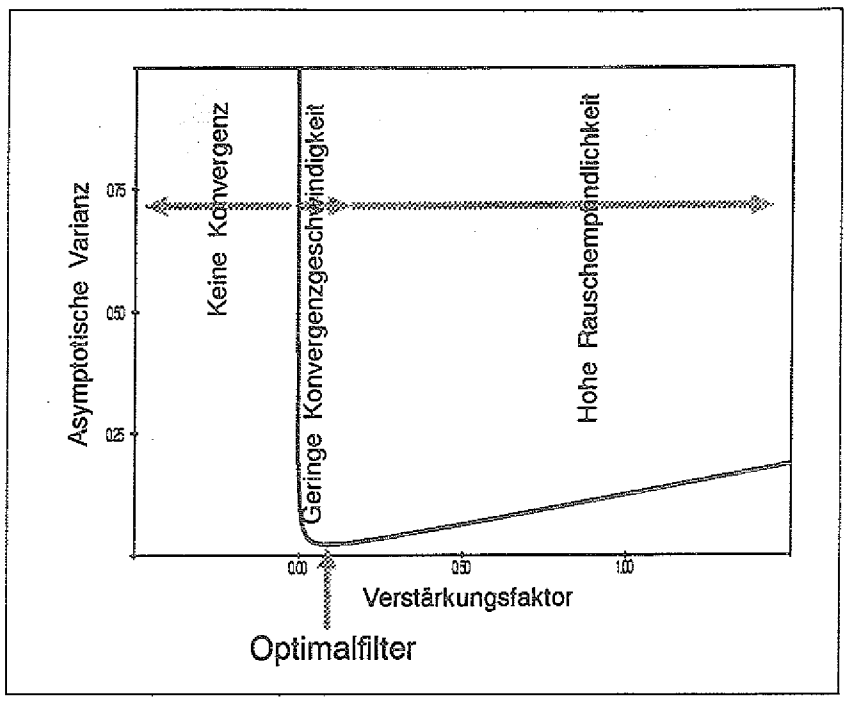


Abbildung 11.3: Asymptotische Varianz eines linearen Filters 1.Ordnung

- Am linken Rand $k = \frac{a}{c}$ hat $p_\infty(k)$ einen Pol, der durch die Nicht-Konvergenz $\hat{x} \rightarrow x$ an dieser Stelle bedingt ist.
- Am rechten Rand ($k \rightarrow \infty$) befindet sich eine zweite Polstelle, die der zunehmenden Rauschempfindlichkeit des Filters für große Werte von k entspricht.
- Zwischen den Polen ist $p_\infty(k)$ stetig mit genau einem Minimum bei:

$$k_{opt} = \frac{a}{c} + \sqrt{\frac{a^2}{c^2} + \frac{\sigma_x^2}{\sigma_y^2}}$$

Die zugehörige Varianz ist:

$$p_\infty(k_{opt}) = \frac{\sigma_y^2}{c}\left(\frac{a}{c} + \sqrt{\frac{a^2}{c^2} + \frac{\sigma_x^2}{\sigma_y^2}}\right)$$

Die optimale Lösung zeigt einen interessanten Effekt: Meß- und Systemrauschen wirken sich genau entgegengesetzt auf k_{opt} aus. Dies entspricht der intuitiven Vorstellung von den Zusammenhängen. Bzgl. der asymptotischen Varianz $p_\infty(k_{opt})$ wirken sich beide Rauschterme — wie zu erwarten — gleichsinnig aus. Die Abbildung 11.4 zeigt das Verhalten des Filters für einige Werte von k.

11.2.2 Nichtlineare Theorie

Die Techniken zur Konstruktion nichtlinearer Filter unterscheiden sich wenig von denen für nichtlineare Beobachter (vgl. 9.2.3). Ein Aspekt, der neu hinzutritt, sind mögliche Modellfehler bzgl. der Rauschterme. Beim linearen Filterentwurf hat die korrekte Beschreibung des Rauschens keinen Einfluß auf die Konvergenz des Filters, da die Annahmen über die Störterme erst bei der Berechnung der optimalen Verstärkungsmatrix benötigt werden. Ganz allgemein beeinflußt die Wahl der Rauschterme im wesentlichen die asymptotische Kovarianzmatrix des Zustandsschätzers und weniger die Konvergenz.

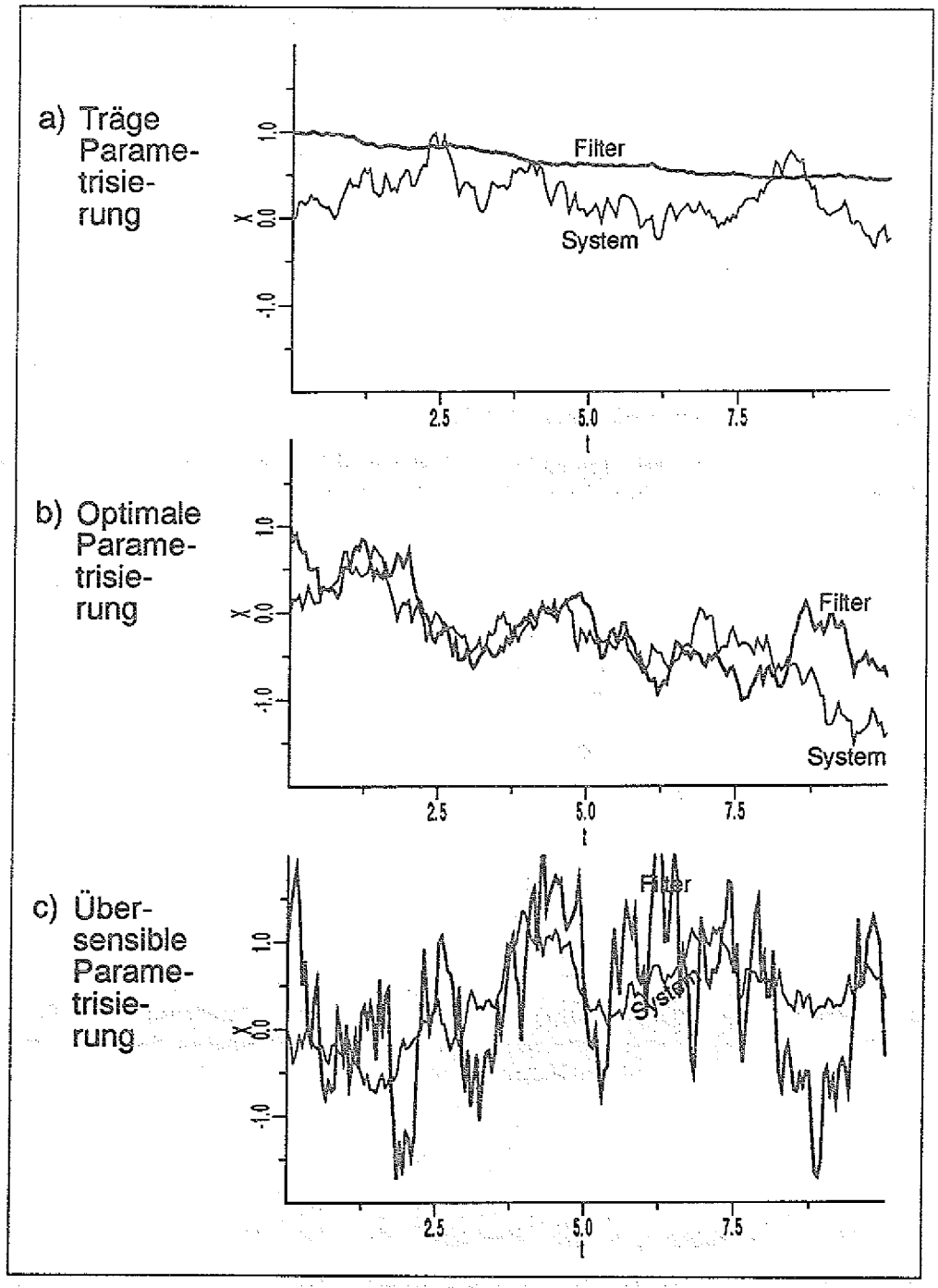


Abbildung 11.4: 1-dimensionales lineares Filter

11.3 Kalman-Filter

11.3.1 Die Filter-Gleichungen

Das Kalman-Filter ist ein spezielles Filter mit einer zeitabhängigen Verstärkungsmatrix $K(t)$, die automatisch angepaßt wird [Kre80] [Jaz80] [BS75]. Dabei wird die Matrix-Riccati-Gleichung für das Filter zusätzlich zu den eigentlichen Filtergleichungen gelöst, um aus der Kovarianzmatrix die nötigen Informationen für die optimale Wahl von K abzuleiten. Die Filter-Gleichungen haben somit die allgemeine Form (vgl. 11.2.1.1)

$$\begin{aligned} \dot{x} &= Ax + Bu + \sqrt{R} \cdot e_x \\ y &= Cx + \sqrt{S} \cdot e_y \end{aligned} \qquad \begin{aligned} K &= K(P) \\ \dot{\hat{x}} &= A\hat{x} + Bu + K \cdot (y - \hat{y}) \\ \hat{y} &= C\hat{x} \\ \dot{P} &= (A - KC) \cdot P + P \cdot (A - KC)^T + K\,S\,K^T + R \end{aligned}$$

mit $P = \mathrm{Cov}\,[x - \hat{x}]$. Hier nimmt die Matrix-Riccati-Gleichung durch die Rückkopplung bzgl. K eine nichtlineare Gestalt an.

11.3.1.1 Arbeitsweise des semi-diskreten Filters

Die Herleitung der optimalen Verstärkungsmatrix $K(P)$ ist am einfachsten für den semi-kontinuierlichen Fall zu durchschauen. Bei einem solchen System kann in einem Zeitraum, in dem keine Meßwerte eintreffen, auch keine neue Information aus dem System aufgenommen werden, so daß in dieser Situation die Filter-Gleichungen von den Systemgleichungen entkoppeln. Die Filter-Gleichungen im Zeitraum zwischen zwei Abtastungen sind:

$$\begin{aligned} \dot{\hat{x}} &= A\hat{x} + Bu \\ \dot{\hat{y}} &= C\hat{x} \\ \dot{P} &= AP + PA^T + R \end{aligned}$$

Erst zu dem Zeitpunkt, an dem ein neuer Meßwert y eintrifft, kann dieser zur Korrektur von $\hat{x}$ verwendet werden. Um diesen sogenannten *Updating*-Schritt präzise formulieren zu können, muß (vorübergehend) die Terminologie verfeinert werden:

Mit

$$\hat{x}^-,\ \hat{y}^-,\ P^-$$

seien die *a priori* Schätzungen bezeichnet, die zum Zeitpunkt des Eintreffens eines Meßwertes y bestehen. In die a priori-Schätzwerte ist die in y enthaltene Information noch nicht eingegegangen. Es sind gerade die aus der Matrix-Riccati-Gleichung gewonnenen Schätzer.
Mit

$$\hat{x}^+,\ \hat{y}^+,\ P^+$$

seien die *a posteriori* Schätzungen nach Verarbeitung von y bezeichnet.

Mit dieser Bezeichnungsweise formuliert man den Updating-Schritt als:

$$\begin{aligned} \hat{x}^+ &= \hat{x}^- + K(P^-) \cdot (y - \hat{y}^-) \\ P^+ &= P^- + L(P^-) \cdot P^- \end{aligned}$$

mit zwei noch näher zu bestimmenden Verstärkungsmatrizen $K(P^-)$ und $K(L^-)$. Dieser Ablauf ist in Abbildung 11.5 an einem zweidimensionalen Beispiel illustriert.

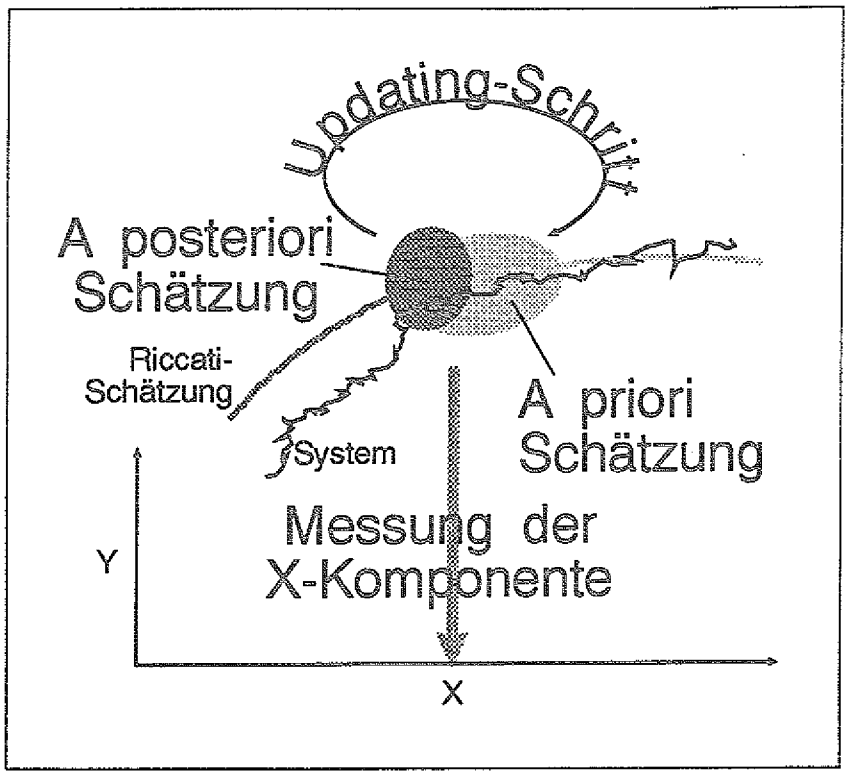


Abbildung 11.5: Arbeitsweise des semikontinuierlichen Kalman-Filters

11.3.1.2 Herleitung der Updating-Gleichungen

Die Berechnung optimaler Verstärkungsmatrizen $K(P^-), L(P^-)$ gestaltet sich sehr einfach, wenn man sich auf lineare unverzerrte Schätzer beschränkt. Dann stehen alle Ergebnisse der linearen Schätztheorie, insbesondere das Gauß-Markov-Theorem (vgl. 10.2.2.1) zu Verfügung. Ein lineares Modell zur Schätzung von x ist gegeben durch:

$$\begin{aligned} \hat{x}^- &= 1 \cdot x + (\hat{x}^- - x) \\ y &= C \cdot x + \sqrt{S} \cdot e_y \end{aligned}$$

oder in Matrixschreibweise:

$$\begin{pmatrix} \hat{x}^- \\ y \end{pmatrix} = \begin{pmatrix} 1 \\ C \end{pmatrix} \cdot x(t) + \begin{pmatrix} x - \hat{x}^- \\ \sqrt{S} \cdot e_y \end{pmatrix}$$

Dieses Modell verdient eine nähere Betrachtung: Es faßt in der ersten Zeile exakt die Vergangenheitsinformation zusammen, wobei die Kovarianzmatrix $\mathrm{Cov}[x - \hat{x}^-]$ nichts anderes als die mit Hilfe der Riccati-Gleichung berechnete Matrix P^- ist. Die zweite Zeile enthält dagegen die Information über die Messung, wobei die Kovarianzmatrix S vorgegeben ist:

$$\mathrm{Cov} \begin{bmatrix} x - \hat{x}^- \\ \sqrt{S} \cdot e_y \end{bmatrix} = \begin{pmatrix} P^- & 0 \\ 0 & S \end{pmatrix}$$

Wendet man auf dieses lineare Modell das Gauß-Markov-Theorem an (was einige Matrizen-Rechentechnik verlangt, aber im Prinzip eine einfache Gleichungsumformung ist), so erhält man zwei Gleichungen für den

$$\text{BLUE } x^+ \text{ und dessen Kovarianzmatrix } P^+,$$

die sich schreiben lassen als [6]:

$$\begin{aligned} K &= P^- C^T S^{-1} \\ L &= K C \\ P^+ &= P^- + L P^- \\ \hat{x}^+ &= \hat{x}^- + K(y - \hat{y}^-) \end{aligned}$$

Nun können die soeben eingeführten terminologischen Unterscheidungen wieder fallen gelassen werden, so daß die Updating-Gleichungen die Gestalt

$$\begin{aligned} K &:= P C^T S^{-1} \\ P &:= (1 - KC) P \\ \hat{x} &:= \hat{x} + K(y - \hat{y}) \end{aligned}$$

annehmen Die Gleichungen sind jetzt im Sinne einer Wertzuweisung in einer Programmiersprache zu lesen!

11.3.1.3 Der kontinuierliche Fall

Die Herleitung der Filter-Gleichungen für das kontinuierliche System beruht auf einem Grenzübergang, bei dem das Abtastintervall Δt des semi-kontinuierlichen Systems gegen 0 läuft. Dabei verschmelzen die Updating-Gleichungen mit den Gleichungen für die Zeiträume zwischen dem Eintreffen der Meßwerte zu den kontinuierlichen Kalman-Filter-Gleichungen:

$$\begin{aligned} \dot{\hat{x}} &= A\hat{x} + Bu + K \cdot (y - \hat{y}) \\ K &= P C^T S^{-1} \\ \dot{P} &= AP + PA^T + K S K^T + R \end{aligned}$$

Dies sind exakt die Gleichungen, die sich ergeben, wenn in die in Abschnitt 11.3.1 genannte allgemeine Form der Filter-Gleichungen die Verstärkung

$$K = K(P) = P C^T S^{-1}$$

eingesetzt wird:

$$\begin{aligned} \dot{P} &= (A - KC) \cdot P + P \cdot (A - KC)^T + K S K^T + R \\ &= AP + PA^T - 2 P C^T S^{-1} C P + P C^T S^{-1} S S^{-1} C P + R \\ &= AP + PA^T - K S K^T + R \end{aligned}$$

Die kontiniuierlichen Gleichungen haben für die biotechnische Situation wenig Bedeutung, da von einer diskreten Abtastung mit nicht vernachlässigbar kleiner Abtastrate ausgegeangen werden muß. Das semi-kontinuierliche Kalman-Filter ist somit die Methode der Wahl. Insbesondere kann die Länge des Abtastintervalls bei der semi-kontinuierlichen Form auch wechseln. Ferner können Meßwerte fehlen, was mit einer Meßgleichung wechselnder Dimension leicht kompensiert werden kann.

[6] Die Gleichungen präzisieren, auf welche Weise ein Kalman-Filter die über den Momentan-Meßwert $y(t)$ hinaus gehende Vergangenheits-Information $\hat{x}^-$ auswertet (vgl. dazu auch 6.6.3.1).

11.3.2 Beispiele

11.3.2.1 Eindimensionaler Fall

Beim eindimensionalen Beispiel ergeben sich die kontinuierlichen Kalman-Filter-Gleichungen

$$\begin{aligned} \dot{x} &= ax + bu + \sigma_x \cdot e_x \\ y &= cx + \sigma_y \cdot e_y \end{aligned} \qquad \begin{aligned} \dot{\hat{x}} &= a\hat{x} + bu + k(y - \hat{y}) \\ k &= pc/\sigma_y^2 \\ \dot{p} &= ap + pa - k\sigma_y^2 k + \sigma_x^2 \end{aligned}$$

und nach Einsetzen des Terms für k:

$$\begin{aligned} \dot{\hat{x}} &= a\hat{x} + bu - pc/\sigma_y^2 \cdot (y - \hat{y}) \\ \dot{p} &= 2ap - p^2c^2\sigma_y^2 + \sigma_x^2 \end{aligned}$$

Daraus läßt sich die asymptotische Varianz und die zugehörige asymptotische Verstärkungsmatrix berechnen als

$$\begin{aligned} p_\infty &= \frac{\sigma_y^2}{c}\left(\frac{a}{c} + \sqrt{\frac{a^2}{c^2} + \frac{\sigma_x^2}{\sigma_y^2}}\right) \\ k_\infty &= \frac{a}{c} + \sqrt{\frac{a^2}{c^2} + \frac{\sigma_x^2}{\sigma_y^2}} \end{aligned}$$

Das Kalman-Filter konvergiert somit gegen das bereits in Abschnitt 11.2.1.4 berechnete Optimalfilter für das lineare System! Insbesondere ist das Filter stabil.

11.3.2.2 Ein zweidimensionales Beispiel

Als zweidimensionales Beispiel soll erneut das schwingende System (vgl. 7.3.3.1 und 9.2.2.3)

$$\begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \end{pmatrix} = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + \sigma_x \begin{pmatrix} e_1 \\ e_2 \end{pmatrix}$$

mit der Meßgleichung

$$y = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + \sigma_y \cdot e_y = x_1 + \sigma_y \cdot e_y$$

herangezogen werden. Die Abbildung 11.6 zeigt einen Durchlauf des semi-diskreten Filters zusammen mit den jeweiligen a priori und a posteriori-Schätzungen, dargestellt durch die 1Σ-Ellipsen. Darunter ist ein Lauf mit sehr dichter Abtastung zu sehen, der im Prinzip dem kontinuierlichen Filter entspricht. Am wechselnden Abstand zwischen wahrem Wert und Schätzer ist sehr gut zu erkennen, daß über die Messung nur eine Information über die x-Koordinate zur Verfügung steht: Bei einer geringen Abweichung in x-Richtung kann die y-Abweichung nur schwer korrigiert werden.

11.3.3 Parametrisierung eines Kalman-Filters

Die Abbildung 11.7 zeigt die Auswirkungen eines Modellfehlers bzgl. $\sigma_y^2 = \mathrm{Var}[e_y]$. Der Parameter bestimmt entscheidend die Stärke der Rückkopplung von der Vorhersagedifferenz $y - \hat{y}$ auf den Systemzustand. Ein zu kleiner Parameter täuscht eine sehr genaue Messung vor und hat daher eine starke Korrekturwirkung, ein großer Parameter macht das Filter träge im Folgeverhalten.

Das Beispiel zeigt, daß die Auswirkungen, die eine Veränderung des Parameters σ_y auf das Verhalten des Filters hat, aufgrund des Modells sehr genau vorausgesagt werden können. Dasselbe gilt für die übrigen Rauschterme des Modells. Diese Eigenschaft ist für den Anwender bei der Parametrisierung eines Kalman-Filters von großem Vorteil.

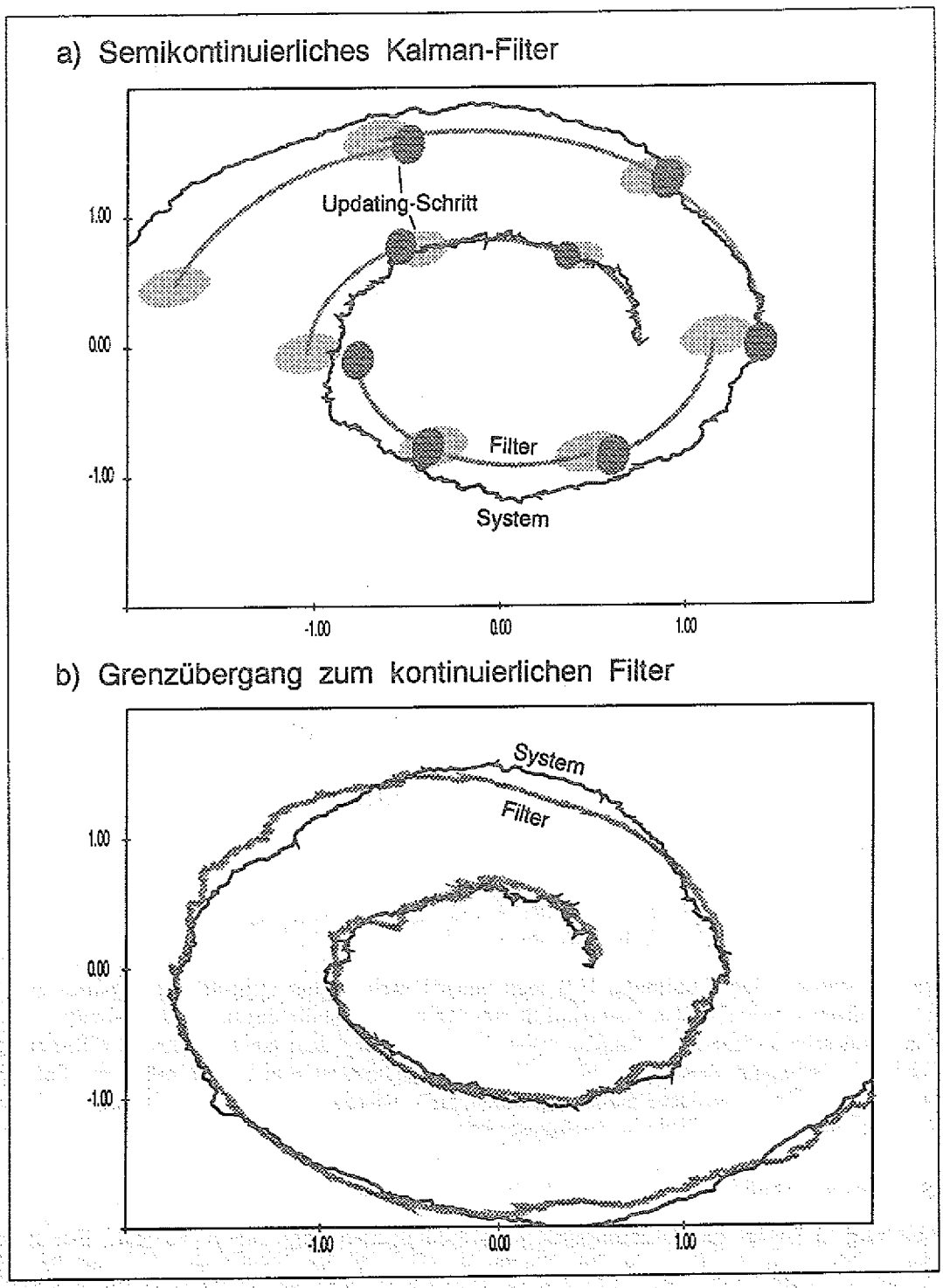


Abbildung 11.6: Durchlauf eines 2-dimensionalen linearen Kalman-Filters

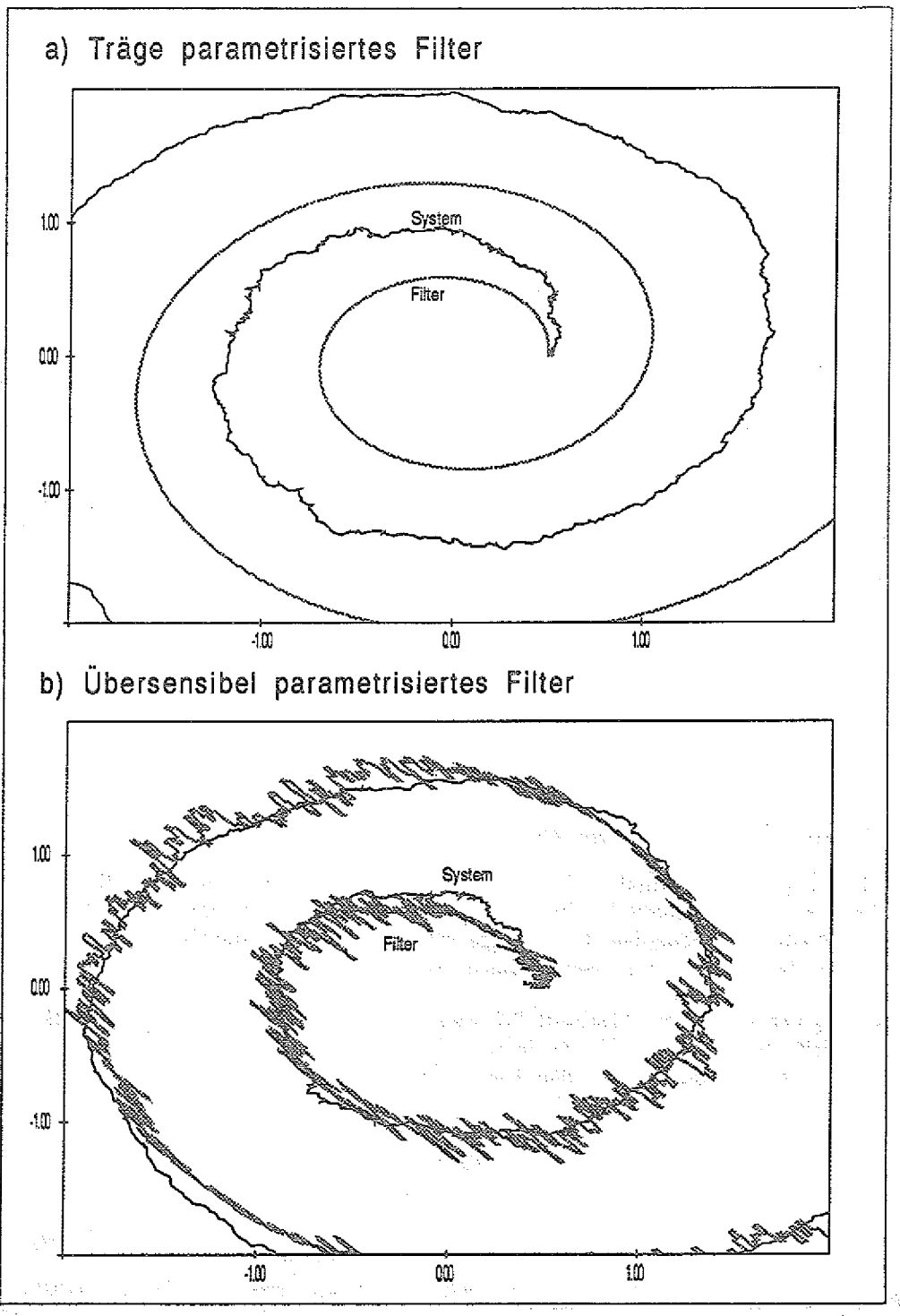


Abbildung 11.7: Durchlauf eines 2-dimensionalen linearen Kalman-Filters bei inkorrekter Modellierung des Meßrauschens.

Im Vergleich dazu ist ein Beobachter nur schwer und auf eine sehr indirekte Weise über die Eigenwerte parametrisierbar. Die Auswirkung einer willkürlichen Veränderung der Verstärkungsmatrix bei einem Beobachter kann erst nach einer aufwendigen Rechnung ermittelt werden!

Dieser Vergleich zeigt, warum das Kalman-Filter eine so herausragende Rolle unter den Beobachtern und Filtern hat. Beim praktischen Einsatz sollte ihm nach Möglichkeit der Vorzug vor einem einfachen Filter gegeben werden.

11.3.4 Erweiterte und adaptive Kalman-Filter

Die Linearisierungstechniken für nichtlineare Beobachter (vgl. 9.2.3.1) können ohne Modifikationen auch auf das Kalman-Filter übertragen werden:

- Das mit Hilfe einer Linearisierung konstruierte Kalman-Filter mit nichtlinearer Modellgleichung und linearer Rückkopplung wird als *Erweitertes Kalman-Filter* bezeichnet. Die Gleichungen des Erweiterten Kalman-Filters sind bereits in Abschnitt 2.3.2 angegeben worden.
- Kommt durch Hinzufügen weiterer Zustandsvariablen eine Parameteranpassung hinzu (vgl. 9.2.4), so spricht man von einem *adaptiven Erweiterten Kalman-Filter*. Beispiele hierfür gibt es im Rahmen dieser Arbeit genug (vgl. auch Abschnitt 2.3, 7.4, 22.1).

Ein interessanter Aspekt der nichtlinearen Filterung soll noch kurz angeschnitten werden: Die Herleitung der Updating-Gleichungen beruht im Kern auf einer linearen Schätzung und dem Gauß-Markov-Theorem (vgl. 10.2.2.1). Im Prinzip kann diese Schätzung auch für ein nicht normalverteiltes Meßrauschen durchgeführt werden. Dabei können auch andere Schätzer zum Einsatz kommen, die die Charakteristik des Rauschens besser berücksichtigen als der BLUE. Besonders interessant wird dieser Ansatz bei unsymmetrischen Verteilungen (vgl. z.B. die Trübungsmessung in Abschnitt 6.3) oder Ausreißerverteilungen. Sie können dann zur Konstruktion spezieller *robuster* Filter führen. Dieser Gedanke unterstreicht erneut die Bedeutung einer Modellierung des Meßrauschens (vgl. 6.3).

Vom theoretischen Standpunkt aus gestaltet sich der Ansatz schwierig, zumal mit nicht normalverteiltem Meßrauschen auch der Systemzustand seine Normalverteilung verliert! Das Auffinden geeigneter Approximationen für derartige Systeme ist zur Zeit Gegenstand der Forschung (vgl. z.B. [Haz88] [Yav85]).

11.3.5 Numerik des Kalman-Filters

Bei der Realisierung des Filters muß die Positiv-Definitheit der $n \times n$-System-Kovarianzmatrix P gewährleistet bleiben. Aufgrund numerischer Instabilitäten bei Vorliegen von Eigenwerten $\lambda_i \sim 0$ kann diese Bedingung durchaus verletzt werden. In der Folge wird auch das Filter instabil. Zwei einfache Techniken zur Stabilisierung bzw. zur Lösung dieses Problems sollen kurz genannt werden:

Symmetrisierung der Matrix: In Abschnitt 7.2 wurde gezeigt, daß die Matrix-Riccati-Differentialgleichung bereits durch $\frac{1}{2}n(n+1)$ Gleichungen bestimmt ist, da die Matrix (bei exakter Rechnung) symmetrisch bleibt. Diese Redundanz kann ausgenutzt werden, um die Matrix nach jedem Einzelschritt des Differentialgleichungs-Lösers nachträglich zu symmetrisieren:

$$P := \frac{1}{2}\left(P + P^T\right)$$

Die Mittelwertbildung führt zu einer gewissen numerischen Stabilisierung (vgl. [Sch75]).

Algebraische Umformung: Die Gleichungen des diskreten Kalman-Filters können so umgeformt werden, daß die Symmetrie und Positiv-Definitheit von P bei jedem Rechenschritt auch numerisch garantiert bleiben (vgl. hierzu [Lew86]).

Weitere numerische Studien finden sich in [Yav85].

Kapitel 12

Logische und heuristische Methoden

In Entsprechung zu den in Abschnitt 8 vorgestellten logischen und heuristischen Modellen geht es nun darum, eine kurze Vorstellung von möglichen Methoden zu geben:

12.1 Aussagenlogische Methoden

Ausagenlogische Modelle sind mit den Rechenregeln des ausagenlogischen Kalküls behandelbar. Das Problem der automatischen Folgerung von Aussagen aus einem Satz von aussagenlogischen Ausdrücken ist theoretisch und praktisch gelöst. Die entsprechenden Verfahren beruhen im wesentlichen auf bestimmten Normalformen für ausagenlogische Ausdrücke (z.B. die *Konjunktive Normalform* [BN77]). Damit kann eine beliebige Aussage aus gegebenen Fakten abgeleitet oder aber als nicht ableitbar bzw. falsch erwiesen werden. Insbesondere ist die Konsistenz und Widerspruchsfreiheit von aussagenlogischen Systemen beweisbar.

12.2 Beweismethoden für Regelsysteme

Im Gegensatz zur Aussagenlogik gerät man bei der Anwendung der Prädikatenlogik in den Gültigkeitsbereich des Gödelschen Unvollständigkeitssatzes [Her78]. D.h. es ist prinzipiell unmöglich, gewisse Aussagen innerhalb des Kalküls als wahr zu erweisen, insbesondere ist eine Konsistenzprüfung nicht durchführbar.

Für spezielle Unterklassen der prädikatenlogischen Ausdrücke sind allerdings effektive Beweisverfahren bekannt. Sie beruhen in der Regel auf der sogenannten *Universellen Normalform*, welche wiederum auf die *Klauselform* der Logik führt. Spezielle Formen der Klausellogik, wie z.B. die *Horn-Klauseln*, standen bei der Entwicklung von Sprachen der Künstlichen Intelligenz Pate (vgl. auch 8.2). Das bekannteste praktisch einsetzbare Beweisverfahren für Klausellogiken ist die *Resolventenmethode* [BN77].

Im allgemeinen Fall kann die Folgerung von Aussagen aus einem Regelsystem nur auf heuristischen Strategien beruhen, deren Erfolg im allgemeinen nicht nachgewiesen werden kann. Im wesentlichen handelt es sich bei allen diesen Strategien um Suchmethoden, die versuchen, eine geschlossene Folgerungskette von den Fakten zu den zu beweisenden Aussagen zu konstruieren. Ein großer Teil der heutigen Forschungen auf dem Gebiet der Künstlichen Intelligenz beschäftigt sich mit der Entwicklung solcher Heuristiken [Win79] [Pea84]. Die beiden bekanntesten Suchstrategien sind

das Backward-Reasoning, womit versucht wird, von der zu beweisenden Aussage aus rückwärts eine Beweiskette zu konstruieren. Das Verfahren terminiert, wenn alle Prämissen in der zu beweisenden Aussage als wahr erwiesen werden können. Die Programmiersprache PROLOG beruht auf diesem Konzept als strukturbildendem Element [CM87]. Eine typische Fragestellung für das Backward-Reasoning lautet:

> "Die Symptome sind gegeben. Wo liegen die Ursachen?"

das Forward-Reasoning, womit genau in entgegengesetzter Richtung von den Fakten hin zu den Folgerungen vorgegangen wird:

"Gegeben sind die Tatsachen. Was wird die Folge sein?"

Heutige Expertensysteme und Sprachen der Künstlichen Intelligenz beinhalten oft beide Mechanismen, die durch eine sogenannte *Inferenzmaschine* implementiert werden. Die Widerspruchsfreiheit eines Regelsystems stellt ein besonderes Problem dar, da solche Regelsysteme in der Praxis sehr schnell anwachsen und schwer überschaubar werden können.

12.3 Algorithmische Modelle

Man kann einen Compiler als eine Methode zur Behandlung algorithmischer Modelle ansehen. Gewissermaßen ist der Compiler zusammen mit dem Run-Time-System das Analogon zu einem Simulationssystem für Differentialgleichungs-Modelle.

12.4 Syntaktische Klassifikation

Wie bereits in Abschnitt 8.4 beschrieben, beruht eine syntaktische Klassifikation auf einer formalen Grammatik für die zu erkennenden Wortklassen. Die Methode, die unmittelbar der Grammatik entspricht, ist der zugehörige *Scanner* bzw. *Parser*. Er kann für bestimmte Sprachklassen auf automatischem Wege aus der Grammatik erzeugt werden. Dazu mehr in 18.3.3.

12.5 Weitere Methoden

Methoden zu den übrigen in Kapitel 8 erwähnten Modellen sind nachzuschlagen in:

- [Rei82] [JV87] für Netzwerkmodelle,
- [HRWL83] [Neg85] [Beu88] für Expertensysteme,
- [DP80] [BS85] [Doh85] für Fuzzy Models und
- [CERS88] für neuronalen Netze.

Kapitel 13

Graphische Methoden

Wie schon in 3.4 erläutert, nehmen die graphischen Methoden eine Sonderstellung ein, da sie kein explizit formuliertes Modell zur Grundlage haben. Sie sind vielmehr direkt an die visuelle Wahrnehmung des Datenanalytikers gekoppelt. Implizit ist darin allerdings ein Modell enthalten, welches sich unsere Wahrnehmung von den Daten macht. Graphische Methoden bilden somit die eigentlich kreativen Elemente unter den Datenanalysebausteinen. Ihre Bedienung erfolgt zunächst rein intuitiv; die getroffenen Entscheidungen sind erst nachträglich in explizit formulierte Modelle zu fassen.

Graphische Methoden sind im Umfeld der Präsentationsgraphik, der interaktiven Computergraphik und der Explorativen Datenanalyse in großer Zahl entwickelt worden [CCKT83] [Cle85] [CM88] [McC87] [DT82]. Es genügt hier eine kurze Aufzählung und Bewertung der Anwendbarkeit im biotechnologischen Aufgabenbereich.

13.1 Graphische Präsentation

13.1.1 Visuelle Perzeption

Die Gestaltung graphischer Darstellungen sollte nicht erst zum Zeitpunkt des Verfassens einer Publikation oder eines Vortrags erfolgen. Ganz im Gegenteil hat die graphische Präsentation zu allen Zeitpunkten — also insbesondere auch während einer Datenanalyse-Sequenz — prinzipiell dieselbe Aufgabe: Sie soll

- das schnelle Erfassen des Informationsgehaltes eines Datensatzes fördern,
- den Blick auf Wesentliches lenken und
- sekundäre Informationen in den Hintergrund stellen.

Die Frage, welche Art der Präsentation diese Aufgaben optimal löst, führt unmittelbar zu einer anderen Fragestellung, nämlich der nach der Funktionsweise der menschlichen visuellen Wahrnehmung. Die Untersuchung der *visuellen Perzeption* gehört in die Gebiete der Wahrnehmungspsychologie, aber auch der Sinnesphysiologie.

Ein einfaches Modell für die visuelle Perzeption numerischer Daten stellt das Schema

$$\text{Daten} \xrightarrow{\textit{Codierung}} \text{Präsentation} \xrightarrow{\textit{Decodierung}} \text{Auge und Gehirn}$$

dar [Cle85]. Das Ziel einer Präsentation muß sein, die Informationsverluste beim Vorgang der Codierung und Decodierung möglichst gering zu halten. Hier gilt ganz allgemein [Cle85]:

- graphische Methoden stellen die Daten in ihrer Gesamtheit dar (Informationserhaltung),
- während numerische Methoden stets eine Datenreduktion vornehmen (Informationsverlust).

In diesem Unterschied liegt die Wichtigkeit einer graphischen Präsentation begründet.

Eine hervorragende systematische Darstellung der Methoden und Kriterien zur wahrnehmungsgerechten graphischen Präsentation von Daten sind die "Elements of Graphing Data" von Cleveland [Cle85]. Die Vorarbeiten zu diesem Buch umfaßten unter anderem umfangreiche Vergleiche verschiedenster graphischer Darstellungen und wahrnehmungspsychologische Testreihen bzgl. verschiedener Kriterien [CM84].

13.1.2 Konfiguration graphischer Bausteine

Cleveland weist an vielen Beispielen nach, daß es keine generell "beste" Präsentationstechnik geben kann, sondern stets nur problemspezifisch ausgerichtete Lösungen. Z.B. erfordert die graphische Darstellung des Resultats einer Kurvenanpassung eine andere Gestaltungsmittel als die Vergleichsdarstellung zweier Verteilungsfunktionen.

Die Konsequenz für ein Datenanalysesystem muß die Forderung nach größtmöglicher Flexibilität zur Gestaltung graphischer Darstellungen auch innerhalb des Systems sein. Nur auf diese Weise können Visualisierungen auf spezielle Probleme "maßgeschneidert" werden. Die Konfiguration der graphischen Bausteine muß im Rahmen des allgemeinen Konzepts vollständig auf interaktivem Wege möglich sein.

Ein graphischer Baustein hat somit eine ebenso so große Konfigurations-Komplexität wie ein modellgestützter Datenanalysebaustein: Er benötigt eine vollständige Beschreibung der graphischen Darstellung in einer formalisierten Gestalt! Diese Beschreibung betrifft nicht nur das Layout von Koordinatenachsen und Beschriftungen, sondern auch Angaben über die Darstellung von Linien, Datenpunkten, Markierungen etc..

13.2 Visualisierungstechniken

Verschiedene graphische Techniken zur Visualisierung sind bereits in Kapitel 2 vorgestellt worden. Sie sollen an dieser Stelle kurz zusammengestellt und um einige weitere Methoden aus der Literatur ergänzt werden.

13.2.1 Hervorhebung und Inspektion von Daten

Hierunter sollen Techniken verstanden werden, die dazu dienen, Teile eines gegebenen Datensatzes deutlicher hervorzuheben, zu inspizieren oder getrennt von den anderen Daten zu betrachten. Die meisten Methoden wurden schon im Rahmen des Beispiels in Abschnitt 2.2 vorgestellt:

- Vergrößern von Teilbereichen (*Zoom In*) sowie die umgekehrte Operation (*Zoom Out*),
- Verschieben des aktuellen Sichtfensters (*Zentrierung*),
- Ein- und Ausblenden von zusammenhängenden Daten (z.B. Kurven),
- Einfärben oder Umfärben von Datenpunkten oder Teildatensätzen,
- Abfrage des expliziten numerischen Wertes eines Datenpunktes.

13.2.2 Graphische Dateneingabe und -veränderung

Einige Visualisierungstechniken, welche die Eingabe oder Veränderung von Datensätzen ermöglichen, sind (vgl. 2.2 und 2.3.7):

13.2.2.1 Dateneingabe

Es muß bemerkt werden, daß in der freien Dateneingabe die Gefahr der willkürlichen Umgestaltung von Meßdaten liegt. Dies muß das System auf jeden Fall verhindern! Es sollte daher lediglich die Neueingabe einzelner Werte (z.B. als Startwerte einer Simulation) gestattet werden, keinesfalls aber das Hinzufügen von Werten zu einem bestehenden Datensatz. Möglichkeiten der Eingabe sind:

- Graphische Parametrisierung mit Hilfe von Skalen und Balkendarstellungen,

- Eingabe von Punkten im Koordinatensystem (z.B. mit einem Fadenkreuz),
- Eingabe ganzer Profile als freihand-gezeichnete Kurven.

13.2.2.2 Datenveränderung

Hierunter soll — aufgrund des genannten Vorbehaltes — nicht die Veränderung der numerischen Werte, sondern das Hinzufügen von Attributen verstanden werden. Dazu gehören:

- das (temporäre) Löschen von Datenpunkten (*Brushing*),
- das (temporäre) Löschen ganzer Bereiche,
- das Einfärben von Punkten.

13.2.3 Auskoppeln von Teildatensätzen

Speziell

- die äquidistante Abtastung einer Zeitreihe,
- die nicht äquidistante Auskopplung einzelner Werte,
- das Auskoppeln eines (zeitlichen) Teilbereichs,
- die Phasensetzung als Spezialfall einer Abtastung der Zeitachse.

13.2.4 Dynamische Graphik

Eine dynamische Graphik veranschaulicht den Ablauf eines Datenanalyse-Verfahrens mit Hilfe von Bewegung. Das Ergebnis ist auf einem Drucker nicht reproduzierbar, da hier die Information über die zeitliche Abfolge verloren geht. Eine dynamische Präsentation kann mehrere Zwecke erfüllen:

- Unmittelbare Veranschaulichung dynamischer Vorgänge (dynamische Systemsimulation).
- Hinzufügen einer weiteren Dimension durch die Zeit. Durch das Verfolgen des zeitlichen Ablaufs kann in zwei Dimensionen durchaus eine dritte mit zur Verfügung gestellt werden. Ein Beispiel ist das Phasenraumdiagramm, wo nur über die Zeichengeschwindigkeit der Kurven die Zeitinformation mitgeliefert wird [SW89].
- Illustration eines Datenanalyseverfahrens. Ein schönes Beispiel ist die Darstellung einer Kurvenanpassung, bei der nach jedem Iterationsschritt die Anpassungskurve neu eingezeichnet wird (vgl. auch 22.3). Dabei wird unter anderem ein Eindruck von der Güte der Anpassung vermittelt [Gol89] [SSW90].
- Veranschaulichung zufälliger Vorgänge (beispielsweise durch den dynamischen Aufbau einer Punktwolkendarstellung für eine Verteilung). Der Nutzen solcher Darstellungen sollte nicht unterschätzt werden, da das menschliche Vorstellungsvermögen für zufällige Vorgängen nicht sehr stark ausgeprägt ist.
- Verkürzung der Wartezeit: Ein interaktives System ist nur so lange interaktiv, wie ein ungehinderter Informationsfluß zwischen Mensch und Maschine besteht. Lange Wartezeiten beeinträchtigen die Konzentration und führen beim Benutzer erfahrungsgemäß zu Unruhe.

Dies stellt hohe Anforderungen an die Fähigkeiten des Graphik-Systems (vgl. 13.3).

13.2.5 3-dimensionale Darstellungen

Alle bis hierhin genannten graphischen Techniken lassen sich durchaus in einem einzigen universellen frei programmierbaren graphischen Baustein realisieren. In der Tat verfügt der in Kapitel 2.2 vorgestellte Dateneditor bereits über einige der leistungsfähigsten Hilfsmittel. 3D-Graphiken sind im Gegensatz dazu eher spezielle Darstellungsformen (vgl. 13.3.4). Die Techniken sind bekannt; es genügt eine kurze Aufzählung:

- Gitterdarstellung von Flächen im Raum (mit oder ohne verdeckten Linien),
- Flächendarstellung mit ausgefüllten Flächen,
- Niveaulinienplots,
- Punktwolkendarstellung von Dichten,
- 3-dimensionale Balkendiagramme,
- Stereoskopische Bilder, die über einen hochauflösenden Drucker schnell ausgegeben und betrachtet werden können ([CM88] enthält einige Beispiele).

13.2.6 Höhere Dimensionen

Versuche zur Visualisierung höherdimensionaler Information wurden bisher — mit wechselndem Erfolg — vor allem im Umfeld der Explorativen Datenanalyse unternommen. Sie beruhen meist auf verschiedenen Projektionen in die Ebene und sind z.B. in [CCKT83] zusammengestellt.

13.3 Anforderungen an die graphischen Gestaltungsmöglichkeiten

Aus dem Visualisierungdbedarf leiten sich unmittelbar die Anforderungen an ein Graphik-System ab. Die folgenden Ausführungen beziehen sich vor allem auf das Problemfeld der Bioprozeß-Datenanalyse:

13.3.1 Koordinatensysteme

Beim Layout von Koordinatensystemen muß größtmögliche Freiheit bestehen. Der Benutzer soll die Achsenaufteilung, Zahlenformate und Beschriftung vollständig in der Hand haben und auch über komplexere Anordnungen wie logarithmische Skalen, gekreuzte Koordinatenachsen oder Polarkoordinaten verfügen können. Marktgängige Systeme bieten meist nur einen Satz von festen Koordinatenlayouts an, unter denen der Benutzer das am besten Geeignete auszuwählen hat.

13.3.2 Darstellung von Punkten und Linien

Es muß eine Vielzahl von Möglichkeiten zur Darstellung und Markierung von Daten zur Verfügung stehen. Dazu gehören zunächst die verschiedenen Punktmarkierungen, Linientypen und -farben. Des weiteren müssen aber auch komplexere graphische Objekte wie Balkendarstellungen oder Fehlergrenzen frei konfigurierbar sein.

13.3.3 On-Line-Graphik

Benötigt wird eine On-Line-Graphik, d.h. ein System, daß in der Lage ist, mehrere Objekte gleichzeitig auf den Schirm zu zeichnen und zwar sukzessive mit dem Eintreffen der Daten. Die meisten Systeme sehen in der Tat nur das Zeichnen einer einzelnen Kurve bei Vorliegen des gesamten Datensatzes vor. Zum Erzielen dynamischer Effekte müssen graphische Objekte auf der anderen Seite auch wieder auslöschbar sein.

Diese Forderung zeigt, daß graphische Bausteine nicht nur bzgl. ihrer Konfiguration, sondern auch bzgl. ihres Datenflusses den Datenanalysebausteinen gleichgestellt sind. Beide Arten von Bausteinen sind in einem Netzwerk von Analysekomponenten formal nicht voneinander zu unterscheiden!

13.3.4 Nicht gestellte Anforderungen

Einige Forderungen, die bei Geschäftsgraphiken und einigen wissenschaftlichen Problemstellungen durchaus bestehen, treten bei der interaktiven Datenanalyse eher in der Hintergrund. Hierzu gehören z.B.:

Geschwindigkeitsanforderungen: Liniengraphik ist nicht zeitaufwendig. Der größte Teil der Rechenzeit wird für die Koordinatentransformation benötigt, was im Vergleich zum Rechenaufwand einer Simulation kaum ins Gewicht fällt.

3-D-Anforderungen: 3-dimensionale Graphiken sind sehr spezielle Darstellungsformen. Sie lassen eine weit geringere Zahl von Freiheitsgraden zur Gestaltung als gewöhnliche X-Y-Auftragungen. So steht man z.B. nicht vor dem Problem, mehrere Datensätze gleichzeitig zu visualisieren, da das Ergebnis in der Regel verwirrend ist. Ferner ist die Problematik der dynamischen Visualierung von Daten nicht gegeben, da eine 3D-Darstellung meist erst aufgrund eines ganzen Datensatzes berechnet werden kann (Problem der verdeckten Linien [Fel88]). 3-D-Graphiken können daher als separates Problem betrachtet werden.

Geschäftsgraphische Anforderungen: Beliebt sind die Tortendiagramme und diverse Balkendarstellungen mit 3-D-Effekten. Diese sind im naturwissenschaftlichen Rahmen höchstens zu Präsentationszwecken geeignet [1].

13.4 Implementierungsansätze

Die genannten Anforderungen haben in den letzten Jahren zu verstärkten Bemühungen bei der Entwicklung geeigneter Software geführt. Drei wichtige Problembereiche sollen angesprochen werden:

13.4.1 Graphisches Basissystem

Zur Realisierung graphischer Bausteine, die den genannten Anforderungen genügen, wird zunächst ein graphisches Basissystem benötigt. Dieses muß nicht nur in der Lage sein, die üblichen Punkte, Linien und Flächen sowie Text zu zeichnen — ein solches Elementargraphik-System ist z.B. das Graphische Kern-System GKS [BB86] —, sondern ferner auch graphischen Primitive enthalten, die zum Aufbau von Koordinatensystemen benötigt werden. Das Erreichen maximaler Flexibilität stellt ein schwieriges Problem dar. Hier hat vor allem die Technik der objektorientierten Programmierung große Fortschritte gebracht (vgl. 14.2). Einen Lösungsansatz stellt das Graphiksystem "OnLiGraph" dar, welches in Kapitel 17 beschrieben ist.

13.4.2 Interaktive Gestaltung

Inzwischen sind einige sehr elegante Methoden zur interaktiven Konfiguration von Graphiken entwickelt worden, die auch in einer Datenanalyse-Umgebung Anwendung finden können. So kann z.B. ein Koordinatensystem durch Anwahl der graphischen Objekte (Koordinatenachsen, Skalen etc.) mit einem Zeiger interaktiv entworfen und verändert werden [CM88]. Ein systematischer Ansatz zur Gestaltung von Koordinatensystemen ist in der Konzeption von "OnLiGraph" enthalten und wird in 17.6 beschrieben.

13.4.3 Konfiguration

Auch für das Konfigurationsproblem gibt es erprobte Lösungen: Die Informationen zum Aufbau einer graphischen Darstellung müssen in einer Graphik-Sprache formuliert werden. Diese muß allerdings über die bloße Protokollierung der elementaren Zeicheninstruktionen (Meta-Files) deutlich hinausgehen und einfache Programmierkonstrukte wie z.B. Unterprogramme zur Verfügung stellen. Solche Graphiksprachen sind z.B. die Sprachen GRAP [BK84] und PostScript [Ges85]. Liegt eine solche Graphik-Sprache vor, so können auch komplexere Elementaroperationen (z.B. das Zeichnen von Fehlerbalken oder problemspezifischen Hilfslinien) vom Benutzer frei definiert und wiederverwendbar gespeichert werden.

[1] Auch in dieser Arbeit wurde ein Tortendiagramm zur Visualisierung verwendet (Abb. 14.1).

Teil IV

Technische Details der Programmierung

Die im Rahmen dieser Arbeit erstellte Software ist äußerst umfangreich, wie die Statistik in Abbildung 14.1 zeigt. Allein das in Kapitel 21 beschriebene frei programmierbare Erweiterte Kalman-Filter umfaßt

20500	Zeilen Quellcode (ohne das benutzte Elementar-Graphikpaket und die zugeladenen Numerik-Routinen) in
600	aufgerufene Funktionen in
35	Modulen in
6	Programmbibliotheken. Dies entspricht
195	KB optimiertem Code (ohne Debugging-Information) !

Eine ausführliche Beschreibung der Programme würde — ebenso wie die Quelltexte selbst — einen weiteren Band füllen und ist im Rahmen dieser Arbeit uninteressant. Stattdessen sollen grundlegende Konzepte in Kürze zusammengestellt werden. Bei Detailfragen wird auf weiterführende Literatur verwiesen.

Kapitel 14

Programmierkonzepte

Die Anforderungen an die Programmierung leiten sich unmittelbar aus den grundlegenden Anforderungen an ein biotechnisches Datenanalysesystem ab, wie sie in Kapitel 1.3 formuliert wurden. Die Programmierung interaktiver Systeme dieser Größenordnung bedarf einer sorgfältig durchdachten Strukturierung, die sich vor allem auf die Modularisierung des Programmcodes auswirkt. Bei der Lösung der Detailprobleme hilft der Einsatz moderner Programmiertechniken, die insbesondere eine zukünftige Erweiterbarkeit gewährleisten. Nach einem kurzen Überblick über die Organisation der Module werden programmiertechnische Grundkonzepte aus den Bereichen

- Objektorientierte Programmierung,
- Deklarative Programmierung und
- Programmierung paralleler kommunizierender Prozesse

abgehandelt. Für die Konzepte des Compilerbaus ist das Kapitel 18 reserviert.

14.1 Modularisierung

Erreicht ein Projekt eine gewisse Größenordnung, so wird eine Aufteilung des Codes auf verschiedene, möglichst strikt voneinander getrennte Module unumgänglich. Die Vorteile einer weitreichenden Modularisierung sind hinreichend bekannt:

- Jeder Modul kann getrennt von den anderen entwickelt und gewartet werden.
- Projekte lassen sich auf mehrere Programmierer aufteilen.
- Sind die einzelnen Module gut auf Fehler geprüft, so können die Fehlerquellen des Gesamtsystems enger eingegrenzt werden.
- Die Portierung des Codes auf andere Rechnertypen und Betriebssysteme wird erleichtert. Dabei müssen meist nur die Ein/Ausgabe-Module (Graphik, Benutzeroberfläche) geändert werden.
- Bei Anbindung des Datenanalysesystems an ein spezielles Prozeßüberwachungssystem stellt letzteres bereits einige Komponenten zur Verfügung, so daß nicht alle Module gebraucht werden.

14.1.1 Überblick über die Module

Im Rahmen dieser Arbeit wurden Programmbibliotheken erstellt, die jeweils mehrere Module enthalten und in den nachfolgenden Kapiteln beschrieben werden:

- Datentypen (siehe 15):
  - Strings (siehe 15.1),

  - Vektoren und Matrizen (siehe 15.2),
  - Sequentielle Speicher (Puffer, Kellerspeicher, Warteschlangen, siehe 15.3),
  - Fehlerbehandlung (siehe 15.4).

- Benutzer-Führung (siehe Kapitel 16):
  - Bildschirm-Ausgabe,
  - Datenformatierung,
  - Feld-Editor,
  - Hilfsfunktionen,
  - Maskenaufbau.

- On-Line-Graphik (siehe Kapitel 17):
  - Setzungen und Abfragen,
  - Hilfsfunktionen,
  - Graphik-Treiber,
  - Zeichenoperationen,
  - Koordinatensysteme,
  - Metafile-System,
  - Metafile-Konvertierung.

- Compiler (siehe Kapitel 18):
  - Scanner,
  - Bezeichnerverwaltung,
  - Operatorverwaltung,
  - Parser, Differenzierer, Vereinfacher,
  - Formelauswertung
  - Maschinencode-Übersetzung.

- Numerik (siehe Kapitel 19):
  - Verfahren der linearen Algebra,
  - Zufallsverteilungen,
  - Gewöhnliche Differentialgleichungen,
  - Zufallszahlengeneratoren und Verteilungen,
  - statistische Verfahren,
  - Stochastische Differentialgleichungen.

- Datenflußmanagement und -Verwaltung (Bisher nur im Konzept, siehe Kapitel 20):
  - Hilfsfunktionen,
  - Kanäle,
  - Kanalverwaltung,
  - Kommunikationssteuerung,
  - Datenflußmanagement.

Für jeden Modul wurde ein Testprogramm erstellt, mit dem alle Funktionen überprüfbar sind. Damit wird eine Portierung sehr erleichtert!

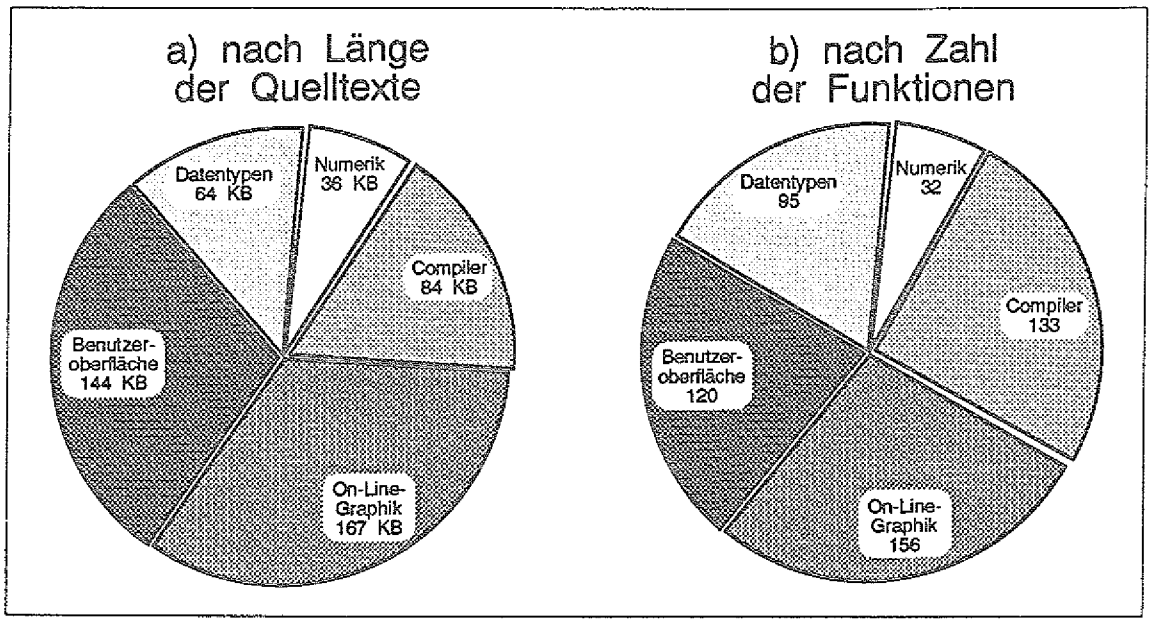


Abbildung 14.1: Statistik der im Rahmen dieser Arbeit erstellten Programm-Module

14.1.2 Code-Statistik

Eine Statistik nach Programmzeilen und Funktionen über die fünf großen Bibliotheken (Datentypen, Benutzerführung, On-Line-Graphik, Compiler, Numerik) zeigt die Abbildung 14.1 [1]. Die Abbildung bringt den enormen Anteil der Programmteile, die für die Präsentation und Interaktion mit dem Benutzer verantwortlich sind — das sind alle außer den Datentypen und den numerischen Routinen — deutlich zum Ausdruck. Es kann grob gesagt werden, daß bei der interaktive Programmierung etwa 80%-90% der Zeit für den Benutzerdialog aufgewendet werden müssen!

14.1.3 Einfluß verschiedener Programmiertechniken auf die Module

Zu den wegweisenden EDV-Entwicklungen der 80er Jahre gehören die verschiedenen Programmiertechniken und Werkzeuge zur Gestaltung von Mensch-Maschine-Schnittstellen [FPS85]. Sie haben die Einsatzmöglichkeiten des Computers um ein Vielfaches erweitert: Der Computer hat sich von einer Rechenmaschine mit schnellem Datenzugriff zu einem intuitiv steuerbaren Hilfsmittel für wechselnde komplexe Aufgabenstellungen entwickelt. Zu dieser Entwicklung haben (neben den technischen Fortschritten) vor allem moderne Programmiermethoden, wie die Objekt-Orientierte und die Deklarative Programmierung beigetragen. Bezeichnenderweise ist im Jahre 1980 das erste SmallTalk-System auf den Markt gebracht worden [GR89].

Ohne die Objektorientierte Programmierung ist heute an eine effiziente Realisierung von Benutzeroberflächen nicht mehr zu denken. Die Abbildung 14.2 zeigt, welche Programmierkonzepte vornehmlich in die einzelnen Module Einzug gefunden haben:

- Es wird deutlich, daß die Objektorientierte Programmierung in erster Linie die Programmierung der Datentypen, Benutzerschnittstelle und On-Line-Graphik beeinflußt hat.
- Dagegen waren viele wesentliche Entwicklungen im Compilerbau und der Programmierung paralleler Prozesse (Bertriebssysteme) bereits in den 70er Jahren abgeschlossen.

[1] Die Diskussion über geeignete Maße für die Programmkomplexität soll hier nicht aufgenommen werden (vgl. dazu [Pre87]). Weitere Statistiken wie Zeitaufwand, Größe des übersetzten Codes, Vernetzungsgrad, Abstraktionsgrad verändern zwar die prozentualen Anteile der einzelnen Module, jedoch bleibt die Größenrelation im wesentlichen erhalten.

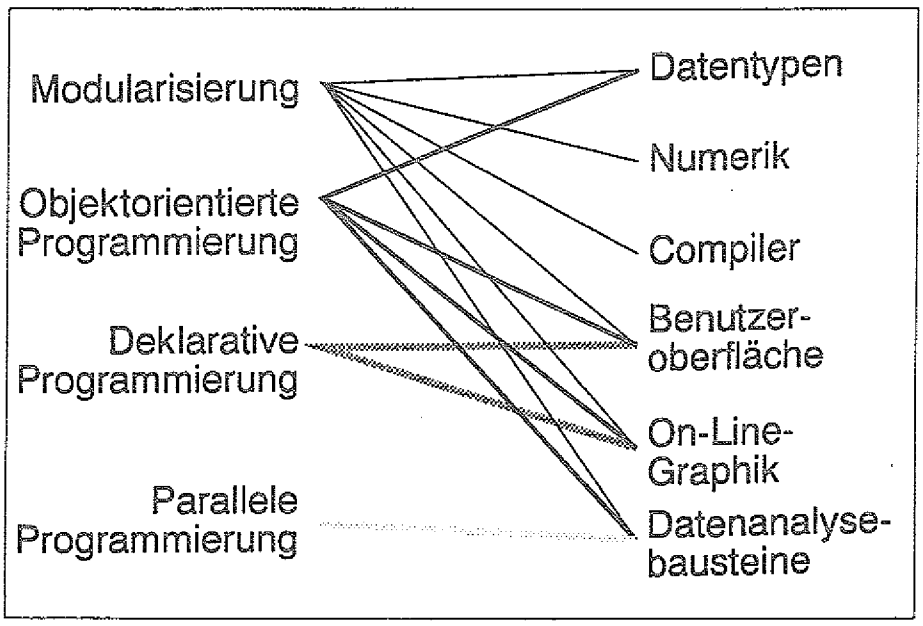


Abbildung 14.2: Einfluß der verschiedenen Programmiertechniken auf die erstellten Programm-Module

- Die Art, numerische Verfahren zu programmieren, hat sich schließlich seit den 60er Jahren nicht wesentlich verändert [2].

14.2 Objektorientierte Programmierung

Über Objektorientierte Programmierung ist in jüngster Zeit so viel berichtet worden ([Sch88] [Sch90b]), daß hier eine kurze Skizze der Grundgedanken genügt. Das Konzept soll anhand eines zentralen Beispiels aus dem Kontext der interaktiven Datenanalyse diskutiert werden: Es handelt sich um die Realisierung einer frei konfigurierbaren Verschaltung von Datenanalysebausteinen zu Netzwerken, wie sie in der Abbildung 14.3 schematisch dargestellt ist, und in Kapitel 2 an Beispielen illustriert wurde [3].

14.2.1 Grundbegriffe

14.2.1.1 Klassen und Methoden

Von außen gesehen haben alle Datenanalysebausteine ähnliche Merkmale, die im wesentlichen durch

- den Konfigurationsdatenstrom,
- den Eingangsdatenstrom und
- den Ausgangsdatenstrom

beschrieben werden können (vgl. hierzu das Schema 2.5 des Erweiterten Kalman-Filters). Auch die Operationen, die jeder Baustein auf den Daten durchführt, lassen sich nach einem allgemeinen Schema klassifizieren [4]:

- Bei der *Initialisierung* fordert jeder Baustein den benötigten Speicherplatz an und belegt alle Daten-Felder mit Default-Werten.

[2] Der Name FORTRAN der bei der Numerik-Programmierung maßgeblichen Programmiersprache ist eine Abkürzung für Formula Translator. Sie ist für die Realisierung modernerer Konzepte aufgrund der veralteten Konzeption wenig geeignet.

[3] Es sei hier an die Tatsache erinnert, daß der "Urvater" der Objektorientierten Programmierung, die Programmiersprache SIMULA — wie der Name schon sagt — zunächst zur Simulation paralleler, vernetzter Prozesse konzipiert war.

[4] Es könnten noch weitere Operationen genannt werden. Für das Beispiel sollen diese genügen.

- Bei der *Konfiguration* liest der Baustein Daten, Modelle, Parameter etc. aus einer Eingabedatei ein und belegt seine internen Datenfelder.
- Bei der *Konsistenzprüfung* werden alle Konfigurationsdaten auf Konsistenz, Vollständigkeit und Fehlerfreiheit geprüft.
- Bei der *Implementierung* wird der Baustein in den Datenfluß eingebunden.
- Bei der *Startwertbelegung* werden die Startwerte und -bedingungen der Datenanalysemethode aus dem Kontext ermittelt.
- Beim *Dateninput* werden die Eingabedaten für den nächsten Rechenschritt gelesen.
- Beim *Rechenschritt* wird ein Iterationsschritt für das nächste Zeitintervall durchgeführt.
- Beim *Datenoutput* werden die berechneten Daten nach außen weitergegeben.
- Bei der *Destruktion* wird der Baustein gelöscht und der Speicherplatz freigegeben.

Alle Datenanalysebausteine bilden nun eine *Klasse*, bestehend aus Daten und Operationen (= Methoden) auf diesen Daten. Sie kann abstrakt in der Form

```
class Datenanalysebaustein
    begin
        data:
            Konfigurations-Datenstrom ;
            Eingangs-Datenstrom ;
            Ausgangs-Datenstrom ;
        method:
            Initialisierung ;
            Konfiguration ;
            Konsistenzprüfung ;
            :
            Destruktion ;
    end ;
```

formuliert werden kann [5].

14.2.1.2 Vererbung

Die Klasse Datenanalysebaustein ist bis hierhin nur ein abstraktes Schema: Wie die Datenströme genau aussehen und wie die Operationen im Detail ausgeführt werden sollen, ist in keiner Weise konkretisiert. Man spricht hier von *virtuellen Methoden*. Ein konkreter Datenanalysebaustein ist nun eine Struktur, die alle Methoden der abstrakten Klasse Datenanalysebaustein übernimmt, dabei jedoch mit Inhalt füllt. Dieser Mechanismus wird als *Vererbung* bezeichnet. Die Datenstrukturen für die internen Daten des Bausteins müssen dazu ebenfalls spezifiziert werden. Sie sind — im Gegensatz zu den Methoden — nicht verallgemeinerbar.

Wie die internen Daten und die Konkretisierung der Methoden eines Datenanalysebausteins beschaffen sein können, zeigen zwei Beispiele:

a) Ein Kalman-Filter ist eine Tochterklasse von Datenanalysebaustein, die zusätzlich ein Modell mit Zustandsdaten enthält:

[5] Im folgenden wird zur Erläuterung der programmiersprachlichen Elemente ein Meta-Code verwendet, der sich etwas an die Pascal-Notation anlehnt und im wesentlichen selbsterläuternd sein soll.

```
class Kalman-Filter
    begin
        inherit :
            Datenanalysebaustein ;
            { Virtuelle Methoden und Datenströme werden hiermit übernommen }
        data :
            Modell ;
            Zustand ;
            Kovarianzmatrizen ;
    end ;
```

Die vorher nur abstrakt spezifizierten virtuellen Methoden (Initialisierung, Konfiguration usw.) können nun mit Inhalt gefüllt werden. Dazu muß im Einzelnen festgelegt werden, wie die konkreten Methoden des Kalman-Filters auf den Daten operieren, was am Beispiel zweier Methoden etwa wie folgt geschehen kann:

```
method Kalman-Filter.Konfiguration
begin
    Konfigurations-Datenstrom Öffnen ;
    Modell Einlesen ;
    Zustand Einlesen ;
    Kovarianzmatrizen Einlesen ;
    Konfigurations-Datenstrom Schliessen ;
end ;

method Kalman-Filter.Zeitschritt
begin
    Matrix-Riccati-Differentialgleichung Einzelschritt ;
    Kalman-Verstärkungsmatrix Berechnen ;
    Updating-Schritt Durchführen ;
end ;
;
```

b) Ein Graphik-Baustein muß dagegen Zeichenattribute und Graphik-Layout speichern:

```
class Graphik-Baustein
    begin
        inherit :
            Datenanalysebaustein ;
        data :
            Zeichenattribute ;
            Graphik-Layout ;
    end ;
```

Entsprechend erfahren die Methoden eine ganz andere Realisierung:

```
method Graphik-Baustein.Konfiguration
    begin
        Konfigurations-Datenstrom Öffnen ;
        Zeichenattribute Einlesen ;
        Graphik-Layout Einlesen ;
```

```
        Konfigurations-Datenstrom Schliessen ;
    end ;

method Graphik-Baustein.Implementierung
    begin
        Koordinatensystem Zeichnen ;
    end ;

method Graphik-Baustein.Zeitschritt
    begin
        Linien Verbinden ;
        Marker Zeichnen ;
    end ;
:
```

14.2.1.3 Objekte

An diesen Beispielen dürften die Begriffe *Klasse, Methode* und *Vererbung* sowie die Beziehung von *virtuellen* und *konkreten* Methoden klar geworden sein. Ein konkreter Repräsentant aus einer der Klassen heißt ein *Objekt*. Objekte werden z.B. in der Form

Datenanalysebaustein	DAB_1, DAB_2 ;
Kalman-Filter	KF_1 , KF_2 ;
Graphik-Baustein	GB_1 , GB_2 ;

definiert. Man beachte, daß die Objekte KF_1 und KF_2 sowohl Repräsentanten der Klasse Kalman-Filter als auch der Klasse Datenanalysebaustein sind, da erstere eine Tochterklasse der zweiten ist. Die Objekte DAB_1 und DAB_2 sind nur formal definiert: Der Versuch, die Methode DAB1.Zeitschritt zu benutzen, würde zu einer Fehlermeldung der Art "Methode nicht implementiert !" führen.

14.2.2 Das Prinzip der späten Bindung

Bis hierhin können (bis auf die abstrakte Klasse Datenanalysebaustein) alle Bausteine auch konservativ mit Hilfe von Datenstrukturen und Prozeduren programmiert werden. Um den wesentlichen Aspekt der objektorientierten Programmierung zu demonstrieren, sei nun angenommen, daß ein Netzwerk von Datenanalysebausteinen bereits aufgebaut und vollständig konfiguriert ist. Die beteiligten Bausteine seien über eine Liste von Zeigern verfügbar (vgl. 14.3).

Bemerkung: Es wird (in Anlehnung an PASCAL oder C [KR83]) die folgende Zeigernotation verwendet:
Wenn

Zeiger

ein Zeiger auf eine Datenstruktur ist, so bezeichnet die *Dereferenzierung*

&Zeiger

die Struktur selbst. Die Struktur kann *umtypisiert* werden in der Form:

(Neuer_Typ) &Zeiger

und dann wie eine Struktur des Typs Neuer_Typ verwendet werden.

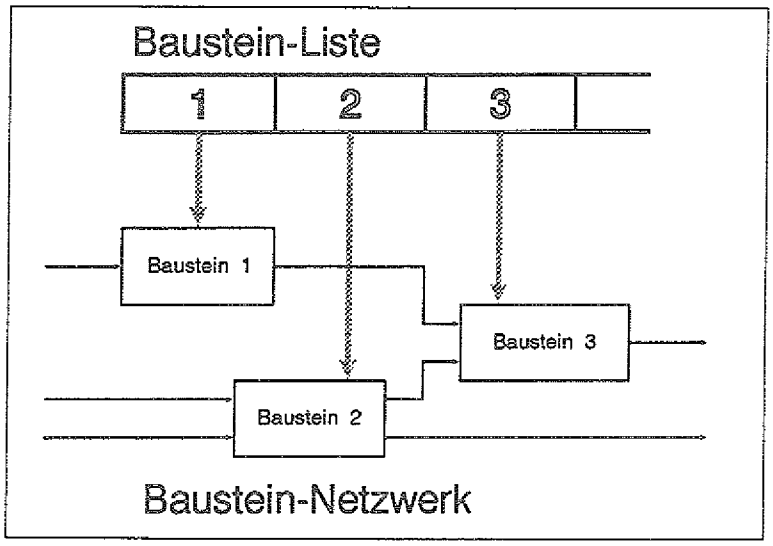


Abbildung 14.3: Zeigerliste mit den Bausteinen eines Netzwerks

14.2.2.1 Prozeduraler Programmierstil

Ein Programmabschnitt, in dem die Steuerung einer zeitlichen Iteration stattfindet, sieht im traditionellen prozeduralen Programmierstil beispielsweise wie folgt aus:

```
:
{ Startwerte }
forall Zeiger in Baustein-Liste do
    case &Zeiger is of type
        Kalman-Filter     : (Kalman-Filter )(&Zeiger).Startwerte ;
        Graphik-Baustein : (Graphik-Baustein)(&Zeiger).Startwerte ;
        :
    end ;

{ Iteration }
t := 0.0 ;
while (t < t_end)
begin
    forall Zeiger in Baustein-Liste do
        case &Zeiger is of type
            Kalman-Filter     : (Kalman-Filter )(&Zeiger).Zeitschritt ;
            Graphik-Baustein : (Graphik-Baustein)(&Zeiger).Zeitschritt ;
            :
        end ;
        t := t + dt ;
    end ;
end ;
:
```

Dieser Programmierstil hat mehrere Nachteile:

- Die Typ-Information für jeden Baustein muß separat abgespeichert werden. (Dies kann mit Hilfe einer zweiten Liste oder aber an der ersten Speicherstelle eines jeden Bausteins geschehen.)

- Bei einer Erweiterung des Baustein-Sortiments müssen *alle* Fallanweisungen im Programm um die neuen Einträge verlängert werden!
- Das extensive Abarbeiten von Fallanweisungen ist ein enormer Zeitfaktor und (zumal bei der zeitlichen Iteration) kaum vertretbar.

Kurzum: Ein solches Programm ist schwer lesbar, kaum zu warten und langsam!

14.2.2.2 Objektorientierter Programmierstil

Bei Verwendung einer objektorientierten Programmiersprache entfallen diese Probleme. Dann sieht das obige Programm einfach wie folgt aus:

```
:
forall Zeiger in Baustein-Liste
    &Zeiger.Startwerte ;

t := 0.0 ;
while (t < t_end)
begin
    forall Zeiger in Baustein-Liste
        &Zeiger.Zeitschritt ;
    t := t + dt ;
end ;
:
```

Das objektorientierte System verwaltet die Baustein-Typen selbst und entscheidet zur Laufzeit, welche Funktionen in Abhängigkeit vom Baustein-Typ aufzurufen sind. Dieses geänderte Aufrufschema wird als *späte Bindung* (im Gegensatz zur *frühen Bindung* beim Übersetzen) bezeichnet und ist ein Wesensmerkmal der objektorientierten Programmiersprachen [Cox86].

Da alle Objekte derselben Oberklasse Datenanalysebaustein angehören, ist garantiert, daß die gesuchten Methoden für jeden Baustein vorhanden sind und dasselbe Aufrufschema haben. Des weiteren ist klar, daß jederzeit neue Bausteine zum System hinzugefügt werden können, ohne daß an dem obigen Fragment irgend etwas geändert werden müßte! Es können also Teile an der Basis eines Programmsystems hinzugefügt, verändert oder ausgetauscht werden, ohne Veränderungen im übrigen Programmtext nach sich zu ziehen.

Dieses Konzept ist bei der Computer-Hardware schon lange verwirklicht: Integrierte Schaltkreise und Einschübe können von verschiedenen Firmen zugekauft werden. Einzige Bedingung ist, daß die Außenanschlüsse einer gewissen Norm genügen und der Elektronik-Baustein ein festgelegtes Ein/Ausgabeprotokoll hat. In diesem Kontext bezeichnet B.J. Cox in seinem Standardwerk "Object Oriented Programming" [Cox86] die Klassen auch als *Software-IC's*.

14.2.3 Einsatzgebiete und Sprachen für die Objektorientierte Programmierung

Die Objektorientierte Programmierung ist heute zur Voraussetzung für die Realisierung großer interaktiver Systeme geworden. Hierzu gehören in erster Linie :

- Interaktive graphische Systeme,
- Benutzeroberflächen,
- Textsatzsysteme.

14.2.3.1 Programmiersprachen

Bei den objektorientierten Programmiersprachen unterscheidet man zwischen

- klassischen prozeduralen Programmiersprachen, die um einige objektorientierte Konzepte erweitert wurden, ohne den klassischen Rahmen zu sprengen. Prominente Vertreter sind
  - C++ (das ein Kandidat für den zukünftige C-Standard ist) [Str87],
  - einige Pascal-Dialekte (z.B. Turbo Pascal 5.5, MacIntosh-Pascal),
  - Ada.
- radikal objektorientierten Sprachen, die das objektorientierte Konzept zum strukturbildenden Element machen: Ein Problem wird allein durch die Kommunikation zwischen Objekten gelöst. Der herausragende Vertreter dieses Typs ist
  - SmallTalk, bei dem allerdings deutlich das ältere
  - LISP Pate gestanden hat [BSS84].

  Solche Sprachen sind ebenso einfach wie flexibel, benutzerfreundlich und fehlertolerant. Auf die Nachteile eines enormen Speicherplatzbedarfs und langer Rechenzeiten wird noch eingegangen werden.
- Hybridsprachen, bestehend aus einer Standard-Programmiersprache und Smalltalk-Elementen. Sie sollen die Vorteile beider Konzepte vereinen. Hierzu gehört z.B.
  - Objective C [Cox86].

  Im Gegensatz zu den "reinrassigen" Lösungen scheinen sich die Hybridsprachen jedoch nicht durchzusetzen, da sie das Erlernen zu vieler heterogener Strukturelemente und Konzepte erfordern.

14.2.3.2 Umsetzung objektorientierter Konzepte in einer klassischen prozeduralen Programmiersprache

Es muß an dieser Stelle betont werden, daß die Objektorientierte Programmierung ein Programmierkonzept ist, nicht aber die Programmierung in einer speziellen objektorientierten Sprache! Vielmehr können objektorientierte Konstrukte auch in prozeduralen Standard-Programmiersprachen verwirklicht werden, wenn diese auch nicht den vollen Komfort einer eigens dazu ausgelegten Programmier-Umgebung bieten. So sind z.B. die meisten derzeit auf dem Markt erhältlichen C++-Compiler — unter anderem auch der originale AT&T-Compiler — lediglich *Präprozessoren*, die ein C++-Programm in ein normales C-Programm übersetzen.
Es bestehen prinzipiell zwei Möglichkeiten für die Umsetzung objektorientierter Konstrukte in prozeduralen Programmiersprachen, die anhand des Beispiels verdeutlicht werden können:

a) Jeder Datenanalysebaustein erhält als ersten Dateneintrag einen Objektcode, der angibt, welcher Typ von Baustein vorliegt. Der Code dient einem Baustein-Verteiler zur Erkennung des jeweiligen Baustein-Typs. Dieser Weg wurde im wesentlichen bei dem Programmier-Beispiel für die traditionelle Programmierung beschritten (vgl. 14.2.2.1).

   Der Verteiler muß dabei für jede Methode nur einmal programmiert werden, wenn er in eine eigene Prozedur verlegt wird. Ferner wird auf diese Weise das Hauptprogramm von den unübersichtlichen Fallunterscheidungen befreit. Durch Zentralisierung der Methoden zusammen mit ihrem jeweiligen Baustein-Verteiler in einem Modul, hält sich der Aufwand für die Einführung neuer Methoden oder Bausteine in Grenzen. Allerdings bleibt der Nachteil des langsamen Abarbeitens der Fall-Anweisungen bestehen, so daß diese Technik bei zeitkritischen Problemen nicht angewendet werden sollte.

b) Wesentlich schneller, jedoch mit Sorgfalt zu programmieren, ist eine andere Technik. Sie stellt in der Tat auch die interne Lösung vieler objektorientierter Systeme (u.a. der C++-Präprozessoren) dar: Jeder Baustein erhält eine Liste von Prozedurzeigern, die den Methoden entsprechen. In dieser Situation sind nicht die Bausteine, sondern die Methoden mit einem Code zu versehen. Aufgrund der Codenummer und der Methodenliste kann dann jeder Baustein die für ihn spezifischen Methoden auswählen.

Die jeweils gültigen Prozedurzeiger müssen mit Hilfe einer Initialisierungsoperation vorbelegt werden. Dabei genügt es, die Liste der Prozedurzeiger für jede Klasse einmal zentral abzuspeichern. Der einzelne Baustein wird dann mit einem Zeiger auf die Prozedurliste seiner Klasse initialisiert.

14.2.3.3 Weitere Anmerkungen

i) In der Regel haben die Methoden noch Parameter, die aus Gründen der Übersichtlichkeit weggelassen wurden.

ii) Offenbar benötigt die Zeigervariante etwas mehr Speicherplatz. Dies ist ganz allgemein ein Kennzeichen der Objektorientierten Programmierung, denn die Zusatzinformationen, die sonst vom Benutzer zum Zeitpunkt der Übersetzung bereitgestellt werden, müssen jetzt auch zur Laufzeit verfügbar sein. Bei den radikal objektorientierten Systemen ist dies besonders stark ausgeprägt.

iv) Die Verwendung eines C++-Präprozessors ist problematisch, da hierbei das Debugging auf der C-Code-Ebene erfolgen muß. Der erzeugte C-Code ist jedoch im allgemeinen kryptisch! Moderne Hilfsmittel wie Hierarchie- und Struktur-Browser [Gol83] sind ebenso schwierig zu implementieren (vgl. hierzu auch [RNG90]). Erst zum Zeitpunkt der Fertigstellung dieser Arbeit ist mit Turbo C++ die ersten brauchbare C++-Programmierumgebung für den PC auf den Markt gekommen [TCP90]!

14.3 Weitere Konzepte der Objektorientierten Programmierung

Im Wesentlichen unterscheidet sich eine objektorientierte Programmiersprache von einer prozeduralen Sprache lediglich durch die Möglichkeit der späten Bindung. Als Konsequenz dieser Verschiebung resultiert ein großes Spektrum von neuen Möglichkeiten, das sich erst bei extensiver Anwendung des Prinzips erschließt. Vom Programmierer verlangt dies einen Umdenkprozeß!

Die Vorteile der Objektorientierten Programmierung machen sich erst bei Programmen einer gewissen Größe bemerkbar, da sie vor allem den Vernetzungsgrad der Module reduziert. Die Kosten sind eine längere Vorlaufzeit zur Programmkonzeption — was sich noch nie nachteilig ausgewirkt hat — und ein oft stark vergrößerter Schreibaufwand bei der Codierung. Auf der anderen Seite erhält der Code auf höheren Programm-Ebenen eine sehr kurze und prägnante Struktur (vgl. 14.2.2.2).
Einige weitere Möglichkeiten der Objektorientierten Programmierung sollen kurz vorgestellt werden.

14.3.1 Konstruktion, Kapselung und Destruktion

Die meisten prozeduralen Programmiersprachen erlauben die Programmierung von Datenstrukturen variabler Länge nur unter Verwendung elementarer Konstrukte auf der Basis von Zeigern (Felder variabler Länge, verkettete Listen etc.). Dabei stößt man immer wieder auf kaum zu lokalisierende Zeigerfehler:

- uninitialisierte Zeiger, vergessene oder falsche Initialisierung des referenzierten Speicherplatzes, was zum Überschreiben anderer Daten oder Programmteile führt,
- unerlaubte Benutzung des Speicherplatzes über die zulässigen Grenzen hinaus, erneut mit der Folge des Überschreibens von Daten,
- nicht zurückgegebener Speicherplatz, der auf Dauer zu einem Speicherüberlauf führt.

Diese Fehlerquellen lassen sich in objektorientierten Programmiersprachen weitgehend ausschalten und zwar durch die Verwendung zweier spezieller Methoden, die *Konstruktion* und *Destruktion* genannt werden, sowie durch die *Kapselung* der internen Daten.

14.3.1.1 Konstruktion

Der Konstruktor erhält keinen eigenen Namen, sondern trägt den Namen seiner Klasse. Mit der Deklaration von Objekten einer solchen Klasse ist dann zugleich der Aufruf des Konstruktors verbunden, so daß mit der Deklaration *automatisch* auch die Initialisierung verbunden ist.
Beispiel: Vektoren variabler Länge können beschrieben werden durch die Klasse:

```
class Vektor
begin
    data :
        Dimension ;
        Element-Zeiger ;
    method :
        Vektor ( Dimension ) ; { der Konstruktor }
        ⋮
    end ;
```

Dabei zeigt Element-Zeiger auf einen eigens zu reservierenden Speicherbereich, in dem die Elemente des Vektors abgelegt sind. Der Konstruktor wird wie folgt programmiert:

```
method Vektor.Vektor ( Dimension )
begin
    Vektor.Dimension := Dimension ;
    Vektor.Element-Zeiger := Speicherplatzreservierung (Dimension) ;
    &(Vektor.Element-Zeiger) := ( 0.0 , 0.0 , ... , 0.0 ) ;
end;
```

Von nun ab ist mit der Deklaration

```
Vektor v(2) , w(4) ;
⋮
```

zugleich auch die Speicherplatzbelegung und Initialisierung mit Nullen verbunden. Bei prozeduralen Programmiersprachen muß dagegen die Initialisierung an anderer Stelle im Programm erfolgen als die Deklaration. Durch diese räumliche Trennung kommen häufig Unterlassungsfehler zustande!

14.3.1.2 Destruktion

Das Gegenstück zum Konstruktor ist der Destruktor, der mit dem Klassennamen und einem Zusatzsymbol (hier ~) versehen wird. Der Destruktor der Klasse Vektor gibt den reservierten Speicherplatz wieder frei:

```
method Vektor.~Vektor
    begin
        Speicherplatzfreigabe (Vektor.Element-Zeiger, Vektor.Dimension) ;
    end;
```

Der Destruktor taucht im Programmtext nirgends explizit auf, sondern wird *automatisch* aufgerufen, wenn der Gültigkeitsbereich eines Bezeichners verlassen wird.

Bemerkung 1: Bei objektorientierten Programmiersprachen gibt es eigentlich keine Deklaration von Variablen mehr. An ihre Stelle tritt die Konstruktion, welche eine Operation wie jede andere ist. Insbesondere ist die Verwendung temporäre Variablen, deren Gültigkeitsbereich sich nur über einige Befehle erstreckt, möglich.

Bemerkung 2: In SmallTalk ist die Destruktion radikaler gelöst: Hier bleibt jedes Objekt automatisch so lange im Speicher bis keine Referenz mehr auf dieses Objekt besteht. Die Sammlung des nicht mehr benötigten Speicherplatzes (*garbage collection*) [Knu73] ist sehr zeitaufwendig, schließt jedoch Zeigerfehler vollkommen aus.

14.3.1.3 Kapselung

Die internen Daten einer Klasse sind von außen nicht sichtbar und können nur über die Methoden verändert und abgerufen werden. Dieses Prinzip wird als *Kapselung* der internen Daten bezeichnet. Die Kapselung bewirkt einen kontrollierbaren Zugriff von außen.
Bei Vektoren können damit z.B. Indizierungsfehler ausgeschlossen werden:

```
method Vektor.LeseElement ( Index )
    begin
        if ( Index < 1 ) or ( Index>Vektor.Dimension) then
            Fehlerbehandlung ;
        else
            return &(Vektor.Element-Zeiger) [ Index ]
    end ;

method Vektor.SchreibeElement ( Index , Zahl )
    begin
        if ( Index< 1 ) or ( Index>Vektor.Dimension ) then
            Fehlerbehandlung ;
        else
            &(Vektor.Element-Zeiger) [ Index ] := Zahl ;
    end ;
```

Es wird dadurch verhindet, daß der Benutzer der Klasse direkt auf die Elemente des Datenfeldes zugreift und damit eventuell fremde Daten zerstört [6].

14.3.2 Überladen

14.3.2.1 Überladen von Funktionen

Die späte Bindung verhilft den objektorientierten Systemen zur Fähigkeit der kontextabhängigen Unterscheidung zwischen Methoden gleichen Namens, die aber verschiedenen Klassen eigen sind: Der Ausdruck Baustein.Zeitschritt bedeutet etwas anderes, je nachdem ob man einen Kalman-Filter oder einen Graphik-Baustein vor sich hat. Dieses Konzept läßt sich weiter verallgemeinern: Auch innerhalb derselben Klasse kann es verschiedene Methoden mit gleichem Namen geben, sofern diese durch Parameteranzahl und -typ voneinander unterschieden sind. Man spricht vom *Überladen* von Funktionen.
Beispiel: In der Klasse Vektor können verschiedene Initialisierungsoperationen bereitgestellt werden:

```
class Vektor
    begin
        data :
            Dimension ;
```

[6] Die C++-Sprachdefinition [Str87] ermöglicht für zeitkritische Operationen die *Inline-Deklaration* von Methoden. Der Compiler generiert hierfür keine Prozeduraufrufe, sondern setzt entsprechende Befehle direkt in den Programmtext ein. Der erzeugte Code hat somit keine Geschwindigkeitseinbußen gegenüber der herkömmlichen Programmierung.

```
        Element-Zeiger ;
    method :
        Vektor ( ) ;                    { Ein Konstruktor }
        Vektor ( Dimension ) ;          { Noch ein Konstruktor }
        Vektor ( Dimension , Zahl ) ;   { Und noch ein Konstruktor }
        :
end ;
```

Dabei erzeugt:

Vektor ( ) einen 0-dimensionalen Vektor, der später noch umdimensioniert werden kann.

Vektor (Dimension) einen Vektor der angegebenen Dimension, der mit 0.0 belegt wird.

Vektor (Dimension , Zahl) einen Vektor der angegebenen Dimension, der mit Zahl belegt wird.

Im Programmtext kann deklarieren werden:

```
Vektor u ( ) , v ( 5 ) , w ( 10 , 1.0 ) ;
```

Das objektorientierte System erkennt an den Parametern, welche Methode jeweils gemeint ist.

14.3.2.2 Überladen von Operationen

Das Überladen von Funktionen ist zunächst nur eine angenehme Erleichterung, welche die Suche nach immer neuen Prozedurnamen für ähnliche Operationen beendet. Bedeutsamer wird das Überladen erst bei den Operatoren, beispielsweise bei +, -, *, / und := . Für ein objektorientiertes System sind Operatoren lediglich eine andere Notation für Funktionen mit zwei Argumenten:

$$x + y \quad \Leftrightarrow \quad +(x, y);$$
$$x := y \quad \Leftrightarrow \quad :=(x, y);$$

Die Operatoren werden von prozeduralen Programmiersprachen seit jeher überladen, d.h. abhängig von den Parametern verschieden interpretiert. So gibt es z.B. stets die vier verschiedenen Funktionen

```
+  ( ganzzahlig  , ganzzahlig  ) ;
+  ( ganzzahlig  , dezimal     ) ;
+  ( dezimal     , ganzzahlig  ) ;
+  ( dezimal     , dezimal     ) ;
```

mit einem jeweils verschiedenen Ergebnistyp (Pascal kennt "+" sogar für Zeichenketten). Auch die Zuweisungen

```
x  :=   1;
x  :=   1.0;
x  :=  "1.0";
```

bedeuten jeweils etwas anderes. Die objektorientierten Programmiersprachen gestatten nun auch die Neudefinition und Überladung von Operatoren. Damit wird ein erheblicher Gewinn an Übersichtlichkeit und Prägnanz erreicht.

Ein Beispiel: Definiert man die Klasse Vektor so wie oben und die Klasse Matrix auf analoge Weise, so können alle Operationen der Vektor- und Matrizenalgebra mit den üblichen Schreibweisen eingeführt werden:

$$v + w,\ M + N,\ c \cdot v,\ c \cdot M,\ M \cdot v,\ M * N,\ v * w\ .$$

Die rechte Seite RS der Matrix-Riccati-Differentialgleichung

$$\dot{P} = AP + PA^T + R$$

muß z.B. in einer prozeduralen Programmiersprache wie folgt umgesetzt werden (Die Dimensionierung der Matrizen ist $N \times N$):

```
mxm   ( RS , A , P ,     N , N , N ) ;   { RS   = A * P }
mxmt  ( Tmp , P , A ,    N , N , N ) ;   { Tmp  = P * trp(A), temporäre Matrix }
maddm ( RS , RS , Tmp ,  N , N ) ;       { RS   = RS + Tmp }
maddm ( RS , RS , R ,    N , N ) ;       { RS   = RS + R }
```

Nach Überladen der Operatoren schreibt man ganz einfach:

```
RS := P * trp ( A ) + A * P + R ;
```

Die Übereinstimmung der Vektor- und Matrizen-Dimensionen kann dabei implizit geprüft werden!

14.3.3 Nachrichten

Der Aufruf einer Methode (z.B. KF.Zeitschritt für ein Kalman-Filter) kann auch anders interpretiert werden: Das Objekt KF erhält die Nachricht "Zeitschritt durchführen", worauf das Objekt selbst entscheiden muß, wie es diesen Auftrag ausführt.

14.3.3.1 Nachrichten in SmallTalk

Dieses Prinzip des Übertragens von Nachrichten ist so mächtig, daß es zum alleinigen Strukturprinzip von Smalltalk werden konnte: In einem Smalltalk-System wird ein Problem — nomen est omen — dadurch gelöst, daß die verschiedenen Objekte dieses untereinander "ausdiskutieren". Erhält ein Objekt eine Nachricht, so versucht es, den Auftrag zu erfüllen; gelingt ihm das aufgrund der ihm zur Verfügung stehenden Informationen nicht, so besorgt es sich diese durch entsprechende Anfragen bei anderen Objekten, die zu diesem Zweck oft eigens erst geschaffen werden. Schon bei kleinen Anfragen findet man wenig später im Speicher des Rechners eine ganze "Population" von untereinander kommunizierenden Objekten.

Ein Smalltalk-System erreicht auf diese Weise eine erstaunliche Fähigkeit zur Problemlösung auf der Basis von relativ wenig Programmcode. Dazu kommt eine bei konventionellen Programmiersprachen ungekannte Fehlertoleranz: Falls ein Objekt zu einer Nachricht keine Methode bereithält, so wird dies durch das System toleriert. Die Art der Fehlerbehandlung kann dabei vom Benutzer frei programmiert werden [GR89]. Der traditionelle Begriff eines Programmfehlers ist in einer SmallTalk-Umgebung unbekannt!

Auf der anderen Seite benötigt ein Smalltalk-System für seine Objekte und deren Verwaltung einiges an Rechenzeit, so daß es für rechenintensive Probleme, die mit numerischen Standard-Algorithmen gelöst werden können, ungeeignet ist [7]. Die gemäßigt objektorientierten Sprachen verwirklichen das Nachrichten-Konzept daher nur eingeschränkt.

Das Nachrichten-Konzept wird heute bei allen interaktiven Systemen angewandt. So wird z.B. eine Bildschirmmaske aufgebaut, indem allen darin befindlichen Objekten den Auftrag erteilt wird, auf dem Schirm zu erscheinen. Die nachrichtenbasierte Steuerung der Kommunikation mit einem Benutzer ist in der Lage, falsche Eingaben zu tolerieren und kontextsensitive Hilfestellungen zu erteilen. SmallTalk erlaubt dabei auch das Versenden ganzer Programme als Nachricht an Menüobjekte, womit die Ablaufsteuerung äußerst effektiv gestaltet werden kann.

14.3.3.2 Programmdokumentation und versionsunabhängige Konfiguration

Es ist guter Stil, den Programmtext immer gleich mit kurzen Kommentaren zu versehen. Damit ist aber noch lange keine Dokumentation erstellt, die auch einem anderen Benutzer an die Hand gegeben werden kann! Die Erstellung von Dokumentationen ist äußerst zeitaufwendig und hält in der Regel nicht mit der Programmentwicklung Schritt [Bie81].

Im Idealfall sollte ein Programm daher so gestaltet sein, daß es zugleich alle Informationen zu seiner Dokumentation (z.B. in einer geeigneten formalen Notation) enthält. Die Dokumentation kann dann *automatisch* aus dem Programm generiert werden. Dieser Gedanke, der von Donald E. Knuth stammt [Knu84], hat viele der hier vorgestellten Konzepte beeinflußt.

[7] Die Ausführungsgeschwindigkeit ist gegenüber einer prozeduralen Programmierung deutlich geringer [Ude90].

Das Dokumentationsproblem ist analog zum Konfigurationsproblem bei Datenanalysebausteinen: Ein solcher Baustein sollte in der Lage sein, seine gesamten Konfigurationsdaten lesbar in einer Text-Datei abzuspeichern. Umgekehrt muß er sich aus dieser Datei rekonfigurieren können. Dabei spielt die Versionskompatibilität eine große Rolle: Alte Konfigurationsdateien sollen auch bei Versionsänderungen noch einwandfrei lesbar sein. In Abschnitt 16.3.4 ist dargestellt, wie dieses Problem in einer traditionellen strukturierten Programmiersprache gelöst werden kann [8].

14.3.4 Offenheit objektorientierter Systeme

Objektorientierte Systeme sind per definitionem offen, d.h. ein Benutzer erhält über die Methoden Zugriffsmöglichkeiten auf alle Komponenten des Systems. Er kann Klassen und Methoden hinzufügen oder erweitern. Alle dazu benötigten Informationen werden in Form neuer Klassen und Methoden automatisch publiziert. Trotzdem können der Programmcode und die internen Daten nach außen hin gekapselt werden, indem nur ein definierter Zugriff gestattet wird. In bezug auf die Datenanalyse heißt das: Jeder Anwender kann neue Techniken einführen und so in das System einbinden, daß sie von den ursprünglich vorhandenen Methoden nicht mehr zu unterscheiden sind!

Auch die Flexibilität eines objektorientierten Programms bei Erweiterungen und Veränderungen an der Basis ist bemerkenswert. So können z.B. Teile eines Systems zunächst auf einfache Weise realisiert werden, um schnell einen laufenden Prototyp zu produzieren (*rapid prototyping* [Pre87]), und erst später durch effizientere Techniken ersetzt werden, z.B. durch eine veränderte interne Datenrepräsentation oder schnellere Algorithmen. Im Vergleich dazu muß ein konservativ realisiertes System meist in großen Teilen neu geschrieben werden, wenn Teile der Basis — insbesondere der Datenrepäsentation — geändert werden!

14.4 Deklarative Sprachkonzepte

14.4.1 Deklarative Programmierumgebungen

Der Aufbau eines Programms in einer deklarativen Programmiersprache unterscheidet sich fundamental von der prozeduralen Strukturierung: Bei einer prozeduralen Programmierweise wird die Problemlösung durch Algorithmen vorgegeben, welche mit Hilfe von Kontrollstrukturen den genauen Ablauf vorschreiben. Im Gegensatz dazu besteht ein deklarativ formuliertes Programm aus einer Abfolge von beschreibenden Aussagen (*Deklarationen*), die festlegen wie die Beziehungen zwischen den verschiedenen Objekten im Problembereich aussehen. Es ist Aufgabe einer deklarativen Programmierumgebung, den Lösungsweg für ein gegebenes Problem aufgrund der gegebenen Beziehungen selbst zu suchen.

Der Hauptvertreter der deklarativen Programmiersprachen ist PROLOG [CM87]. PROLOG benötigt als Eingabe die Formulierung des Beziehungsgeflechts in Form prädikatenlogischer Ausdrücke. Aus diesem Grunde ist die Sprache besonders zur Behandlung logischer und kombinatorischer Probleme geeignet (vgl. dazu auch die Ausführungen in 8 und 12). Solche Systeme sollten keinesfalls angewendet werden, wenn eine algorithmische Lösung bekannt ist. Ihre Stärke liegt im Auffinden von Lösungswegen bei nicht klar strukturierten Zusammenhängen.

14.4.2 Anregungen für die prozedurale Programmierung

Die konkrete Arbeitsweise einer deklarativen Programmier-Umgebung ist hier nebensächlich. Wichtig sind dagegen Anregungen für die Programmiermethodik, die sich aus dem Umgang mit einem deklarativen System ergeben. Die Programmierung in einer deklarativen Programmiersprache kennt keine sequentielle Abfolge der Befehle. Zu jedem Zeitpunkt müssen alle Deklarationen bzgl. ihres Informationsgehaltes für die aktuelle Fragestellung überprüft werden. Die Konsequenz ist, daß sich der Programmierer keine Gedanken mehr über die Reihenfolge der Eingabe machen muß [9]! Er kann vielmehr die Deklarationen in einer Weise anordnen, die der *logischen* Struktur des Problems entspricht. Solche Programme sind —

[8] Ein SmallTalk-System sieht diese Möglichkeit zur Konfiguration von vorn herein für alle Objekte des Systems vor!

[9] Ganz richtig ist diese Aussage nicht, wie das Problem von Endlos-Iterationen in PROLOG bei Nennung der Abbruchbedingung *hinter* der Rekursionsbedingung zeigt (vgl. [CM87]).

da die Umsetzung in die algorithmische Sprache entfällt — im allgemeinen sehr leicht zu konzipieren, einfach zu notieren und ebenso problemlos zu warten.

14.4.2.1 Beispiel für den deklarativen Programmierstil

Dies soll am Beispiel der deklarativen Formulierung einer Eingabemaske verdeutlicht werden:

Menü enthält $Feld_1$, $Feld_2$, $Feld_3$;
Menü steht an Position (5,5);
Menü hat Höhe 10 und Breite 20;
Menü hat Farbe blau;
Menü wird gelöscht, falls Taste ESC gedrückt;

$Feld_1$ enthält Variable X_1;
$Feld_1$ steht an Position (2,2) ;
$Feld_2$ enthält Variable X_2;
$Feld_2$ steht an Position (2,4) ;
$Feld_3$ enthält Variable X_3;
$Feld_3$ steht an Position (2,6) ;

Untermenü erscheint, wenn (X_3 = "Auswahl A")

Selbstverständlich bedarf es auch zur Realisierung eines solchen Programms einiger (deklarativer) Programmierarbeit im Detail und einer stärker formalisierten Notation.

14.4.2.2 Imitation in einer prozeduralen Umgebung

Ein deklarativer Programmier-Still kann bis zu einem gewissem Grade in einer prozeduralen Umgebung imitiert werden. Vor allem für die Oberflächenprogrammierung resultiert ein hohes Maß an Fehlertoleranz und Wartungsfreundlichkeit der Programme. Der Grundgedanke bei der Imitation ist, alle Deklarationen als Prozeduraufrufe zu formulieren und in linearer Folge — ohne Berücksichtigung der Reihenfolge — zu notieren. Ein Satz zusammengehöriger Deklarationen (z.B. die Beschreibung einer Eingabemaske) wird dann zu einer umfassenden Prozedur geklammert.
Am Beispiel der Maskenprogrammierung könnte dies folgendermaßen aussehen:

```
procedure Menü;
begin
    MenuPosition (5,5);
    MenuSize (10,20);
    MenuColor (blau);
    MenuClear (ESC);

    FieldVariable (Feld1,X1);
    FieldPosition (Feld1,2,2);
    FieldVariable (Feld2,X2);
    FieldPosition (Feld1,2,4);
    FieldVariable (Feld3,X3);
    FieldPosition (Feld1,2,6);

    SubMenuCall (UnterMenü,X3 = "Auswahl A");
end ;
```

Wie ein derartiges Konstrukt konkret zu realisieren ist, wird in Kapitel 16 anhand der Ablaufsteuerung für die Benutzeroberfläche beschrieben.

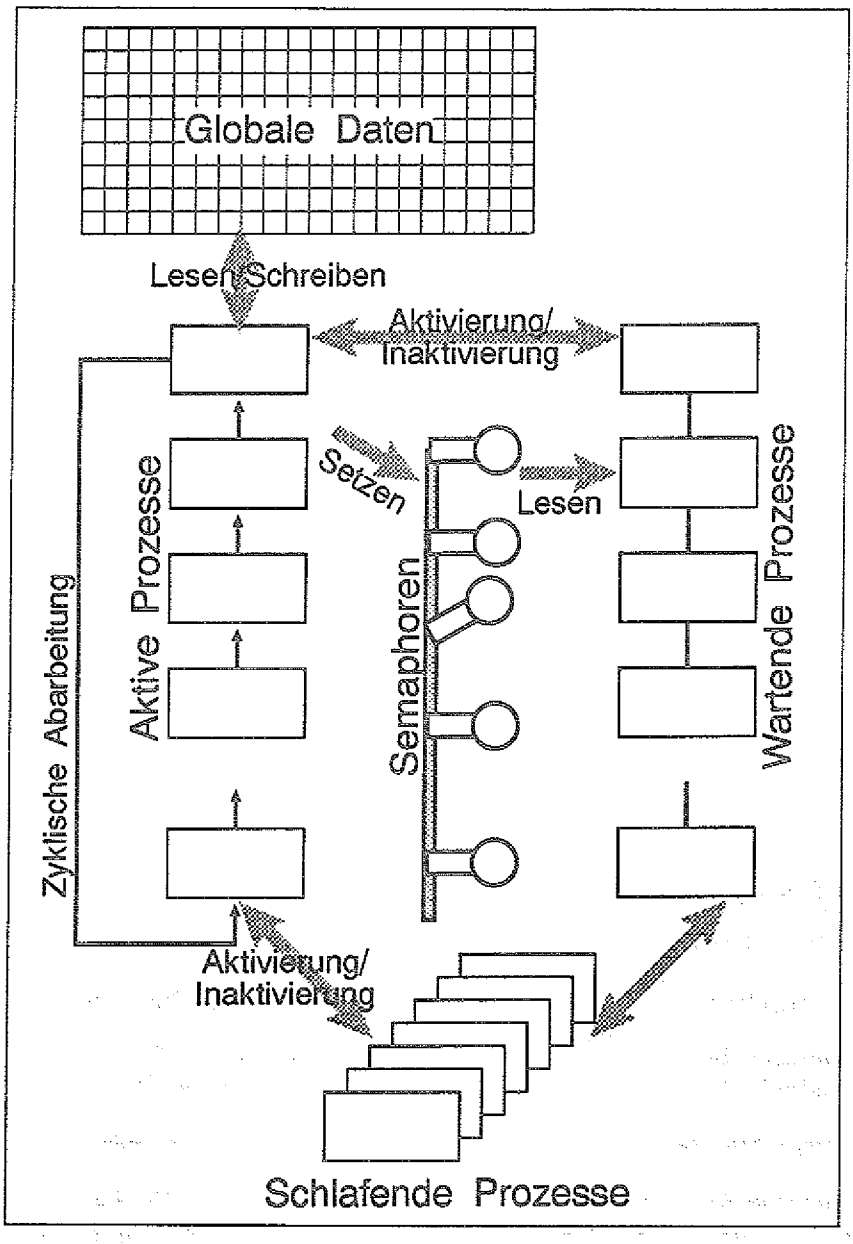


Abbildung 14.4: Konzepte und Strukturen für die Programmierung paralleler Prozesse

14.5 Programmierung paralleler Prozesse

Bei der Konzeption der Ablaufsteuerung für Netzwerke von Datenanalysebausteinen sollten die Lösungskonzepte der parallelen Programmierung berücksichtigt werden [Zim80] [ZH88]. Interessant sind hier vor allem die Aspekte, die auch beim Entwurf von Multi-Tasking-Betriebssystemen auftreten [10]. Zwar treten bei der Ablaufsteuerung viele der gewichtigsten Probleme von Betriebssystemen (Speicherverwaltung, Real-Time-Anforderungen, Interrupt-Handling, Prioritätensetzung, Pipelining [Hab81]) nicht in dieser Komplexität auf; die Lösungsansätze sind jedoch weitgehend übertragbar. An dieser Stelle soll auf das Warteschlangenkonzept, das Statuskonzept und die Interprozeß-Kommunikation eingegangen werden. Die einzelnen Abläufe sind in Abbildung 14.4 schematisch dargestellt.

[10] Es ist denkbar, die gesamte Ablaufsteuerung von vorn herein einem Multi-Tasking-Betriebssystem zu übertragen. Jedoch dürfte der Verwaltungs-Overhead für einer Vielzahl von parallel betriebenen Bausteinen erheblich zu hoch werden!

14.5.1 Warteschlangen

Betriebssysteme betreiben stets ein extensives Warteschlangenmanagement. Dabei werden alle Prozesse der gleichen Kategorie in eine Warteschlange eingereiht. Nachdem der erste Prozess in der Schlange eine Rechenzeitzuteilung erhalten hat, gibt es prinzipiell zwei Möglichkeiten:

- Wiedereinreihung an das Ende der Schlange.
- Einordnung in eine andere Kategorie.

Das Konzept einer zyklischen Warteschlange wurde in dem Programmierbeispiel in Abschnitt 14.2.2.2 bereits verwendet. Die Prozesse entsprechen hier den Datenanalysebausteinen und ein Rechenzyklus entspricht einem Iterationsschritt. Die dort als Implementierung bezeichnete Operation besteht gerade in der Einreihung eines Bausteins in die Schlange.

14.5.2 Prozeßstatus

Die Kategorie eines Prozesses wird durch seinen Status gekennzeichnet. Einige Möglichkeiten, die bei der Ablaufsteuerung der Datenanalyse eine Rolle spielen, sollen genannt werden [ZH88]:

Aktive Prozesse sind in die Warteschlange der Prozesse eingereiht, die ständig abgearbeitet werden.

Wartende Prozesse befinden sich in einer anderen Schlange und können durch ein genau spezifiziertes Ereignis aktiviert werden.

Schlafende Prozesse bedürfen dagegen einer Aktivierung durch einen anderen Prozeß.

Diese Unterscheidungen bedürfen noch einer weiteren Differenzierung, die hier nicht mehr erfolgen soll.

14.5.3 Inter-Prozeß-Kommunikation

Zwei Prozesse können auf mehrfache Weise miteinander kommunizieren [ZH88]:

- Indirekt über die globalen Daten des Systems, auf die alle Prozesse Zugriff haben.
- Mit Hilfe spezieller Signale (sogenannter *Semaphoren*), die von jedem Prozess gesetzt werden können. Schlafende Prozesse können auf das Setzen eines Signals warten.
- Über eine direkte Statusänderung, die ein Prozeß für einen anderen Prozeß veranlassen kann.

Der Unterschied zwischen dem Setzen einer Semaphore und einer direkten Statusänderung besteht darin, daß im ersten Fall der aufgerufene Prozeß für Statusänderungen verantwortlich ist, im zweiten Fall der Aufrufer. Nach Möglichkeit sollten Semaphoren bevorzugt werden. Sie greifen weniger stark in die globale Ablaufsteuerung ein. Es sei darauf hingewiesen, daß Semaphoren — die auch Parameter haben können — dem Nachrichtenkonzept der Objektorientierten Programmierung sehr nahe stehen.

14.6 Auswahl der Programmiersprache für diese Arbeit

Die im folgenden beschriebenen Programmbibliotheken wurden ausnahmslos in C programmiert. Die Gründe für diese Wahl liegen nach der Diskussion der Programmierkonzepte nahe:

- C ist eine prozedurale Standard-Programmiersprache mit mächtigen und flexiblen Sprachkonstrukten.
- C ist weitgehend standardisiert und kann mit wenig Aufwand auf andere Betriebssysteme portiert werden [Lap90].
- Bereits von der Konzeption her ist C stark modular aufgebaut.

- Die Effizienz von in C programmierten numerischen Routinen steht der von FORTRAN-Programmen in nichts nach (bei geschickter Realisierung sind sogar Geschwindigkeitsvorteile zu erzielen).
- Im Bereich der Systemprogrammierung hat sich C als weltweiter Standard durchgesetzt (UNIX und X-Windows sind in C programmiert).
- Die Erweiterung von C zum zukünftigen C++-Standard erlaubt die spätere Übertragung der Programme in einen objektorientierten Kontext.
- Eine hybride Programmierung unter Verwendung mehrerer Sprachen gestaltet sich auf dem PC sehr aufwendig. Für die spätere Weiterentwicklung ist jedoch daran gedacht. Dann wird der Umstand wichtig, daß alle heutigen Compiler eine Mixed-Language-Schnittstellen für C-Programme zur Verfügung stellen.

Alle Strukturen wurden von vornherein so konzipiert, daß mit der C++-Umsetzung bald begonnen werden kann. Die C++-Sprachdefinition [Str87] wurde daher stets mit berücksichtigt. Bei den nun folgenden Ausführungen soll dies nicht mehr eigens betont werden.

Kapitel 15

Datentypen

Die grundlegenden Datenstrukturen treten in den verschiedenen Teilen eines größeren Programmsystems immer wieder auf. Es lohnt sich daher, die Basisklassen und -methoden an zentraler Stelle einmal sorgfältig zu konzipieren. Dabei entsteht insbesondere das Problem, daß Strukturen wie Vektoren, Matrizen, Kellerspeicher und Warteschlangen zwar stets gleich aufgebaut sind, aber auf verschiedenen Grunddatentypen (ganze Zahlen, Fließkommazahlen, Zeichenketten etc.) operieren. Hier bietet die Objektorientierte Programmierung entscheidende Vorteile, da solche Klassen abstrakt, d.h. unabhängig vom Grunddatentyp definiert werden können. Die im folgenden beschriebenen Realisierungen in C sind daher objektorientiert in dem in Abschnitt 14.2 beschriebenen Sinne. Die Übertragung nach C++ wird die noch bestehenden Umständlichkeiten vollständig beseitigen!

15.1 Strings

Stringoperationen sind in allen prozeduralen Programmiersprachen enthalten. Allerdings bergen sie stets die Gefahr der Speicherbereichsüberschreitung, insbesondere bei der Übergabe an Prozeduren. Da dies einer der am schwersten lokalisierbaren Fehler überhaupt ist, wurde ein gegen Speicherbereichsüberschreitung gesicherter String-Typ geschaffen.

15.1.1 Pascal- und C-Strings

Ein C-String ist nicht direkt eine Folge von Druckzeichen, sondern ein Zeiger auf das erste Zeichen einer solchen Folge. Das String-Ende ist durch ein spezielles End-of-String-Symbol EOS gekennzeichnet. Im Gegensatz dazu steht ein Pascal-String, der im nullten Byte die Länge mitführt (Abb. 15.1).
Beide Darstellungsweisen haben Vor- und Nachteile:

- Bewegt man einen C-String-Zeiger um einige Zeichen weiter, so repräsentiert der Zeiger einen Teilstring des anfänglichen Strings. Diese Methode zur Bildung von Teilstrings [1] soll als *Quereinstieg* bezeichnet werden. Beim Versuch des Quereinstiegs in einem Pascal-String würde dagegen das Längenbyte abhanden kommen! Durch die Möglichkeit des Quereinstiegs lassen sich Stringoperationen, die mit Teilstrings operieren sowie das sequentielle Abarbeiten eines Strings in C sehr elegant umsetzen.

- Wird ein C-String an ein Unterprogramm übergeben, wo er z.B. mit Zeichen gefüllt wird, so kann das Unterprogramm nicht feststellen, wieviel Speicherplatz zur Verfügung steht. Bei nachlässiger Programmierung kommt es daher oft zur Speicherbereichsüberschreitung (was zu tagelangen Fehlersuchaktionen führen kann ...). Einem Pascal-String ist dagegen die Längeninformation inhärent; der Fehler kann vom System abgefangen werden.

[1] Der Begriff wird an anderer Stelle (15.3) auch auf beliebige sequentielle Strukturen angewandt

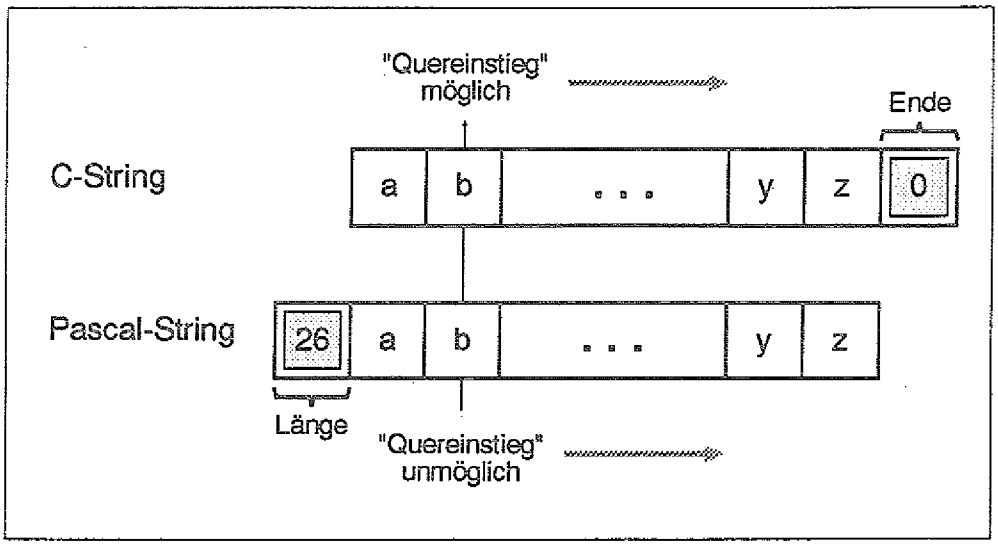


Abbildung 15.1: Unterschiede zwischen Pascal- und C-Strings

15.1.2 Erweiterung der C-Strings

Der neue Stringtyp soll die positiven Eigenschaften von Pascal- und C-Strings in sich vereinen:

- Die Möglichkeit zum Quereinstieg bleibt erhalten.
- Die Information über den maximal verfügbaren Speicherplatz bei Schreiboperationen ist in der Datenstruktur enthalten.
- Zusätzlich ist der neue Stringtyp kompatibel mit den normalen C-Strings bzgl. Leseoperationen. D.h. die Bibliothek der Standardopertionen (Suchen, Scannen, Längenbestimmung, Konvertierung in numerische Werte etc.) bleibt weiterhin einsetzbar.

Es folgt unmittelbar:

- Ein String ist weiterhin ein Zeiger auf eine Zeichenkette.
- Die Zeichenkette ist nach wie vor mit EOS terminiert.
- Am Anfang der Datenstruktur dürfen keine Zusatzinformationen eingefügt werden.

D.h. es verbleibt prinzipiell nur genau eine Möglichkeit zur Realisierung. Die Längeninformation muß *hinter* dem EOS-Symbol angehängt werden:

⟨Zeichenkette⟩ EOS ⟨Längeninformation⟩

Die Längeninformation kann nicht die Zahl der insgesamt zur Verfügung stehenden Speicherplätze sein, weil dieses Datum nicht invariant unter der Quereinstiegs-Operation ist. Vielmehr muß gelten:

⟨Längeninformation⟩ = ⟨Zahl der verbleibenden Speicherplätze⟩

Der neue Datentyp wird als *Savestring* bezeichnet. Der Speicherplatzbedarf für einen Savestring ist

Kapazität + 2 Byte,

wobei die *Kapazität* eines Strings die maximale Zahl von Zeichen ist, die er aufnehmen kann.

15.1.3 Restrisiken

Einige Restrisiken, die bei der Verwendung von Savestrings verbleiben, sind eher unproblematisch:

- Wird der Zeichenketteneiger über das abschließende EOS-Symbol hinausbewegt, so findet er undefinierte Werte vor und das Stringende sowie der freie Speicherplatz sind nicht mehr zu ermitteln. Normalerweise wird ein String jedoch Zeichen für Zeichen abgearbeitet, so daß bei EOS abgebrochen werden kann.
- Strings können nicht länger als $254 = 2^8 - 2$ Zeichen lang sein, solange nur ein Byte für die Längeninformation verwendet wird. Abhilfe kann durch Erweiterung auf 2 Byte geschaffen werden, was etwas rechenzeitaufwendiger ist.
- Operationen, die den zur Verfügung gestellten Speicherplatz überschreiten, schneiden die überschüssigen Zeichen ab (tolerante Fehlerbehandlung). Im interaktiven Betrieb wird die Zahl der eingebbaren Zeichen ohnehin meist durch die Feldlänge in Eingabemasken bestimmt.

15.1.4 Operationen auf erweiterten C-Strings

Savestrings können genau wie normale C-Strings initialisiert werden. Dazu ist sind lediglich als letzte Zeichen der initialisierenden Zeichenkette das EOS-Symbol und das Längenbyte explizit anzugeben. Die übrigen realisierten Operationen gehören den folgenden Kategorien an:

- Spezielle Initialisierungsoperationen.
- Dynamische Speicherplatzbelegung.
- Setzen, Auffüllen und Abschneiden.
- Kopieren und Löschen.
- Eingabe von der Tastatur.

15.1.5 Objektorientierte Erweiterung

Ein Savestring ist von vornherein eine objektorientierte String-Implementierung, da die Längeninformation unabhängig vom Kontext stets zur Verfügung steht (Kapselung). In einer C++-Implementierung kann weiterhin die Bezeichnervielfalt durch Überladen prägnanter Funktionsnamen (Init, Copy, Delete, Insert Set, Fill, Cut) erheblich reduziert werden. Alle diese Funktionen arbeiten dann — abhängig vom Typ der Argumente — auf normalen C-Strings, Savestrings und einzelnen Druckzeichen. Für die Konkatenation bietet sich ferner die Realisierung des in Pascal übliche Verkettungsoperators + an.

15.2 Vektoren und Matrizen

Vektoren und Matrizen von Elementen beliebigen Typs (Strings, Zahlen, arithmetische Operations-Codes etc.) sind zentrale Objekte für die modellgestützte Datenanalyse. Ihre Dimensionierung erfolgt erst zur Laufzeit, so daß die Programmierung dynamischer Strukturen erforderlich wird.

15.2.1 Anforderungen an dynamische Vektoren und Matrizen

- Die Dimensionierung muß zur Laufzeit veränderbar sein. Dazu werden Dimensionsoperationen wie das Löschen, Einfügen und Anhängen von Elementen oder Zeilen/Spalten benötigt.
- Ein Vektor oder eine Matrix muß beliebige Datentypen fassen können.
- Zahlen müssen im Speicher genau so abgelegt sein, wie bei einer statischen Vektor- oder Matrixdefinition. D.h. die Werte stehen in aufeinanderfolgenden Speicherzellen. Bei Matrizen läuft hierbei der Spaltenindex schneller als der Zeilenindex (siehe Abb. 15.2). Nur so kann zugekaufte Konfektionssoftware für die Vektor- und Matrizenrechnung weiterhin verwendet werden.

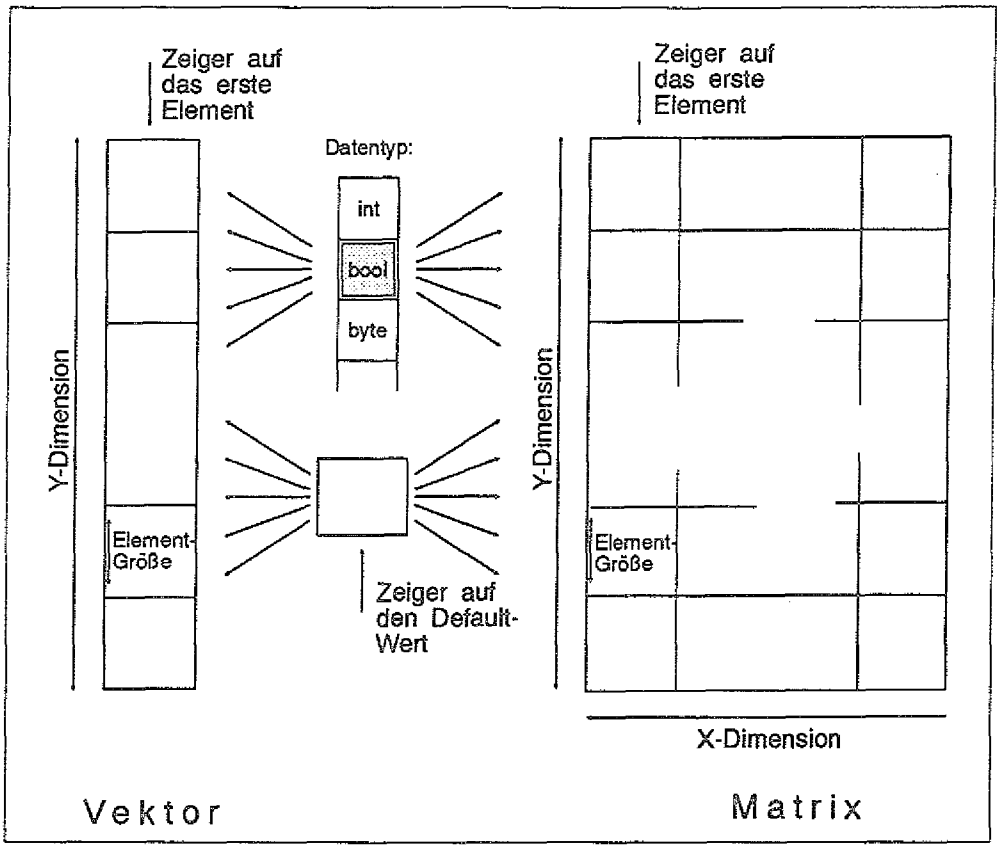


Abbildung 15.2: Realisierung von dynamischen Vektoren und Matrizen

- Im Anschluß an die Konfigurationsphase darf die dynamische Speicherplatzverwaltung keine Rechenzeit mehr in Anspruch nehmen. Insbesondere muß für numerische Routinen ein schneller Zugriff auf das Datenfeld gewährleistet sein.

15.2.2 Realisierung als Datenstruktur

15.2.2.1 Einträge in der Struktur

Ein Vektor (eine Matrix) wird dargestellt durch eine Datenstruktur (vgl. Abb. 15.2), die folgende Einträge enthält:

- ein Zeiger auf das Datenfeld,
- die Angabe, ob es sich um einen Vektor oder eine Matrix handelt [2]: Matrizen werden als Unterklasse der Vektoren mit Zusatzdaten aufgefaßt Die Vektoroperationen sind (wenn auch nicht immer sinnvoll) auch auf Matrizen angewendbar.
- die Zeilen- und Spaltendimension (im folgenden stets *Y-Dimension* und *X-Dimension* genannt). Vektoren haben nur eine Y-Dimension (sie werden also "stehend" notiert).
- die Anzahl der Elemente im Datenfeld: bei Vektoren ist dies gleichzeitig die Y-Dimension, bei Matrizen das Produkt X-Dimension × Y-Dimension.
- die Länge eines Eintrags in Bytes: Alle Standardoperationen beruhen nur auf dem Verschieben der Elemente im Speicher und benötigen daher in erster Linie diese Größe [3].

[2] Die Typinformation kann in einer objektorientierten Umgebung wie z.B. C++ entfallen
[3] Dies kann in C++ anders geregelt werden. Beim folgenden Typ-Eintrag ebenso.

- eine Angabe über den Typ der Einträge (Typcode): Es gibt einige Operationen die typabhängig programmiert sind (z.B. Vergleichen mit einem String oder einer Zahl); andere sind nur für bestimmte Typen zulässig (z.B. das Belegen mit aufsteigenden Zahlenwerten) und erzeugen sonst eine Fehlermeldung.
- einen Default-Wert-Zeiger: Der Default-Wert wird bei allen dimensionsvergrößernden Operationen automatisch zur Initialisierung der freiwerdenden Felder benutzt.
- Dimensionslistenzeiger und Dimensionsoperations-Codes (bei Vektoren je einen, bei Matrizen je zwei davon). Die Bedeutung dieser Einträge wird in 15.2.4 erklärt.

15.2.2.2 Unterstützte Typen und Default-Werte

Dynamische Vektoren und Matrizen sind zunächst vorgesehen für

- ganze Zahlen,
- Dezimalzahlen,
- Savestrings,
- Wahrheitswerte,
- Arithmetische Operations-Codes,
- Maschinencodesequenzen.

Weitere Typcodes können vom Benutzer jederzeit hinzugefügt werden. Unter C++ wird die Typ-Beschränkung vollkommen entfallen.
Folgende häufig benötigte Default-Werte zur Initialisierung sind bereits vorprogrammiert:

- Alle explizit unterstützten Datentypen reservieren einen speziellen Wert, um anzuzeigen, daß kein Eintrag vorhanden ist (bei Zahlen oft als NaN = Not a Number bezeichnet):
  - Bei ganzen Zahlen ist hierfür der größtmögliche Wert reserviert, der darstellbar ist ($2^8 - 1$ bzw. $2^{16} - 1$).
  - Bei Dezimalzahlen wird nach IEEE-Standard ein Zahlenwert mit größtmöglichem Zehnerexponent verwendet (10^{39} bzw. 10^{309} bzw. 10^{4933}) [CG87]
  - Boolesche Werte benötigen in C von vornherein 1 Byte Speicherplatz, so daß 255 = FF (hexadezimal) benutzt werden kann.
  - Bei Strings bietet sich der Leerstring an.
  - Bei Operations-Codes eine Codesequenz der Länge 0.
  - Bei Maschinencodesequenzen ein befehlsloses Unterprogramm mit Rücksprung.
- Die Null gibt es in verschiedenen Formen:

ganzzahlig	:	0
dezimal	:	0.0
boolsch	:	FALSE
String	:	"0",

- ganz analog die Eins.

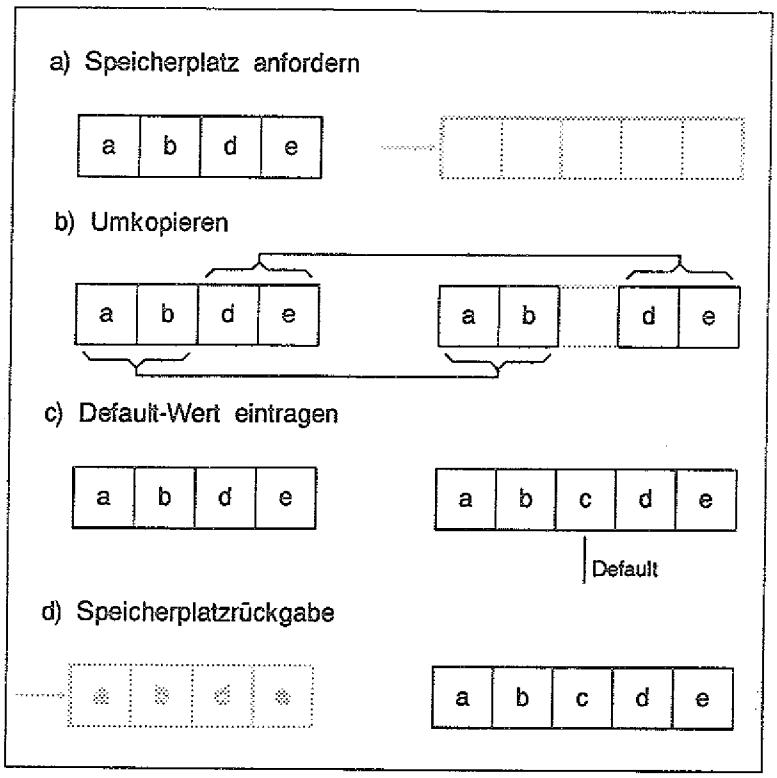


Abbildung 15.3: Durchführung einer Dimensionsoperation (Einfügens eines Elementes)

15.2.3 Operationen

15.2.3.1 Dimensionsoperationen und ihre Umsetzung

Zu den dimensionsverändernden Operationen (im folgenden kurz *Dimensionsoperationen*) zählen:

- Dimensionssetzung oder Umdimensionierung,
- Einfügen und Anhängen von Elementen (bzw. Zeilen oder Spalten),
- Löschen von Elementen.

Sie werden programmtechnisch wie folgt realisiert (vgl. Abb. 15.3):

1. Anforderung des Speicherplatzes für die neuen Abmessungen.
2. Umkopieren der Teile, die erhalten bleiben.
3. Belegen neu hinzugekommener Teile mit dem Default-Wert.
4. Rückgabe des alten Speicherplatzes.

Dimensionsoperationen können unterbunden werden, wenn Objekte fest dimensioniert bleiben sollen.
Weitere Operationen auf Vektoren und Matrizen gehören den folgenden Kategorien an:

- Initialisierungsoperationen.
- Setzen und Zugreifen.
- Kopieren und Zusammensetzen.
- Spezielle Operationen (Suchen, Sortieren, Dateiverzeichnis einlesen etc.).

Bemerkung: Operationen, die Vektoren in Matrizenzeilen oder -Spalten kopieren und umgekehrt wurden nicht programmiert, da sie meist nur für Zahlen benötigt werden. Damit gehören sie zu den numerischen Operationen und werden im Numerikteil (Kapitel 19) abgehandelt.

Alle genannten Operationen prüfen die Korrektheit von Typen und Dimensionen sowie die Indexbereiche und nehmen eine entsprechende Fehlerbehandlung vor. Ein korrekt programmiertes System sollte solche Fehler eigentlich nicht mehr aufweisen. Daher kann die Fehlerprüfung bei vollständig ausgetesteten Programmen über einen Schalter aus dem Code genommen werden (vgl. dazu auch 15.4).

15.2.4 Dimensionslisten

15.2.4.1 Ein Beispiel

Modellgestützte Methoden benötigen eine ganze Reihe von Vektoren und Matrizen, deren Dimensionen untereinander in Beziehung stehen. Dies wird am Beispiel des Kalman-Filters sehr deutlich: Die Abbildung 15.5 zeigt alle Funktionen, Vektoren und Matrizen, die zur Definition eines Erweiterten Kalman-Filters benötigt werden, zusammen mit ihrer Dimensionierung. Zusätzlich zu den im Bild dargestellten numerischen Größen müssen noch die String-Vektoren für Funktionsterme und Startwerte vorgesehen werden, ganz abgesehen vom Parametervektor und dem Vektor der Eingangsgrößen, die in der Abbildung weggelassen wurden. Insgesamt kommt man bei der Realisierung des Filters auf

	35	Vektoren
und	7	Matrizen
mit	4	verschiedenen Dimensionszahlen
sowie	4	Datentypen.

Eine solche Vielzahl von Dimensionen und Dimensionsbeziehungen explizit zu verwalten, garantiert nahezu das Auftreten von Zeigerfehlern! Daher wurde eine implizite Repräsentation der Dimensionsbeziehungen mit in die Datenstruktur implementiert.

15.2.4.2 Arbeitsweise von Dimensionslisten

Eine *Dimensionsliste* ist eine zyklisch gekettete Liste, die Vektoren und Matrizen gleicher Dimension miteinander verbindet. In dem Moment wo ein Glied der Kette seine Dimension ändert, wird automatisch die entsprechende Dimensionsoperation bei allen anderen Gliedern veranlaßt. Die zyklische Verkettung der Liste erspart hierbei die Suche nach dem Listenanfang. Auf diese Weise ist garantiert, daß stets alle Elemente einer Dimensionsliste dieselbe Dimension haben.

Eine Matrix gehört in der Regel zwei Dimensionslisten an. Einzige Ausnahme ist eine quadratische Matrix, die in ihrer Liste mit der X- und Y-Dimension zweimal vertreten ist. Vektoren benötigen hingegen nur jeweils einen Dimensionslistenzeiger. Zu jedem Zeiger muß vermerkt werden, ob dieser sich auf die X- oder Y-Dimension des nachfolgenden Listenelementes bezieht. Dies erklärt die Bedeutung von Dimensionslistenzeigern und Dimensionsoperations-Codes in der Datenstruktur von Vektoren und Matrizen. Die Abbildung 15.5 zeigt, wie die Dimensionslisten des Kalman-Filter-Beispiels konkret aussehen.

15.2.4.3 Definition von Dimensionslisten

Formal können die Dimensionsbeziehungen des Filters wie folgt formuliert werden:

$$
\begin{aligned}
\text{Y-dim } x &= \text{Y-dim } f \\
&= \text{Y-dim } P \\
&= \text{X-dim } P \\
&= \text{X-dim } f_x \\
&= \text{Y-dim } f_x \\
&= \text{Y-dim } R \\
&= \text{X-dim } R \\
&= \text{X-dim } g_x \\
&= \text{Y-dim } K
\end{aligned}
\qquad
\begin{aligned}
\text{Y-dim } y &= \text{X-dim } K \\
&= \text{Y-dim } g_x \\
&= \text{Y-dim } Q \\
&= \text{X-dim } Q \\
&= \text{X-dim } S \\
&= \text{Y-dim } S
\end{aligned}
$$

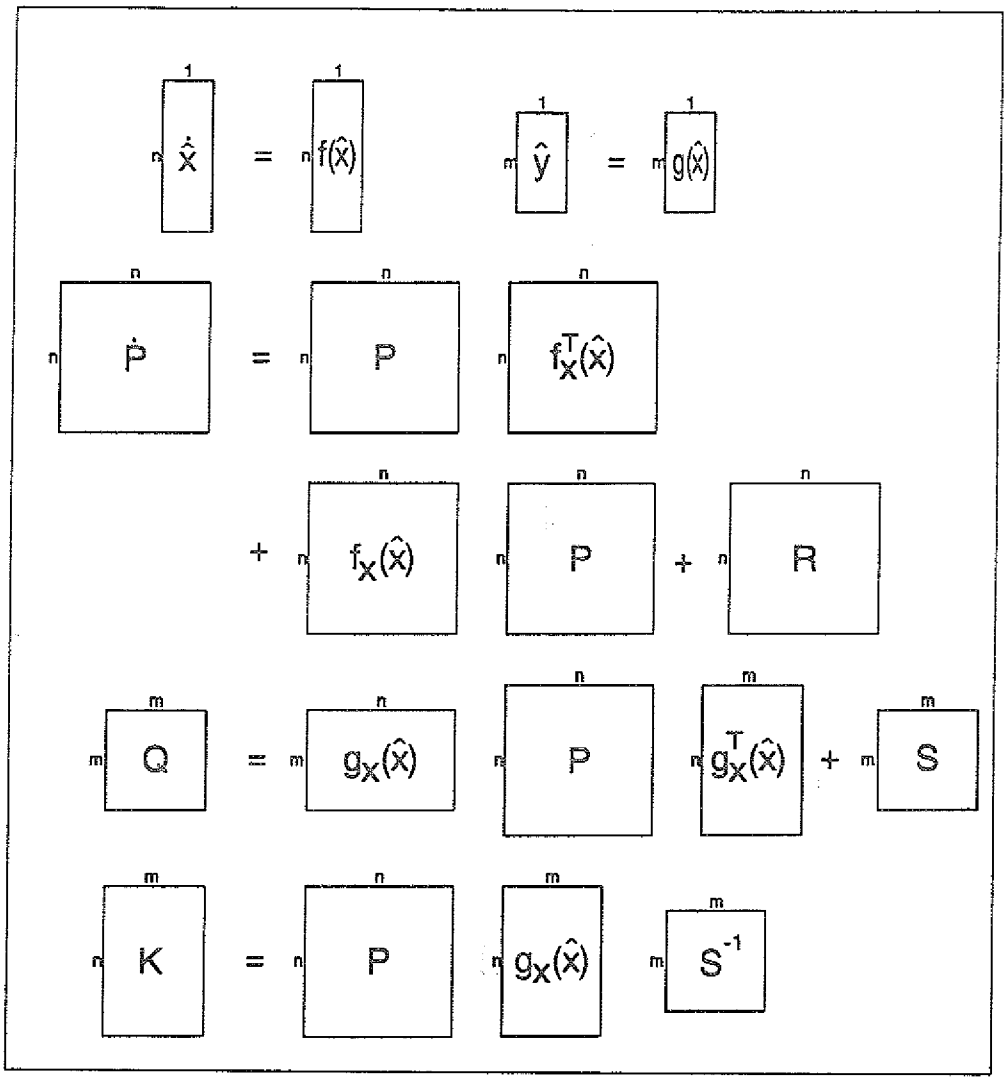


Abbildung 15.4: Funktionen, Vektoren und Matrizen eines Erweiterten Kalman-Filters mit ihren Dimensionierungen

Für die Formulierung der obigen Liste von Gleichheitsbedingungen in der Programmiersprache wurde ein deklarativer Stil gewählt. Am Beispiel des Filters:

```
DimList ( x , Y-Dim , f ,  Y-Dim ) ;
DimList ( x , Y-Dim , P , Y-Dim ) ;
DimList ( x , Y-Dim , P , X-Dim ) ;
   ⋮
```

Diese Vereinbarungen müssen einmal im Initialisierungsteil des Programms erfolgen.

15.2.4.4 Constraint Propagating

Nebenbedingungen wie die in diesem Abschnitt behandelten Dimensionsbeziehungen werden in der Fachsprache als *Constraints* bezeichnet. Die automatische Aufrechterhaltung derartiger Beziehungen heißt *Constraint Propagating*. Effiziente Methoden für das Constraint Propagating haben für die Implementierung interaktiver Systeme eine ständig wachsende Bedeutung [BD86] [BDFB+87].

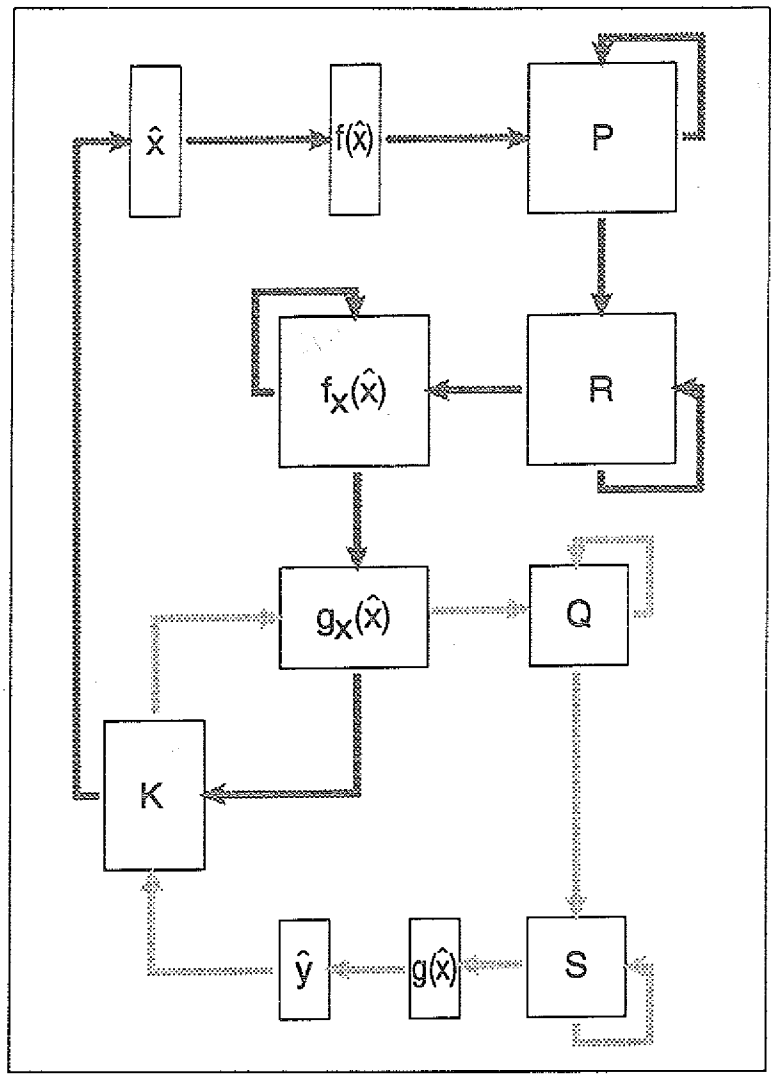


Abbildung 15.5: Dimensionsliste für das Kalman-Filter

15.2.5 Vorteile und Einschränkungen

Die gewählte Repräsentation von Vektoren und Matrizen hat bei der Gestaltung von Benutzeroberflächen sichtbare Konsequenzen: Der Benutzer kann mit Hilfe entsprechender Kommandos jederzeit Zeilen oder Spalten in Vektoren und Matrizen einfügen. Alle Objekte gleicher Dimension werden dann automatisch mit umdimensioniert. Hat sich dabei die Zahl der Elemente vergrößert, so sind die neu hinzugekommenen Einträge bereits mit Default-Werten initialisiert. Verwendet man z.B. NaNs als Default-Werte, so wird sofort optisch angezeigt, an welchen Stellen in der Maske noch Informationen einzutragen sind. Auf diese Weise ist bereits den dynamischen Klassen selbst ein gewisses Maß an Benutzerführung eigen!

Einige Einschränkungen für die Verwendung der Klassen sollten genannt werden:

- Dimensionsoperationen sind mit häufigem Umkopieren von Speicherblöcken verbunden. Allerdings wird diese Operation von der Hardware der meisten Rechner eigens unterstützt. Zudem treten Umdimensionierungen nur während des Benutzerdialogs auf, der keine hohen Anforderungen an die Rechenleistung stellt.

- Die Elemente von Vektoren und Matrizen können selbst keine variable Länge haben und müssen daher ausreichend groß ausgelegt werden. Dieses Problem tritt vor allem bei Strings und arithme-

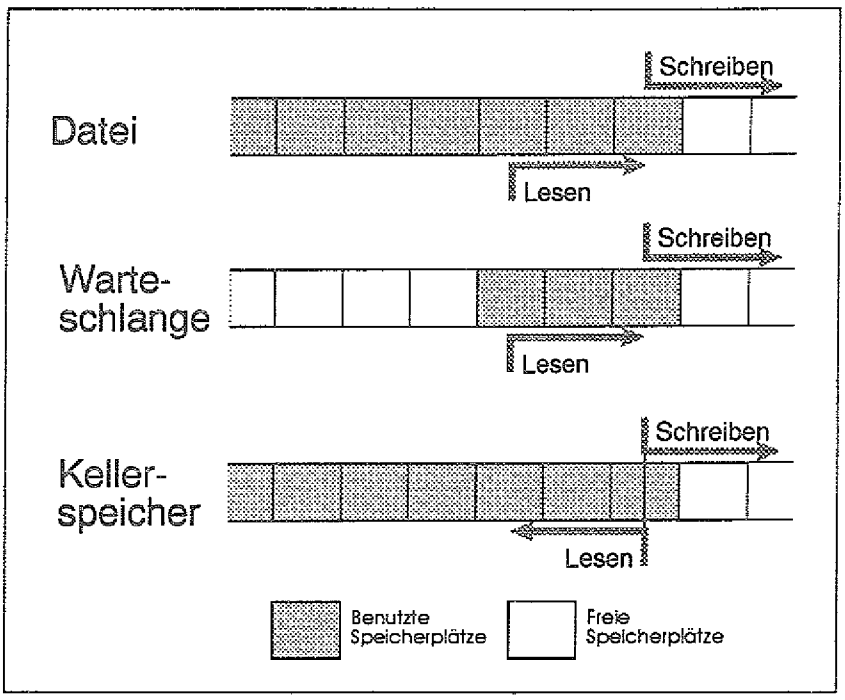


Abbildung 15.6: Verschiedene Arten von sequentiellen Speichern

tischem Operations-Code auf [4].

- Bei Umdimensionierungsoperationen wechseln alle Einträge eines Vektors oder einer Matrix ihren Platz im Speicher. Aus diesem Grunde dürfen keine Zeiger auf Vektor- oder Matrixelemente zeigen, solange sich die Dimensionen verändern! Andernfalls entstehen freihängende Zeiger.

15.3 Sequentielle Speicher

Sequentielle Speicher sind Speicherbereiche, die sequentiell gelesen und geschrieben werden. Sie nehmen aufeinander folgend eintreffende Elemente eines festen oder auch variierenden Typs auf und geben diese in derselben oder in umgekehrter Reihenfolge weiter. Lese- und Schreiboperationen können dabei durchaus an verschiedenen Enden des Speichers stattfinden. Typische Beispiele sind (siehe auch Abb. 15.6):

Dateien: Schreiben und Lesen erfolgt in gleicher Richtung, wobei nicht gleichzeitig geschrieben und gelesen werden kann (Konzept des Schreib-Lese-Kopfes).

Warteschlangen: Schreiben und Lesen erfolgt gleichsinnig, und zwar gleichzeitig am vorderen und hinteren Ende der Schlange.

Kellerspeicher: Schreiben und Lesen erfolgt in entgegengesetzter Richtung. Es kann dabei stets nur das zuletzt geschriebene Element gelesen werden.

In Gegensatz dazu steht der wahlfreie Zugriff auf beliebige Elemente, wie er bei Vektoren und Matrizen oder Random-Access-Datein verwirklicht ist.

15.3.1 Anforderungen an sequentielle Speicher

15.3.1.1 Typ der Elemente

Die im Speicher abzulegenden Elemente können

[4] Ein Ausweg wäre die Verwendung eines Vektors von Zeigern. Dies würde jedoch die Bedingung der sequentiellen Ablage der Elemente im Speicher verletzen (vgl. 15.2.1)

- von einem festen und zur Zeit der Übersetzung bekannten Typ sein,
- von einem festen, aber erst zur Laufzeit des Programms bekannten Typ sein,
- von variablem Typ sein. In diesem Fall wird allerdings vorausgesetzt, daß die zu speichernden Elemente eine Längeninformation in sich tragen (so wie z.B. die Savestrings).

15.3.1.2 Operationen

Auf einem sequentiellen Speicher sind eine ganze Reihe von Operationen denkbar, von denen je nach Anwendung nur ein Teil gebraucht wird:

a) Initialisieren und Löschen:

Initialisierung, ggfs. mit Speicherplatzanforderung, wenn der benötigte Speicherplatz für die Struktur erst zur Laufzeit bekannt ist.

Löschen und Rückgabe des benötigten Speicherplatzes.

Löschen eines Abschnitts am Anfang oder Ende des Speichers.

b) Schreiben und Lesen:

Schreiben eines Elements.

Lesen eines Elements.

Lesen und Löschen des gerade gelesenen Elements aus dem Speicher.

c) Positionierung:

Aufsetzen des Schreibkopfes am Ende.

Aufsetzen des Lesekopfes am Anfang bzw. Ende (je nach Speichertyp).

Suchen durch sequentielles Lesen.

Referenzierung , d.h. Anforderung eines Zeigers auf einen Eintrag.

d) Speicherplatzanfragen:

Gesamtspeicherplatz in Byte oder speicherbaren Elementen,

Restspeicherplatz, d.h. die Zahl der noch freien Plätze.

Speicherende-Abfrage.

e) Spezielle Operationen:

Verketten zweier sequentieller Speicher desselben Typs.

Verschieben eines sequentiellen Speichers im Arbeitsspeicher [5].

Quereinstieg in dem bei den Savestrings erläuterten Sinne (vgl. 15.1.1) [6].

15.3.1.3 Performanz

Zu den Performanz-Kriterien gehören:

- Rechenzeitaufwand,
- Programmieraufwand,
- Speicherplatzbedarf.

Welche Anforderung dabei vorrangig ist, hängt sehr von der Problemstellung ab. Eine ausführliche Diskussion findet sich in [Knu73].

[5] Achtung: Speicherbereiche, die verschiebbar sind, können nicht gleichzeitig von anderswoher mit Zeigern referenziert werden! Zum Problem der Kopieropertionen auf Datenstrukturen vgl. auch 15.5.2.

[6] Quereinstiege sind nur bei bestimmten Datenrepräsentationen möglich. Z.B. ist ein Quereinstieg in eine Warteschlange undenkbar!

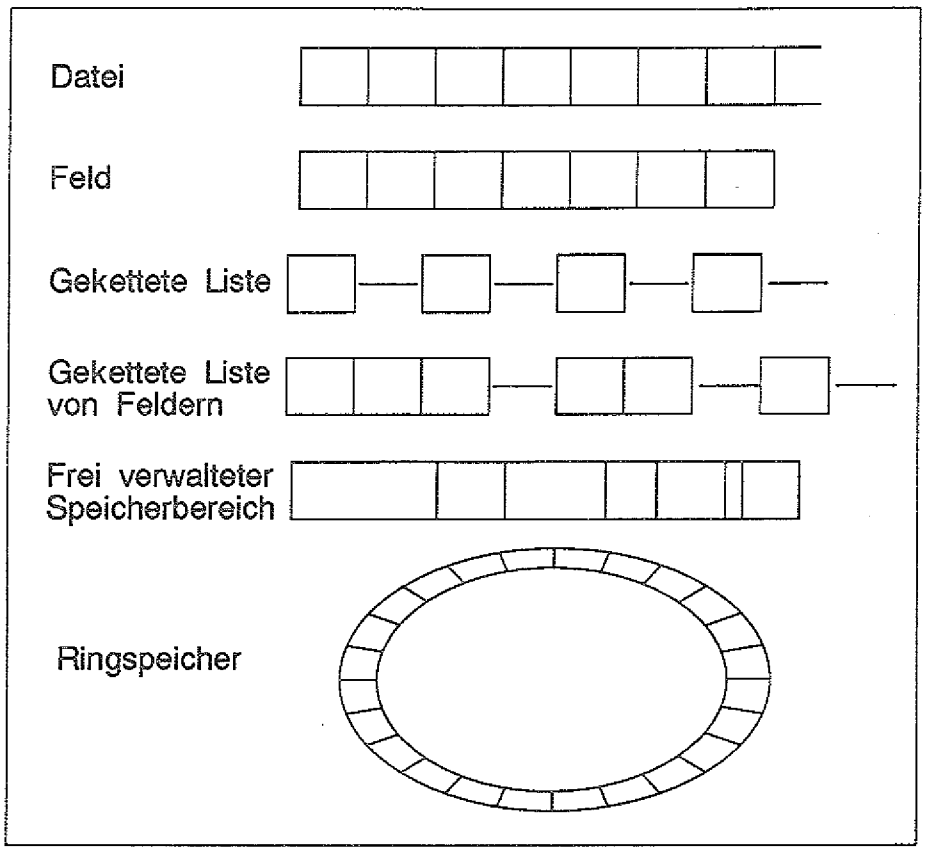


Abbildung 15.7: Realisierung von sequentiellen Speichern

15.3.2 Implementierung von sequentiellen Speichern

15.3.2.1 Mögliche Datenstrukturen

Sequentielle Speicher können je nach Anforderungsprofil auf verschiedene Weise realisiert werden [Knu73]. Unter anderem sind folgende Datenstrukturen denkbar (vgl. Abb. 15.7):

- gewöhnliche sequentielle Dateien,
- indizierte Felder,
- gekettete Listen,
- gekettete Listen von indizierten Feldern (Hybridrepräsentation),
- Ringspeicher,
- frei verwaltete Speicherabschnitte.

Die wesentlichen Unterschiede faßt die Tabelle 15.3.2.1 zusammen. Sie zeigt, daß keine der Repräsentationen alle Vorteile in sich vereinigt. Je nach Anforderungsprofil muß man sich für eine Lösung entscheiden.

15.3.2.2 Umgesetzte Typen von sequentiellen Speichern

Jede der Strukturen wurde in den noch zu beschreibenden Modulen gebraucht. Die Auflistung der Anwendungsbereiche zeigt, wie die erwähnten Speichermodelle im Rahmen der Datenanalyse Anwendung finden:

Datenstruktur:	Sequentielle Datei	Indiziertes Feld	Gekettete Liste	Ring-Speicher	Frei verwalteter Bereich
Speicherkapazität unbegrenzt	+	−	+	−	−
Lesen mit Speicherfreigabe möglich	−	0	+	+	−
Zeiger-Referenzierung möglich	−	+	+	+	+
Verkettung ohne Kopieroperation möglich	−	−	+	−	−
Zusammenhängender Speicherbereich	−	+	−	+	+
Quereinstieg möglich	+	−	+	−	−
Verschiedene Element-Typen möglich	+	−	+	−	+
Dauerhafte Speicherung	+	−	−	−	−
Geringe Zugriffszeit	−	+	0	+	0

+ ja, 0 bedingt, − nein

Tabelle 15.1: Unterschiede zwischen den verschiedenen Realisierungen eines sequentiellen Speichers

Dateien:

- Konfigurationsdateien,
- Hilfstexte, Fehlermeldungen,
- Prozeßdaten.

Indizierte Felder:

- Ablage von Daten in einer Aufrufhierarchie,
- Nachrichtenstack der Oberflächensteuerung,
- Operatorliste des Parsers,
- Auswertung arithmetischer Operations-Codes.

Gekettete Listen von indizierten Feldern:

- Zwischenspeicher für den Formelparser und -Differenzierer.

Ringspeicher:

- Zwischenpuffer des Scanners.

Frei verwalteter Bereich:

- Bezeichnerliste des Parsers,
- Maschinencode-Sequenzen.

15.4 Fehlerbehandlung

Die Fehlerprüfung und -behandlung ist eine wichtige, aber lästige Aufgabe bei der Programmierung großer Systeme, da sie im Falle eines korrekt funktionierenden Systems und korrekter Eingaben eigentlich überflüssig ist. Meist erhält jeder Modul seine eigene Fehlerbehandlung. Spätestens wenn man diese zum dritten Mal neu programmiert, wird klar, daß zwar nicht die Fehlerprüfung, wohl aber die Fehlerbehandlung zentralisierbar ist [7]. Ein solcher Ansatz wird kurz vorgestellt:

15.4.1 Aufgaben einer Fehlerprüfung und -behandlung

Im Prinzip gibt es zwei Arten von Fehlern:

- Programmierfehler im System: Die Fehlerprüfung dient dazu, diese aufzudecken, und wird nach Ausmerzen aller Programmierfehler überflüssig. Die meisten professionellen Programme enthalten aber nach wie vor einige Prüfungen, was sich in Meldungen des Typs: "Serious error! Please consult ... " niederschlägt.

- Bedienungsfehler von Seiten des Benutzers: Solche Fehler müssen stets diagnostiziert und mit einer möglichst detaillierten Meldung versehen werden. Nur bei schwerwiegenden Fehlern sollte ein Abbruch erfolgen. Ansonsten ist es meist die einfachste Lösung, die falsche Eingabe zu ignorieren.

15.4.2 Realisierung der Fehlerbehandlung

15.4.2.1 Fehlerdaten

Jeder Modul erhält ein Exemplar einer allgemein definierten Fehlerstruktur, in der alle Daten darüber abgespeichert sind, wie Fehler in diesem Modul zu behandeln sind (objektorientierter Ansatz). Die Struktur enthält:

- einen Code für den Fehlererkennungsmodus,
- den Namen des Moduls, in dem der Fehler aufgetreten ist,
- den Namen der Funktion, bei welcher der Fehler aufgetreten ist,
- einen Fehlercode,
- den Namen einer Datei, in der die Fehlercodes mit Text erklärt sind (optional),
- den Namen einer Log-Datei, in der Fehlermeldungen protokolliert werden (optional),
- eine Einsprungadresse zu einer anderen Stelle des Programms, wo die Fehlerbehandlung erfolgt (optional) [8],
- einen Funktionszeiger für eine externe Fehlerbehandlungsprozedur (optional),
- einen Fehlerzähler,
- ein Limit, das angibt, wieviele Fehler maximal toleriert werden, bis ein Programmabbruch erfolgt.

[7] Auch dieses Problem ist in SmallTalk durch eine freiprogrammierbare Fehlerbehandlung für jede einzelne Klasse elegant gelöst [GR89].

[8] C verfügt über die Möglichkeit eines *Far Jump*, d.h. es kann aus einer tieferen Ebene einer Rekursion heraus auf eine beliebige höhere Ebene gesprungen werden.

15.4.2.2 Fehlerbehandlungsmodi

Bisher sind folgende Behandlungsmodi realisiert:

Ignorieren : Fehlerbehandlung völlig ausschalten.

Tolerieren : Fehler registrieren (Log-Datei, Fehlerzähler), aber weiter nicht behandeln.

Melden : Anzeige einer Meldung auf dem Bildschirm, die mit Tastendruck quittiert werden muß.

Far Jump : Einsprung in ein anderes Programmstück über die vorgegebene Einsprungadresse.

Behandlung : Aufruf der externen Fehlerbehandlungsfunktion.

Halt : Fehlermeldung und Programmstop.

15.4.3 Operationen zur Fehlerbehandlung

Zu Anfang sollte die Modul-eigene Fehlervariable Error explizit initialisiert werden. Dies kann in C statisch zur Übersetzungszeit erfolgen. Ein auftretender Fehler kann von da ab jederzeit mit einem Aufruf der Prozedur

HandleError (Error, Module, Function, Number)

behandelt werden. Weitere Operationen gehören in die Kategorien:

- Initialisierung und Modus-Setzung,
- Hilfsfunktionen zur benutzereigenen Fehlerbehandlung.

Mit Hilfe einer derartigen — schnell zu implementierenden — Fehlerstruktur für jeden Modul sollten Unterlassungsfehler, die dadurch zustande kommen, daß eine Fehlerbehandlung "auf später" verschoben wird, nicht mehr auftreten.

15.5 Objektorientierte Realisierung der Datentypen

Die Vielfalt der Realisierungsmöglichkeiten und Anforderungen an die verschiedenen Datentypen zeigt deutlich die Notwendigkeit der Entwicklung von Programmierhilfsmitteln zur einheitlichen Behandlung solcher immer wiederkehrender Strukturen! Hier bietet sich die Objektorientierte Programmierung als ideale Lösung an. Ansatzweise wurde dies für die dynamischen Vektoren und Matrizen durchgeführt. Eine extensive Umsetzung objektorientierter Konzepte in einer prozeduralen Programmiersprache mit den Techniken aus Abschnitt 14.2.3.2 gestaltet sich jedoch aufgrund der großen Unterschiede im Detail extrem schwierig, so daß im Rahmen dieser Arbeit davon Abstand genommen wurde.

15.5.1 Klassenbibliotheken

Das SmallTalk-80-System [GR89] enthält im Standardumfang bereits eine umfangreiche Klassenbibliothek für die grundlegenden Datentypen. Alle Ansammlungen von Objekten — dies können z.B. verkettete und sortierte Listen, Keller, Warteschlangen oder Felder sein — werden hierbei unter der abstrakten Oberklasse Collection zusammengefaßt. Gemeinsame Operationen für alle Kollektionen sind:

- die Initialisierung,
- das Hinzufügen oder Herausnehmen eines Elements,
- die Angabe der Zahl der Elemente.

Die abgeleiteten Tochterklassen haben ein sich mit der Tiefe der Klassenhierarchie vergößerndes Methodenspektrum. Ganz ähnliche Klassenbibliotheken sind für andere objektorientierte Programmiersprachen verfügbar (vgl. z.B. [TCP90]). Es ist zu erwarten, daß sich durch die Verwendung solcher Bibliotheken in Zukunft die Entwicklung großer Programmsysteme entscheidend beschleunigen wird.

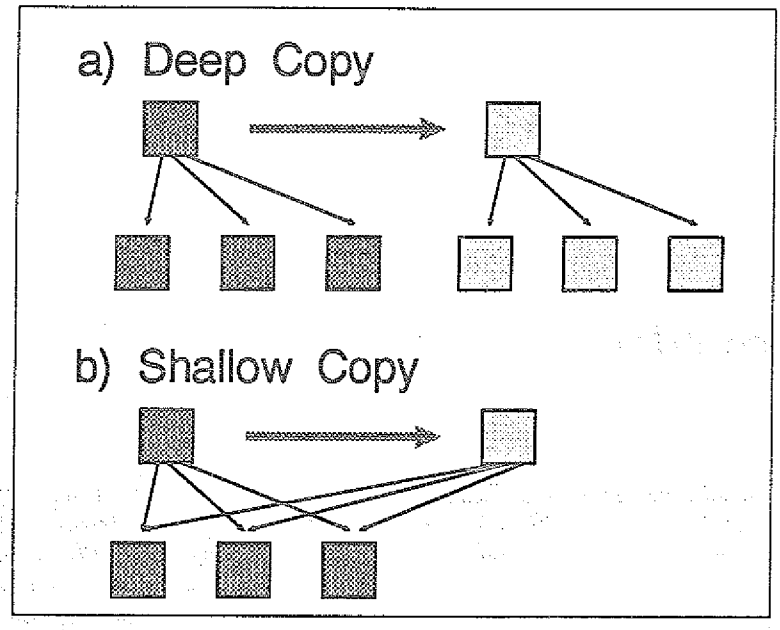


Abbildung 15.8: Kopieroperationen unterschiedlicher Tiefe

15.5.2 Kopieroperationen

Ein besonderes Problem bei Kollektionen werfen die Kopieroperationen auf. Bei Zeigerstrukturen unterscheidet man prinzipiell zwei Varianten [GR89] (vgl. auch Abb. 15.8):

Deep Copy: Kopiert wird die gesamte Struktur einschließlich der referenzierten Elemente.

Shallow Copy: Kopiert werden lediglich die Zeiger, wobei die referenzierten Elemente dieselben bleiben.

Je nach Anwendung ist meist eine der beiden Varianten sinnvoller. Dabei ist darauf zu achten, daß im Falle einer Shallow Copy die "Besitzrechte" (*Ownership*) an den Elementen auf mehrere Kollektionen übergehen: Elemente, die zu mehreren Kollektionen "gehören" dürfen nicht aus dem Speicher gelöscht oder im Speicher verschoben werden, da andernfalls freihängende Zeiger entstehen [9].

[9] Die Smalltalk-Speicherverwaltung verhindert auch diesen Zeigerfehler, indem sie über alle Zeigerverbindungen Buch führt und bei Verschiebung von Objekten die Zeiger nachträgt. Dies macht den eigentlichen Komfort eines solchen Systems aus und gleichzeitig den größten Zeitfaktor!

Kapitel 16

Benutzeroberflächen

Es gibt derzeit eine Vielzahl von kommerziellen Maskengeneratoren, die teils zu den entsprechenden graphischen Oberflächen (MS-Windows, OS/2 Presentation Manager, X Windows) gehören und teils als eigenständige Systeme im Textmodus betrieben werden [GDW88] [TCP91]. Die meisten Systeme sind allerdings in erster Linie für die Kommunikation mit dem Betriebssystem oder zur Formulargenerierung im Wirtschafts- und Verwaltungssektor entworfen wurden: Es mangelt nicht an Eingabeformaten für Finanzdaten oder Prüfmechanismen für die Vollständigkeit von Eingaben, wogegen die Darstellungung von Dezimalzahlen in wissenschaftlicher Notation sowie die Verwaltung von Vektoren und Matrizen variabler Dimension nicht vorgesehen ist! Da eine graphische Oberfläche aus Speicherplatz-, Zeit- und Kostengründen zunächst nicht in Betracht kam, wurde eine Eigenentwicklung durchgeführt, die im Textmodus arbeitet. Die gewonnenen Konzepte sollen später auf graphische Oberflächen übertragbar sein.

16.1 Anforderungen und Lösungsansätze

16.1.1 Anforderungen an eine Benutzeroberfläche für Datenanalyse und Prozeßüberwachung

Es sollen hier nur spezielle Probleme angedeutet werden, die bei den auf dem Markt befindlichen Systemen Probleme bereiten (vgl. auch [HDF+85]):

Übersichtliche Codierung: Die Codierung der Menüs muß sich so einfach wie möglich gestalten, Fehler tolerieren und die Struktur der Maske prägnant wiedergeben. Insbesondere sollte der Programmcode zu einem Menü übersichtlich an einer Stelle im Programm zusammengefaßt und nicht etwa fragmentiert sein. Dies vermeidet die Hauptfehlerquellen bei der Programmierung von Benutzerschnittstellen.

Wissenschaftliche Datentypen: Die meisten Systeme unterstützen weder eine wissenschaftliche Notation noch die Darstellung fehlender Daten (NaN's).

Dynamische Strukturen: Nicht nur die interaktive Datenanalyse, sondern auch die Prozeßüberwachung (Bezeichnertabellen, Kanalverwaltung, etc.) erfordert eine Datenrepäsentation in Form dynamischer Strukturen, wie z.B. den in 15.2 beschriebenen dynamischen Vektoren und Matrizen. Dies zieht die Forderung nach einer dynamisch veränderbaren Maskenstruktur nach sich. Insbesondere sollten die Dimensionierungsoperationen fest zum System gehören und schnell bedient werden können. Bei Veränderungen von Dimensionen muß das Layout automatisch neu berechnet werden (vgl. auch 15.2.4.4).

Konfiguration: Die Möglichkeit zum Abspeichern aller Maskeninhalte in eine Datei muß im System enthalten sein. Entscheidend ist hierbei die Erhaltung der Versionskompatibilität: Verändert sich im Laufe der Entwicklung eines Programms die Menüstruktur, so müssen alte Konfigurationsdateien nach wie vor einlesbar und die Veränderungen diagnostizierbar sein. Nur so kann die Weiterverwendbarkeit älterer Ergebnisse auch für die Zukunft garantiert werden!

Hilfestellung: Gerade bei der Umsetzung komplexerer Methoden sind Hilfestellungen unumgänglich. Die Hilfestellung muß kontextsensitiv und erneut versionsunabhängig implementiert sein.

Dokumentation: Die Fähigkeit zur Dokumentation muß dem System immanent sein. D.h. die Ausgabe von Konfigurationsdaten auf einen Drucker soll nicht eigens programmiert werden müssen. Das Updating von Dokumentationsroutinen — bei Versionsumstellungen ein erheblicher Zeitfaktor — entfällt.

16.1.2 Verschiedene Ansätze zur Beschreibung von Benutzeroberflächen

Es gibt eine Reihe von Ansätzen zur Spezifikation von Benutzeroberflächen, die sich aber nicht gegenseitig ausschließen. Sie sollen in Hinblick auf die Anwendungbarkeit bei der Datenanalyse und Prozeßüberwachung verglichen werden:

16.1.2.1 Interface-Definitions-Sprachen

Alle Angaben zum Aufbau von Masken (Position und Art der vorkommenden Datenfelder, Namen von Prozeduren zur Ereignisbehandlung etc.) werden in einer speziellen Definitionssprache (*User Interface Language*) [Gil91] formuliert. Die Konfigurationsdatei wird dann von einem eigenen Compiler (*Resource Compiler*) verarbeitet und zum Anwender-Programm hinzugeladen.

Vorteile :

- Mit Hilfe eines Maskeneditors können Oberflächen nach dem WYSIWYG-Prinzip erstellt und in einer Konfigurationsdatei abgespeichert werden.

Nachteile :

- Es ist keine dynamische Maskenstruktur denkbar!
- Die Prozeduren müssen separat codiert werden [Gil91].

16.1.2.2 Deklarative Formulierung

Bei dieser sehr eleganten Beschreibungsart werden alle Objekte der Benutzerschnittstelle zusammen mit ihren Eigenschaften in Form einer Liste von Klauseln notiert. Eine solche Liste kann natürlicher Sprache sehr ähnlich sehen (vgl. dazu auch 14.4).

Vorteile :

- Kompakte, einfache und äußerst fehlertolerante Formulierung.
- Dynamische Strukturen sind problemlos realisierbar.

Nachteile :

- In der Reinform kann die deklarative Beschreibung nur in Sprachen wie PROLOG oder Smalltalk verwirklicht werden [1].
- Die Abarbeitung ist zeitaufwendig.

[1] SmallTalk erscheint heute als *die* Sprache zur schnellen Umsetzung interaktiver Strukturen.

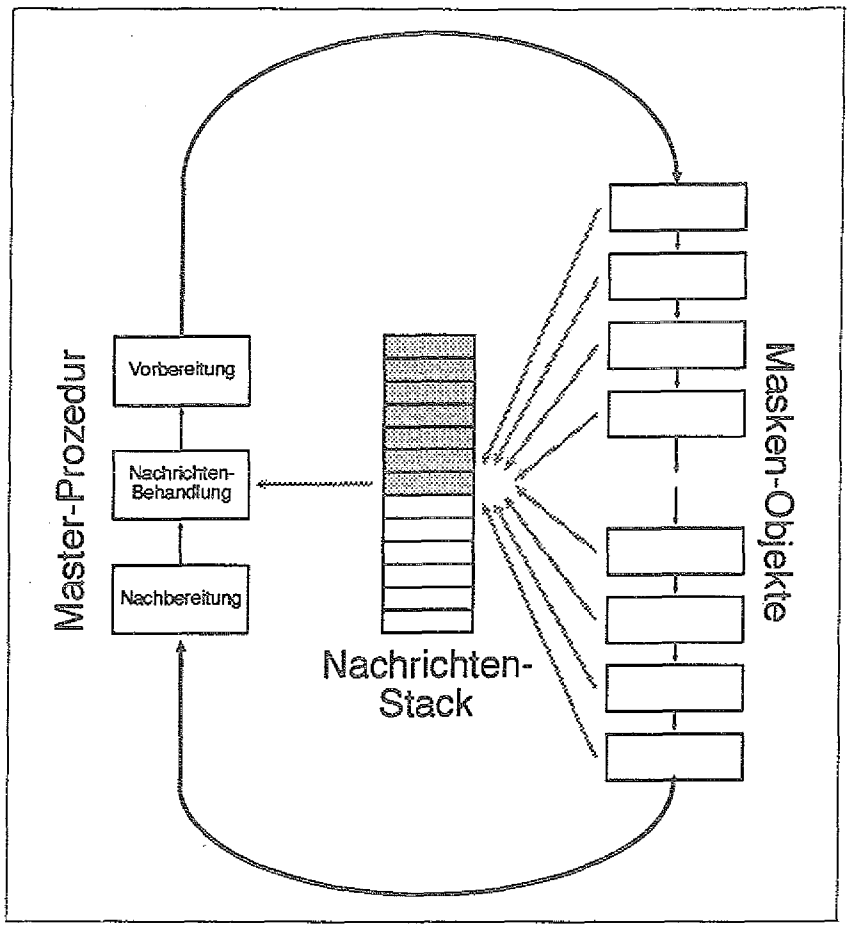


Abbildung 16.1: Nachrichtenbasierte Kontrolle einer Benutzeroberfläche

16.1.2.3 Datenstrukturen aus Menüobjekten

Solche Strukturen sind in der Regel die rechnerinterne Darstellung der Benutzeroberfläche [2]. Hierbei kommen die Vorteile der objektorientierten Programmierung voll zum Tragen.

Vorteile :

- Schnelle Abarbeitung.
- Das System kann auf der Ebene des Programmcodes durch dynamische Strukturen erweitert werden.

Nachteile :

- Die Datenstrukturen sind auf der Ebene des Quellcodes aufwendig aufzubauen und zu verwalten.
- Operationen müssen erneut separat codiert werden.

16.2 Konzeption und Arbeitsweise des Maskengenerators

Der vorgestellte Maskengenerator soll nicht den Perfektionsansprüchen professioneller Systeme genügen. Vielmehr stellt es einen Versuch zur prinzipiellen Lösung der skizzierten Probleme im Zusammenhang

[2] Für das X-Windows-System wurden die Begriffe *Widgets* und *Gadges* geprägt [Gil91].

mit wissenschaftlichen Benutzeroberflächen dar. Im Mittelpunkt stehen daher dynamische Strukturen, übersichtliche Codierung, automatische Konfiguration und Dokumentation sowie kontextsensitive Hilfestellungen.

16.2.0.4 Nachrichtenbasierte Kontrolle

Nahezu alle heutigen Maskengeneratoren beruhen auf dem Nachrichtenkonzept der Objekt-Orientierten Programmierung. Dabei werden in einer Schleife ständig alle Maskeninträge abgearbeitet und mit Nachrichten bzgl. Status und Benutzereingaben beschickt. Die Objekte senden entsprechende Antworten an eine Kontrollprozedur (*Master-Routine*) oder auch andere Menüobjekte zurück. Eine nachrichtenbasierte Ablaufsteuerung ist stets ein offenes System, da vom Programmierer eigene Nachrichten und Operationen hinzugefügt werden können.

Auch die hier vorgestellte Steuerung des Benutzerdialogs ist nachrichtenbasiert: Eine Maske ist dabei vollständig definiert durch eine Menüprozedur, die von der Master-Routine zyklisch aufgerufen wird (siehe Abb. 16.1) . In der Prozedur ist der Reihe nach in Form von Funktionsaufrufen aufgelistet, welche Elemente das Menü enthält. Die Art der Formulierung imitiert somit einen deklarativen Programmierstil (vgl. 14.4).

16.2.0.5 Ein Beispiel für eine Maskendefinition

Operationen werden ebenfalls durch Funktionsaufrufe bewirkt, die bedingt oder unbedingt stattfinden können. Bedingte Operationen werden in erster Linie dann ausgeführt, wenn auf Eingabefeldern bestimmte Tasten gedrückt werden. Zu diesem Zweck gibt jede Funktion, die ein Eingabefeld definiert, eventuell eingegebene Steuerzeichen als Wert zurück. Das Parameterschema für die Prozeduren ist soweit möglich stets gleich.
Nachfolgend ein Beispiel für eine Menüdefinition (vgl. auch Abb. 2.6):

```
procedure menu ;
{ Menüdefinition }

begin
   { Platzbedarf auf dem Textbildschirm, Farben, Rahmen, Kopf- und Fusstexte }
   DiaSize    ( 10 , 5 , 60 , 15 ) ;
   DiaColor   ( schwarz , grau , magenta ) ;
   DiaLayout  ( Rahmen + Einfach + Kopftext + Fußtext ) ;
   DiaHead    ( "Kopftext" ) ;
   DiaFoot    ( "Fußtext" ) ;

   { Textfeld }
   DiaText    ( 1 , 1 , "Ein erklärender Text");

   { Eingabefelder verschiedener Typen mit Position, Format und Feldnamen }
   DiaType    ( 1,  3,  N,  "Nummer : %3u (<)",    "Nummer" ) ;
   DiaType    ( 1,  5,  Z,  "Zähler : %6i (<)",    "Zähler" ) ;
   DiaType    ( 1,  7,  t,  "Zeit   : %10.5f(>)",  "Zeit" ) ;
   DiaType    ( 1,  9,  x,  "Wert   : %12.5d(>)",  "Wert " ) ;
   DiaType    ( 1,  11, D,  "Diskret ? %5b (=)",   "Diskret" ) ;
   DiaType    ( 1,  13, A,  "Abfrage : %3q (<)",   "Abfrage" ) ;
   DiaType    ( 1,  15, K,  "Kennung : %c ",       "Kennung" ) ;
   DiaType    ( 1,  17, T,  "Titel   : %8S (<)",   "Titel" ) ;

   { Variabel dimensionierbare Vektoren und Matrizen, ggfs. transponiert }
   DiaVct     ( 15,1,  v ,  "Zustand  : %12.5d(T)", "Zustand" ) ;
   DiaMat     ( 15,3,  M ,  "Kovarianz: %12.5d" ,   "Kovarianz" ) ;
```

```
    { Bedingtes Öffnen eines Untermenüs bei Eingabe einer Returntaste }
    { DiaType gibt einen Return-Code zurück. DiaKey fragt die gedrückte Taste ab. }
    if (        DiaType ( 1 , 19 , "UnterMenü1" , "%5S(B)" ,"" )
       and      DiaKey = CarriageRetrurn )
      then DiaEnter ( UnterMenü1 );

    { Unbedingtes Öffnen eines anderen Untermenüs bei Eingabe der F1-Taste }
    if ( DiaKey = F1_Key )
      then DiaEnter ( UnterMenü2 ) ;

    { Auf Tastendruck Schreiben und Lesen des Menüinhalts in eine Datei }
    if ( DiaKey = Alt_W_Key )
      then DiaWrite ( "Menu.Cfg" ) ;
    if ( DiaKey = Alt_R_Key )
      then DiaRead ( "Menu.Cfg" ) ;
end ;
```

16.2.1 Nachrichten

Die Menü-Definitionsprozedur wird von der Master-Prozedur so lange immer wieder aufgerufen, bis ein Verlassen des Menüs über eine entsprechende Anforderung veranlaßt wird. Bei jedem Aufruf wird dabei eine Nachricht an alle Objekte der Maske gesendet, die dann von jedem Objekt, das angesprochen ist, ausgeführt wird. Umgekehrt können die Objekte des Menüs auch Nachrichten versenden. Eine Nachricht besteht aus:

- einer Adresse, die den Empfänger spezifiziert. Dies kann sein:

All,	d.h. alle Objekte der Maske sind adressiert,
Master,	d.h. die aktuelle Nachricht wurde von einem der Menüelemente an die Master-Routine gesendet,
First,	d.h. das erste Element des Menüs nimmt die Nachricht an,
Next,	d.h. die Nachricht ist für das zum aktuellen Zeitpunkt nächste in der Folge der aufgelisteten Elemente bestimmt.

- einem Operations-Code, der die auszuführende Aktion spezifiziert. Definiert sind:

Field Output	: Bildschirmausgabe aller Einträge.
Field Input	: Benutzereingabe auf dem derzeit aktiven Feld.
Field Leave	: Derzeit aktives Feld verlassen.
Field Enter	: Feld aktivieren.
Field Goto	: Feld anspringen.
Field Read	: Feld aus einer Konfigurationsatei einlesen.
Field Write	: Feld in eine Konfigurationsdatei schreiben.
Field Print	: Dokumentation aller Einträge.
Field Help	: Hilfestellung zum aktuellen Eintrag.
Menu Enter Up	: Menü "von unten" betreten.
Menu Enter Down	: Menü "von oben" betreten.
Menu Init Up	: Menü beim Betreten "von unten" initialisieren.
Menu Init Down	: Menü beim Betreten "von oben" initialisieren.

Menu Leave Up	: Menü "nach oben" verlassen.
Menu Leave Down	: Menü "nach unten" verlassen.
Menu Jump	: Anderes Menü anspringen.
System Exit	: Menüsystem verlassen.
Done	: Aktion ausgeführt.
Dimension Op	: Dimensionsoperation bei dynamischen Vektoren und Matrizen.
Window Layout	: Platzauslegung für das aktuelle Menü.
Cursor Move	: Cursor-Bewegung ausführen.
Key Pressed	: Taste eingegeben.
Key Define	: Operation zu einer eingegebenen Taste in Form einer Nachricht definieren.
Error Beep	: Akustisches Fehlersignal auslösen.

- Parametern, die in Anzahl und Typ vom Operations-Code abhängen.

16.2.1.1 Der Nachrichten-Stack

Oft setzen sich Operationen aus Teiloperationen zusammen, so daß der Bedarf entsteht, mehrere Nachrichten in Folge abzusenden und der Reihe nach bearbeiten zu lassen. Zu diesem Zweck wurde ein Nachrichten-Stack aufgebaut, in dem Folgen von Nachrichten abgelegt werden (siehe auch Abb. 16.1). Die zuletzt gesendete Nachricht wird somit zuerst wieder aus dem Stack gelesen, so daß Anweisungsfolgen in umgekehrter Reihenfolge geschickt werden müssen. Dies erscheint gegenüber einer Warteschlange etwas umständlich, jedoch hat der Stack den großen Vorteil, daß Operationen vorgezogen, aber auch für später zurückgestellt werden können.
Beispiele :

- Eine Fehlerbehandlung kann zu jedem Zeitpunkt vorrangig ausgeführt werden.
- Eine Nachricht kann vor dem Betreten eines Menüs abgelegt werden, um erst zu dem Zeitpunkt wirksam zu werden, wo dieses Menü wieder verlassen wird. Auf diese Weise können komplexe Abläufe realisiert werden (vgl. 14.3.3.1).

16.3 Lösung der gestellten Probleme

Die zu Anfang beschriebenen Probleme der Oberflächengestaltung für den Einsatz in der Datenanalyse können mit dem vorgestellten Ansatz sehr elegant gelöst werden:

16.3.1 Übersichtliche Codierung

Die Vorteile eines deklarativen Programmierstils wurden bereits diskutiert (vgl. 14.4). Im übrigen spricht das angeführte Beispiel für sich: Alle benötigten Informationen sind in einer Prozedur konzentriert. Lediglich die Initialisierung der Strukturen muß in einer klassischen Programmiersprache noch an anderer Stelle erfolgen (unter C++ wird sich auch dieses Problem nicht mehr stellen). Bei der Codierung können nahezu keine Fehler gemacht werden, die nicht sofort am optischen Eindruck der fertigen Maske erkannt werden könnten.

16.3.2 Wissenschaftliche Datentypen

Das System unterstützt die verschiedenen Arten der dezimalen Notation. Insbesondere können bei Vektoren und Matrizen die Dezimalpunkte bündig untereinander gesetzt werden, was die Lesbarkeit erheblich steigert. Aufgrund einer objektorientierten Realisierung der Grunddatentypen sind neue Datentypen und Notationen jederzeit hinzufügbar.

Wichtig ist auch die durchgehende Unterstützung fehlender Werte (NaNs). Ein solcher Wert kann in einem eigenen Format ((NaN)) angezeigt werden.

16.3.3 Dynamische Strukturen

Alle Vektoren und Matrizen können vom Benutzer frei dimensioniert werden. Dies geschieht durch Anwahl einer Zeile/Spalte des Objekts auf dem Bildschirm und Eingabe einer Tastenkombination:

Alt I, Alt X/Y : Einfügen einer Spalte/Zeile.

Alt D, Alt X/Y : Löschen einer Spalte/Zeile.

Alt A, Alt X/Y : Anhängen einer Spalte/Zeile.

Alt C, Alt X/Y : Kopieren einer Spalte/Zeile (Duplikat).

Alt Q, Alt X/Y : Übernehmen der daneben/darüberliegenden Spalte/Zeile (Quotation).

Nun folgen der Reihe nach folgende Aktionen des Systems:

1. Zuordnen einer Dimensionsoperation zur Tastatureingabe.
2. Durchführung der Dimensionsoperation vom derzeitig aktiven Objekt, resultierend in einer Umdimensionierung aller Einträge in der zugehörigen Dimensionsliste.
3. Neuberechnung des Maskenlayouts.
4. Löschen der Maske und Neuausdruck.
5. Neupositionierung des Cursors — das Objekt könnte den Ort geändert haben — und Reaktivierung des Eingabefeldes.

16.3.4 Konfiguration

Für die Konfiguration entscheidend sind die Namen, die zusätzlich für jedes Feld vergeben werden (oben z.B. "Zeit", "Zustand", "Kovarianz" usw.). Diese Namen erscheinen nicht auf dem Bildschirm, sondern dienen der internen Ablaufsteuerung. Sie sind der Schlüssel zu einer versionsunabhängigen Verwaltung von Konfigurations- und Hilfsdateien und dürfen daher bei einem Versionsupdating als einziges nicht verändert werden.

16.3.4.1 Dateiformat

Eine Konfigurationsdatei ist eine Textdatei; sie kann im Prinzip auch von einem Editor erstellt werden. Die Datei enthält Einträge, die jeweils das folgende Format haben:

a) Name,

b) Dimensionierung bei Vektoren und Matrizen,

c) Werte bzw. Liste der Werte.

Die Reihenfolge der Einträge ist frei.

16.3.4.2 Abspeichern

Zum Abspeichern der Konfiguation wählt der Benutzer einen Dateinamen, worauf das System (neben der üblichen Sicherheitsanfrage) folgende Aktionen ausführt:

1. Öffnen der Datei.
2. Übersenden einer Meldung und des Dateizeigers an alle Eingabefelder, die sich auch in mehreren Masken befinden können.
3. Jedes Feld schreibt daraufhin seine Daten in dem genannten Format in die Datei.
4. Schließen der Datei.

16.3.4.3 Einlesen

Das Einlesen gestaltet sich etwas schwieriger, da hier das Problem der Versionskompatibilität gelöst werden muß, d.h. die Reihenfolge der Einträge kann sich geändert haben, es können neue Daten hinzugekommen sein und alte fehlen:

1. Öffnen der Datei. Danach zyklische Ausführung der Punkte 2a - 2d bis zum Erreichen des Dateiendes.

2a. Einlesen des nächsten Namens.

2b. Senden des Namens zusammen mit der aktuellen Dateiposition an alle Eingabefelder.

2c. Sobald ein Feld mit dem gesendeten Namen identifiziert ist, liest es die zugehörigen Informationen aus der Datei ein und gibt eine entsprechende Meldung zurück.

2d. Falls keines der Eingabefelder die Daten verwerten konnte, kann die Master-Routine eine entsprechende Fehlerbehandlung vornehmen.

3. Schließen der Datei. Neuberechnung und Ausgabe der Bildschirmmaske.

4. Eingabefelder, für die keine Informationen in der Datei zu finden waren, tragen jetzt noch ihre NaN-Werte. Sie können aber auch — auf eine entsprechende Nachricht hin — eine Warnung anzeigen.

16.3.5 Hilfestellung

Die kontextsensitive Hilfestellung gestaltet sich aufgrund der vergebenen Feldnamen sehr einfach. Eine Hilfedatei muß dazu lediglich zu jedem Namen einen Hilfetext bereithalten. Zusätzlich können im Kopf globale Informationen abgelegt werden. Der Ablauf nach der üblichen Hilfe-Anforderung des Benutzers durch die F1-Taste ist:

1. Absenden einer Aufforderung zur Hilfestellung an den derzeit aktiven Menüeintrag.

2. Das angesprochene Feld öffnet die Hilfe-Datei, sucht seinen Namen heraus und zeigt die zugehörige Meldung auf dem Schirm an.

3. Nach Bereit-Meldung übernimmt die Master-Prozedur die Kontrolle.

16.3.6 Dokumentation

Die Dokumentation einer Konfiguration auf dem Drucker ist ebenso unproblematisch, wenn man sich an den folgenden Grundsatz hält:

> "Die Präsentation der Daten in den Eingabemasken muß zugleich die logische Struktur der Daten hervorheben."

Dieser Forderung kann meist durch eine geeignete Unterstrukturierung der Masken und entsprechende Überschriften nachgekommen werden.

Ist dies gewährleistet, so gestaltet sich die Dokumentation denkbar einfach: Die Eingabemasken erhalten der Reihe nach eine Aufforderung zur Ausgabe, die vom Bildschirm auf den Drucker umgeleitet wird. Diese Operation ist aus folgendem Grund nicht identisch mit einer einfachen Bildschirm-Hardcopy: Eine Maske kann wesentlich größer sein als eine Druckseite (auf dem Schirm ist dann nur ein Ausschnitt sichtbar). Das Problem läßt sich durch eine Seitenaufteilung lösen, wie sie bei Tabellenkalkulationsprogrammen anzutreffen ist. Die Aufteilung muß von der Master-Routine gesteuert werden.

Kapitel 17

Graphik

Genau wie bei den Benutzeroberflächen ist die Vielzahl graphischer Systeme kaum noch überschaubar. Aber auch hier engt sich das Gesichtsfeld schnell ein, wenn der Einsatz im wissenschaflichen Bereich gefragt ist: Die meisten marktgängigen Systeme sind Elementargraphikpakete für die Durchführung von Elementaroperationen (Zeichnen von Strecken, Punkten, Flächen etc.) [BB86] [MW886] [TCP91]. Insbesondere umfaßt jede graphische Oberfläche eine Anwenderschnittstelle für die graphischen Primitive (API = "Application Programmers Interface" genannt). Andere Systeme unterstützen stark den Bereich der Illustration und Animation (Bewegen von Elementen, Zeichnen, Verändern, Colorieren etc.). Eine dritte Gruppe dient schließlich zur Datenpräsentation, jedoch können diese Programme in der Regel nicht aus einer Programmiersprache heraus angesprochen und so in die eigene Software integriert werden. Auf dem PC fehlen Programmodule zum Zeichnen in wissenschaftlichen Koordinatensystemen fast gänzlich!

17.1 Anforderungen an ein Graphiksystem für die Datenanalyse

17.1.1 Erfahrungen aus der Anwendungspraxis

In der Abteilung Theoretische Biologie an der Universität Bonn bestand die Forderung nach einem geeigneten Graphikpaket schon bei der Gründung im Jahre 1986: Die Studenten sollten am Computer eine Programmiersprache (Pascal) erlernen und zugleich die Modellierung und Simulation biologischer Systeme üben. Der hochgradig interaktive Charakter dieses Lernprozesses erfordert ein leicht zu bedienendes Graphik-Paket, das es ermöglicht, die laufende Simulation graphisch auf dem Schirm zu verfolgen: Für das Verständnis der Dynamik eines Systems ist die dynamische Visualisierung der Resultate das adäquate didaktische Hilfsmittel!

Da zu diesem Zeitpunkt auf dem PC kein brauchbares System existierte, wurde eine Eigenentwicklung betrieben. Die Turbo-Pascal-Bibliothek *MathGraph* (= "Mathematisch/Naturwissenschaftliches Graphikpaket) hat seither mehrere Generationen durchlaufen und verfügt inzwischen über weit ausgereifte Möglichkeiten, z.B. beliebig viele frei positionierbare Koordinatensysteme, Mehrfachachsen mit Beschriftungen, Kurven beliebiger Farbe und Strichelung, Mausunterstützung, einfache Menüs und ein Metafile-System mit Ausgabemöglichkeiten auf einen Plotter oder PostScript-Drucker. Trotzdem wurde der Grundsatz der einfachen Bedienbarkeit durch die Studenten gewahrt, was inzwischen zu einer Verbreitung über Bonn hinaus geführt hat (Jülich, Dortmund, Heidelberg, Tübingen, Aachen, Karlsruhe).

Auf der anderen Seite hat eben diese Einfachheit zu der Entscheidung geführt, das Paket nicht in die Sprache C zu übertragen, sondern ein flexibleres Programm neu zu entwickeln, welches damit zwangsläufig eine größere Komplexität hat und daher mehr Sorgfalt bei der Programmierung erfordert. Da C-Programmierer im allgemeinen keine Anfänger mehr sind, sollte die höhere Komplexität kein Problem darstellen. Die Schwerpunkte bei der Neuentwicklung wurden vor allem auf ein komfortables Koordinatenlayout und die Portierbarkeit zwischen verschiedenen Compilern und Betriebssystemen gelegt. In die Konzeption von *OnLiGraph* (= "On-Line-Graphik") fließen die mit MathGraph gemachten Erfahrungen, aber auch Anregungen von professionellen Systemen [MW886] [GM89] ein. Insbesondere ist an

eine spätere Anbindung an das Graphische Kern-System (GKS [BB86]) gedacht [1].

17.1.2 Ein Anforderungskatalog

Der folgende Anforderungskatalog bezieht sich vor allem auf das Problemfeld der Simulation und Datenanalyse:

On-Line-Graphik: Das System muß in der Lage sein, mehrere Kurven gleichzeitig auf dem Schirm zu zeichnen und zwar sukzessive mit dem Eintreffen der Werte. Die meisten Systeme sehen in der Tat nur das Zeichnen einer einzigen Kurve bei Vorliegen des gesamten Datensatzes vor (Polylines in GKS [BB86]). Zum gleichzeitigen Zeichnen von Linien benötigt man einen Linienfortsetzungsbefehl vom Typ : "Verlängere die Kurve K nach (x, y)"

Obwohl diese "On-Line-Forderung" banal klingt, hat sie doch weitreichende Konsequenzen für die Strukturkonzeption des Graphikpakets. Bei dem Versuch, eine On-Line-Graphik mit einem verfügbaren System, z.B. dem Borland Graphics Interface BGI oder dem Graphics Kernel System GKS), zu realisieren, treten prinzipielle Schwierigkeiten auf (mehr dazu in 17.8).

Koordinatensysteme: Das Zeichnen muß mit Hilfe der Koordinaten erfolgen, die auch an den Achsen aufgetragen sind. Es handelt sich dabei um Dezimalzahlen und nicht um ganze Zahlen. Insbesondere muß ein Abschneiden der Linien bei Überschreitung der Ränder erfolgen (Clipping). Bei dem Versuch, diese Bedingung zu erfüllen, sind selbst bei professionellen Systemen Fehler gemacht worden (vgl. hierzu 17.8).

Layout: Beim Layout von Koordinatensystemen muß größtmögliche Freiheit bestehen. Der Benutzer soll die Achsenaufteilung, Zahlenformate und Beschriftungen frei wählen können und auch über kompliziertere Anordnungen wie logarithmische Skalen, gekreuzte Koordinatenachsen oder Polarkoordinaten verfügen können. Marktgängige Systeme bieten meist nur einen Satz von festen Koordinatenlayouts an, unter denen der Benutzer das am besten geeignete auszuwählen hat.

Platzbedarfsberechnung: Mit dem Koordinatenlayout einher geht das Problem der Platzbedarfsberechnung: Der Benutzer sollte lediglich den Bereich auf dem Schirm bestimmen, den das Bild einnehmen soll. Alle Berechnungen im Zusammenhang mit dem Platzbedarf der Anordnung müssen dann vom System übernommen werden. Dieses Problem ist bei einem flexiblen Layout keineswegs trivial!

Darstellungsmöglichkeiten: Eine Vielzahl von Möglichkeiten zur Darstellung und Markierung von Daten muß zur Verfügung stehen. Dazu gehören zunächst die verschiedenen Punktmarkierungen, Linientypen und -farben, aber auch Balkendarstellungen oder Fehlergrenzen.

Geräteunabhängigkeit: Das System sollte weitgehend geräteunabhängig arbeiten. Dazu ist es erforderlich, die wirkliche Punktauflösung des verwendeten Gerätes vor dem Benutzer zu verbergen und stattdessen ein für alle Geräte einheitliches Koordinatenfenster über die Zeichenfläche zu legen (virtuelle Koordinaten). Die Umrechnung in die Koordinaten des Ausgabegerätes (Bildschirme verschiedener Auflösung, Plotter, Matrixdrucker, PostScript-Drucker etc.) übernimmt ein Gerätetreiber.

Portierbarkeit Die Zusammenfassung aller geräteabhängigen Funktionen in einen Treiber erleichtert eine Portierung in andere Umgebungen, da nur der Treiber neu geschrieben werden muß.

Hardcopies und Meta-Files: Der Benutzer muß die Möglichkeit haben, Hardcopies und Metafiles zu erstellen. Hardcopies sind Eins-zu-Eins-Kopien der Bildschirmpixel auf einen Drucker (siehe z.B. Abb. 10.2). In einem Meta-File werden dagegen alle Zeichenbefehle, die zur Entstehung eines Bildes geführt haben, protokolliert. Auf dieser Basis kann das Bild in ein anderes Vektor-Graphikformat (HPGL, PostScript, etc.) konvertiert werden. Diese Anforderungen entstehen zwangsläufig dann, wenn der Benutzer fragt, wie er nun die programmierte Graphik in eine Publikation übernehmen kann.

[1] GKS ist inzwischen auch für den PC erhältlich [GKS90], stand aber im Rahmen dieser Arbeit aus Kostengründen nicht zur Verfügung.

Anforderungen, die nicht an das Graphik-Paket gestellt werde (Geschäftsgraphik, 3D-Graphik etc.) wurden bereits in Abschnitt 13.3) zusammengestellt. Die konkrete Lösung der oben formulierten Probleme in OnLiGraph wird im folgenden beschrieben:

17.2 Offene und portable Konzeption

Im Gegensatz zu MathGraph ist OnLiGraph ein offenes System, d.h. dem Benutzer kann auf alle Daten zugreifen, die auch auch dem System zur Verfügung stehen. Die Realisierung der graphischen Grundobjekte folgt dabei weitgehend dem objektorientierten Paradigma. Eine gewisse Anlehnung an die Konzeption des MetaWindow-Pakets [MW886] ist dabei nicht zu verleugnen [2].

OnLiGraph setzt ein vorhandenes Elementargraphik-Paket für die Bildschirmausgabe voraus. Derzeit gibt es Versionen für:

- die Borland BGI-Graphik (Turbo C),
- die MicroSoft C Graphik,
- das MetaWindow-System (hier sind nur wenige Anpassungen erforderlich).
- das GKS-System, welches allerdings auf einem wesentlich höheren Niveau aufsetzt.

Auf der Grundlage der Elementargraphik ist ein in seinen Funktionen genau spezifizierter geräteabhängiger Graphik-Treiber zu programmieren [3]. Im wesentlichen umfaßt dieser die Initialisierungsoperationen für das Gerät (Graphik-Modus) und die elementaren Zeichenoperationen (Punkte, Linien, Text, Balken). Alle Operationen müssen auf der Basis virtueller Koordinaten (siehe 17.3) arbeiten. In einer C-Realisierung besteht der Treiber in der Regel lediglich aus einer Reihe von Präprozessor-Anweisungen, welche die Korrespondenz zwischen den OnLiGraph-Funktionsnamen und den Funktionsnamen des Elementargraphik-Pakets herstellen (*Generic File*). Eine schnelle Portierbarkeit ist damit gegeben.

17.3 Koordinaten

17.3.1 Koordinatenebenen

OnLiGraph kennt drei Koordinatenebenen [BB86] (vgl. auch Abb. 17.1):

Gerätekoordinaten, die den realen Koordinatenbereich des Ausgabegerätes charakterisieren. Bei Bildschirmen ist dies die horizontale und vertikale Zahl der Pixel. Gerätekoordinaten sind rein interne Größen für den Graphiktreiber.

Virtuelle Koordinaten ermöglichen die geräteunabhängige Positionierung von Objekten auf der Zeichenfläche. Der bei OnLiGraph zugrunde liegende virtuelle Koordinatenbereich ist standardmäßig (gannzzahlig) $[0, 10000] \times [0, 10000]$.

Weltkoordinaten sind schließlich die reellen Koordinaten, die innerhalb eines benutzerdefinierten Koordinatensystems gültig sind. Dieser Koordinatenbereich ist frei wählbar.

17.3.2 Koordinatentransformationen

Zu transformieren sind:

- Weltkoordinaten in virtuelle Koordinaten:

$$\begin{aligned} x_{virt} &= x_{welt} \,/\, m_x + b_x \\ y_{virt} &= y_{welt} \,/\, m_y + b_y \end{aligned}$$

[2] Die offene Konzeption ist im Gegensatz dazu im GKS-System leider nicht ganz verwirklicht worden. Der Mangel an Datenstrukturen weist auf die FORTRAN-Vergangenheit dieses Standards hin.

[3] Vgl. hierzu den Computer Graphics Interface-Standard (CGI) [GM89] oder das BGI-Graphiktreiber-Konzept [TCP90].

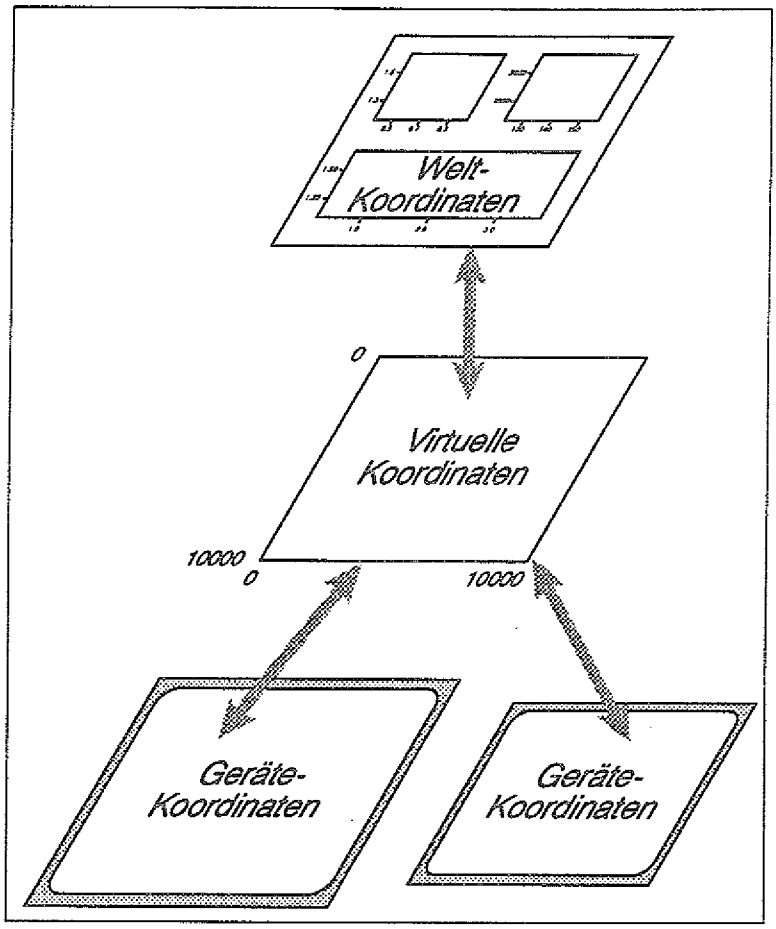


Abbildung 17.1: Koordinatenebenen

- Virtuelle Koordinaten in Gerätekoordinaten:

$$\begin{aligned} x_{device} &= x_{virt} \text{ div } M_x + B_x \\ y_{device} &= y_{virt} \text{ div } M_y + B_x \end{aligned}$$

(alle Werte sind ganzzahlig, div bezeichnet die ganzzahlige Division). Der Vorteil der ganzzahligen Transformation besteht in der hohen Rechengeschwindigkeit. Einige Fehlerquellen, die damit verbunden sind, werden in 17.8 beschrieben.

17.3.3 Aspekt, Rotation und Scherung

Zusätzlich zu den Umrechnungskonstanten wird noch der Aspekt benötigt, d.h. das Verhältnis des physikalischen Abstands zweier vertikal benachbarter Pixel zum Abstand zweier horizontal benachbarter Pixel. Erst mit dieser Information ist man in der Lage, Objekte unter Erhaltung der Größenverhältnisse und Winkel zu rotieren. Der Aspekt wird aus Geschwindigkeitsgründen in Form eines ganzzahligen Quotienten

$$\frac{x_{asp}}{y_{asp}} \quad (\geq 1 \text{ bei üblichen Bildschirmen})$$

ausgedrückt. Das Rotieren und Scheren bzgl. virtueller Koordinaten kann nicht einfach über die üblichen Matrizen

$$\begin{pmatrix} \cos\phi & -\sin\phi \\ \sin\phi & \cos\phi \end{pmatrix} \text{ und } \begin{pmatrix} 1 & \tan\alpha \\ 0 & 1 \end{pmatrix}$$

erfolgen, sondern muß zusätzlich den Aspekt berücksichtigen. Mathematisch formuliert muß die Norm

$$\|(x,y)\|_{(x_{asp},y_{asp})} \stackrel{\text{def}}{=} \sqrt{\frac{x^2}{x_{asp}^2} + \frac{y^2}{y_{asp}^2}}$$

bei der Rotation und in der vertikalen Richtung der Scherung erhalten bleiben. Daraus bestimmt man die korrekten Matrizen als

$$\begin{pmatrix} \cos\phi & -\frac{y_{asp}}{x_{asp}} \cdot \sin\phi \\ \frac{x_{asp}}{y_{asp}} \cdot \sin\phi & \cos\phi \end{pmatrix} \quad \text{und} \quad \begin{pmatrix} 1 & \frac{y_{asp}}{x_{asp}} \cdot \tan\alpha \\ 0 & 1 \end{pmatrix}.$$

Die korrekte Behandlung des Aspekts ist Aufgabe des Graphik-Treibers.

17.3.4 Welten

Eine *Welt* umfaßt

- das Zeichenfenster auf dem Schirm, das in virtuellen Koordinaten angibt, wohin die Ausgabe erfolgt,
- das Weltkoordinatenfenster,
- einen Positionsmerker (*Cursor*) in Weltkoordinaten für die aktuelle Position im Weltkoordinatensystem,
- einen Flag, der angibt, ob das zuletzt gezeichnete Objekt den Fensterbereich überschritten hat und daher abgeschnitten werden mußte (vgl. 17.8.1),
- die Koeffizienten der Koordinatentransformation in virtuelle Koordinaten.

Welten ist als Datenstrukturen realisiert und bilden die Bezugsrahmen für das Zeichnen in Weltkoordinaten.

17.4 Graphische Grundelemente

Die graphischen Grundelemente sind wie folgt durch ihre Attribute charakterisiert (vgl. die graphischen Primitive in GKS [BB86]):

Linien durch

- Farbe,
- Breite,
- Strichelung.

Buchstaben durch

- Schrifttyp (*Font*),
- Breite des Buchstabens 'X',
- Höhe des Buchstabens 'X',
- Drehwinkel,
- Neigungswinkel,
- Abstand zwischen den Zeichen,
- Breite eines Leerzeichens,
- Abstand zwischen den Zeilen,
- Bündigkeit (rechts, links, zentriert, oben, unten etc.),

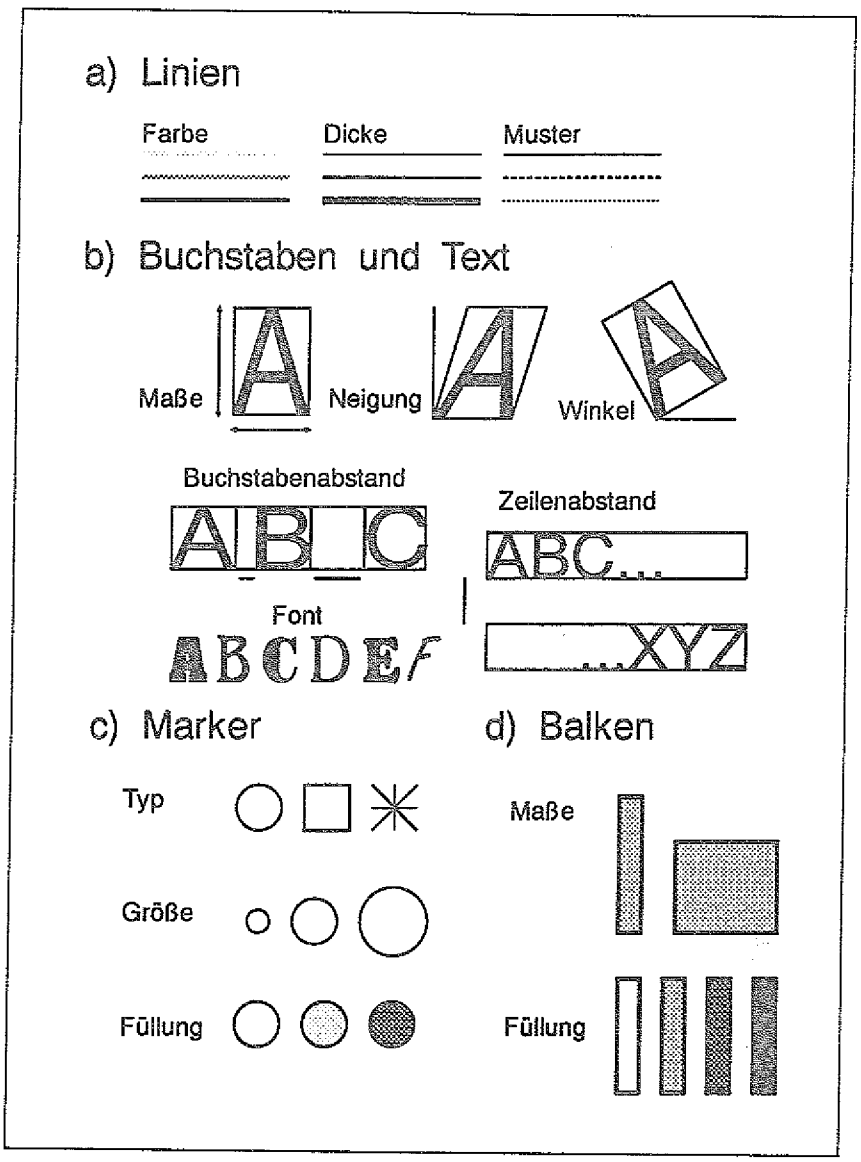


Abbildung 17.2: Graphische Grundelemente und ihre Attribute

- Proportionalschrift oder feste Buchstabenbreite,
- Unterstreichung.

Marker durch

- Typ (Kreuz, Stern, Kreis, etc.),
- Größe,
- Füllung.

Balken durch

- Höhe,
- Breite,
- Füllung.

Der Graphiktreiber muß Befehle zur Positionierung aller Grundelemente in der virtuellen Ebene enthalten. Dabei sind Buchstaben, Marker und Balken erst durch zusätzliche Angabe einer Linie mit Farbe, Strichmuster und Linienbreite vollständig charakterisiert.

17.4.1 Verwendung von Vektorfonts

Druckzeichen werden in OnLiGraph hauptsächlich durch Vektorfonts realisiert [Gre90], da diese eine beliebige Rotation, Scherung und Skalierung erlauben. Dieser Vorteil wiegt den hohen Zeitaufwand zum Zeichnen von Vektorzeichensätzen auf. Zur schnellen (horizontalen) Textausgabe steht jedoch gleichzeitig ein pixelbezogener Zeichensatz zur Verfügung.

Vektorfonts sind in vielen Elementargraphiksystemen bereits enthalten. Trotzdem ist in OnLiGraph auch eine portabler Standard implementiert. Er beruht auf den CHR-Vektorfonts der Firma Borland (vgl. auch [Gre90]), die hinsichtlich des Speicherplatzbedarfs zur Laufzeit leicht modifiziert wurden [4].

17.5 Zeichenstifte

Mit Hilfe von Zeichenstiften können die Grundobjekte bezogen auf virtuelle Koordinaten oder Welt-Koordinaten ausgegeben werden. Außerdem besitzen alle Zeichenstifte einen Positionsmerker (*Cursor*). Die Cursor sind der Schlüssel für die gleichzeitige Fortsetzbarkeit mehrerer Linien (On-Line-Bedingung). OnLiGraph kennt zwei Klassen von Zeichenstiften, wobei die zweite als Unterklasse der ersten realisiert ist (Vererbungskonzept):

17.5.1 Virtuelle Zeichenstifte

17.5.1.1 Beschreibungsdaten

Virtuelle Zeichenstifte enthalten:

- einen Meta-File-Zeiger. Falls dieser gesetzt ist, werden alle Ausgaben, die über den virtuellen Zeichenstift erfolgen, protokolliert,
- eine laufende Nummer, um die Zeichenstifte in einem Meta-File-Protokoll voneinander zu unterscheiden,
- einen virtuellen Cursor, welcher die aktuelle Position des Stiftes speichert.

17.5.1.2 Operationen

Mit virtuellen Zeichenstiften kann man nun die folgenden Zeichenoperationen durchführen:

- Positionieren,
- Pixel zeichnen,
- Linie zeichnen oder verlängern,
- Marker zeichnen,
- Rechtecke zeichnen,
- Balken zeichnen,
- Buchstaben und Text zeichnen.

Alle Funktionsaufrufe haben ein ähnliches Schema. Beispielsweise:

[4] Für die Überlassung der grundlegenden Software danke ich Herrn Prof. Dr. B. Goldschmidt aus Halle.

```
DrawLine   ( VirtPen, Line,          X1, Y1, X2, Y2    ) ;
DrawLineTo( VirtPen, Line,          X, Y              ) ;
DrawMarker( VirtPen, Line, Marker,  X, Y              ) ;
DrawBar    ( VirtPen, Line, Bar,    X, Y              ) ;
DrawText   ( VirtPen, Line, Font,   X, Y, "Text"      ) ;
```

In einer objektorientierten Umgebung reduzieren sich alle diese Funktionen zu einer einzigen überladenen Funktion Draw!

17.5.2 Welt-Zeichenstifte

17.5.2.1 Attribute

Ein Welt-Zeichenstift erbt alle Eigenschaften und Methoden eines virtuellen Zeichenstiftes und erhält dazu noch ein Weltkoordinatensystem. Man beachte, daß das Weltkoordinatensystem einen zusätzlichen Cursor enthält. Der virtuelle und der Welt-Cursor können durchaus differieren, nämlich dann, wenn die letzte Welt-Zeichenoperation zu einem Abschneiden der Linie geführt hat. In diesem Fall ist der Clip-Flag der Welt gesetzt.

17.5.2.2 Operationen

Die Operationen, die mit Welt-Zeichenstiften ausgeführt werden können, sind identisch mit denen von virtuellen Zeichenstiften. Einziger Unterschied ist der Typ der Koordinaten. Die universelle Funktion Draw kann somit noch weiter überladen werden! Die Operationen werden ausgeführt, indem die Weltkoordinaten in virtuelle Koordinaten transformiert und dann die entsprechende Funktion für virtuelle Zeichenstifte aufgerufe wird.

17.6 Koordinatensysteme

Das flexible Layout von Koordinatensystemen erweist sich als ein schwieriges Kapitel der Graphik-Programmierung. Die wichtigsten Grundregeln für die graphische Präsentation von Daten sind in Cleveland's "The Elements of Graphing Data" [Cle85] hervorragend zusammengefaßt. Es wurde versucht, alle in diesem Buch vorgestellten Methoden auch in OnLiGraph zu realisieren. Außerdem wurden graphische Darstellungen, die von den Programmen Turbo Graphics Toolbox, Lotus 1-2-3, PlotIt!, MS Excel, MS Chart und Harvard Graphics erzeugt wurden, gesichtet. Es kann gesagt werden, daß diese Layouts im wesentlichen mit OnLiGraph nachvollzogen werden können!

17.6.1 Grundelemente eines Koordinatensystems

Um dem Benutzer beim Aufbau seiner Koordinatensysteme völlige Freiheit einzuräumen, wurde ein Aufbau aus frei konfigurierbaren Grundelementen konzipiert. Die Abbildung 17.3 illustiert dieses Vorgehen. Die Grundelemente sind:

X- und Y-Achsen mit Strichunterteilung und den Attributen:

- Zahl der großen und kleinen Unterteilungsstriche,
- Lage des ersten großen Unterteilungsstrichs,
- Art der Unterteilung (z.B. äquidistant oder logarithmisch),
- Ausführung von Strichen an den Achsenrändern.

X- und Y-Skalenbeschriftungen mit den Attributen:

- Zahlennotation,
- Vor- und Nachkommastellenzahl,

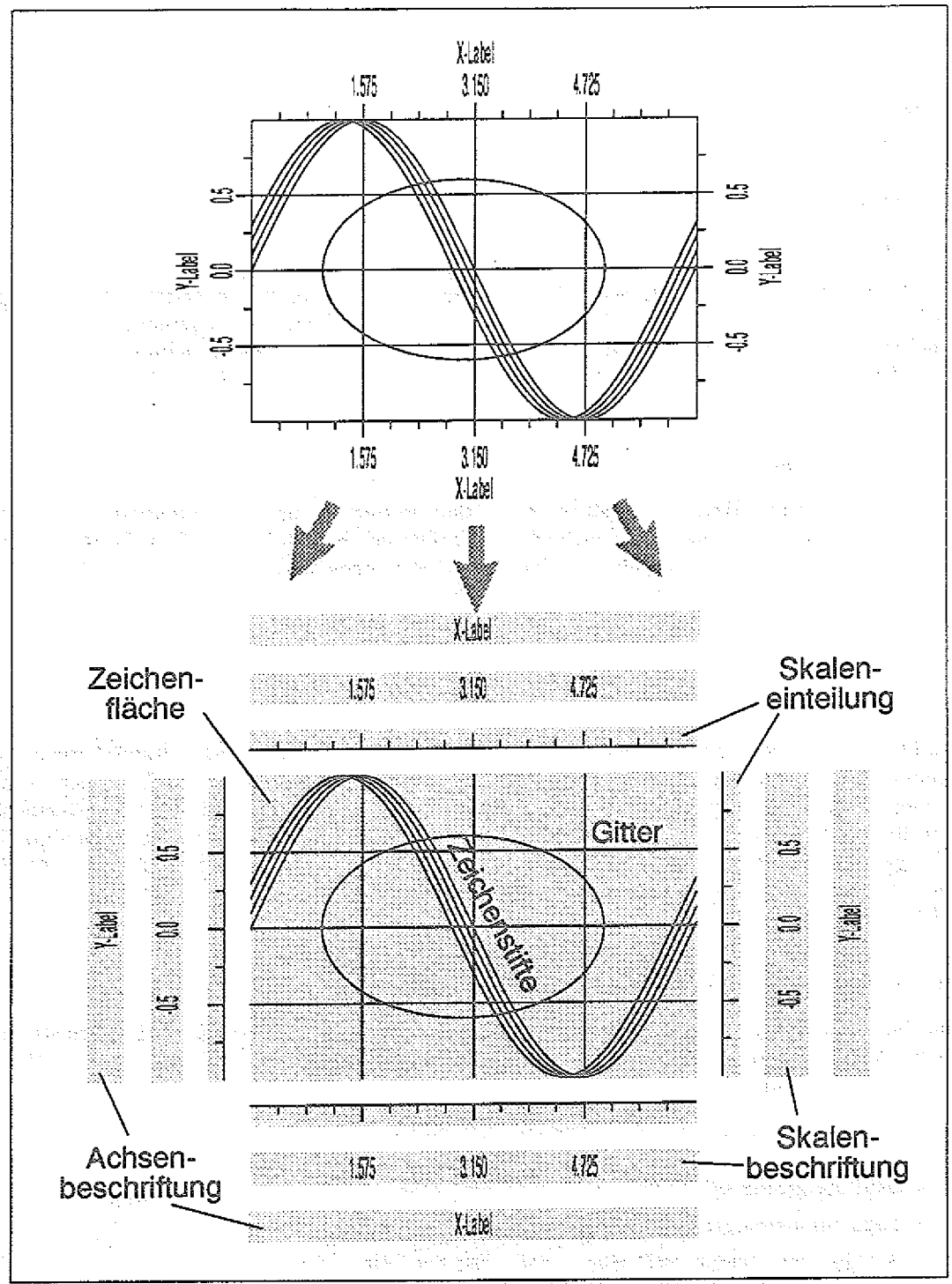


Abbildung 17.3: Aufbau von Koordinatensystemen ("Explosionszeichnung")

- Koordinatenbereich,
- Font für die Ziffern.

X- und Y-Achsenbeschriftung mit den Attributen:

- Beschriftungstext,
- physikalische Einheit,
- Font für den Text.

Aufteilung der Zeichenfläche mit den Attributen

- X- und Y-Gitterlinien, entsprechend den großen Skalenstrichen,
- X- und Y-Null-Linien, falls 0.0 im jeweiligen Koordinatenbereich liegt,
- Rahmen um das Gesamtsystem,
- Farben der Zeichenfläche und des äußeren Rahmens.

17.6.2 Formulierung des Koordinatenlayouts

Zur Formulierung eines Koordinatenlayouts wurde — analog zum Maskengenerator — ein deklarativer Stil imitiert [5]. Die zu einem Koordinatensystem gehörigen Grundelemente werden — von innen nach außen — in einer Prozedur aufgelistet. Das kann für ein Koordinatensystem mit zwei zugehörigen Zeichenstiften und Beschriftungen auf allen vier Seiten z.B. so aussehen:

```
procedure CoordDef ;
{ Definition eines Koordinatensystems }

begin
   { Bildschirmbereich, Farben und Rahmen }
   CoordWindow ( 1000 , 9000 , 1000 , 9000 ) ;
   CoordColor    ( blau , rot ) ;
   CoordFrame    ;

   { Erster Zeichenstift mit zugehörigen Koordinatenachsen }
   CoordPen ( Pen1 ) ;

      { X-Achse, Gitter, Skala und Beschriftung zu Pen1 }
      CoordXAxis    ( Unten , 200 , 5 , 3 , 100 ) ;
      CoordXGrid    ;
      CoordXScale   ( Unten , GothicFont , 5 , 3 , 100 ) ;
      CoordXLabel   ( Unten , GothicFont , "X-Label" , 100 ) ;

      { Y-Achse, Gitter, Skala und Beschriftung zu Pen1 }
      CoordYAxis    ( Links , 200 , 5 , 1 , 100 ) ;
      CoordYGrid    ;
      CoordYScale   ( Links , GothicFont , 5 , 1 , 100 ) ;
      CoordYLabel   ( Links , GothicFont , "Y-Label" , 100 ) ;

   { Zweiter Zeichenstift mit zugehörigen Koordinatenachsen }
   CoordPen ( Pen2 ) ;

      { X-Achse, Gitter, Skala und Beschriftung zu Pen2 }
      CoordXAxis    ( Oben , 200 , 5 , 3 , 100 ) ;
```

[5] Die Reihenfolge der Deklarationen ist hier allerdings nicht ganz frei.

```
        CoordXScale    ( Oben , GothicFont , 5 , 3 , 100 ) ;
        CoordXLabel    ( Oben , GothicFont , "X-Label" , 100 ) ;

        { Y-Achse, Gitter, Skala und Beschriftung zu Pen2 }
        CoordYAxis     ( Rechts , 200 , 5 , 1 , 100 ) ;
        CoordYScale    ( Rechts , GothicFont , 1 , 100 ) ;
        CoordYLabel    ( Rechts , GothicFont , "Y-Label" , 100 ) ;
end ;
```

Die Abbildung 17.3 zeigt das Resultat, zusammen mit einigen von Pen1 und Pen2 gezeichnetet Kurven.

17.6.3 Realisierung des Layouts

Der Realisierung des Layouts liegt — ganz ähnlich wie beim Maskengenerator — das Nachrichtenkonzept der Objektorientierten Programmierung zugrunde: Die vom Benutzer definierte Layout-Prozedur wird mehrmals von einer Master-Routine aufgerufen, wobei jeweils eine andere Aufgabe erledigt wird. Insgesamt finden die folgenden vier Durchläufe statt, mit denen jeweils eine Nachricht an die einzelnen Grundelemente verbunden ist:

1. Fenster-Abmessung: Hiervon wird einzig die Prozedur CoordWindow angesprochen, die daraufhin in einer globalen Variablen die äußeren Fensterabmessungen ablegt.

2. Layout-Berechnung: Jedes Grundelement, muß nun seinen Flächenbedarf vom äußeren Fenster abziehen, so daß die eigentliche Zeichenfläche nach Abzug aller Beschriftungen und Achsen übrigbleibt. Auf diese Nachricht reagieren alle Prozeduren, die einen Platzbedarf haben (CoordAxis, CoordScale, CoordLabel).

3. Koordinatentransformation: Nachdem nun die Abmessungen der Zeichenfläche bekannt sind, können die Transformationskonstanten für die Umrechnung von Weltkoordinaten in virtuelle Koordinaten für alle beteiligten Zeichenstifte berechnet werden. Dies betrifft ausschließlich die CoordPen-Prozedur.

4. Zeichnen: Zum Abschluß zeichnet jede Prozedur, die etwas zu zeichnen hat, ihr Grundelement auf den Schirm, und zwar in der Folge von innen nach außen angeordnet.

17.6.4 Mögliche Erweiterungen

- Das Nachrichten-System ist leicht so weit ausbaubar, daß ein interaktives Layout möglich wird: Dabei wählt der Benutzer die Grundelemente mit einem graphischen Zeiger an woraufhin die zugehörigen Konfigurationsdaten auf dem Bildschirm erscheinen und geänder werden können [6].

- Gerade bei der automatischen Berechnung von Intervallgrenzen ergeben sich oft unbrauchbare Zahlenwerte (z.B. $[-123.673, 33.729]$). Durch Einführung einer weiteren Prozedur CoordAdjust kann der Bereich mit wählbarer Präzision auf vernünftige Grenzen erweitert werden.

  Beispiel: $[-123.673, 33.729] \rightarrow [-150, 50.0]$

- Zuweilen sollen Weltkoordinatenpunkte exakt auf Pixel-Punkte des Ausgabegerätes abgebildet werden. Z.B. sind die Niveauliniendiagramme in den Abbildungen 10.2 und 10.4 auf diese Weise entstanden. Eine weitere Prozedur (CoordWindow) kann daher die Zeichenfläche auf Abmessungen bringen, bei denen die sich die Pixelabstände des Ausgabegerätes mit vorgebbaren Koordinatenintervallen decken.

[6] Dieser Ansatz wurde auch im Rahmen des CIF-Projektes am Institut für Technische Chremie der Universität Hannover [CIF89] teilweise verwirklicht.

17.7 Meta-Files und Hard-Copies

17.7.1 Meta-Files

17.7.1.1 Realisierung

Die Realisierung eines Meta-Files ist verhältnismäßig einfach, weil alle Zeichenbefehle letztlich über die virtuellen Zeichenstifte ablaufen: Bei Benutzung eines solchen Stiftes protokolliert OnLiGraph automatisch einen entsprechenden Eintrag in den zugehörigen Meta-File. Ein Meta-File-Eintrag hat das Format

⟨Operations-Code⟩⟨Zeichenstift Nr.⟩⟨Zahl der Argumente⟩⟨Argument 1⟩⟨Argument 2⟩...,

analog zum GKS-Metafile-Format [GM89] und zum X-Windows-Protokoll [SG86]. Das gewählte Format kann — aufgrund der zusätzlich abgespeicherten Zahl der Argumente — später um neue Codes erweitert werden, ohne die Versionskompatibilität zu verlieren.

17.7.2 Konvertierung in andere Graphikformate

Konverter für die folgenden Graphikformate sind geplant oder bereits implementiert (vgl. [Poh90] [GM89]):

PS : PostScript (für einen PostScript-Drucker),

AI : Adobe Illustrator Format (AI) (für PC-Illustrationsprogramme),

DXF : Data Exchange Format (für CAD-Systeme),

HPGL : Hewlett Packard Graphics Language (für einen Plotter),

SCR : Screen Format (für die Bildschirm-Vorschau),

TXT : Textformat (zu Prüfzwecken).

Einen Eindruck von den Möglichkeiten, die sich dadurch für Publikationen bieten, geben die Abbildungen in dieser Arbeit.

17.7.3 Hard-Copies

17.7.3.1 Realisierung und Format

Eine Bildschirm-Hard-Copy in eine Datei wird auf dem PC nach Drücken der Shift-PrintScreen-Tastenkombination oder durch die Software ausgelöst. Dabei werden die Bildpunkte zeilenweise in das Meta-File-Format übertragen. Auf diese Weise können in Meta-Files Vektor- und Pixelgraphiken gemischt werden. Die Farben sind geräteabhängig codiert.

17.7.4 Konvertierung in andere Graphikformate

Zunächst wird nur PC-spezifische Hardware (EGA/VGA-Karte) unterstützt. Die Konvertierung kann erfolgen in die Formate

PCX : MicroSoft Paintbrush (für PC-Malprogramme),

TIFF : Tag Image File (für Desktop-Publishing-Systeme),

PCL : Printer Control Language (für Laserdrucker).

17.8 Fallstricke

Es gibt einige grundlegende Fehler, die bei der Programmierung einer On-Line-Graphik gemacht werden können und deren Beseitigung oft eine völlige Neukonzeption erfordert. Diese Fehlerquellen sollen kurz beschrieben werden:

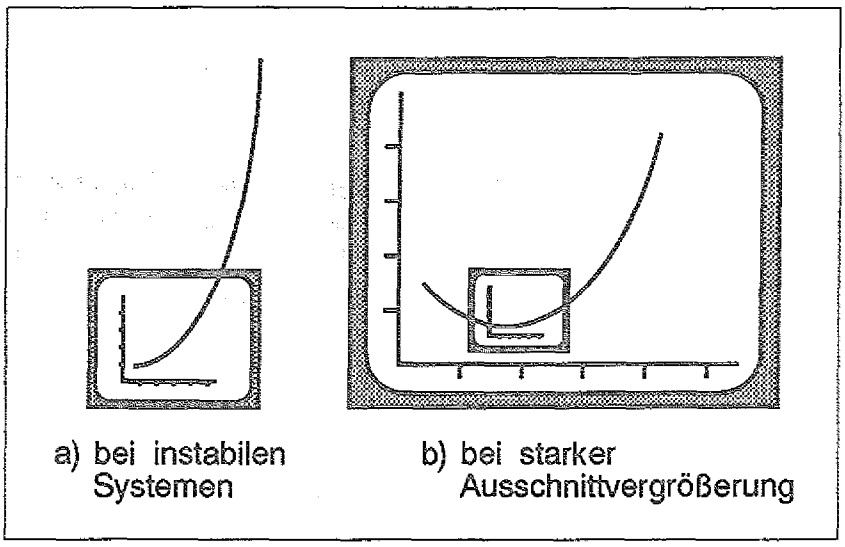


Abbildung 17.4: Ganzzahliger Überlauf bei der Koordinatentransformation

17.8.1 Clipping

Mit *Clipping* bezeichnet man das Abschneiden von graphischen Elementen am Rand eines Bildschirmfensters. Ein Clip-Algorithmus auf der Basis von virtuellen oder Geräte-Koordinaten ist bei fast allen Elementargraphik-Paketen eingebaut und arbeitet sehr effizient. Folgender Algorithmus für das Zeichnen der Linie

$$(x_1, y_1) - (x_2, y_2) \text{ in Weltkoordinaten}$$

ist naheliegend:

1. $(x_1, y_1) - (x_2, y_2)$ in virtuelle Koordinaten umrechnen.
2. Am (virtuellen) Fensterrand abschneiden.
3. Linie in virtuellen Koordinaten zeichnen.

Diese Implementierung ist leider fehlerhaft, denn bei einer On-Line-Graphik muß stets mit folgenden Situationen gerechnet werden (vgl. Abb. 17.4) :

- Ein simuliertes System erweist sich als instabil. Die Kurven streben im Zeitverlauf gegen unendlich.
- Ein gezeichnetes Bild ragt um mehrere Größenordnungen über die Clip-Grenzen des Bildschirms hinaus, wenn es sich um einen Ausschnitt eines viel größeren Bildes handelt (z.B. die mehrfache Vergrößerung der Umgebung eines stationären Punktes im Phasenraum).

Die Anwendung des obigen Algorithmus scheitert in diesen Fällen an der Koordinatenumrechnung: Es kommt zu einem ganzzahligen Überlauf, da virtuelle Koordinaten nur etwa 3 Zehnerpotenzen umfassen.

Der Fehler könnte durch eine einfache Fehlerabfrage (z.B. mit Hilfe der matherr-Routine in C) abgefangen und die entsprechende Linie einfach nicht gezeichnet werden. Aber auch diese Lösung ist nicht korrekt, denn beispielsweise läuft die Linie

$$(-10^6, -10^6) - (10^6, 10^6) \text{ durchaus durch das Fenster } [0,1] \times [0,1] \text{ !}$$

Ersetzt man stattdessen die überlaufenden Werte einfach durch den maximal darstellbaren ganzzahligen Wert, so können falsche Linien entstehen, die den Benutzer irritieren. Das Beispiel der Linie

$$(-0.5 \cdot 10^6, -10^6) - (0.5 \cdot 10^6, 10^6) \text{ im Fenster } [0,1] \times [0,1] \text{ !}$$

produziert diesen Fehler.
Fazit: Die einzig mögliche Lösung besteht darin, das Clipping auf der Ebene der Weltkoordinaten durchzuführen, d.h. wie folgt vorzugehen:

1. Am Weltkoordinaten-Fensterrand abschneiden. Dies erfordert zeitaufwendige Floating-Point-Arithmetik.
2. In virtuelle Koordinaten umrechnen
3. Linie in virtuellen Koordinaten zeichnen.

Wahlweise sollte dem Benutzer ermöglicht werden — auf eigene Gefahr — das Clipping abzuschalten.

17.8.2 Virtuelle Koordinaten

Es stellt sich die Frage, ob die rechenzeitaufwendige Zwischenstufe der virtuellen Koordinaten zwischen der Weltkoordinatenebene und der Gerätekoordinaten-Ebene wirklich notwendig ist (vgl. Abb. 17.1). Die Geräteunabhängigkeit virtueller Koordinaten ist dabei kein Grund, die virtuelle Ebene in Form von Datentypen und Operationen auch explizit zu realisieren. Schließlich könnten Weltkoordinaten auch direkt in Gerätekoordinaten umgerechnet werden.

Der entscheidende Aspekt kommt hinzu, wenn der Wunsch besteht, Meta-Files zu erzeugen. Nun könnte auch der Meta-File lediglich die Zeichenbefehle in Gerätekoordinaten aufnehmen, und das Ausgabegerät entsprechend umskaliert werden. Wenn man das Ergebnis auf einem hochauflösenden Ausgabegerät darstellt, so ist das Ergebnis enttäuschend: Die Kurven erhalten bei diesem Verfahren Ecken! Der Grund liegt darin, daß Pixelkoordinaten ein Bild immer nur in der Auflösung des Bildschirms reproduzieren, wogegen Plotter und Laserdrucker um den Faktor 3 - 10 höher auflösen können.

Folglich müssen virtuelle Koordinaten eine möglichst hohe Auflösung gewährleisten. Viele Systeme verwenden deshalb den maximal darstellbaren ganzzahligen Zahlbereich oder sogar Dezimalzahlen [BB86]. Dezimalzahlen erfordern jedoch einen hohen Rechenzeit- und Speicherplatzaufwand.

Es gibt einen weiteren Grund dafür, daß der virtuelle Koordinatenbereich möglichst groß zu wählen ist: Die Transformation in Gerätekoordinaten wird mit einer schnellen ganzzahligen Division realisiert [MW886] (vgl. 17.3). Dabei gehen zwangsläufig einige Pixel verloren, wie folgendes Beispiel zeigt:

Der virtuelle Koordinatenbereich $[0,1000]$ kann auf die VGA-Auflösung mit 480 vertikalen Punkten nur mit dem (ganzzahligen) Faktor 3 abgebildet werden. Erreicht werden damit

$$1000 \text{ div } 3 = 333 \text{ Pixel.} \quad \text{Der Verlust beträgt } 480 - 333 = 147 \text{ Pixel !}$$

Bei dem Bereich $[0,10000]$ ist dies der Faktor 21. Erreicht werden

$$10000 \text{ div } 21 = 476 \text{ Pixel.} \quad \text{Der Verlust beträgt nur 4 Pixel.}$$

17.8.3 Metafile-Formate

Beim Erstellen eines Meta-Files werden alle Zeichenanweisungen protokolliert. Es erscheint zunächst überflüssig, auch noch eine Information darüber abzulegen, welcher der vielen Zeichenstifte eine graphisches Element gezeichnet hat. In der Tat ist diese Information für eine exakte Reproduktion eines Bildes auf einem Bildschirm nicht erforderlich.

Das zusätzliche Anspeichern einer Kennnummer (1 Byte) für jeden Zeichenstift sollte trotzdem nicht vergessen werden, denn diese Information erspart viel Arbeit beim Entwurf eines Konvertierprogramms in andere Grapikformate. Zwei Beispiele:

- Im Gegensatz zu einem Bildschirm ist ein Plotter nicht in der Lage, on-line mehrere Linien in verschiedenen Farben zu zeichnen, weil dazu ein Wechsel des Zeichenstifts erforderlich wäre. Bei Übersetzung in das HPGL-Format müssen zunächst alle Linien, die zu einem Linienzug gehören, gesammelt werden, was mit Hilfe von Kennnummern ganz einfach ist.

- PostScript erlaubt das Zeichnen und sogar die Spline-Glättung zusammenhängender Linienzüge. Verwendet man diese Möglichkeit, so wird die Ausgabe um einiges kompakter.

Es muß hierbei stets bedacht werden, daß Metafiles einen hohen Speicherbedarf haben (ca. 5-40 KB). Die Translate werden oft noch um den Faktor 2-4 größer!

17.8.4 Ganzzahliges Rechnen

Beim Dividieren und Multiplizieren von ganzen Zahlen (virtuelle und Geräte-Koordinaten) ist daran zu denken, daß die Assoziativ- und Distributivgesetze nicht mehr gelten! Einige typische Fehler:

- Die Multiplikation

$$a \cdot \frac{b}{c}$$

ist stets in der Form

$$(a \cdot b) \text{ div } c$$

durchzuführen.

Beispiel: $(5 \cdot 2) \text{ div } 3 = 3 \quad \text{aber} \quad 5 \cdot (2 \text{ div } 3) = 0$

Bei virtuellen Koordinaten verläßt man dabei oft den integer-Zahlbereich. Zwischenergebnisse müssen dann als long integer repräsentiert werden.

- Beim Zeichnen von symmetrischen Markierungssymbolen geht bei der Koordinatentransformation von virtuellen auf Geräte-Koordinaten die Symmetrie verloren. Beispiel: Das Intervall

$$[\, x - \Delta x \;,\; x + \Delta x \,]$$

wird umgerechnet zu

$$[(x - \Delta x) \text{ div } M_x \;,\; (x + \Delta x) \text{ div } M_x \,].$$

Dieses Intervall ist im Gegensatz zu

$$[\, x \text{ div } M_x - \Delta x \text{ div } M_x \;,\; x \text{ div } M_x + \Delta x \text{ div } M_x \,]$$

im Allgemeinen nicht symmetrisch um x !

Der Effekt führt im ungünstigsten Fall dazu, daß sich die Linien eines Sterns auf der Geräteebene nicht mehr in einem Punkt schneiden, was den Benutzer sehr irritiert! Legt man Wert auf die Symmetrieerhaltung bei Symbolen, so muß auch die zweite (rechenaufwendigere) Variante der Koordinatenumrechnung implementiert werden.

17.8.5 Koordinatenlayout

Es wurde schon betont, daß ein fest vorgegebenes Koordinatenlayout auf Dauer nicht befriedigen kann. Die Entwicklung verläuft dann typischerweise so, daß so lange neue Layout-Varianten hinzugefügt werden, bis diese nicht mehr überblickt werden können. Erforderlich ist deshalb ein flexibles Konzept, das die Komposition beliebiger Koordinatensysteme aus Grundelementen erlaubt. Wie das dabei entstehende Problem der Platzbedarfsberechnung zu lösen ist, wurde in Abschnitt 17.6 beschrieben. Entscheidend ist die Zerlegung des Koordinatensystems in disjunkte rechteckige Bereiche (vgl. Abb. 17.3).

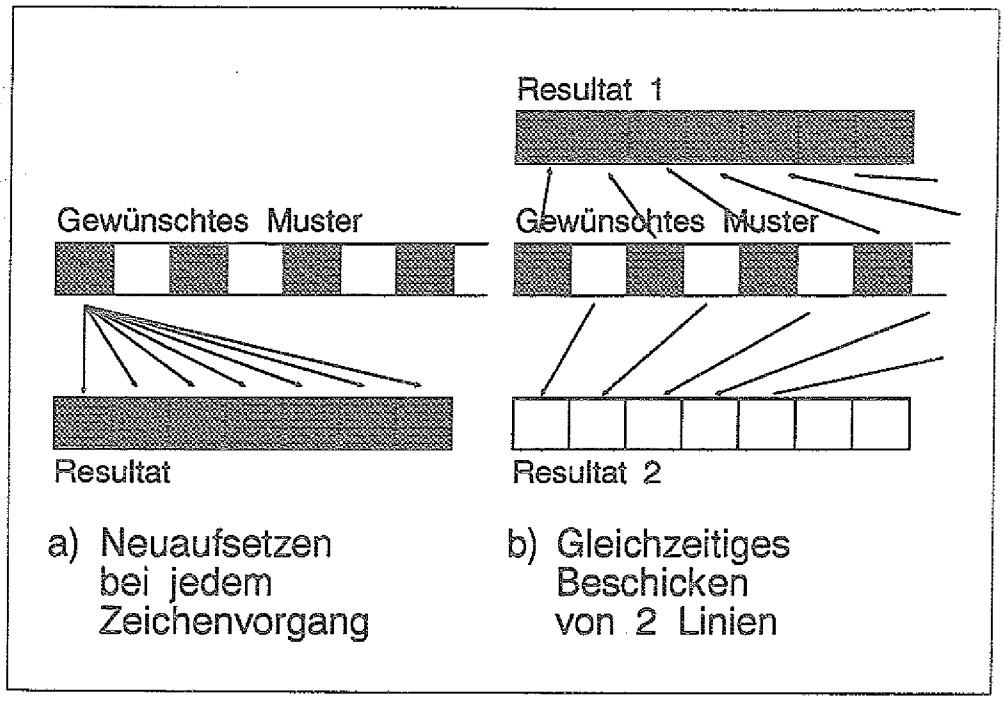


Abbildung 17.5: Interferenzmuster bei Verwendung von Linienmusterzählern

17.8.6 Cursor

Der für eine On-Line-Graphik charakteristische Befehl ist der Linienverlängerungsbefehl
DrawLineTo (Pen, Line, X, Y),
der eine Linie zu einem neuen Punkt hin verlängert. Sollen mehrere Linien gleichzeitig gezeichnet werden, so geschieht das mit einer Befehlssequenz der Form

```
begin
    :
  DrawLineTo ( Pen1, X1, Y1 ) ;
  DrawLineTo ( Pen2, X2, Y2 ) ;
  DrawLineTo ( Pen3, X3, Y3 ) ;
    :
end;
```

die in einer Schleife immer wieder durchlaufen wird. Ein solches Konstrukt ist nur mit Hilfe eines Cursors für jede Linie realisierbar. Viele Elementargraphikpakete sehen nur einen einzigen Liniencursor vor, so daß mehrfache Positionsmerker eigens implementiert werden müssen.

Es sollte erwähnt werden, daß bei Verwendung von Weltkoordinaten sogar je ein Cursor für die beiden Koordinatenebenen erforderlich ist. Dies hängt damit zusammen, daß beim Clipping (vgl. 17.8.1) der virtuelle Cursor aufgrund eines Überlaufs verloren werden kann.

17.8.7 Linienmusterzähler

Bei Verwendung von gemusterten Linien treten im On-Line-Einsatz oft sehr sonderbare Effekte auf, die mit dem internen Mechanismus der Mustererzeugung zusammenhängen: Ein Linienmuster wird durch zyklisches Abtasten einer (meist 16 bit langen) Bitfolge erzeugt. Dies kann auf zwei Arten geschehen:

a) Bei jedem Zeichenbefehl wird das Muster von vorn abgetastet. Die Folge: Setzt man Kurven aus kurzen Stücken zusammen (wie bei allen On-Line-Simulationen), so erscheinen wiederholt immer nur die ersten Bits des Muster auf dem Schirm. Im Extremfall von Stücken mit Länge 1 Pixel heißt das: Die "gemusterte" Linie erscheint entweder ohne Muster oder gar nicht (vgl. Abb. 17.5a)!

b) Das Muster wird nach jedem Zeichenbefehl zyklisch rotiert, so daß das Muster dort aufgenommen wird, wo die letzte Linie aufgehört hat. Jetzt kommt es zu Störungen beim gleichzeitigen Zeichnen mehrerer Kurven mit demselben Muster: Der Linienmusterzähler versorgt dann alternierend mehrere Kurven. Im Extremfall zweier Kurven, die beide aus Stücken der Länge 1 Pixel aufgebaut werden und Linienmuster 1010101.. erscheint die eine Linie ohne Musterung, die andere gar nicht (vgl. Abb. 17.5b)! Es können so beliebige Interferenzmuster erzeugt werden ...

Leider erlaubt kaum ein Elementargraphiksystem den Zugriff auf den Linienmusterzähler, so daß der obige Effekt nicht einfach zu beseitigen ist. Spätestens an dieser Stelle wird klar, ob der Hersteller bei der Konzeption an einen On-Line-Einsatz gedacht hat!

Kapitel 18

Compiler

Wie schon in Abschnitt 3.2 erläutert wurde, enthält jeder Datenanalysebaustein als zentrales Element einen Compiler. Dieser muß nicht nur arithmetische Ausdrücke übersetzen, sondern in weiteren Ausbaustufen auch Elemente von formalen Sprachen mit derzeit noch nicht definierbarer Syntax. Bei einer derart weit ausbaubaren Konzeption sollten die Prinzipien und praktischen Lösungskonzepte des Compilerbaus berücksichtigt werden. Die Grundprobleme des Compilerbaus — als einer der ältesten Disziplinen der Informatik — können seit den 70er Jahren als gelöst betrachtet werden, so daß die bereits vorliegenden Konzepte eine große Tragfähigkeit haben. Ein Standardwerk ist das hervorragende "Drachenbuch" von Aho, Sethi und Ullman [ASU88], das vor allem für die praktische Umsetzung von Compilern eine Fundgrube ist. Der hier vorgestellte Ansatz lehnt sich eng an die dort gemachten Vorschläge an. In einigen Punkten mußte jedoch eine Anpassung an die Bedürfnisse der Datenanalyse vorgenommen werden.

18.1 Anforderungen

Fernziel soll die Realisierung eines echten Compilers [1] für eine kleine Programmiersprache sein. Nur auf diese Weise lassen sich komplexe Steuerungsabläufe mit maximaler Flexibilität umsetzen. Angestrebt wird eine echte Teilmenge der Sprache C, welche (in Anlehnung an die Benennung von C++ als Erweiterung von C) "C−−" getauft werden soll und über folgende Konstrukte verfügt:

Datentypen: Lediglich die Datentypen

double und int

sowie

Felder und Zeiger

sind verfügbar.

Ausdrücke: Arithmetische und boolesche Ausdrücke können die binären Operatoren

+, −, *, /, ↑, <, >, ≤, ≥, ≥, or, and, etc.

benutzen sowie einen Satz von arithmetischen Funktionen, z.B.

sin, cos, tan, sqr, sqrt, log, ln, ld, exp, max ,min .

[1] Ein echter Compiler erzeugt im Gegensatz zu einem Interpreter schnellen Operations-Code erzeugt, wie er für die die modellgestützte Datenanalyse benötigt wird.

Zuweisungen: Zuweisungen haben die Gestalt:

⟨Variable⟩ = ⟨Arithmetischer bzw. Boolescher Ausdruck⟩ .

Kontrollstrukturen: Zugelassen sind die Verzweigungen

if, if else

sowie die Schleifen

for, while und **do while** .

Funktionen : C−− erlaubt die Definition von Funktionen mit

- Wertübergabe der Parameter (Referenzaufruf erfolgt über die Zeigernotation [KR83]),
- einem Funktionswert (optional),
- rekursiver Aufrufbarkeit.

Kommentare sind möglich.

Diese Teilmenge von C ist verhältnismäßig klein, denn im Kontext der Datenanalyse werden benutzerseitig keine Ein/Ausgabefunktionen oder komplexe Datenstrukturen benötigt. Die im folgenden beschriebenen Werkzeuge sind für die Realisierung des C−−-Compilers sowie einfacher Kommando- und Deklarationssprachen ausgelegt, werden zunächst aber nur zur Erstellung eines Formelcompilers verwendet. Es wird jedoch deutlich, daß der Schritt bis zur Vollversion nur noch gering ist.

18.2 Prinzipieller Aufbau eines Compilers

Typischerweise durchläuft ein Compiler Phasen (vgl. Abb. 18.1):

Lexikalische Analyse: Der Eingabetext wird sequentiell abgearbeitet und in seine syntaktischen Grundbestandteile, die sogenannten *Lexeme*, zerlegt. Bei einer Programmiersprache sind die Lexeme i.a. Bezeichner, Konstanten, Operatoren, reservierte Wörter, Klammern usw. Ein Programm, das die lexikalische Analyse durchführt, wird als *Scanner* bezeichnet.

Syntaxanalyse: Ein *Parser* baut aus den Lexemen, die der Scanner anliefert, den Syntaxbaum des Programms auf und besorgt damit die eigentliche Strukturanalyse [May82].

Semantische Analyse: Als nächstes überprüft die semantische Analyse den Baum auf Konsistenz bzgl. deklarierten Variablen und Typen, Funktionsaufrufen etc. Dabei werden zusätzliche Attribute (z.B. Typumwandlungen) in den Syntaxbaum eingetragen.

Zwischencodegenerierung: Nun wird der attributierte Baum in einen Meta-Code für eine abstrakte Maschine übersetzt (in der Regel in einen sogenannten *Drei-Adress-Code* [ASU88]).

Codeoptimerung: Oft erfolgt nachträglich noch eine Optimierung bzgl. Speicherplatzbedarf und Rechenzeit.

Codegenerierung: Dies ist die eigentliche Übersetzung in den Maschinencode des jeweiligen Rechners und damit der einzige maschinenabhängige Teil eines Compilers.

Diese Phasen laufen bei einem modernen Compiler nicht nacheinander (in sogenannten *Passes* = Durchläufen) ab, sondern gleichzeitig, gesteuert durch den Scanner (*Syntaxgesteuerte Übersetzung*). Codeoptimierung und semantische Analyse sind bei einem C−−-Compiler nicht oder nur rudimentär erforderlich.

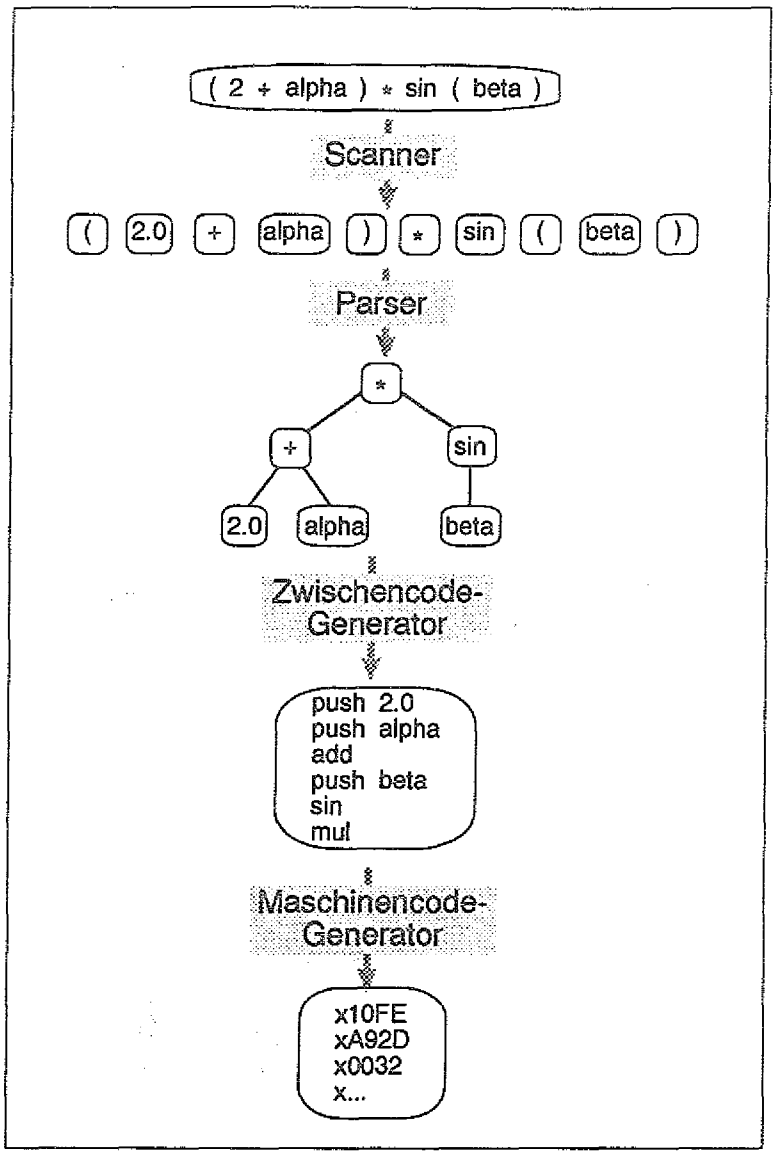


Abbildung 18.1: Ablaufphasen eines Compilers

18.3 Sprachklassen und Sprachanalyse

Scanner und Parser sind Algorithmen, die gewisse Sprachklassen analysieren können. Bei einem Scanner sind dies die relativ einfachen Buchstaben- und Ziffernmuster von Bezeichnern, Zahlen etc., bei einem Parser die Baumstrukturen des Programmaufbaus. Die Strukturmerkmale der verschiedenen Sprachklassen und damit verbunden die optimal angepaßten Algorithmen zu ihrer Erkennung sind schon früh untersucht worden. Man sollte diese Grundlagen kennen, bevor man die Werkzeuge auswählt.

18.3.1 Grammatiken

Chomsky ging schon in den 50er Jahren von einer Sprachdefinition durch Grammatiken aus [Cho56]. Dabei besteht eine Grammatik aus Ableitungsregeln der Form

$$\alpha \rightarrow \beta,$$

wobei α und β Zeichenfolgen (sogenannte *Wörter*) sind, die aus

terminalen Symbolen, das sind z.B. die Symbole

$$A, B, C, \ldots, a, b, c, \ldots, 0, 1, 2, \ldots$$

des zugrunde liegenden Zeichensatzes,

nichtterminalen Symbolen, das sind Meta-Symbole wie z.B.

$$\langle Bezeichner \rangle,\ \langle Ausdruck \rangle,\ \langle Anweisung \rangle,$$

(sie bilden Zwischenstufen bei der Ableitung von Zeichenfolgen)

und auf der rechten Seite ggfs. auch aus dem

leeren Wort ϵ

zusammengesetzt sind. Ein Ableitungsschritt sieht dann so aus, daß α in einem Wort durch β ersetzt wird. Es können so lange Ableitungen durchgeführt werden, bis ein *terminales Wort* (ein Wort, das nur Terminalsymbole enthält) entsteht.

Beispiel: Arithmetische Ausdrücke mit den Operatoren $+$, $-$, $*$, $/$, einzelnen Buchstaben als Bezeichner und ganzen Zahlen werden beschrieben durch folgende Grammatik (das Symbol | bezeichnet eine Oder-Verknüpfung):

$$\begin{array}{lcll}
\langle expr \rangle & \rightarrow & \langle expr \rangle + \langle expr \rangle & (1a) \\
 & | & \langle expr \rangle - \langle expr \rangle & (1b) \\
 & | & \langle expr \rangle * \langle expr \rangle & (1c) \\
 & | & \langle expr \rangle \,/\, \langle expr \rangle & (1d) \\
 & | & \langle ident \rangle & (1e) \\
 & | & \langle number \rangle & (1f) \\
\langle ident \rangle & \rightarrow & A|B|C|...|Y|Z & (2) \\
\langle number \rangle & \rightarrow & \langle number \rangle \langle number \rangle & (3a) \\
 & | & 0|1|2|...|8|9 & (3b)
\end{array}$$

Eine Ableitung des Ausdrucks $A + 10$ könnte dann z.B. so aussehen:

$$\begin{array}{lcll}
\langle expr \rangle & \rightarrow & \langle expr \rangle \ + \langle expr \rangle & (1a) \\
 & \rightarrow & \langle Ident \rangle + \langle expr \rangle & (1e) \\
 & \rightarrow & \langle Ident \rangle + \langle Number \rangle & (1f) \\
 & \rightarrow & A + \langle Number \rangle & (2) \\
 & \rightarrow & A + \langle Number \rangle \langle Number \rangle & (3a) \\
 & \rightarrow & A + \langle Number \rangle 0 & (3b) \\
 & \rightarrow & A + 10 & (3c)
\end{array}$$

Der entsprechende *Syntaxbaum* ist in 18.2 abgebildet.

18.3.2 Chomsky-Hierarchie

Chomsky gelang es anhand der Grammatiken eine Sprachhierarchie aufzubauen, die bis heute fundamentale Bedeutung hat [Cho56]:

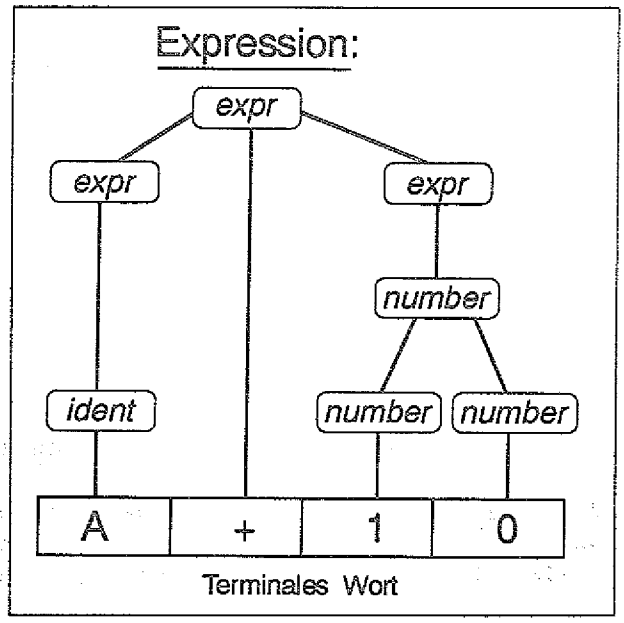


Abbildung 18.2: Syntaxbaum zu einer kontextfreien Grammatik

Die Chomsky Typ-3-Sprachen oder auch *regulären Sprachen* lassen nur Ableitungsregeln des Typs

$$\langle A \rangle \rightarrow w \langle B \rangle$$

mit Nichtterminalsymbolen $\langle A \rangle$, $\langle B \rangle$ und einem terminalem Wort w (ggfs.= ϵ) zu. Z.B. lassen sich Bezeichner in einer Programmiersprache konstruieren durch:

$$\begin{array}{lcl}
\langle ident \rangle & \rightarrow & A\langle alphanum \rangle \\
& | & B\langle alphanum \rangle \\
& | & \vdots \\
& | & Z\langle alphanum \rangle \\
\langle alphanum \rangle & \rightarrow & A\langle alphanum \rangle \\
& | & B\langle alphanum \rangle \\
& | & \vdots \\
& | & Z\langle alphanum \rangle \\
& | & 0\langle alphanum \rangle \\
& | & 1\langle alphanum \rangle \\
& | & \vdots \\
& | & 9\langle alphanum \rangle \\
& | & \epsilon
\end{array}$$

Den zugehörigen *Syntaxgraphen* zeigt die Abbildung 18.3. Mit regulären Ausdrücken darstellen lassen sich ganze und dezimale Zahlen, reservierte Worte, Operatoren, kurzum alles, was in den Aufgabenbereich eines Scanners fällt. Ein Scanner ist definiert als ein Algorithmus, der eine reguläre Sprache erkennt.

Die Chomsky Typ-2-Sprachen oder auch *konteztfreien Sprachen* werden erzeugt durch Ableitungsregeln des Typs

$$\langle A \rangle \rightarrow \alpha$$

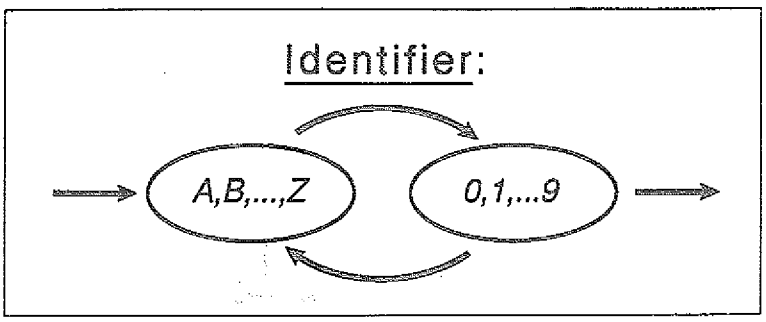


Abbildung 18.3: Syntaxgraph zu einer regulären Grammatik

mit einem Nichtterminalsymbol $\langle A \rangle$ und einem Wort α, das aus terminalen und nichtterminalen Symbolen zusammengesetzt sein kann. Solche Regeln entsprechen genau den Knoten eines Syntaxbaums (vgl. auch Abb. 18.2). Es ist daher nicht verwunderlich ist, daß zu den kontextfreien Sprachen unter anderem die arithmetischen und booleschen Ausdrücke sowie im wesentlichen [2] alle Programmiersprachen gehören. Die in 18.3.1 angegebene Beispiel-Grammatik für einfache arithmetische Ausdrücke ist kontextfrei. Die Aufgabe des Parsers ist der Aufbau des Syntaxbaums zu einem Eingabetext aufgrund der vorgegebenen Grammatik. Benötigt werden dazu rekursive Algorithmen und Kellerspeicher.

Die Chomsky Typ-1-Sprachen oder auch *kontextsensitiven Sprachen* beruhen auf Regeln der Form

$$\alpha_1\langle A\rangle\alpha_2 \rightarrow \alpha_1\langle\beta\rangle\alpha_2$$

mit nichtterminalem $\langle A \rangle$ und beliebigen Wörtern α_1, α_2 und β. D.h. bei kontextsensitiven Sprachen kann die Ersetzung von $\langle A \rangle$ vom Kontext abhängig gemacht werden.

Chomsky nahm an, daß alle natürlichen Sprachen kontextsensitiv sind (eine nicht beweisbare Aussage). Die Erkennung solcher Sprachen ist nur noch mit Methoden der künstlichen Intelligenz möglich und hat für den Compilerbau keine Relevanz.

Die Chomsky Typ-0-Sprachen (oder auch nicht eingeschränkten Sprachen) machen keine Einschränkungen mehr für die Form der Regeln. Sie sind von großem theoretischen Interesse, zumal sie in einer Äquivalenzbeziehung zu den Turing-Maschinen und damit zum Problemkomplex Aufzählbarkeit, Entscheidbarkeit, Berechenbarkeit und Komplexität stehen [Her78] [HU88].

18.3.3 Unterscheidung der kontextsensitiven Sprachen

Der Compilerbau ist unmittelbar mit den regulären Sprachen (Scanner) und den kontextfreien Sprachen (Parser) verknüpft. Die Bemühungen um möglichst effiziente Übersetzer auf der einen Seite und möglichst einfach zu übersetzende Sprachen auf der anderen Seite haben zu einer weiteren Unterteilung der Sprachklassen geführt. Moderne Compiler sind bei geeigneter Sprachdefinition in der Lage, ein Programm in einem einzigen Durchlauf und mit hoher Geschwindigkeit zu übersetzen und zu optimieren. Einige wichtige Unterklassen der kontextfreien Sprachen sind die Sprachfamilien ($k = 0, 1$).

LL(k) Lesen von links nach rechts unter Konstruktion einer Linksableitung mit Lookahead k.

LR(k) Lesen von links nach rechts unter Konstruktion einer Rechtsableitung mit Lookahead k.

LALR(k) Lookahead LR(k)-Grammatik. Eine Abschwächung der LR(k)-Sprachen.

[2] Einige grammatikalische Regeln von Programmiersprachen sind nicht kontextfrei, sondern kontextsensitiv. Dazu gehört z.B. die Forderung, daß verwendete Bezeichner zuvor deklariert sein müssen. Derartige Bedingungen werden bei der semantischen Analyse geprüft, so daß die Syntaxanalyse immer von einer kontextfreien Grammatik ausgeht.

und diverse feinere Unterscheidungen. Eine Hierarchie zwischen den oben genannten Klasse ist nicht gegeben. Grob gesprochen sind die LR(k)-Sprachen schwerer zu übersetzen als die LL(k)-Sprachen und LALR(k)-Sprachen liegen in ihrer Komplexität etwa dazwischen. Wichtig für die Konstruktion eines Compilers für eine kleine Programmiersprache ist nur, daß sich die Konstrukte einer solchen Sprache im wesentlichen in drei Ebenen abhandeln lassen, die jeweils mit einer speziellen Technik effizient behandelt werden. Bei diesen Ebenen handelt es sich um:

Reguläre Ausdrücke, welche die Lexeme (Bezeichner, Konstanten, usw.) der Sprache festlegen. Diese werden mit Hilfe eines Scanners abgearbeitet, dessen effiziente Umsetzung für ganz beliebige reguläre Sprachen heute kein Problem mehr darstellt.

Arithmetische und boolesche Ausdrücke, die in der Tat die syntaktisch komplexesten Teile einer prozeduralen Programmiersprache sind. Sie gehören der Klasse LALR(1) an, sind aber nicht mehr mit LL(1)-Methoden zu erfassen. Wegen der spezifischen Gestalt der arithmetischen und booleschen Ausdrücke muß trotzdem kein allgemeinen LR- oder LALR-Parser konstruiert werden. Es genügt vielmehr ein sogenannter *Operator-Präzendenz-Parser*, der leicht automatisch generiert werden kann.

Kontrollstrukturen und Anweisungen sind wiederum von einfacherer Gestalt, da sie bereits am ersten Wort (**if**, **for**, **while**, etc.) eindeutig erkannt werden. Diese Eigenschaft kennzeichnet die LL(1)-Sprachen und ermöglicht die Anwendung eines *prädiktiven Parsers*, der meist durch die *Methode des rekursiven Abstiegs* verwirklicht wird.

Der C−− Compiler kann somit am einfachsten durch Kombination eines Scanners, eines Operator-Präzedenz-Parsers und eines prädiktiven Parsers implementiert werden.

18.3.4 Scanner- und Parser-Generatoren

Schon in den 60er Jahren wurden die ersten Scanner- und Parser-Generatoren entwickelt, die nach Eingabe einer regulären bzw. LR(1)-Grammatik automatisch einen Scanner bzw. Parser erzeugen. Zu erwähnen sind hier vor allem die eng zusammenarbeitenden Generatoren lex (= lexical analysis) und yacc (= yet another compiler compiler) auf UNIX-Systemen [MB89].

So unkompliziert der Einsatz eines Scanner-Generators ist, so problematisch ist der Umgang mit einem Parser-Generator: Er erwartet eine Grammatik, die frei von sogenannten reduce/reduce- und shift/reduce-Konflikten ist, welche Mehrdeutigkeiten in der Grammatik kennzeichnen (vgl. den unten beschriebenen "dangling else"-Konflikt). Wird ein solcher Konflikt diagnostiziert, so ist die Grammatik so zu transformieren, daß er nicht mehr auftritt. Dabei kann sich ihre Struktur stark verzerren, so daß sich die anschließende Codegenerierung aufgrund des Syntaxbaums schwierig gestaltet. Aus diesem Grunde dürfte kaum ein professioneller Compiler mit Hilfe eines Compiler-Generators erzeugt worden sein. Vielmehr wird in der Regel der bereits skizzierte Weg beschritten. Dies soll die Nützlichkeit von lex und yacc für die Erzeugung von Benutzerdialogsprachen oder Textformatierern nicht in Frage stellen.

18.4 Der Scanner

Ein Scanner hat die Aufgabe, das nächste Lexem des Eingabetextes zu erkennen und als Ganzes an den Parser weiterzureichen. Bei den meisten Programmiersprachen kann der Scanner selbst entscheiden, um welchen Typ von Lexem (z.B. ganzzahlige, dezimale oder boolesche Konstante, Operator oder reserviertes Wort) es sich handelt. Trotzdem gibt es zuweilen Mehrdeutigkeiten. So kann ein Scanner z.B. ohne Kontextinformation nicht erkennen, ob es sich bei einem Minuszeichen um den unadischen oder den duadischen Minus-Operator (-1 oder $2-1$) handelt. Zu diesem Zweck wird in der Praxis meist eine Rückverbindung vom Parser zum Scanner hergestellt, über die Informationen zur Konfliktlösung angefragt werden können.

Der hier entwickelte Scanner soll universell einsetzbar sein, d.h. er soll später auch bei der Analyse von Dialogsprachen oder zur Textformatierung eingesetzt werden können. Aus diesem Grunde ist eine feste Rückkopplung mit dem Parser nicht realisierbar. Der nachfolgend beschriebene Scanner prüft daher nur

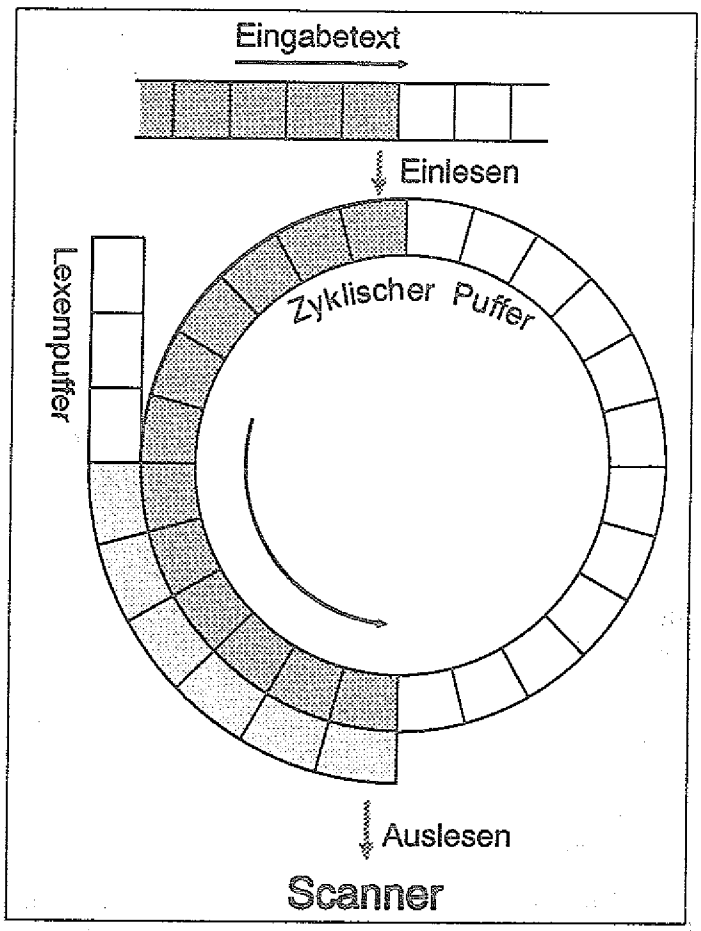


Abbildung 18.4: Pufferanordnung des Scanners

auf Anforderung des Parsers, ob die als nächstes anfallenden Zeichen des Eingabetextes einen bestimmten Typ von Lexem konstituieren. Er entscheidet nicht selbst über die Klassenzugehörigkeit der Lexeme. Diese Aufgabe wird an den Parser übertragen.

18.4.1 Ringpuffer und Lookahead-Puffer

Das eingelesene Lexem wird in einem Puffer an den Parser übergeben, der hier als *Lookahead*-Puffer bezeichnet wird. Der Name erklärt sich dadurch, daß mit Hilfe des Puffers das nächste Lexem der Eingabe "vorausgeschaut" werden kann. Wichtiger ist, daß der Lookahead auch wieder in die Eingabe zurück gelegt werden kann.

Dies wird durch einen Zwischenpuffer ermöglicht, der zwischen die Eingabe und den Lookahead-Puffer geschaltet ist. Da dieser Zwischenpuffer einen sehr hohen Durchsatz hat, wurde die übliche langsame Realisierung durch ein Schieberegister vermieden und stattdessen eine zyklische Architektur gewählt. Die Anordnung von Ringpuffer und Lookahead-Puffer ist in Abbildung 18.4 schematisiert. Während der lexikalischen Analyse arbeiten die beiden Puffer wie folgt zusammen:

Der Parser fordert ein Lexem aus einer bestimmten Wortklasse an. Daraufhin werden so lange Zeichen vom Ringpuffer in den Lexempuffer übertragen bis kein Wort der gewünschten Klasse mehr vorliegt. Beispielsweise stoppt der Scanner bei Anforderung einer Zahl und dem Eingabetext 12345ABC hinter der Ziffer 5. Wichtig ist, daß bei der Kopieroperation die gelesenen Zeichen im Ringpuffer 3wir *noch nicht* gelöscht werden. Es gibt nun zwei Fälle:

a. Der Parser benötigt das erkannte Lexem zum Aufbau des Syntaxbaums. Der Lookahead wird dann *akzeptiert* und nach Verwendung durch den Parser gelöscht. Die entsprechenden Zeichen im

Ringpuffer können anschließend durch neue Zeichen aus der Eingabe überschrieben werden.

b. Die Erkennung des Lexems bedeutet für den Parser eine Negativentscheidung, d.h. es muß ein anderer Analyseweg eingeschlagen werden. Dann werden die Zeichen aus dem Lexempuffer in den Ringpuffer zurückgelegt, was durch einfaches Zurücksetzen des Lesekopfs geschieht.

Die Pufferanordnung ist in C als Datentyp mit Hilfe von dynamischer Speicherbelegung, Zeigern für die Schreib/Lese-Köpfe und einer End-Of-Buffer-Marke (EOB) realisiert. Die Pufferverwaltung sorgt dafür, daß bei Überschreitung der Pufferkapazitäten eine entsprechende Fehlermeldung erfolgt. Sind die Bezeichnerlängen der zu erkennenden Sprache beschränkt, so können die Puffer so dimensioniert werden, daß ein Überlauf bei einer korrekten Eingabe nicht auftreten kann.

18.4.2 Funktionen

Für die Erkennung der elementaren Konstrukte von Programmiersprachen steht ein Funktionsvorrat zur Verfügung, der mit der Zeit beliebig erweiterbar ist. Die Funktionen sind mit Hilfe von einfachen Entscheidungsbäumen auf der Basis der Syntaxgraphen realisiert. Sie können Argumente haben; ihre Namen sind prägnant gewählt. Z.B.:

```
IsAlpha ;          IsIdentifier ;   IsSymbol (Symbol) ;
IsDigit ;          IsBoolean ;
IsAlphaDigit ;     IsUnsigned ;
IsEndOfSource ;    IsInteger ;
                   IsFloat ;
```

Als Ergebnis wird jeweils ein Wahrheitswert zurückgeliefert, der angibt, ob ein entsprechendes Wort erkannt werden konnte. Im Anschluß kann der Lookahead mit der Befehlen

```
AcceptLexeme ;   bzw.   DisposeLexeme ;
```

akzeptiert oder zurückgelegt werden.

18.4.3 Zusätzliche Wahlmöglichkeiten bei der Initialisierung

Um möglichst universell einsetzbar zu sein, hat der Scanner zusätzliche Optionen bei der Initialisierung, mit denen festgelegt wird, wie der Text zu verarbeiten ist:

Eingabequelle : Der in den Ringpuffer eingespeiste Text kann wahlweise einer Datei, einem String oder einem Feld von Strings (aufgefaßt als Zeilen eines Textes) entstammen.

Ignorierte Zeichen : Welche Zeichen als erstes Zeichen eines Lexems zu ignorieren sind, kann eigens spezifiziert werden. In C-- gehören dazu z.B. das Leerzeichen und das Zeilenendesymbol EOL (End Of Line).

Groß- und Kleinschreibung kann wahlweise unterschieden werden (Pascal- oder C-Konvention).

Sonderzeichen wie z.B. ä, ü, ö, ß können zum alphanumerischen Zeichensatz hinzugefügt werden. Dabei muß ggfs. jedes Sonderzeichen zusammen mit der entsprechenden Großschreibung angegeben werden (ä-Ä, ö-Ö, ü-Ü).

Kommentare im Eingabetext werden vom Scanner ignoriert. Dazu ist anzugeben, wie die öffnenden und schließeden Kommentarklammern aussehen (Pascal: {...}, C: /*...*/).

18.5 Der prädiktive Parser

Nach dem Scanner soll nun zunächst der prädiktive Parser für die Kontrollstrukturen von C-- erklärt werden, da dieser sich mit Hilfe der Methode des rekursiven Abstiegs auf der Basis des Scanners sehr einfach implementieren läßt [Wir78].

18.5.1 Eine prädiktive Grammatik

Eine prädiktive Grammatik ist dadurch gekennzeichnet, daß beim Lesen des Eingabewortes von links nach rechts, anhand des nächsten eingelesenen Terminalsymbols stets eindeutig zu erkennen ist, welche der Grammatik-Regeln in der Folge anzuwenden ist.

Ein typisches Beispiel ist die Grammatik für die Kontrollstrukturen von C−−: Bezeichner sowie arithmetische und boolesche Ausdrücke werden in diesem Kontext als terminale Symbole angesehen. Die Kontrollstrukturen haben dann die Syntax:

⟨*block*⟩	→	{ ⟨*sequence*⟩ }	(1*a*)
	|	⟨*statement*⟩ ;	(1*b*)
⟨*sequence*⟩	→	⟨*statement*⟩ ; ⟨*sequence*⟩	(2*a*)
	|	ϵ	(2*b*)
⟨*statement*⟩	→	**for** (⟨*assign*⟩; ⟨*boolexpr*⟩; ⟨*assign*⟩) ⟨*block*⟩	(3*a*)
	|	**while** (⟨*boolexpr*⟩) ⟨*block*⟩	(3*b*)
	|	**if** (⟨*boolexpr*⟩) ⟨*block*⟩	(3*c*)
	|	*assign*	(3*d*)
⟨*assign*⟩	→	⟨*ident*⟩ = ⟨*arithexpr*⟩	(4)

18.5.2 Rekursiver Abstieg

Die Analyse eines C−−-Programmstücks erfolgt von der Wurzel des Syntaxbaums zu den Blättern (*Top Down Analyse*) mit Hilfe der Methode des rekursiven Abstiegs. Zu diesem Zweck wird zu jeder Ableitungsregel eine eigene Prozedur programmiert. Sie untersucht, ob das erste Symbol der Regel als nächstes Wort im Eingabetext vorkommt. Wenn dies zutrifft, arbeitet die Prozedur die zugehörige Regel ab und liefert den Wahrheitswert "wahr" zurück. Andernfalls wird der Lookahead in die Eingabe zurückgelegt und das Ergebnis ist "falsch".
Setzt man die Funktionen IsSymbol (Symbol) und IsIdent des Scanners, eine spezielle Fehlererkennungsprozedur ErrorExit sowie die Funktionen IsArithExpr und IsBoolExpr als gegeben voraus, so hat der rekursive Abstieg für die oben angegebene Grammatik folgende Gestalt:

```
function IsBlock ;
begin
   { Regel 1a }
   if IsSymbol ( "{" )
   begin { Erwartet wird "Sequence" }
                                AcceptLexeme ;
      if IsSequence        then AcceptLexeme else ErrorExit;
      if IsSymbol ( "}" )  then AcceptLexeme else ErrorExit;
      return true ;
   end ;

   { Regel 1b }
   if IsStatement
   begin { Erwartet wird: ";" }
                                AcceptLexeme ;
      if IsSymbol ( ";" )  then AcceptLexeme else ErrorExit;
      return true ;
   end ;

   { Keine Regel anwendbar }
   return false ;
end ;
```

```
function IsSequence ;
begin
   { Regel 2a }
   if IsStatement
   begin { Erwartet wird: "; Sequence" }
                                  AcceptLexeme ;
      if IsSymbol ( ";" )  then AcceptLexeme else ErrorExit;
      if IsSequence        then AcceptLexeme else ErrorExit;
      return true ;
   end ;

   { Regel 2b : Leeres Wort akzeptieren }
   return true ;
end ;

function IsStatement ;
begin
   { Regel 3a }
   if ( IsSymbol ( "for" ) )
   begin { Erwartet wird: "( Assign ; BoolExpr ; Assign )" }
                                  AcceptLexeme ;
      if IsSymbol ( "(" )  then AcceptLexeme else ErrorExit;
      if IsAssign          then AcceptLexeme else ErrorExit;
      if IsSymbol ( ";" )  then AcceptLexeme else ErrorExit;
      if IsBoolExpr        then AcceptLexeme else ErrorExit;
      if IsSymbol ( ";" )  then AcceptLexeme else ErrorExit;
      if IsAssign          then AcceptLexeme else ErrorExit;
      if IsSymbol ( ")" )  then AcceptLexeme else ErrorExit;
      return true ;
   end ;

   { Regel 3b, 3c analog }
   if IsSymbol ( "while" ) begin ... end ;
   if IsSymbol ( "if" ) )  begin ... end ;

   { Regel 3d }
   if IsAssign
   begin { Keine weitere Erwartung }
      AcceptLexeme ;
      return true ;
   end ;

   { Keine Regel anwendbar }
   return false ;
end ;

function IsAssign ;
begin
   { Regel 4 }
   if IsIdent
   begin { Erwartet wird: "= ArithExpr" }
                                  AcceptLexeme ;
      if IsChar ( "=" )    then AcceptLexeme else ErrorExit;
      if IsArithExpr       then AcceptLexeme else ErrorExit;
```

```
        return true ;
    end ;

    { Keine Regel anwendbar }
    return false ;
end ;
```

Es wird deutlich, daß die Programmierung eines rekursiven Abstiegs auf der Basis einer prädiktiven Grammatik und eines geeigneten Scanners eine Routinearbeit ist. Nach Hinzufügen von Anweisungen zur Generierung eines Syntax-Baums sowie Einbeziehung des nachfolgend beschriebenen Operator-Präzedenz-Parsers ist bereits ein vollwertiger Parser für C−− erstellt.

18.5.3 Mehrdeutigkeiten

Die obige Grammatik enthält mit den Regeln (2a) und (2b) eine Mehrdeutigkeit, durch die sie streng genommen aus der Klasse der LL(1)-Grammatiken herausfällt: Es ist nämlich stets möglich, die Regel (2b) anzuwenden, da ein leeres Wort als nächstes Eingabesymbol immer akzeptierbar ist. Dieses Problem ist (außer durch eine Transformation der Grammatik auf eine etwas umständlichere Form) dadurch umgangen worden, daß Regel (2b) nur dann angewendet wird, wenn Regel (2a) versagt.

Eine weitere Mehrdeutigkeit dieses Typs ist das bekannte "dangling else"-Problem beim – oben noch nicht eingeführten — if else-Befehl:

$$\langle statement \rangle \quad \rightarrow \quad \textbf{if} \; (\; \langle boolexpr \rangle \;) \; \langle block \rangle \; \textbf{else} \; \langle block \rangle \quad (3e)$$

Bei dem Programm

```
if ( Bool1 ) if ( Bool2 ) X = 1 else X = 2 ;
```

kommt — je nachdem, ob Regel 3c oder 3e zuerst angewendet wird — eine der beiden Interpretationen

```
if ( Bool1 ) { if ( Bool2 ) X = 1   else X = 2 } ;
if ( Bool1 ) { if ( Bool2 ) X = 1 } else X = 2   ;
```

in Frage. Modernen Programmiersprachen realisieren meist die erste Variante, d.h. Regel (3c) muß vor Regel (3e) geprüft werden.

18.6 Der Operator-Präzedenz-Parser

Arithmetische und boolesche Ausdrücke lassen sich nicht mehr adäquat durch eine prädiktive Grammatik erfassen. Dies liegt an den Operator-Eigenschaften der Präzedenz und Assoziativität:

18.6.1 Präzedenz

Die Regeln des "Punkt-vor-Strich-Rechnens" werden durch die Präzedenzrelation zwischen den Operatoren widergegeben. Diese Hierarchie spiegelt sich nicht in der in Abschnitt 18.3.1 vorgestellten Grammatik für arithmetische Ausdrücke wider. Man kann anhand dieser Regeln nicht entscheiden, ob bei den Eingaben

```
A + B * C   und   A * B + C
```

zuerst die +-Regel oder zuerst die *-Regel anzuwenden ist [3]. Entscheidet man sich wie beim dangling-else-Problem dafür, vorangig die erste Regel zu prüfen, so wird einer der beiden Ausdrücke nicht richtig ausgewertet.

[3] Man spricht von einer *mehrdeutigen* Grammatik.

Diese Probleme mit der Operator-Präzedenz lassen sich durch Transformation der Grammatik auf eine andere Gestalt lösen. Aho, Sethi und Ullman [ASU88] geben eine äquivalente Grammatik an, die prädiktiv ist, und die Operator-Präzedenz-Relation widerspiegelt. Bei Hinzunahme neuer Operatoren und Hierarchieebenen wird dieses Verfahren schnell unübersichtlich. Zudem ist es wenig flexibel bzgl. Erweiterungen.

18.6.2 Assoziativität

Ein anderer Konflikt ist nicht mehr in der LL(1)-Sprachkategorie lösbar. Operatoren können links- und rechts-assoziativ sein, d.h.

$+,\ -,\ *,\ /$ sind linksassoziativ: $a+b+c = (a+b)+c$

und das Potenzieren $\uparrow$ ist rechtsassoziativ: $a^{b^c} = a^{(b^c)}$

Beim Versuch, diese Beziehungen in einer Grammatik ausdrücken, entstehen Regel der folgenden Art:

$$\begin{array}{lcl} \langle sum \rangle & \rightarrow & \langle sum \rangle + \langle number \rangle \\ \langle pot \rangle & \rightarrow & \langle number \rangle \uparrow \langle pot \rangle \end{array}$$

Leider ist die erste Regel — welche Linksassoziativität wederspiegelt — nicht prädiktiv. Der Versuch eines rekusiven Abstiegs würde in eine Endlosschleife führen. Möglich ist allerdings eine Transformation dieser Regel auf die prädiktive Gestalt

$$\begin{array}{lcl} \langle sum \rangle & \rightarrow & \langle number \rangle \langle more \rangle \\ \langle more \rangle & \rightarrow & + \langle number \rangle \\ & | & + \langle number \rangle + \langle more \rangle \end{array}$$

Diese Grammatik generiert aber wieder rechtsassoziative Syntaxbäume, was im übrigen ganz allgemein für prädiktive Grammatiken gilt!

18.6.3 Die Operatortabelle

Aus den genannten Gründen muß für arithmetische und boolesche Ausdrücke eine andere Art von Parser konstruiert werden. Dieser *Operator-Präzedenz-Parser* soll so flexibel formuliert werden, daß er die Tabelle der Operatoren zusammen mit den Präzedenz- und Assoziatsregeln als Eingabe annimmt, also frei programmierbar ist. Die Information über die Operatoren wird dem Parser über eine Tabelle zur Verfügung gestellt, die folgende Angaben für jeden Operator enthält:

Das Symbol, mit dem der Operator bezeichnet wird.
Z.B.: $+,\ -,\ *,\ /$

Die Stelligkeit, d.h. die Zahl der Argumente:
Z.B.: 1 für sin, cos, tan, Vorzeichenfunktion $-$,
2 für min,max,
2 für $+,\ -,\ *,$.

DiePriorität, d.h. die Vorrangigkeit in der Präzedenzrelation $\succ$.
Z.B.: $\uparrow \succ * \succ + \succ$ **and** $\succ$ **or**

Die Notation, d.h. die Schreibweise des Operators mit seinen Argumenten.
Z.B. sind $+,-$ *Infix-Operatoren* geschrieben a + b
und min, max *Postfix-Operatoren* geschrieben max (x , y) .

Die Assoziativität, d.h. Links- oder Rechtsassoziativität.

Die Differentiationsregeln, in Form einer Funktion, die beschreibt, wie der Operator zu differenzieren ist (Dazu mehr in 18.8).

Die Vereinfachungsregeln, in Form einer Funktion, die Rechenregeln zum Vereinfachen von Ausdrücken enthält (mehr in 18.8).

Die Auswertungsvorschrift, d.h. eine Funktion, die einen Operator auswertet, wenn beide Operanden Zahlen sind.

18.6.4 Die Bezeichnertabelle

Auch die Bezeichner werden flexibel gehandhabt. Sie werden in einer Tabelle gesammelt, die vom Benutzer zur Verfügung gestellt wird. Der Benutzer hat zwei Möglichkeiten, diese Tabelle zu handhaben:

- Die Tabelle wird voll gefüllt an den Parser übergeben, d.h. alle Bezeichner sind vordefiniert. Der Parser meldet dann einen Fehler, wenn ein unbekannter Bezeichner auftritt.
- Die Tabelle wird leer oder nur teilweise gefüllt übergeben. Der Parser nimmt dann die neu hinzukommenden Bezeichner in die Tabelle auf.

18.6.5 Arbeitsweise des Operator-Präzedenz-Parsers

Die Funktionsweise eines Operator-Präzedenz-Parsers ist im "Drachenbuch" [ASU88] hervorragend beschrieben, der prinzipielle Algorithmus wird hier kurz widergegeben: Zunächst generiert man aus den Angaben in der Operatortabelle eine dreiwertige Relation mit den Werten

$$\prec, \quad \sim, \quad \succ$$

die über die Operatoren hinaus auf die syntaktischen Grundelemente $< id >$ (für Bezeichner), (,) (Klammern) und $< EOB >$ (für das Eingabeende) ausgedehnt wird. Für die Grundrechenarten inklusive des Potenzierens kann die Relation wie folgt tabelliert werden:

	+	−	*	/	↑	⟨id⟩	(	)	⟨EOB⟩
+	≻	≻	≺	≺	≺	≺	≺	≻	≻
−	≻	≻	≺	≺	≺	≺	≺	≻	≻
*	≻	≻	≻	≻	≺	≺	≺	≻	≻
/	≻	≻	≻	≻	≺	≺	≺	≻	≻
↑	≻	≻	≻	≻	≺	≺	≺	≻	≻
⟨id⟩	≻	≻	≻	≻	≻			≻	≻
(	≺	≺	≺	≺	≺	≺	≺	∼	
)	≻	≻	≻	≻	≻			≻	≻
⟨EOB⟩	≺	≺	≺	≺	≺	≺	≺	≺	∼

Die Einträge in der Diagonalen sind nicht etwa ∼, sondern spiegeln bei den Operatoren die Assoziativität wider. Die leeren Felder entsprechen Situationen, die bei korrekter Eingabe nicht auftreten können (z.B. können nie zwei Bezeichner direkt hintereinander stehen). Der Algorithmus zur Syntax-Analyse ist nun denkbar einfach. Benötigt wird lediglich ein Kellerspeicher (Stapel), der eingelesene Symbole zwischenspeichert:

```
repeat
begin
   if ( Eingabe abgearbeitet und Keller leer )
      return;
   else
   begin
      S = oberstes Symbol auf dem Stapel ;
      E = nächstes Symbol in der Eingabe einlesen ;
      if (S ≻ E)
      begin
```

```
            repeat
            begin
                So = oberstes Symbol auf dem Stapel ;
                Sn = nächstes Symbol auf dem Stapel ;
                So vom Stapel nehmen ;
            end
            until ( Sn ≺ E oder Sn ∼ E )
        end
        else if (keine Relation zwischen S und E)
            Fehler ;
    end;
    E auf den Stapel legen ;
end ;
```

Wie schon beim rekursiven Abstieg fehlen noch die Anweisungen zur Baumerzeugung, um einen vollwertigen Parser zu erstellen. Was weiterhin noch fehlt, sind Operatoren in Frontstellung, die aber im Prinzip wie Bezeichner gehandhabt werden können. Die Argumentliste von Präfix-Operatoren wird dabei mit Hilfe eines prädiktiven Parsers abgearbeitet, der für die Argumente erneut den Operator-Präzedenz-Parser aufrufen muß. Insbesondere lassen sich die Vorzeichen-Operatoren $+, -$ durch eine einfache Modifikation des Algorithmus von den Infix-Operatoren $+, -$ unterscheiden, die hier nicht mehr beschrieben wird.

18.7 Codegenerierung

Bei der Codegenerierung bedient man sich eines maschinennahen Zwischencodes [4]. Für die arihmetischen und booleschen Ausdrücke ist dies die *Umgekehrte Polnische Notation*:

18.7.1 Umgekehrte Polnische Notation

In der Umgekehrten Polnischen Notation wird jeder Operator seinen Argumenten nachgestellt (*Postfix*-Schreibweise). Beispiel:

$$-(x-y)*z\uparrow 2 \quad \text{wird zu} \quad (\,(\,(\,x\,,\,y\,)-\,,\,(\,z\,,\,2\,)\,\uparrow)*)-$$

Die Klammern können auch weggelasen werden, sofern die Zahl der Argumente jedes Operators eindeutig bestimmt ist. Unterscheidet man z.B. unadisches und duadisches $+$ bzw. $-$ durch die Schreibweise $-_1$ bzw. $-_2$ voneinander, so kann das obige Beispiel auch geschrieben werden als

$$x\ y\ -_2\ z\ 2\ \uparrow\ *\ -_1\ .$$

Die Umgekehrte Polnische Notation kann unmittelbar als Ausgabe des Operator-Präzedenz-Parsers erzeugt werden, der dazu lediglich die Elemente, die er aus seinem Kellerspeicher holt, in der Ausgabe aneinanderreihen muß.

18.7.2 Übersetzung und Auswertung der Umgekehrten Polnischen Notation

Der Operator-Präzedenz-Parser generiert zunächst eine Umgekehrte Polnische Notation, in der alle Operatoren durch einen Zeiger auf die Operatordefinition eindeutig repräsentiert sind, Bezeichner mit ihrem Namen auftreten und Konstanten durch ihre numerischen Werte ersetzt wurden [5]. Im nächsten Schritt werden die Operatoren durch ihre Auswertungsfunktion ersetzt sowie die Bezeichner durch die Speicheradressen ihrer Werte. Nach diesem Schritt dürfen sich die Speicherpositionen der Variablen nicht mehr

[4] Vgl. auch die *interpretativen Routinen* bei [Knu73].

[5] Auf dieser Zwischenstufe greifen auch der unten beschriebene Differenzierer und der Vereinfacher an

verändern! Der fertige arithmetische Code kann mit Hilfe eines numerischen Kellerspeichers und dem bekannten Auswertungsalgorithmus schnell und effizient abgearbeitet werden. Benötigt wird dazu die für jeden Operator zur Verfügung gestellte Auswertungsfunktion:

```
repeat
begin
   if ( Eingabe abgearbeitet)
      return oberstes Stapel-Element ;
   else
   begin
      E = aktuelles Symbol in der Eingabe ;
      if ( E ist ein numerischer Wert )
         E auf den Stapel legen ;
      else if (E ist eine Variable)
         Wert von E auf den Stapel legen ;
      else if (E ist ein Operator mit n Argumenten)
      begin
         Auswertungsfunktion auf die obersten n Stapel-Elemente anwenden ;
         n Elemente vom Stapel nehmen ;
         Ergebnis auf den Stapel legen ;
      end
   end
end ;
```

18.7.3 Maschinencode-Generierung für arithmetische Ausdrücke

Ein arithmetischer Koprozessor — wie z.B. der 80x87 oder der 6888x — beruht auf einer Hardware-Implementierung des Rechenstapel-Konzepts. Die Umgekehrte Polnische Notation kann unmittelbar in den entsprechenden Maschinencode umgesetzt werden (vgl. [CG87]). Z.B. ist der 80x87-Maschinenode für das anfangs genannte Beispiel im wesentlichen gegeben durch:

```
FLD     x       ; x laden
FLD     y       ; y laden
FSUB            ; subtrahieren
FLD     z       ; z laden
FLD     2.0     ; 2 laden
FXCH            ; oberste Stack-Elemente vertauschen
FYL2X           ; Y · ld X berechnen
F2XM1           ; 2^(X-1) berechnen
FADD    1.0     ; 1 addieren
FMUL            ; multiplizieren
FCHS            ; Vorzeichen umkehren
```

Zusammen mit einem Prozedurkopf und einer Rücksprunganweisung wird daraus eine funktionsfähige Routine, die allerdings erheblich schneller arbeitet als die algoritmische Abarbeitung des Zwischencodes [MP886]. Ein weiterer Vorteil: Numerische Fehler (Division durch 0 etc.) werden von der Hardware toleriert und lediglich ein Fehlerflag gesetzt, der nachher abgefragt werden kann (z.B. mit der matherr-Funktion vieler C-Compiler).

Mit Hilfe des Kellerspeichers lassen sich auch Funktionen mit Argumenten realisieren, wenn man davon ausgeht, daß die Argumente zunächst (in der richtigen Reihenfolge) auf dem Stapel abgelegt werden. Die Funktion rechnet dann mit diesen Argumenten und ersetzt sie anschließend durch das Ergebnis. Auf

diesem Prinzip einer Stackmaschine beruhen auch einige Programmiersprachen wie z.B. FORTH [Kai88] und PostScript [Ges85].

Die hier skizzierte Realisierung eines Compilers mit Maschinencode-Ausgabe soll aus Effizienzgründen erst nach Abschluß dieser Arbeit in einem Zuge mit der Entwicklung des C−−-Compilers erfolgen. Die Rechengeschwindigkeit ist bereits auf der Ebene des Zwischencodes zufriedenstellend.

18.7.4 Übersetzung von Kontrollstrukturen

Die Umgekehrte Polnische Notation kann ohne Schwierigkeiten in einen Zwischencode eingebettet werden, der für die Kontrollstrukturen einer Programmiersprache erzeugt wird. Ein solcher Zwischencode beruht im wesentlichen darauf, daß alle Verzweigungen und Schleifen durch Sprunganweisungen ersetzt werden ("Spagetti-Codierung"). Die einzigen im Zwischencode definierten Verzweigungsbefehle sind die bedingten Sprungbefehle CondJumpGe und CondJumpEq, die das oberste Element vom Rechenstapel nehmen und abhängig von dessen Wert (≥ 0 oder $= 0$) einen Sprung ausführen. Ein Beipiel:

```
if ( ⟨boolexpr ⟩ )
   X = ⟨arithexpr1⟩
else
   X = ⟨arithexpr2⟩;
```

wird zu

```
        ⟨boolexpr⟩          ; auswerten
        CondJumpEq  Else    ; springen nach Else, falls 0
        ⟨arithexpr2⟩        ; auswerten
        Store       X       ; abspeichern nach X
        Jump        Cont    ; unkonditionierter Sprung
Else:
        ⟨arithexpr2⟩        ; auswerten
        Store       X       ; abspeichern nach X
Cont:
```

Auch hier bietet sich eine Maschinencode-Umsetzung unmittelbar an.

18.8 Differenzieren und Vereinfachen

Viele modellgestützte Methoden benötigen nicht nur ein Modell sondern auch dessen Ableitungen nach allen Zustandsvariablen. Auf der Basis der Umgekehrten Polnischen Notation läßt sich ein explizites Differenzieren der Formeln realisieren [6]

18.8.1 Differenzieren

Jede Operatordefinition enthält zusätzlich eine Ableitungsregel, die mit Hilfe der Argumente des Operators und deren Ableitungen die Ableitung (in Umgekehrter Polnischer Notation) ausrechnet. Ein Beispiel für das Differenzieren eines Produkts nach x [CM87]:

[6] Es soll erwähnt werden, daß es auch eine Technik zur impliziten Differentiation der Modelle gibt. Die Methode der *internen numerischen Differentiation* (IND) [Boc83] beruht gewissermaßen auf einer Differentiation des Diskretisierungsschemas, das einem numerischen Verfahren zugrundeliegt. IND hat eine hohe Effizienz und Genauigkeit. Allerdings muß diese Technik auf jedes verwendete numerische Verfahren erneut angewendet werden!

$$\text{Diff}\ (\ f\ g\ *\ ;\ x\ )\ =\ f\ \text{Diff}\ (\ g\ ;\ x\ )\ *\ \text{Diff}\ (\ f\ ;\ x\ )\ g\ *\ +$$

Die Blätter der resultierenden Bäume werden mit Hilfe der Regeln

$$\text{Diff}\ (\ \langle ident\rangle\ ;\quad x\ )\ =\ \begin{cases} 1 & ,\ \text{falls}\ \langle ident\rangle = x \\ 0 & ,\ \text{sonst} \end{cases}$$

$$\text{Diff}\ (\ \langle number\rangle\ ;\ x\ )\ =\ 0$$

behandelt. Die Ableitung eines Ausdrucks in Umgekehrter Polnischer Notation erfolgt rekursiv mit Hilfe zweier Kellerspeicher. Zu jedem Zeitpunkt sind darin sich die bisher ausgewerteten Teilausdrücke bzw. ihre Ableitungen abgelegt. Nach Terminieren des Algorithmus enthält der erste Kellerspeicher eine Kopie des Eingabeausdrucks, der andere die Ableitung.

18.8.2 Vereinfachen

Beim Differenzieren nach diesem Schema überrascht die Größe der entstehenden Ausdrücke. Einige Beispiele:

$$\begin{aligned} \frac{d}{dx}\left[(x+a)\cdot(x+b)\right] &= (1+0)\cdot(x+b)+(x+a)\cdot(1+0) \\ \frac{d}{dx}\frac{x}{x+y} &= \frac{1\cdot(x+y)-x\cdot(1+0)}{(x+y)^2} \\ \frac{d}{dx}x^y = \frac{d}{dx}e^{y\ln x} &= (0\cdot\ln x + y\cdot 1\cdot\frac{1}{x})\cdot x^y \end{aligned}$$

Offenbar muß eine Vereinfachung erfolgen, welche zumindest die elementaren Rechenregeln berücksichtigt. Ein Satz von Vereinfachungsregeln ist in jeder Operatordefinition enthalten. Z.B.:

$$\text{Simp}\ (\ f\ g\ +\ )\ =\ \begin{cases} g & ,\text{falls}\ \ f\ =\ 0 \\ f & ,\text{falls}\ \ g\ =\ 0 \end{cases}$$

$$\text{Simp}\ (\ f\ g\ \cdot)\ =\ \begin{cases} 0 & ,\text{falls}\ \ f\ =\ 0 \\ 0 & ,\text{falls}\ \ g\ =\ 0 \\ g & ,\text{falls}\ \ f\ =\ 1 \\ f & ,\text{falls}\ \ g\ =\ 1 \end{cases}$$

$$\text{Simp}\ (\ f\ g\ /\ )\ =\ \begin{cases} =0 & ,\text{falls}\ \ f\ \ 0 \\ =\text{Fehler} & ,\text{falls}\ \ g\ \ 0 \\ =f & ,\text{falls}\ \ g\ \ 1 \end{cases}$$

Da die Auswertungsfunktion (Eval) bereits für die Formelauswertung zur Verfügung gestellt werden muß, kann die Regel

$$\text{Simp}\ (\ f\ ,\ g\ ,\ \text{op}\ )\ =\ \text{Eval}\ (\ \text{op}\ ,\ f\ ,\ g\ )\ ,\ \text{falls f, g numerische Werte haben,}$$

automatisch hinzugefügt werden. Bei den obigen Beispielen ergeben sich dann folgende Vereinfachungen:

$$\begin{aligned} \frac{d}{dx}\left[(x+a)\cdot(x+b)\right] &= x+b+x+a \\ \frac{d}{dx}\frac{x}{x+y} &= \frac{(x+y)-x}{(x+y)^2} \\ \frac{d}{dx}x^y &= y\cdot\frac{1}{x}\cdot x^y \end{aligned}$$

Eine weitere Vereinfachung kann erst unter Ausnutzung von Kommutativ- und Assoziativgesetzen erfolgen. Solche Umformungen sind nur noch mit dem Aufwand eines Computer-Algebra-Systems zu bewerkstelligen. Für die Anwendung in der modellgestützten Datenanalyse sind die vereinfachten Terme jedoch bereits zufriedenstellend kurz.

Bemerkung: Besondere Schwierigkeiten bereitet die Regel für das Potenzieren. Sie hat die beiden alternativen Formulierungsmöglichkeiten:

$$\begin{aligned} \frac{d}{dx} f(x)^{g(x)} &= \frac{d}{dx} e^{g(x)\cdot \ln f(x)} \\ &= \begin{cases} [\frac{d}{dx} g(x) \cdot \ln f(x) + g(x) \cdot \frac{d}{dx} f(x) \cdot \frac{1}{f(x)}] \cdot f(x)^{g(x)} \\ \frac{d}{dx} g(x) \cdot \ln f(x) \cdot f(x)^{g(x)} + g(x) \cdot \frac{d}{dx} f(x) \cdot f(x)^{g(x)-1} \end{cases} \end{aligned}$$

Da meistens nur eine der beiden Funktionen $f(x), g(x)$ von x abhängt, erweist sich die zweite Regel in Hinblick auf die Vereinfachung als günstiger. Z.B. ergibt die zweite Regel nach automatischer Vereinfachung das korrekte Ergebnis (vgl. dazu das vorher aufgeführte Beispiel):

$$\frac{d}{dx} x^y = y \cdot x^{y-1}$$

Bei Anwendung der Regel ist auf eine sorgfältige Fallunterscheidung zu achten, um einen Aufruf der Logarithmusfunktion mit negativen Werten oder 0 auszuschließen!

Kapitel 19

Numerik

Die in dieser Arbeit verwendeten numerischen Verfahren sind bereits im Methoden-Teil vorgestellt worden. Aufgrund der Zielsetzung (vgl. 3.6.2) ist eine Weiterentwicklung numerischer Verfahren nicht vorgesehen, so daß hier eine kurze Aufzählung und Quellenangabe genügt:

19.1 Verfahren der linearen Algebra

Bei den Verfahren der Linearen Algebra haben sich die beiden fertigen Pakete

- LINPACK [Don79] für Matrizen- und Vektorrechnung, umfassend
  - die Rechenoperationen für reelle und komplexe Vektoren und Matrizen,
  - die Lösung linearer Gleichungssysteme (inklusive der Matrix-Invertierung) mit Hilfe von LR-Zerlegung und Gauß-Elimination [SB73]

und

- EISPACK [Smi76] für die Behandlung von Eigenwertproblemen, umfassend
  - die Balancierung einer Matrix und Reduktion auf Hessenberg-Form,
  - die QR-Methode zur Berechnung von Eigenwerten und Eigenvektoren,
  - die Behandlung symmetrischer und tridiagonaler Matrizen

allgemein etabliert.

Das für den PC erhältliche Programmpaket MATHPAK87 lehnt sich weitgehend an LINPACK und EISPACK an. Alle Verfahren wurden als Maschinencode für den mathematischen Koprozessor (80x87) aufbereitet. Die damit verbundene optimale Ausnutzung des 80x87-Rechenstacks zieht nach sich, daß die Ausführungsgeschwindigkeit um den Faktor 10-20 über der eines in einer Hochsprache formulierten Programms liegt! MATHPAK umfaßt weiterhin die verschiedenen möglichen Kopieroperationen zwischen Vektoren und Matrizen sowie einen FFT-Algorithmus (reell und komplex), der zur Zeit der schnellste auf dem PC zur Verfügung stehende sein dürfte [Nie90]. Die umgesetzten iterativen Verfahren (Nullstellensuche, Parameteranpassung, etc.) sind dagegen nur so elementar ausgeführt, daß sie in einem interaktiven System praktisch nicht einsetzbar sind (vgl. dazu die Ausführungen in 10.2). Hier ist der Geschwindigkeitsvorteil im übrigen nicht so groß, da die Ablaufsteuerung einer Iteration — im Vergleich zum Aufwand für Vektor- und Matrizenrechnung — nur wenig Zeit in Anspruch nimmt.

19.2 Verfahren für gewöhnliche Differentialgleichungen

Implementiert wurden die Runge-Kutta-Verfahren [PFTV88] der Ordnung

- 1 (Euler-Verfahren) ,
- 2 (Euler-Cauchy-Verfahren) und
- 4 (klassisches Runge-Kutta-Verfahren).

Auf das Verfahren der Ordnung 4 kann wahlweise eine Schrittweitensteuerung (nach [PFTV88]) aufgesetzt werden. Die Programmierung eines Verfahrens höherer Ordnung [HNW87] oder eines speziellen Verfahrens für steife Differentialgleichungen [Gea71] [BD81] [EHL75] erschien bei den auftretenden Problemen noch nicht erforderlich (vgl. auch 9.1).

19.3 Zufallszahlengeneratoren und Verteilungen

Als Basisgenerator für gleichverteilte Zufallszahlen wurde die im Turbo C Runtime-System eingebaute rand-Funktion benutzt. rand liefert ganze Zahlen im Bereich [0...32767]. Der Startwert ist vorgebbar oder mit der Uhrzeit zu belegen. Darauf aufbauend wurden Generatoren für

- gleichverteilte (Skalierung),
- exponentialverteilte (Transformation) und
- normalverteilte (Box-Muller-Transformation)

Zufallszahlen implementiert (vgl. 10.1). In jedem Fall können Erwartungswerte, Standardabweichungen und Dimension eines unabhängig verteilten Zufallsvektors vorgegeben werden. Korrelierte Zufallsvariablen mit vorgebbarer Kovarianzmatrix entstehen mit Hilfe der in 6.2.1.1 angegebenen linearen Transformation.
Für alle generierten Verteilungen sind die Dichtefunktionen abrufbar. Weiter wird aufgrund einer unabhängigen Stichprobe die empirische Dichtefunktion und daraus die empirische Verteilungsfunktion mit vorgegebener Gitterweite berechnet.

19.4 Statistische Verfahren

Der BLUE wurde mit Hilfe der in 10.2.2.1 angegebenen Formel zunächst ohne Rücksicht auf numerische Instabilitäten (die nur bei schlecht konditionierten Problemen auftreten) implementiert.

Die Hauptachsentransformation mußte zur Visualisierung von Kovarianzellipsen durchgeführt werden. In niedrigen Dimensionen gibt es explizite Formeln für die Polynomwurzeln, so daß ein iteratives Verfahren nicht benötigt wurde.

Der Page-Hinkley-Test für das einfache Sprungmodell [BB86] wurde in Prozeduren für

- die Schätzung von Sprunghöhe,
- die Schätzung des Sprungzeitpunkts und
- die Berechnung der kumulativen Summe

zerlegt (vgl. 10.5.3). Der eigentliche Test besteht dann aus einer einfachen Vergleichsoperation. Dasselbe gilt für den einfachen Signifikanztest (vgl. 2.4.1).

19.5 Verfahren für stochastische Differentialgleichungen

Sehr einfach umzusetzen ist das stochastische Euler-Verfahren (Runge-Kutta-Verfahren 1.Ordnung), da es keiner vorherige Umformung der Gleichung bedarf (vgl. 11.1). Dasselbe gilt für ein hybrides Verfahren, bestehend aus einem Runge-Kutta-Verfahren 4.Ordnung für den Drift-Anteil und einem Euler-Verfahren für den diffusiven Anteil. Verfahren höherer Ordnung dürfen ohne vorherige Gleichungsumformung nur bei zustandsunabhängigem Diffusionsanteil verwendet werden (vgl. 11.1).

Das semi-diskrete Kalman-Filter läßt sich zerlegen in Prozeduren

- zur Lösung der Matrix-Riccati-Differentialgleichung,
- zur Berechnung der Verstärkungsmatrix und
- zur Durchführung des Updating-Schrittes.

Die Filter-Prozeduren konnten mit Hilfe stochastischer Simulation, der verschiedenen Visualisierungstechniken und den in 7.3.3.1 angegebenen expliziten Beispiellösungen als korrekt erwiesen werden. Es soll noch einmal darauf hingewiesen werden, daß das Filter — als spezielle Form eines Beobachters — oft auch bei Programmierfehlern noch einwandfrei zu arbeiten scheint, so daß ein sorgfältiger Test unumgänglich ist!

Kapitel 20

Datenflußmanagement und Datenverwaltung

Die Programmierung einer effizienten Datenverwaltung und das Datenflußmanagement wurden nur behelfsmäßig angegangen. Der Grund liegt darin, daß unter dem zur Verfügung stehenden PC-Betriebssystem DOS die bekannte 640K-Grenze für die Größe von Code und Daten besteht. Die erforderliche Überschreitung dieser Grenze wäre mit unnötigem programmiertechnischen Aufwand verbunden, zumal eine spätere Workstation-Implementierung vorgesehen ist (siehe Anhang). Trotzdem können hier schon der Anforderungskatalog und eine Konzeption für die Realisierung vorgestellt werden:

20.1 Statistische und regelungstechnische Ansätze

Statistische und regelungstechnische Datenanalysesysteme unterscheiden sich prinzipiell durch die Art der Daten und den Datenfluß voneinander:

Bzgl. der Art der Daten arbeiten

- regelungstechnische Systeme mit zeitabhängigen Daten, deren zukünftige Werte unbekannt sein können.
- statistische Systeme mit ganzen Datensätzen, die vor der Analyse aufgezeichnet wurden.

Bzgl. des Datenflusses arbeiten

- regelungstechnische Systeme zu einem bestimmten Zeitpunkt immer nur mit einem Datum oder maximal mit einer geringen Zahl von Vergangenheitswerten. Die Abarbeitung erfolgt dabei iterativ, indem die beteiligten Verfahren in eine Zeitablaufsteuerung eingebunden werden (vgl. Abb. 20.1a und 20.1b).
- statistische Systeme mit dem gesamten Datensatz zugleich. Dabei laufen die verschiedenen Verfahren nacheinander ab (vgl. Abb. 20.2a und 20.2b).

Stark vereinfacht kann gesagt werden, daß statistische Datenanalysesysteme das Problem der Speicherplatzverwaltung haben und regelungstechnische das Problem der Zeitablaufsteuerung. Es soll kurz dargestellt werden, wie in regelungstechnischen bzw. statistischen Systemen die jeweiligen Probleme gelöst werden.

20.1.1 Das Kanalkonzept bei regelungstechnischen Systemen

Sowohl regelungstechnische Datenanalysesysteme als auch Prozeßüberwachungssysteme arbeiten mit dem Konzept des *Kanals*, der in Analogie zu einer elektrischen Leitung zu sehen ist (vgl. Abb. 20.1c). Ein

solcher Kanal führt zu jedem Zeitpunkt genau einen Wert (oder maximal eine begrenzte Zahl von Vergangenheitswerten). Er wird von genau einem regelungstechnischen Baustein (bzw. von einem externen Eingang) gespeist und kann von mehreren Bausteinen gelesen (bzw. auf einen externen Ausgang gelegt) werden. Jede Verbindung zwischen Datenanalysebausteinen wird durch einen Kanal realisiert, so daß ein Netzwerk von Bausteinen mit einer Vielzahl von Kanälen entsteht. Man kann sich analog zur Computer-Hardware alle Kanäle zu einem Daten-Bus zusammengefaßt vorstellen, der von den Bausteinen angezapft wird (vgl. Abb. 20.1c).

Das Problem der Datenverwaltung ist im wesentlichen äquivalent mit dem der Kanalverwaltung. Es umfaßt

- die Vergabe und Freigabe von Kanälen sowie die Namengebung.
- die Konfliktlösung bei Zugriffskonflikten (mehrere Ausgänge, aber nur ein Eingang pro Kanal).
- die Auslagerung von Daten in Dateien.

Die Ablaufsteuerung erfolgt mit Methoden der parallelen Programmierung (vgl. Abb. 14.4): Alle benutzten Bausteine werden in eine Liste aufgenommen und zyklisch dazu veranlaßt, einen Iterationsschritt durchzuführen (vgl. auch Abb. 20.1b). Diese Vorgehensweise imitiert die parallele Arbeit mehrerer elektronischer Bauelemente.

20.1.2 Das Transformationskonzept bei statistischen Systemen

Ein statistisches System geht meist von einem Transformationsmodell aus, bei dem aus dem Eingangsdatensatz mit Hilfe einer Transformation der Ausgangsdatensatz generiert wird (vgl. Abb. 20.2a). Der Ausgangsdatensatz kann dabei eine vergleichbar hohe Dimension haben wie der Eingangsdatensatz, aber auch aus nur wenigen statistischen Kenndaten bestehen.

Zu einem bestimmten Zeitpunkt findet stets nur eine Transformation statt, so daß die Datenverwaltung befaßt ist (vgl. 20.2c) mit

- der Ein- und Auslagerung von Datensätzen.
- der Namenverwaltung und Archivierung.
- der Buchhaltung über die Beziehungen zwischen den Datensätzen [VH88] [Hub89].

Eine Ablaufsteuerung kann oft mit Hilfe einer Makro-Programmierung durchgeführt werden. Das Makro legt die Reihenfolge und Parametrisierung der verwendeten Transformationen fest. Meist sind zusätzlich — nicht zeitliche — Iterationen und Verzweigungen möglich (vgl. 20.2b). Auch diese Form der Ablaufsteuerung ist in keiner Weise problematisch, da die Transformationen nicht parallel arbeiten müssen.

20.1.3 Eine Hybridlösung

Die biotechnische Datenanalyse-Situation vereint die Problematik von statistischen und regelungstechnischen Systemen: Einerseits müssen statistische Verfahren möglich sein, die auf einen ganzen Datensatz (oder auf alle Vergangenheitswerte) zugreifen, andererseits müssen alle Verfahren in eine Zeitablaufsteuerung integrierbar sein. Dabei sind folgende Probleme zu lösen:

- Die Behandlung biotechnischer Daten in ihrer vollen Komplexität (inklusive Zeitverzögerungen, fehlender Werte, nicht äquidistanter Abtastung etc.; vgl. dazu auch die Ausführungen in Kapitel 1.2).
- Die effiziente Speicherung und Archivierung ganzer Datensätze mit der ständigen Zugriffsmöglichkeit auf alle Werte.
- Die Ablaufsteuerung für kompliziert vernetzte Systeme, welche insbesondere die Möglichkeit zur Umkonfiguration während eines Durchlaufs enthalten muß.

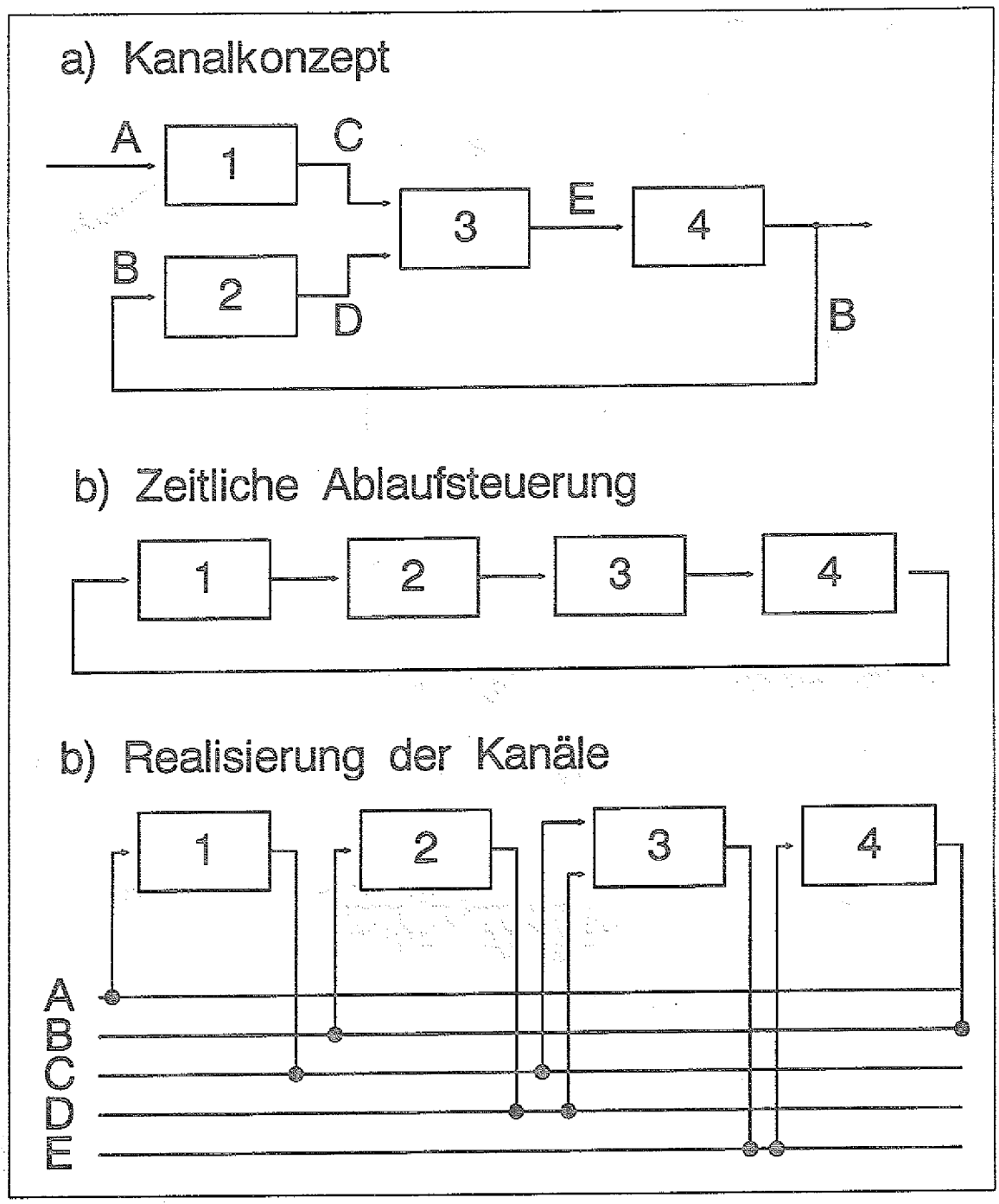


Abbildung 20.1: Kanalkonzept und Ablaufsteuerung in einem regelungstechnischen Datenanalysesystem

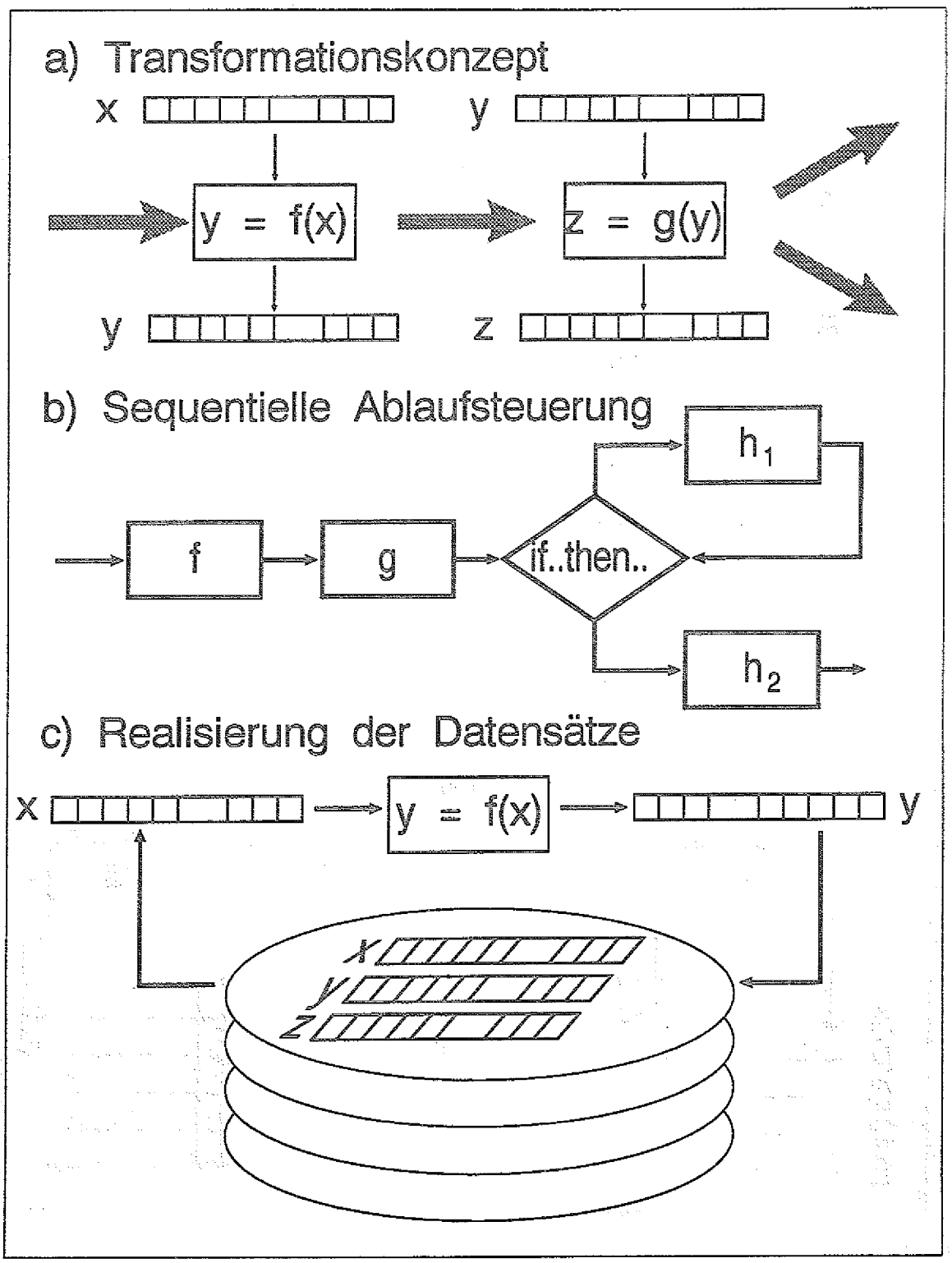


Abbildung 20.2: Transformationskonzept und Ablaufsteuerung in einem statistischen Datenanalysesystem

Diese Anforderungen stellen das eigentlich Neue an der biotechnischen Datenanalyse-Situation dar. Man kann feststellen, daß zur Zeit kein Datenanalysesystem diesen Anforderungen gerecht wird (vgl. [Sch90d]). Die hier erarbeiteten Konzepte stellen einen wesentlichen Schritt in dieser Richtung dar.

20.2 Anforderungen an ein biotechnologisches Datenanalysesystem

Die angestrebte Hybridlösung sollte die Vorteile beider Modelle vereinen, d.h. einerseits die Möglichkeit einer zeitlichen Ablaufsteuerung und andererseits die Möglichkeit zur (bedingten) Anwendung statistischer Verfahren auf ganze Datensätze bieten.

20.2.1 Speicherplatzbedarf

Eine einfache Überschlagsrechnung zeigt, daß die Vereinigung beider Konzepte ohne Rücksichtnahme auf den Speicherplatz nicht zum Ziel führt. Alle Daten müßten dann zu jeder Zeit im Speicher des Rechners gehalten werden: Ein aufwendig instrumentierter Fermenter (vgl. Abb. 1.3) liefert bei jeder Abtastung etwa 20 Meßwerte. Geht man von einem Abtastintervall von einer Minute, einer Laufzeit von einer Woche und einem Speicherplatzbedarf von 4 Byte für einen numerischen Wert aus, so ergibt sich ein Speicherplatzbedarf von

$$20 \times 60 \times 24 \times 7 \times 4 = 0.804 \text{ MB}.$$

Nun wird ein Kalman-Filter mit einem Zustandsraum der Dimension 5 und 3 Messwerten aufgesetzt, resultierend in 5 Zustandsschätzern, 3 Messwertschätzern und 25 Einträgen in der Kovarianzmatrix. Nach einem einzigen Durchlauf des Filters bei gleichzeitiger Aufzeichnung der geschätzten Werte vergrößert sich das Datenmaterial folglich auf

$$(20 + 5 + 3 + 25) \times 60 \times 24 \times 7 \times 4 = 2.137 \text{ MB}\ !$$

Die Rechnung ist übertrieben, aber sie ist ein guter Ausgangspunkt für Überlegungen zu möglichen Einsparungen. Sie münden zusammen mit der Charakterisierung biotechnischer Datensätze aus Kapitel 1.2 in dem nun folgenden Anforderungskatalog:

20.2.2 Ein Anforderungskatalog

20.2.2.1 Datentypen

Jeder Datensatz muß wahlweise

- numerische,
- logische,
- kategorielle,
- oder auch vektorielle bzw. in Matrizenform vorliegende Daten aufnehmen.

20.2.2.2 Zeitausdehnung

Der Bereich eines Datensatzes, der im Speicher verfügbar ist, kann sich

- auf den aktuellen Wert,
- über ein festes Zeitfenster oder
- über ein (in die Vergangenheit oder Zukunft) ausgedehntes verschiebbares Zeitfenster

erstrecken.

20.2.2.3 Zugriff

Zugriffsmöglichkeiten können gegeben sein durch

- Anforderung eines einzelnen Wertes oder
- Anforderung eines Zeigers auf eine Folge von im Speicher aufeinanderfolgenden Werten.

20.2.2.4 Zeitrepräsentation

Ein großer Teil der Daten bezieht sich auf eine Zeitskala, die gegeben sein kann durch

- Startzeitpunkt, Abtastintervall (und eventuell eine Zeitverzögerung) bei äquidistant abgetasteten Daten (implizite Zeitskala) oder
- über eine Referenz auf einen Zeit-Datensatz (explizite Zeitskala).

20.2.2.5 Interpolation

Zeitlich wenig veränderliche Signale können interpoliert werden. Möglichkeiten sind:

- abschnittsweise konstante Interpolation,
- lineare Interpolation oder
- Spline-Interpolation (resultierend in einem zusätzlichen Datensatz mit den Spline-Koeffizienten).

20.2.2.6 Markierung

Datensätze müssen mit zusätzlichen Attributen versehen werden können. In diesen Bereich gehören:

- NaN-Werte für nicht verfügbare Daten,
- Zusatzattribute (z.B. für temporär gelöschte oder graphisch eingefärbte Datenpunkte).

20.2.2.7 Vektor- und Matrizen-Repräsentation

Zeitreihen von Vektoren und Matrizen kann man entweder repräsentieren durch

- einen Datensatz von Vektoren (Matrizen) oder
- einen Vektor (eine Matrix) von Datensätzen.

Damit einher geht das Problem der Namengebung zur Unterscheidung dieser Formen.

20.3 Ein erweitertes Kanalkonzept

20.3.1 Erweiterte Kanäle

Es soll auch weiterhin von (erweiterten) Kanälen gesprochen werden, weil diese Terminologie den einfachsten Zugang zur Realisierung einer Zeitablaufsteuerung bietet. Im Gegensatz zu einem (gewöhnlichen) Kanal muß ein erweiterter Kanal auch ganze Datensätze aufnehmen, wobei ein Wert durch die aktuelle Zeit besonders hervorgehoben ist. Dies wird durch die Abbildung 20.3a veranschaulicht.

Die Datenverwaltung muß dafür Sorge tragen, daß die Datenrepräsentation aller Kanäle stets den Anforderungen der derzeit aktiven Datenanalysebausteinen genügt. Jeder Datenanalysebaustein gibt dazu bei seiner Initialisierung ein Anforderungsprofil (gemäß der obigen Liste) an. Wesentliches Problem der Datenverwaltung ist jetzt die Konfliktlösung:

> Zu jedem Zeitpunkt sollen so wenige Daten wie möglich im Speicher abgelegt werden, ohne daß die Rechenzeit beeinträchtigt wird (vgl. hierzu Abb. 20.3b).

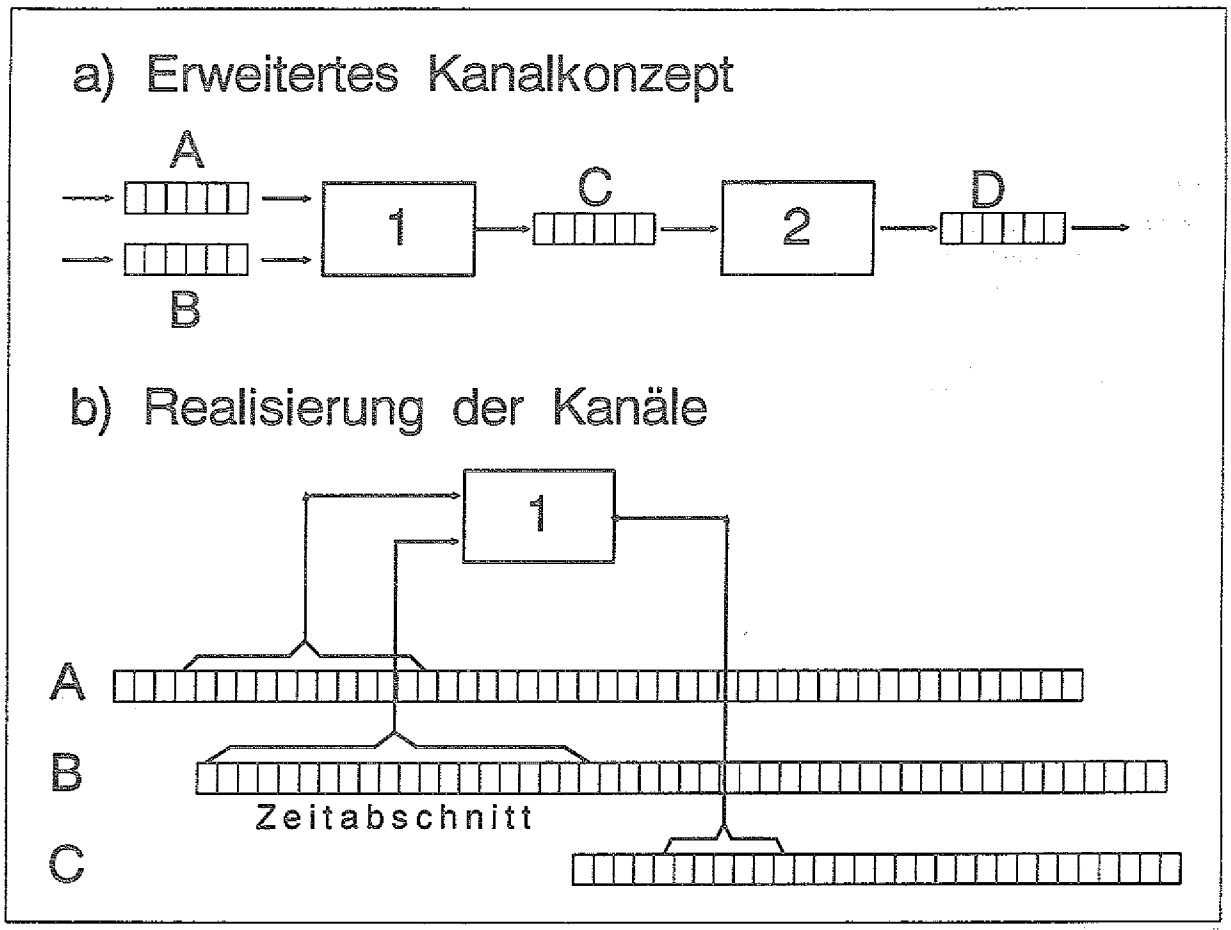


Abbildung 20.3: Erweitertes Kanalkonzept

20.3.2 Kanalverwaltung

Die Datenverwaltung muß somit gewissermaßen den "kleinsten gemeinsamen Nenner" aller Anforderungen finden. Leider ist dies nicht immer möglich, da Anforderungen sich unauflösbar widersprechen können. Z.B. kann nicht Baustein A einen Datensatz von Vektoren verlangen, während Baustein B dieselben Daten als Vektor von Datensätzen wünscht. In dieser Situation gibt es mehrere Möglichkeiten:

a) Einer der Datenanalysebausteine verarbeitet auch eine alternative Repräsentation der Eingangsdaten (möglicherweise mit Geschwindigkeitseinbußen).

b) Der konfliktauslösende Kanal wird dupliziert, was mit Rechenzeit- und Speicherplatzaufwand verbunden ist.

c) Wenn dies nicht möglich ist, kann der Benutzer davon informiert werden, wobei möglicherweise Hilfen für eine Umkonfiguration gegeben werden.

Es gibt eine Datenrepräsentation, die fast allen Anforderungen genügt: Die sequentielle Abspeicherung aller Werte eines Datensatzes mit der kleinstmöglichen Abtastzeit und einer ebensolchen (expliziten) Zeitskala. Sie ist leider von allen Lösungen die speicherplatzaufwendigste, also gewissermaßen das "kleinste gemeinsame Vielfache" aller Anforderungen!

20.3.3 Verwirklichung des Phasen-Konzepts für die Datenanalyse

Die Ablaufsteuerung kann nun phasenweise genau so erfolgen wie bei einem regelungstechnischen System: Alle Bausteine werden in eine Liste gesetzt und zyklisch abgearbeitet. Im einfachsten Fall besteht die

Liste aus einem einzigen statistischen Baustein, der keine zeitliche Steuerung benötigt und lediglich eine Transformation durchführt. Ein solcher Baustein kann aber auch bei jedem Zeitschritt ein ganzes zeitliches Fenster von Daten bearbeiten.

Eine Phase ist genau dann zu Ende, wenn ein Baustein neu in die Liste aufgenommen, herausgenommen oder verändert wird. Dann muß durch eine erneute Initialisierung zunächst eine Konfliktbeseitigung stattfinden. Anschließend wird der iterative Betrieb wieder aufgenommen. Auf diese Weise kann das in Abschnitt 10.3.2.2 formulierte Phasenkonzept für die Datenanalyse konkret verwirklicht werden.

Zur formale Darstellung von Phasenübergängen sind sogenannte *Zustands-Transitions-Diagramme* hervorragend geeignet (vgl. auch [Hal89]). Ein solches Netz ist in Abbildung 20.4 wiedergegeben. Die praktische Realisierung solcher Netze wird durch das Baustein-Konzept unmittelbar ermöglicht.

20.3.4 Kommunizierende Objekte

Offenbar kann man bei der Bewältigung dieser Koordinationsprobleme viel aus der Praxis der Betriebssysteme, vor allem der Multi-Tasking lernen. Verschiedene Möglichkeiten zur Synchronisation und Realisierung einer Interprozeß-Kommunikation sind bereits in Abschnitt 14.5 vorgestellt worden. Sie müssen in geeignetem Umfang zur Verfügung gestellt werden. Insbesondere müssen alle Bausteine mit entsprechenden Steuereingängen versehen werden.

20.3.5 Objektorientierte Realisierung

Die oben beschriebene Vorstellung von Datenanalysebausteinen, Kanälen, Kanalverwaltung und Ablaufsteuerung als miteinander kommunizierende Systeme legt eine objektorientierte Programmierung geradezu zwingend nahe! In einer objektorientierten Umgebung können nicht nur die Bausteine, sondern auch die Kanäle als abstrakte Klassen mit ihren Unterklassen formuliert werden. Dies ist ein weiterer Grund, die Realisierung erweiterter Kanäle aufzuschieben, bis ein brauchbarer C++-Compiler zur Verfügung steht [1].

20.4 Datenbasis

Bzgl. der Datenbasis sei an die Ausführungen in Kapitel 3 erinnert: Es ist erst dann sinnvoll, sich hierüber Gedanken zu machen, wenn ein konkretes Datenbanksystem zugrunde liegt. Eine prinzipielle Entscheidung wird dabei nur bzgl. des Datenmodells der Datenbank zu treffen sein. Dies kann eine relationale Struktur [Wed81], aber auch die modernere objektorientierte Konzeption sein [AE888].

[1] Mit dem Abschluß dieser Arbeit erscheint der neue Turbo C++-Compiler [TCP90], so daß mit der Umsetzung bald begonnen werden kann.

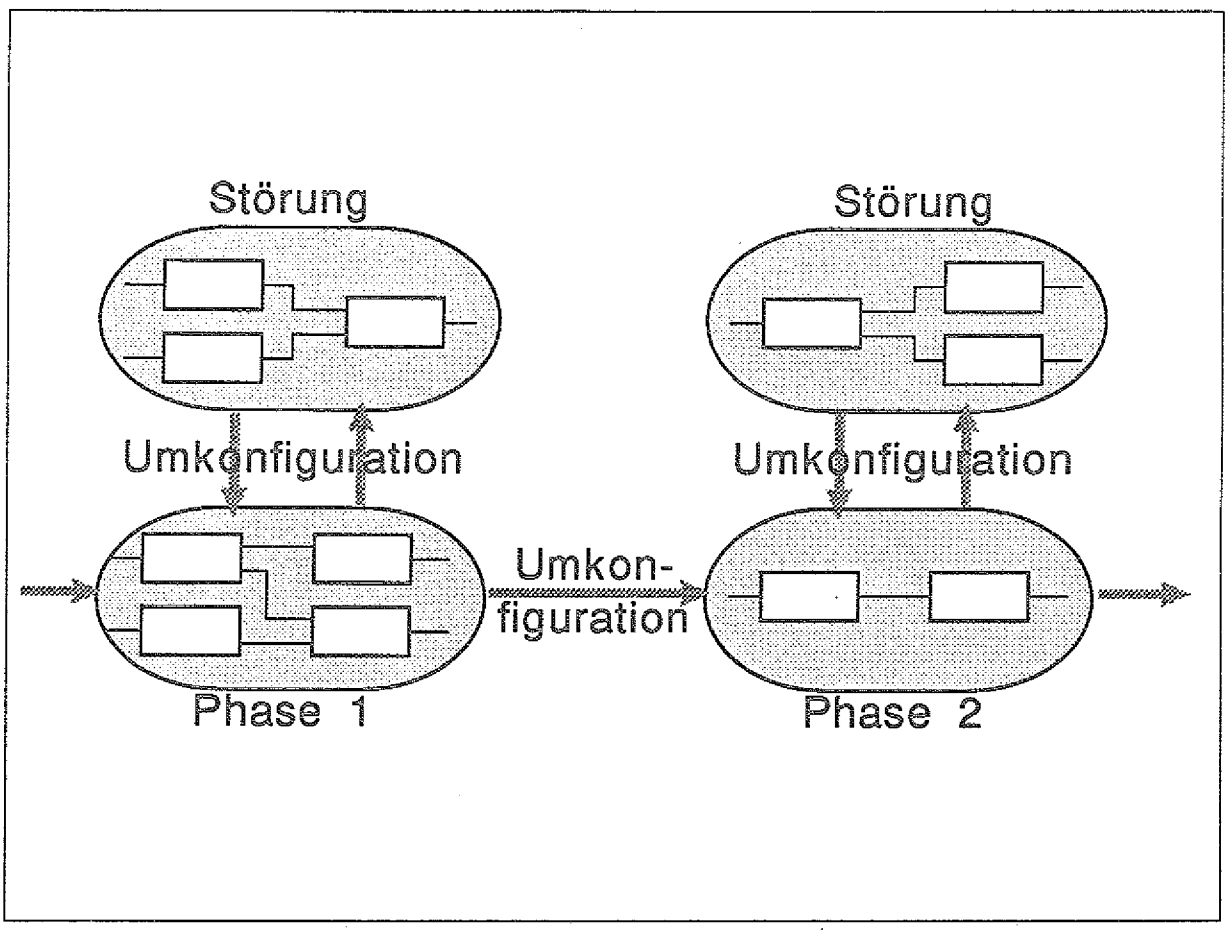
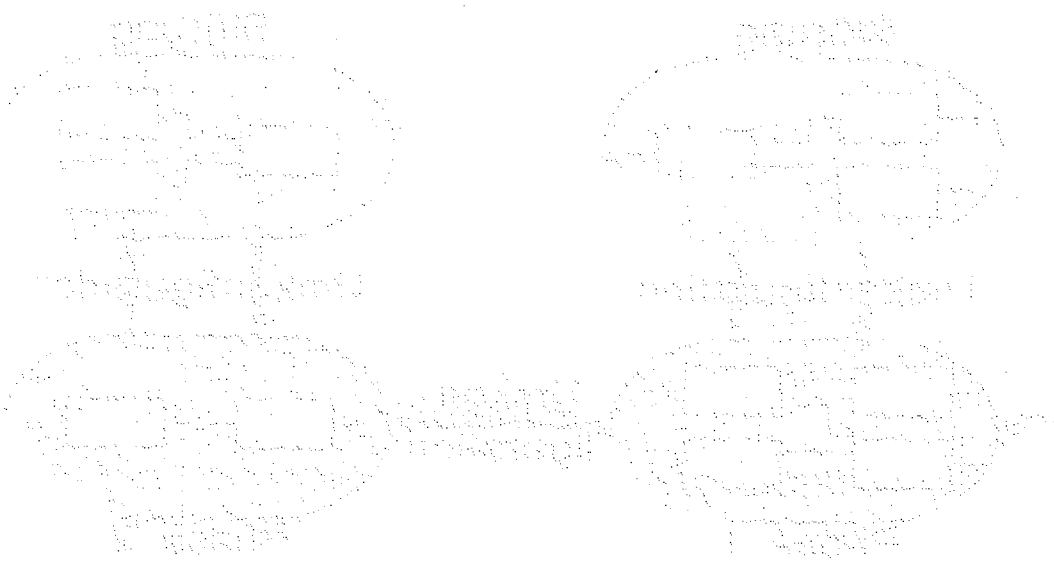


Abbildung 20.4: Zustands-Transitions-Diagramm



Teil V

Anwendungsbeispiele

Den Abschluß der Arbeit bilden eine Reihe von Anwendungsbeispielen, die in keinem festen Zusammenhang stehen. Sie sollen die Vielseitigkeit des Ansatzes und die Breite des Anwendungsgebietes verdeutlichen. Die beiden folgenden Kapitel behandeln dabei zwei Aspekte der Konzeption. Sie ergänzen sich gegenseitig:

Kapitel 21 führt am Beispiel eines frei konfigurierbaren Erweiterten Kalman-Filters die Realisierbarkeit und Tragfähigkeit des Baustein-Konzepts vor. Es geht damit an einer exemplarischen Stelle in die *Tiefe* der programmiertechnischen Realisierung.

Kapitel 22 beschäftigt sich mit der *Breite* des Spektrums und stellt eine Reihe von sehr unterschiedlichen Beispielen vor. Dabei stand nicht die interaktive Realisierung einzelner Methoden im Vordergrund, sondern die Verdeutlichung der Möglichkeiten, die in der Verschaltung verschiedener Bausteine liegen.

Kapitel 21

Ein frei konfigurierbares Kalman-Filter

Das frei konfigurierbare Erweiterte Kalman-Filter ist ein Prototyp für einen vollständig realisierten Datenanalysebaustein gemäß der vorgestellten Konzeption. Um eigenständig betrieben werden zu können, ist es zunächst als integriertes System realisiert worden. D.h. das Programm umfaßt zugleich die Dateiein- und -ausgabe und einen einfachen Graphik-Teil. In späteren Versionen werden diese Bestandteile zu selbständigen Komponenten. In Kombination mit dem in Abschnitt 2.2 beschriebenen graphischen Dateneditor liegt bereits in dieser einfachen Form ein leistungsfähiges Datenanalyse-Instrument vor.

21.1 Benutzeroberfläche und Funktionen

Die Oberfläche des Filters ist der von der Firma Borland entwickelten Oberfläche des Turbo-Pascal-Compilers (Version 4.0 und höher [TCP91]) nachempfunden, die in ihrer Funktionalität richtungsweisend war [1]. Eine Anlehnung an einen Compiler liegt nahe, weil auch die Konfiguration eines Kalman-Filters in den Arbeitsgängen

- Modelleingabe im Editor,
- Übersetzung und Fehlerprüfung und
- Durchlauf

stattfindet. Da die Borland-Oberfläche und ihre Bedienung in weiten Kreisen bekannt ist, genügt hier eine kurze Beschreibung der Menüstrukturen. Alle Menüpunkte sind auch über entsprechende "Hot Keys" schnell zu erreichen. Einen optischen Eindruck von der Oberflächengestaltung verschafft die Abbildung 2.6.

21.1.1 Datei-Eingabe und Ausgabe

Das Untermenü umfaßt die Punkte

- Laden einer Konfigurationsdatei (es erscheint ein Dateiauswahlmenü, vgl. Abb. 2.6a),
- Abspeichern der aktuellen Konfiguration in eine Datei (Warnung bei Gefahr des Überschreibens),
- Auswahl einer Eingabedatei für das Filter (analog zum Laden der Konfigurationsdatei gelöst),
- Auswahl einer Datei zum Ausspulen der Filterergebnisse.

Die Eingabedateien müssen ein bestimmtes Text-Format haben, das insbesondere die Namen und physikalischen Einheiten der Meßgrößen berücksichtigt.

[1] Dies belegt nicht zuletzt die Tatsache, daß auch die Konkurrenzfirma MicroSoft ihre Compiler mit dieser Oberfläche versehen hat.

21.1.2 Modelleingabe

Die Modelleingabe umfaßt

- Systemgleichungen incl. der System-Rauschterme,
- Meßgleichungen incl. der Meß-Rauschterme,
- Kontrollanzeige für die Linearisierung der System-Gleichungen,
- Kontrollanzeige für die Linearisierung der Meß-Gleichungen.

21.1.3 Parameter- und Variableneingabe

Parameter und Variablen untergliedern sich in (vgl. auch Abb. 2.6b)

- die Zeitvariable,
- die Zustandsvariablen,
- die Meßgrößen,
- die Kontrollvariablen und
- die Parameter.

Jede einzelne Größe ist durch die Angabe der physikalischen Dimension, den Wertebereich und einen initialisierenden Wert (der auch eine Formel sein kann) gekennzeichnet.

21.1.4 Bereichsgrenzen für die Rauschterme

Die Bereichsgrenzen werden für eine eventuelle graphische Darstellung benötigt. Berücksichtigt werden müssen

- Obergrenzen für die Varianzen des Systemrauschens,
- Obergrenzen für die Varianzen des Meßrauschens,
- Obergrenzen und Startwerte für die Varianzen des Zustandsschätzers,
- Obergrenzen und Startwerte für die Varianzen des Meßwertprediktors.

21.1.5 Graphik-Konfiguration

Die einzelnen Größen können im Zeitverlauf dargestellt werden. Die Graphik-Variablen sind hierbei frei wählbar. Das Graphik-Layout ist aus den angeführten Gründen nur recht einfach gestaltet.

21.1.6 Übersetzung, Fehlerprüfung und Durchlauf

Dies umfaßt wie bei einem Compiler die Punkte

- Übersetzung und Fehlerprüfung (auftretende Fehler werden diagnostiziert; der Benutzer wird unmittelbar an die Stelle des Auftretens geführt, vgl. Abb. 2.6c).
- Durchlauf bei korrekter Konfiguration, verbunden mit einem Überwechseln in den graphischen Modus.
- Fortsetzung des Durchlaufs nach einer Unterbrechung durch den Benutzer.
- Neustart eines Laufs.

21.2 Implementierung

21.2.1 Datenstruktur

Alle Konfigurationsdaten des Kalman-Filters werden in einer Datenstruktur abgelegt (Kapselung, siehe 14.3). Diese ist sehr umfangreich, wie schon in Abbildung 15.5 illustriert wurde. Sie soll hier exemplarisch für andere Datenanalysebausteine wiedergegeben werden, wobei die Hilfsvariablen zur Datei-, Status- und Fehlerbehandlung, sowie die Bereichsgrenzen weggelassen wurden. Die verwendeten Datentypen sind die schon in Kapitel 15 beschriebenen Typen Double (reele Zahl), Vct (dynamischer Vektor), Mat (dynamische Matrix) und SavStr (speichergesicherter String):

```
Struktur KalmanFilter
begin
    Struktur T { Zeitvariable des Filters }
    begin
        double Val ;        { Wert }
        SavStr Name ;       { Name }
        double Step ;       { Schrittweite }
    end ;
    Struktur X  { Zustandsvariablen }
    begin
        Vct Val ;           { Werte }
        Vct Name ;          { Namen }
        Vct Init ;          { Terme für Startwerte }
        Vct Func ;          { Übersetzte Terme }
    end ;
    Struktur Y  { Meßgrößen }
    begin
        Vct Val ;           { Werte }
        Vct Name ;          { Namen }
    end ;
    Struktur U  { Steuergrößen }
    begin
        Vct Val ;           { Werte }
        Vct Name ;          { Namen }
    end ;
    Struktur A  { Parameter }
    begin
        Vct Val ;           { Werte }
        Vct Name ;          { Namen }
    end ;
    Struktur F  { Modellgleichungen }
    begin
        Vct Term ;          { Terme }
        Vct Func ;          { Übersetzte Terme }
        Mat JacTerm ;       { Jacobi-Matrix }
        Mat JacFunc ;       { Übersetzte Jacobi-Matrix }
    end ;
    Struktur G  { Meßgleichungen }
    begin
        Vct Term ;          { Terme }
        Vct Func ;          { Übersetzte Terme }
        Mat JacTerm ;       { Jacobi-Matrix }
        Mat JacFunc ;       { Übersetzte Jacobi-Matrix }
```

```
        end ;
        Struktur R  { Systemrauschen }
        begin
              Vct Val ;          { Werte }
              Vct Term ;         { Varianzterme }
              Vct Func ;         { Übersetzte Terme }
        end ;
        Struktur S  { Meßrauschen }
        begin
              Vct Val ;          { Werte }
              Vct Term ;         { Varianzterme }
              Vct Func ;         { Übersetzte Terme }
        end ;
        Struktur P  { Kovarianzmatrix der Zustandsschätzer }
        begin
              Mat Val ;          { Werte }
              Vct Init ;         { Terme für die Startwerte }
              Vct Func ;         { Übersetzte Terme }
              end ;
        Struktur Q  { Kovarianzmatrix der Vorhersage }
        begin
              Mat Val ;          { Werte }
        end ;
        Struktur K  { Kalman-Verstärkungsmatrix }
        begin
              Mat Val ;          { Werte }
        end ;
end ;
```

21.2.2 Oberfläche

Die Realisierung der Oberfläche (incl. Hilfestellung, "Hot-Key"-Steuerung und Fehleranzeige) war mit Hilfe des in Kapitel 16 beschriebenen Menügenerators in relativ kurzer Zeit zu bewerkstelligen. Insbesondere das Konzept der variabel dimensionierten Vektoren und Matrizen hat sich dabei als äußerst leistungsfähig erwiesen. Speicherzuordnungsfehler oder Stringüberschreitungsfehler traten aufgrund der objektorientierten Konzeption nicht auf.

21.2.3 Operationen

Die Operationen auf der Datenstruktur sind zunächst wieder als Funktionen in C realisiert. Bei der objektorientierten Erweiterung in C++ werden daraus die Methoden der Klasse Kalman-Filter. Die wichtigsten sind:

- Initialisierung aller Datenfelder (Konstruktion),
- Löschung und Speicherplatzfreigabe (Destruktion),
- Eintritt in die Benutzeroberfläche,
- Einlesen aus einer Datei,
- Ausgabe in eine Datei,
- Übersetzung und Fehlerprüfung,
- Initialisierung der Startwerte vor dem Durchlauf,

- Durchlaufschritt des Differentialgleichungslösers,
- Updating-Schritt.

21.3 Einsatzbereiche

21.3.1 Parameter- und Modellstudien

Das beschriebene frei konfigurierbare Filter kann für Parameter- und Modell-Studien herangezogen werden, die bisher nur recht aufwendig zu betreiben waren (vgl. auch [STSS89]).

21.3.2 Praktische Erfahrungen

Die Konfiguration eines Erweiterten Kalman-Filters für den bereits in Kapitel 1 erwähnten *Candida boidinii*-Prozeß hat gezeigt, daß gerade die Eingabe der Modellgleichungen und deren Ableitungen eine Haupt-Fehlerquelle darstellt. Obwohl das frei konfigurierbare Filter noch nicht zur Verfügung stand, konnten beim *Candida*-Prozeß doch bereits einige Techniken der interaktiven und graphischen Parametrisierung, sowie einige Visualisierungstechniken erprobt werden (vgl. Kapitel 2). Ein Beispiel für die Wachstumsratenschätzung mit Hilfe des an den Prozeß angebundenen Filters zeigt die Abbildung 21.1 [2]. Näheres zur Problemstellung ist in [WNSW90] nachzulesen.

21.3.3 Downloading

Auch das Problem des Downloadings kann bereits auf elementare Art gelöst werden: Da jede Filterkonfiguration in einer Datei im Textformat abgespeichert ist, können die benötigten Terme für Modellgleichungen und Ableitungen mit Hilfe eines Texteditors hieraus entnommen und in ein Programm eingesetzt werden.

Eine elegantere Lösung ergibt sich bei Einsatz eines Präprozessors (wie z.B. des C-Präprozessors [KR83]), mit dessen Hilfe Teile des Programms substituiert werden können, ohne daß der Programmtext verändert werden muß. Dieser Vorgang läßt sich relativ einfach automatisieren. Es sei darauf hingewiesen, daß dies jedoch nicht die angestrebte Form des Downloadings ist. Vielmehr soll das beschriebene Filter (ohne die graphischen Elemente und die Benutzeroberfläche) direkt auf dem Prozeßrechner implementiert und mit Hilfe der Datei konfiguriert werden.

21.4 Vergleich mit anderen Systemen

Auf dem VDI-Workshop "Regelungstechnische Programmpakete für den IBM-PC und kompatible Personal Computer unter MS-DOS" [Sch90d] im März 1990 in Düsseldorf wurde eine ganze Palette von Entwicklungen vorgestellt, von denen einige größere Systeme auch ein Kalman-Filter enthalten (Pilar, CADACS, CYPROS, POS-1 in [Sch90d]). Ein kurzer Vergleich mit hier beschriebenen Implementierung ist lohnend:

- Die für die Biotechnologie üblichen Zustandsraummodelle in Form von Differentialgleichungen finden selten Unterstützung. Stattdessen wird von einer schaltungsartigen Konfiguration mittels regelungstechnischer Grundbausteine (Integrator, Addierglied etc.) ausgegangen.
- Nur wenige Applikationen ermöglichen eine Modelleingabe über eine komfortable Benutzeroberfläche. In der Regel müssen Konfigurationsdateien editiert werden (Problem der Hilfestellung und des Erlernens der Deklarationssprache [Sch82a]), wenn nicht sogar ein FORTRAN-Modul erforderlich ist.
- Das Problem der variablen Dimensionierbarkeit bei gleichzeitiger Verwendung einer Benutzeroberfläche wurde in keinem Fall gelöst.

[2] Ich danke Herrn Mathias Neumann und Herrn Michael Wedy für die Überlassung des Datensatzes.

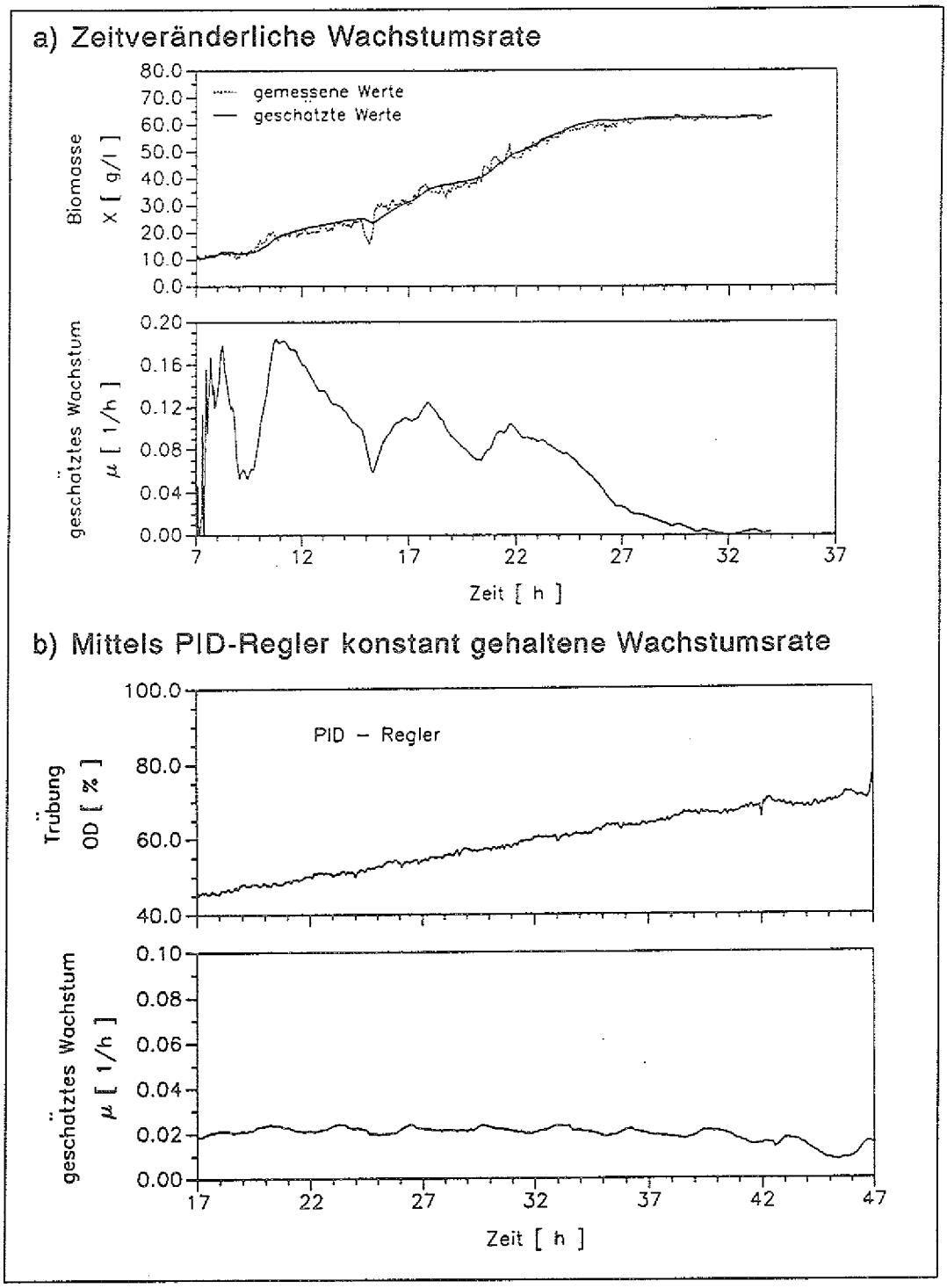


Abbildung 21.1: Einsatz eines Kalman-Filters zur Wachstumsratenschätzung bei einer Fed-Batch-Kultur von *Candida boidinii* (aus [WNSW90])

- Die Linearisierung der Filtergleichungen ist — wenn möglich — nur in einem Fall mittels symbolischer Differentiation gelungen. In der Regel wird numerisch vorgegangen (Problem der Wahl einer Diskretisierungsschrittweite), wenn nicht sogar eine explizite Eingabe erforderlich wird.
- Realisiert wurde das diskrete oder kontinuierliche Filter. Das semikontinuierliche Filter — in Verbindung mit der Problemstellung zeitverzögerter und nicht äquidistanter Meßdaten — ist mit keiner der Konzeptionen vereinbar.

Kapitel 22

Weitere Anwendungsbeispiele

Die abschließenden Beispiele stellen kompliziertere Anwendungsprobleme vor, die als Testfälle für die Leistungsfähigkeit eines Datenanalyse-Konzepts herangezogen werden können. Sie sind derzeit alle noch Gegenstand der Forschung und sollen deshalb nicht bis ins Detail analysiert werden.

22.1 Instationäre Identifikation

22.1.1 Prinzip der instationären Identifikation

22.1.1.1 Stationäre und instationäre Identifikation mittels Parameteridentifikation

Ausgehend vom allgemeinen Stoffwechselmodell (vgl. 5.2.6.1)

$$\dot{X} = N \cdot X_1 \cdot r(X)$$

wurden in 10.4 bzw. 10.3 zwei Techniken zur Identifikation der kinetischen und stöchiometrischen Parameter diskutiert. Die Ergebnisse seien kurz zusammengefaßt:

Stationäre Identifikation in kontinuierlicher Kultur :

Vorteile :

- Eine Annahme spezieller Kinetiken $r(X)$ ist nicht erforderlich. Die Art der Kinetik kann aus den graphischen Auftragungen ersehen werden.
- Zeitverzögerungen und Adaptionsvorgänge wirken sich nicht auf die Auswertung aus.

Nachteile :

- Die Stabilisierung stationärer Zustände muß regelungstechnisch gesichert werden.
- Die Durchführung der Experimente erfordert einen hohen zeitlichen und personellen Einsatz.
- Die Information über den Übergang zwischen verschiedenen stationären Zuständen wird nicht ausgeschöpft.

Instationäre Identifikation mittels Parameteranpassung :

Vorteile :

- Geringer Zeitaufwand.
- Keine Stabilisierung des Systems erforderlich.

Nachteile :

- Das Resultat einer Kurvenanpassung beinhaltet wenig Hinweise auf mögliche Verbesserungen des Modells. Verschiedene Modelle müssen deshalb der Reihe nach erprobt werden ("Erschöpfendes Durchsuchen").
- Bei Störungen des Systems oder Zeitverzögerungen ist das Verfahren nicht anwendbar.

22.1.1.2 Instationäre Identifikation mittels Filterung

Den beiden Identifikationsansätzen soll nun die instationäre Identifikation mittels Filterung gegenübergestellt werden, welche die Vorteile beider Techniken vereint. Im Fall der kontinuierlichen Kultur verallgemeinert sie den stationären Ansatz, der auf der Beziehung

$$\begin{aligned} & N \cdot X_1 \cdot r(X) - D \cdot X = \dot{X} = 0 \\ \Longleftrightarrow \quad & N \cdot X_1 \cdot r(X) = D \cdot X \end{aligned}$$

beruht. Die Einschränkung $\dot{X} = 0$ ist nicht erforderlich, wenn es gelingt, eine zuverlässige Schätzung von $\dot{X}(t)$ zu erlangen. Dann kann zu jedem Zeitpunkt t auch die Beziehung

$$N \cdot X_1 \cdot r(X) = D \cdot X - \dot{X}$$

herangezogen werden, womit ungleich mehr Information zur Verfügung steht. Es ist klar, daß dieser Ansatz auch bei anderen Betriebsweisen des Reaktors anwendbar ist.

Wie mit Hilfe eines Kalman-Filters die kinetischen Daten (Substrataufnahme, Wachstumsrate, etc.) konkret zu schätzen sind, wurde bereits ausführlich erläutert (vgl. Kapitel 2). Die zugrundeliegenden Modelle für ein solches Filter sind durch entsprechende instationäre Erweiterung der in 10.4 dargestellten Ansätze zu gewinnen.

Es muß betont werden, daß ein Filter nur bei hinreichend großer Dichte der Meßdaten anwendbar ist. Es muß daher eine Meßtechnik zur Verfügung stehen, die alle Konzentrationsgrößen mit ausreichend großer Abtastrate erfaßt. Hier bieten moderne Analysengeräte große Vorteile.

22.1.1.3 Voraussetzungen für die Anwendbarkeit der Methode

Zwei Voraussetzungen müssen erfüllt sein:

1. Die Konzentrationen X müssen während des Experiments einen Wertebereich überschreiten, der genügend viel Information für die Bestimmung der Kinetiken bereitstellt.

2. Für die Schätzung der Ableitungen $\dot{X}$ kommen nur lokale Verfahren wie Filterung, lokale Regression, Fensterschätzung etc. in Frage (vgl. auch [SY90]). Diese sind naturgemäß mit einer Trägheit behaftet, so daß die Differentiation nur dann erfolgreich sein kann, wenn sich X nicht zu schnell ändert.

Die erste Forderung muß an jedes Identifikationsverfahren gestellt werden. Das Parameteridentifikationsverfahren liefert als Kriterium lediglich ein numerisches Maß in Form der Kovarianzmatrix der Parameterschätzer (vgl. 10.2.2.1), wogegen die beiden anderen Techniken nach entsprechender graphischer Auftragung auch direkte Hinweise über noch zu erfassende dynamische Arbeitsbereiche des Systems liefern.

Die zweite Forderung setzt eine entsprechende Prozeßführung voraus. Hierzu bietet sich eine kontinuierliche oder eine Fed-Batch-Kultur mit zeitveränderlichem Zufluß an. An diese Prozeßführung werden allerdings nicht die strengen Forderungen der stationären Identifikation gestellt. Anstelle einer Stabilisierung des Prozesses genügt eine Gewährleistung langsamer Veränderungen.

22.1.2 Ein Beispiel

Zur Überprüfung der beiden Voraussetzungen eignet sich hervorragend eine interaktiv gesteuerte Simulation, in der die Steuergröße (hier die Durchflußrate D) vom Bediener on-line geändert werden kann (vgl. [SW88]). Auf diese Weise können in kurzer Zeit grundsätzliche Ergebnisse für die Wahl einer Identifikationsstrategie erzielt werden, die mit exakten Optimierungsansätzen nur mit großem Aufwand zu gewinnen sind [Wal87]. Die erhaltene suboptimale Lösung benötigt etwas mehr Experimentierzeit, ist aber in der Gesamtbilanz (Versuchsvorbereitung + Durchführung + Auswertung) zeiteffizienter.

Die Visualisierung der benötigten Information erfolgt mit Hilfe einer Darstellung der Wachstumsrate im Zeitverlauf (Kontrolle schneller Änderungen) sowie einer Auftragung der Wachstumsrate μ über der

Substratkonzentration S (Kontrolle des überstrichenen Wertebereichs). Die Veranschaulichung der "Informationsdichte" wird hierbei durch diskrete Abstastung von μ in äquidistanten Zeitintervallen erziehlt (vgl. Abb. 22.1, 22.2 und 22.3).

Die Simulation beruht auf dem Modell

$$\begin{aligned} \dot{X} &= \mu(S) \cdot X \quad -D \cdot X \\ \dot{S} &= -\sigma(S) \cdot X \quad -D \cdot (S_{in} - S) \end{aligned}$$

mit einer Inhibierungskinetik

$$\sigma(S) = \sigma_S \cdot \frac{S}{S^2/I_S + S + K_S}$$

für die Substrataufnahme und

$$\mu(S) = Y_{X/S} \cdot \sigma(S)$$

bei vernachlässigtem Erhaltungsstoffwechsel.

22.1.2.1 Batch-Kultur

Die Abbildung 22.1a zeigt die Konzentrationsverläufe einer Batch-Kultur ($D = 0$): Nach einem Zeitraum mit inhibiertem Wachstum durchläuft die Wachstumsrate ihr Maximum, um dann innerhalb sehr kurzer Zeit auf 0 abzusinken. Über die Kinetik bei kleinen Substratkonzentrationen steht daher kaum Information zur Verfügung. Der K_S-Wert ist aus diesem Grunde nach einer Parameteranpassung nur innerhalb weiter Fehlergrenzen festgelegt (vgl. auch [Mun89]).

Mit Hilfe eines Filteransatzes ist die schnelle Änderung der Dynamik nicht zu erfassen. Ein Filter kann ihr nur langsam folgen, wie bereits die auf dem gleichen Modellansatz beruhende Abbildung 2.7 zeigt. Sehr wohl ist jedoch die inhibierte Phase verfolgbar, d.h. mittels der Approximation

$$\sigma(S) = \sigma_S \cdot \frac{S}{S^2/I_S + S + K_S} \sim \sigma_S \cdot \frac{I_S}{S + I_S}$$

(vgl. 5.1.2.2) können aus dem Batch-Experiment die Parameter σ_S, I_S und $Y_{X/S}$ ermittelt werden. Die Wahl einer Inhibierungskinetik für $\sigma(S)$ wird hierbei durch die graphische Auftragung nahegelegt (vgl. Abb. 22.1a).

22.1.2.2 Kontinuierliche Kultur

Im Anschluß an einen Batch-Lauf kann der Reaktor auf kontinuierlichen Betrieb umgestellt werden ($D = \text{const} > 0$). Ein solches Beispiel zeigt die Abbildung 22.1b. Hier überstreicht S nur einen sehr kleinen Konzentrationsbereich bis zum Einschwingen in den stationären Zustand, so daß die Kinetik im Prinzip nur an einer einzigen Stelle auswertbar ist. Durch Hinzufügen des neuen Datums zu den bisher ermittelten kinetischen Daten ist der K_S-Wert enger eingrenzbar.

22.1.2.3 Verlängerte Batch-Kultur mit Pulszufütterung

In [Mun89] wird untersucht, wie der Zufluß $F(t)$ einer Fed-Batch-Kultur beschaffen sein muß, um optimale Fehlergrenzen für die Parameteridentifikation innerhalb einer vorgegebenen Zeit zu erzielen. Mit Hilfe von Methoden der Variationsrechnung konnte gezeigt werden, daß eine optimale Fahrweise $F_{opt}(t)$ mit einer Batch-Kultur beginnt, die dann gegen Ende der exponentiellen Phase durch Zugabe von Substrat verzögert beendet wird. Die Ergebnisse sind auf eine kontinuierliche Kultur übertragbar, da sich die Modelle formal nur durch die Terme D bzw. $\frac{F}{V}$ voneinander unterscheiden (vgl. 5.3.3.1).

Erst mit dem Übergang zu einer variablen Durchflußrate $D(t) > 0$ ist es also möglich, einen größeren Konzentrationsbereich langsam zu überstreichen. Die in [Mun89] für eine Monod-Wachstumskinetik errechnete optimale Lösung ist eine gepulste Substratzufütterung gegen Ende der Wachstumsphase. Eine solche Strategie gibt Abbildung 22.2a wieder: Der Prozeß reagiert auf die gepulste Zufütterung extrem sensibel, so daß die Technik der instationären Identifikation mittels Filterung erneut nicht anwendbar ist.

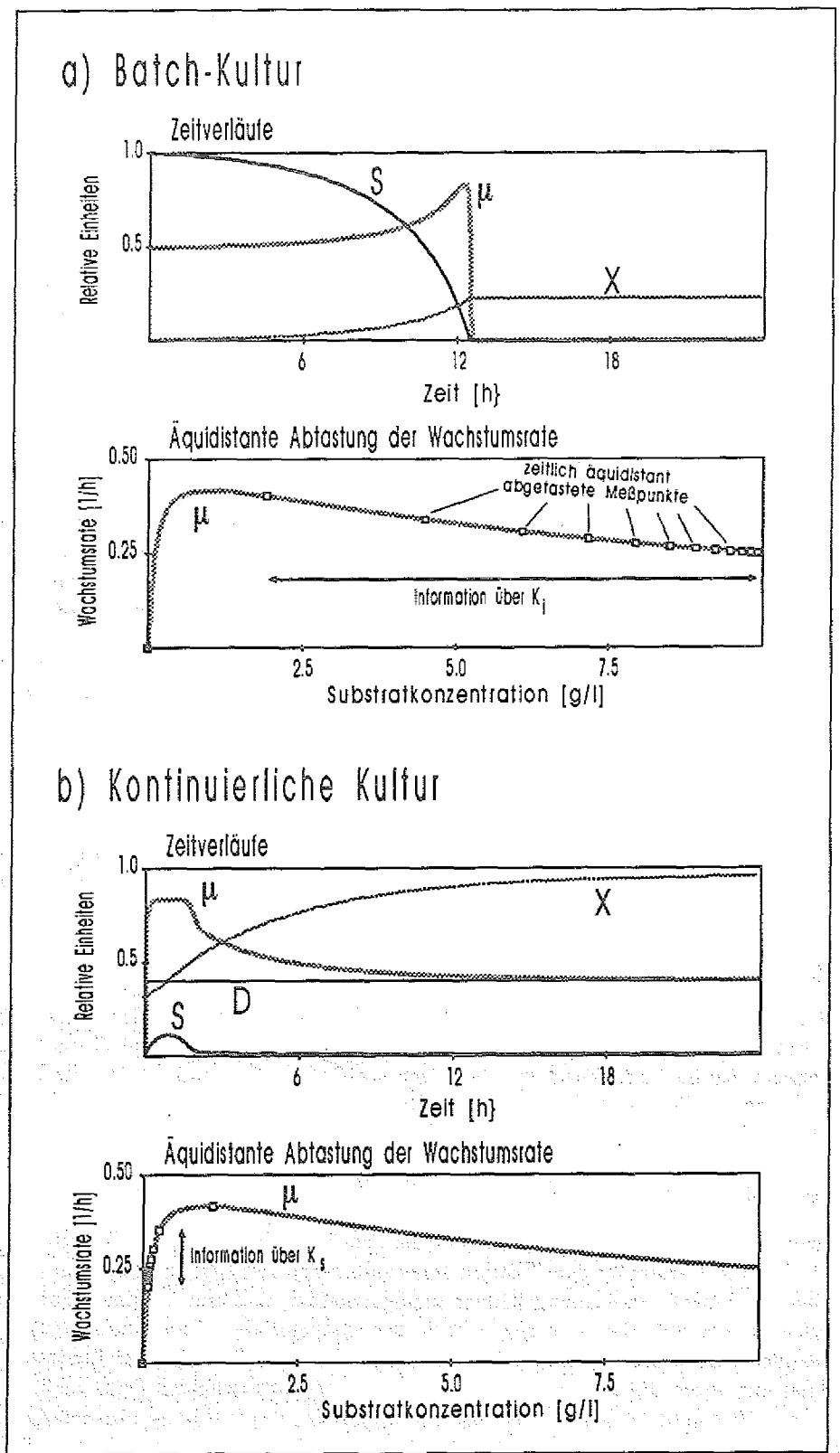


Abbildung 22.1: Instationäre Open-Loop-Identifikation

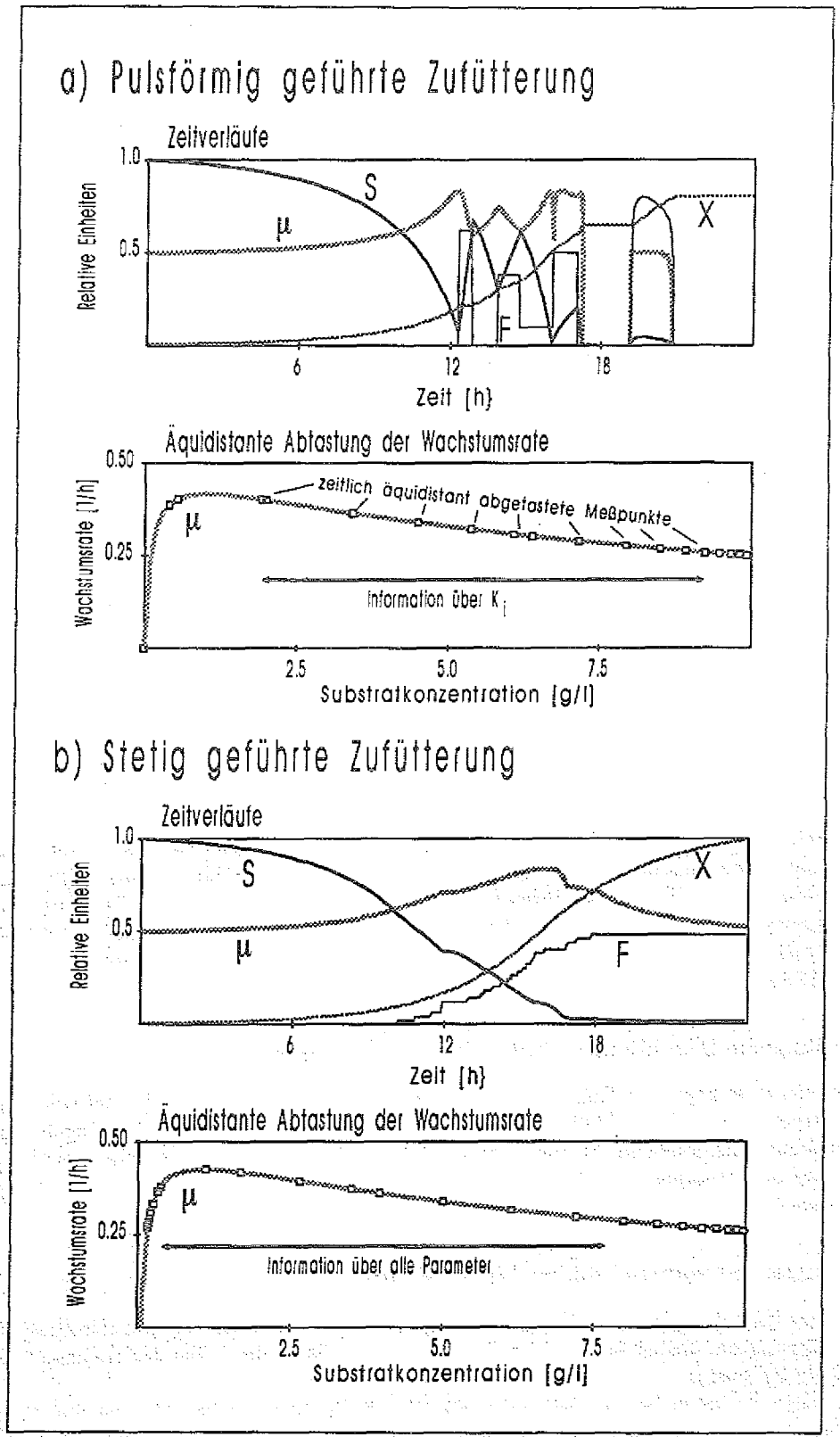


Abbildung 22.2: Identifikation bei einer instationär geführten Fed-Batch-Kultur

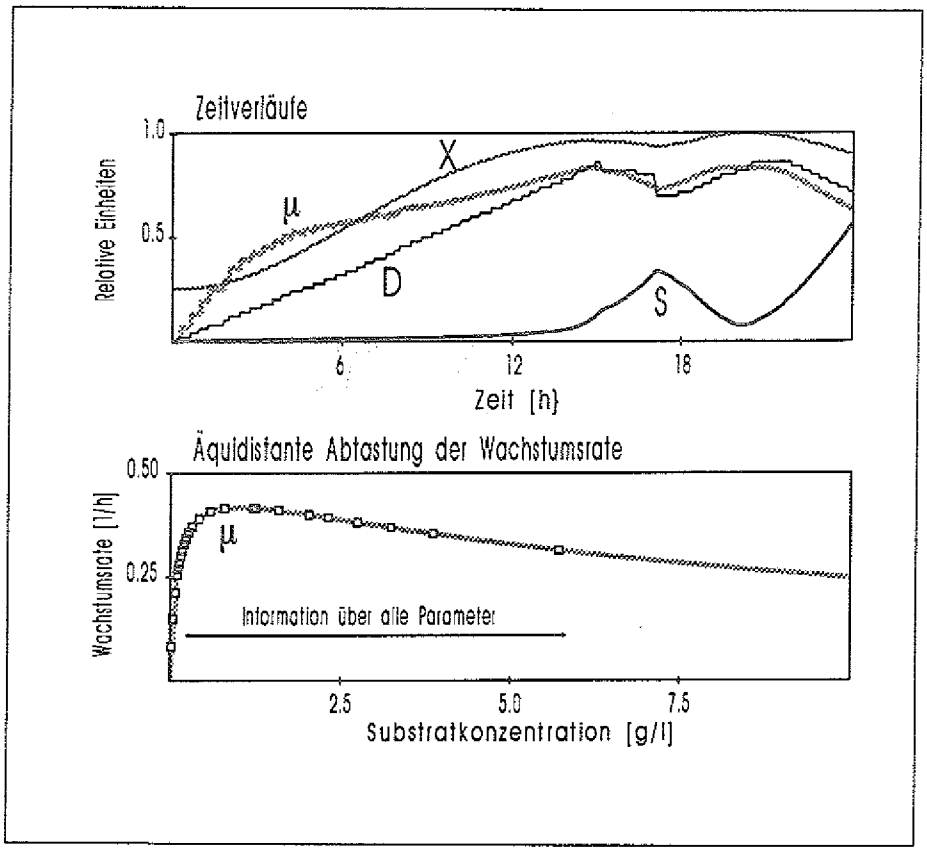


Abbildung 22.3: Identifikation bei einer instationär geführten kontinuierlichen Kultur

An dieser Stelle muß betont werden, daß der in [Mun89] beschriebene Ansatz zur optimalen Identifikation die genaue Kenntnis der Parameter (insbesondere von K_S) bereits voraussetzt! Das Resultat zeigt somit nur, wie der Prozeß hätte gefahren werden müssen, um schon bekannte Parameter optimal zu schätzen. Die Simulation hat gezeigt, daß bei leichter Veränderung der Parameter dieselbe Strategie vollkommen unterschiedliche Ergebnisse liefern kann. Weitere Überlegungen zur Verbesserung des Ansatzes finden sich in [MP89].

22.1.2.4 Verlängerte Batch-Kultur mit stetiger Zufütterung

Beim Übergang von einer gepulsten Zugabe des Substrats zu einer stetigen Veränderung von D lassen sich hervorragende Ergebnisse erzielen (Abb. 22.2b). Voraussetzung ist allerdings eine Regelungsstrategie, die bei Substratkonzentrationen im Bereich des K_S-Werts eine langsame Änderung von S garantiert. Da die Parameter des Systems nicht bekannt sind, besteht hier ein interessantes Anwendungsfeld für parameteradaptive Regler [Åst83] [Ise87].

22.1.2.5 Instationär geführte kontinuierliche Kultur

Für die Wahl einer über den gesamten Verlauf des Experiments veränderlichen Durchflußrate bietet sich eine lineare Steigerung an. Dabei bewegt sich der Prozeß ständig in der Nähe des stationären Zustands wie die Abbildung 22.3 zeigt.

Im Bereich kleiner Durchflußraten ist nun der K_S-Wert leicht bestimmbar: Der gesamte substratlimitierte Bereich der Kinetik wird langsam überstrichen. Beim Überschreiten der kritischen Durchflußrate läuft der Prozeß jedoch in die Auswaschung [YK69]. Aus diesem Grunde ist erneut eine geeignete Regelungsstrategie zu entwickeln. Diese kann z.B. auf dem Prinzip des Turbidostats oder Nutristats [Wat72] [AL84] [SCAGGM89] beruhen. Im Vergleich zum Regelproblem bei der Batch-Kultur ist dieses Problem

jedoch einfacher gelagert, da der Prozeß bei hoher Biomassekonzentration weniger sensibel auf Schwankungen in S reagiert.

Zusammenfassend ist festzustellen, daß eine Kombination aus einer Batch-Phase und einer instationär gefahrenen kontinuierlichen Phase stets die Schätzung aller Parameter ermöglicht.

22.1.2.6 Fed-Batch-Kultur

Da bei der kontinuierlichen Kultur stets das Risiko der Auswaschung besteht, ist es ratsam, das Experiment stattdessen in einer Fed-Batch-Kultur durchzuführen. Bzgl. der Datenauswertung ändert sich gegenüber einer kontinuierlichen Kultur nichts.

22.1.3 Weitere Entwicklungen

Sobald der Zeitbedarf für den Entwurf eines Experiments dessen Dauer überschreitet, ist ein Ad-Hoc-Ansatz ökonomischer. Mit Hilfe von Simulationen können leicht Zeitprofile $D(t)$ entworfen werden, die eine instationäre Parameteridentifikation über einem weiten Bereich von möglichen Parametern gewährleisten. Die Simulationsstudien sind auf interaktivem Wege schnell durchführbar. Die Beispiele zeigen jedoch auch deutlich, daß nur eine adaptive Regelung die instationäre Identifikation aller Parameter bei einem einzigen Durchlauf des Experiments erlaubt.

Die Technik der instationären Identifikation wird derzeit am Institut für Biotechnologie 2 des Forschungszentrums Jülich am Beispiel der kontinuierlichen Fermentation von *Candida boidinii* erprobt [WNSW90]. Die Erstellung geeigneter Strategien ist noch Forschungsgegenstand. Eine vertiefende Darstellung würde hier zu weit gehen.

22.1.3.1 Modularer Aufbau im Rahmen des Datenanalysesystems

Das vorgestellte Datenanalysesystem würde alle für dieses Projekt benötigten Bausteine bereitstellen. Hierzu gehören:

- eine Simulationskomponente für die Prozeßsimulation,
- eine Simulationskomponente für das Meßrauschen,
- ein erweitertes Kalman-Filter zur Schätzung der kinetischen Daten,
- ein Baustein zur Kurvenanpassung und
- eine adaptive Regler-Komponente.

22.2 On-Line-Optimierung

22.2.1 Anwendungsbereiche für heuristische Methoden

Bei vielen biotechnischen Fragestellungen ist eine Beschreibung des Systems mit deterministischen Modellen nicht sinnvoll oder zu aufwendig. Einige typische Beispiele sind:

- Abwasserprozesse mit Mischkulturen von Mikroorganismen. Ein solches System ist weder in seiner Zusammensetzung genau bekannt, noch steht die Meßtechnik zur On-Line-Erfassung relevanter Parameter zur Verfügung.
- Prozesse, die mit wechselnder Substratzusammensetzung gefahren werden, findet man vor allem bei Abwässern, aber auch bei Prozessen mit natürlichen Substraten (z.B. Zuckerrohr zur Äthanolproduktion). Eine modellmäßige Beschreibung des Wachstums in Abhängigkeit von vielen Substraten ist aufgrund der Komplexität des Systems aussichtslos.
- Reihenexperimente, die lediglich einer groben Überwachung bedürfen. Hierzu gehören z.B. die häufig durchzuführenden Testreihen für die Nährmedienzusammenstellung.

- Optimierung eines Prozesses bzgl. solcher Parameter, deren Einfluß auf das System bis heute nur unzureichend bekannt ist. Zu nennen sind hier
  - die Betriebstemperatur, die alle Stoffwechselvorgänge gleichzeitig beeinflußt und somit vielschichtige Veränderungen hervorrufen kann [1]. Die Temperatur wird deshalb meist konstant gehalten, wobei Erfahrungswerte zugrunde gelegt werden.
  - der pH-Wert, für den analoge Aussagen gelten [MOR88] [EUL+76].
  - die Nährmedienoptimierung. Vor allem bei den Spurenelementen fehlen Ansätze zur quantitativen Beschreibung ihres Einflusses. Oft ist nicht einmal bekannt, an welcher Stelle das Spurenelement in den Metabolismus eingreift. Hinzu kommt, daß die einzelnen Komponenten in der Regel nicht unabhängig voneinander wirken, so daß komplexe Abhängigkeiten entstehen. Aus diesen Gründen wird bei einer Verfahrensentwicklung stets eine feste Nährmedienzusammensetzung verwendet. Das Nährmedium ist in Pilotexperimenten so zu komponieren, daß es weder Limitierungen noch Inhibierungen verursacht und zudem kostengünstig ist. Solche Experimente sind mit erheblichem personalen und zeitlichen Aufwand verbunden.

Bei allen genannten Beispielen liegt eine Automatisierung mit Hilfe heuristischer Methoden nahe. Das Prozeßüberwachungssystem muß hierbei das Verhalten eines menschlichen Betreibers imitieren. Hier findet die nachfolgend beschriebene Klassifikationstechnik (vgl. auch [EFSW89] [Wie89] [LSF90]) ein großes Einsatzgebiet.

22.2.2 Das Optimierungsproblem

Als einfaches Beispiel zur Illustration des Ansatzes soll von einem kontinuierlich betriebenen Abwasser-Reaktor mit homogener und immobilisierter Biomasse ausgegangen werden (vgl. z.B. [Sch89]). Es soll angenommen werden, daß die Stärke des Bewuchses konstant bleibt, d.h. Wachstum und Abrieb stehen im Gleichgewicht. Ferner wird davon ausgegangen, daß die im Medium verteilte Biomasse vernachlässigbar ist. Schließlich entsteht bei der Umsetzung des Substrates ein Produkt in Form von Biogas, das gasanalytisch gemessen werden kann. Die Produktbildung sei unmittelbar an die Substrataufnahme gekoppelt. Unter diesen Annahmen ergibt sich das einfache Modell:

$$\begin{aligned} \dot{X} &= 0 \\ \dot{S} &= -\sigma(S) \cdot X + (S_{in} - S) \cdot D \end{aligned} \quad \text{mit } \pi(S) = Y_{P/S} \cdot \sigma(S) \ .$$

Zu optimieren ist nun die Produktbildung

$$P = Y_{P/S} \cdot \sigma(S) \cdot X \ ,$$

wobei als Einflußgröße der Durchfluß D variiert werden kann.

Der Prozeß wird hierbei chargenweise gefahren, so daß für jede neue Zusammensetzung des Abwassers ein neues Optimum anzustreben ist. Die Optimierung kann also nicht statisch erfolgen, da die Zusammensetzung des Mediums veränderlich ist. Bei der nachfolgenden Simulationsstudie für den Optimiervorgang liegt eine Inhibierungskinetik für $\sigma(S)$ zugrunde (vgl. auch 22.1), so daß das System einen optimalen Arbeitspunkt aufweist.

22.2.2.1 Heuristische Vorgehensweise eines Betreibers

Der Betreiber einer solchen Abwasseranlage wird zur Optimierung eine heuristische Suchstrategie verwenden, in die auch frühere Erfahrungen einfließen können. Die Abbildung 22.4 zeigt einen solchen manuell durchgeführten Optimierungsvorgang in der Simulation. Die Durchflußrate D konnte hierzu während des Durchlaufs interaktiv verändert werden.

Die nach mehrfachem Durchlauf der Simulation entwickelte Strategie kann in drei iterativ wiederholte Schritte zerlegt werden:

[1] Bei einigen Systemen wurde eine modellmäßige Beschreibung der Temperaturabhängigkeit des Wachstums versucht [MOR88] [EUL+76]. Eine verfahrenstechnische Anwendung ist jedoch nicht bekannt.

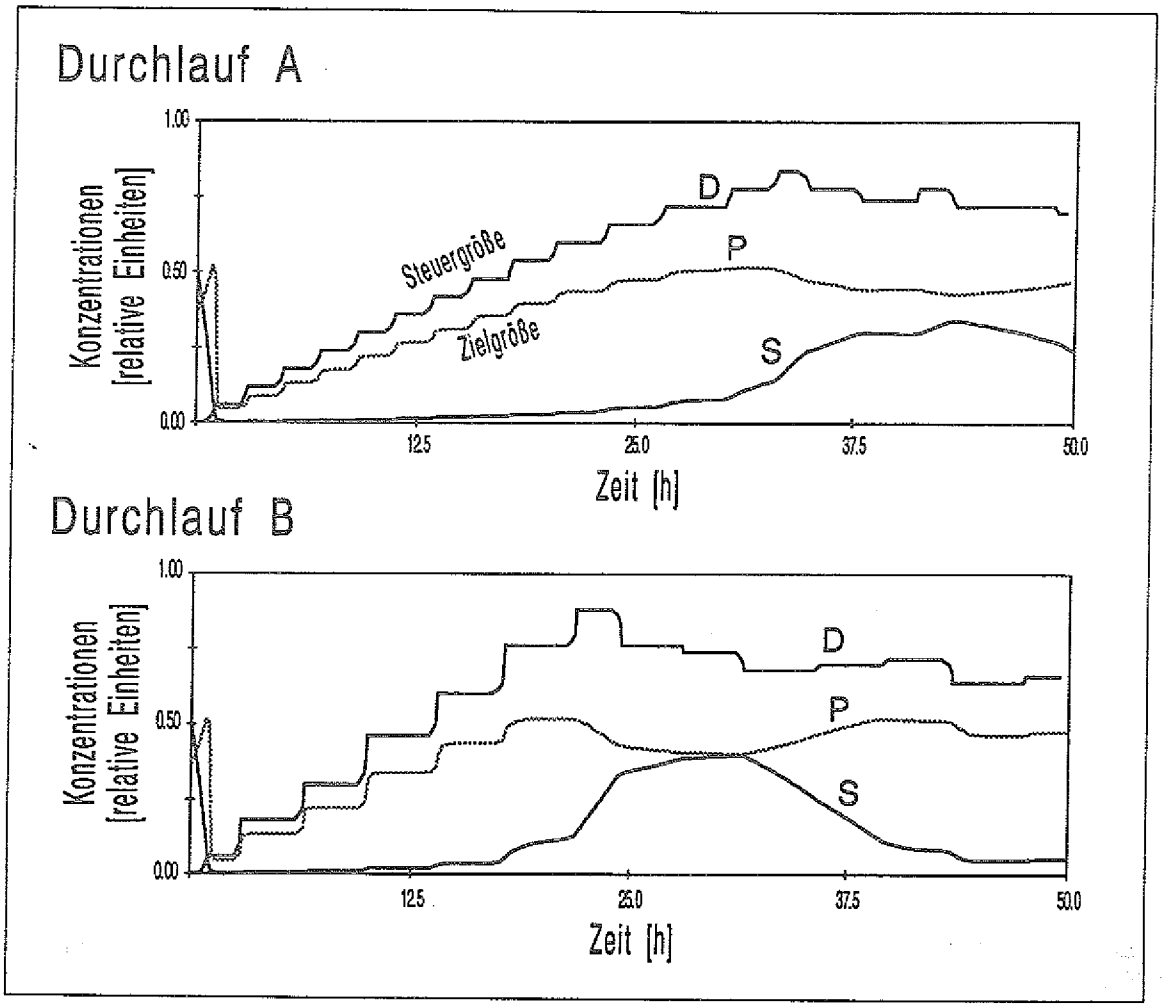


Abbildung 22.4: Zwei Beispiele für die heuristische Optimierung eines Abwasserprozesses durch einen Betreiber

1. Einstellen einer festen Durchflußrate D.
2. Abwarten einer stationären Produktbildung P.
3. Abändern von D nach einem Optimierungsprinzip und erneuter Durchlauf der Schritte 1-3.

Auf Detailprobleme, die mit diesem groben Schema verbunden sind, soll noch nicht eingegangen werden. Wichtig ist zunächst nur, daß die Strategie entscheidend auf einer Mustererkennung im Verlauf von P beruht. Relevant sind Stufen, Plateaus, Maxima und Minima, schnelle Veränderungen etc. Es soll nun versucht werden, diese Mustererkennung zu automatisieren.

22.2.3 Klassifikationsansatz

22.2.3.1 Merkmalsraum

Das erste Problem bei einer heuristischen Klassifikation ist die Festlegung eines geeigneten Merkmalsraums (vgl. 10.7). Da im wesentlichen Trends und Plateaus sowie Maxima und Minima zu erkennen sind, bietet sich ein zweidimensionaler Merkmalsraum an, der von der ersten und zweiten Ableitung von P zu einem Zeitpunkt t aufgespannt wird. Der Merkmalsvektor zum Zeitpunkt t ist also

$$(\dot{P}(t), \ddot{P}(t)) \ .$$

Des weiteren ist festzulegen, welche Klassen unterschieden werden sollen. Die gröbste Unterscheidung, die in Hinblick auf die Problemstellung sinnvoll erscheint, ist gegeben durch:

$$\dot{P}(t) \begin{cases} \gg 0 & \text{(Aufwärtstendenz)} \\ \sim 0 & \text{(Plateau, Minimum oder Maximum)} \\ \ll 0 & \text{(Abwärtstendenz)} \end{cases}$$

und

$$\ddot{P}(t) \begin{cases} \gg 0 & \text{(Positive Krümmung)} \\ \sim 0 & \text{(Linearer Verlauf)} \\ \ll 0 & \text{(Negative Krümmung)} \end{cases}$$

Durch Kombination ergeben sich hieraus 9 Klassen, die mit

$$\begin{array}{lll} \Omega_{\gg\gg}, & \Omega_{\gg\sim}, & \Omega_{\gg\ll}, \\ \Omega_{\sim\gg}, & \Omega_{\sim\sim}, & \Omega_{\sim\ll}, \\ \Omega_{\ll\gg}, & \Omega_{\ll\sim}, & \Omega_{\ll\ll}, \end{array}$$

bezeichnet werden, wobei der erste Index für $\dot{P}(t)$ und der zweite für $\ddot{P}(t)$ steht. Die exakte Bedeutung der Relationen $\ll$, $\sim$, $\gg$ bleibt noch zu präzisieren.

22.2.3.2 Berechnung des Merkmalsvektors

Zur Schätzung von $\dot{P}(t)$ und $\ddot{P}(t)$ stehen diverse Techniken zur Verfügung (Regression, Filterung, Fensterschätzer). Um den Blick auf die wesentlichen Aspekte nicht unnötig zu verstellen, wurde bei der Simulation kein Rauschen aufgeschaltet. Es genügt daher ein einfacher lokaler Regressionsansatz in einer Umgebung von t bzgl. des Parabel-Modells

$$P(t+\tau) = \alpha + \beta \cdot \tau + \frac{1}{2}\gamma \cdot \tau^2 + \epsilon \quad \text{auf} \quad [t-\Delta t, t+\Delta t] \,.$$

D.h. es wird jeweils auf einem kleinen Zeitintervall $[t-\Delta t, t+\Delta t]$ um den derzeitigen Zeitpunkt t eine Regressionsparabel angepaßt. Deren Steigung β und Krümmung γ dienen dann als Schätzer für die tatsächliche Steigung und Krümmung von $P(t)$:

$$\hat{\dot{P}}(t) := \beta, \quad \hat{\ddot{P}}(t) := \gamma$$

22.2.3.3 Lernstichprobe

Zur Parametrisierung des heuristischen Klassifikators wird nun eine Lernstichprobe benötigt (vgl. 10.7). Diese erhält man aus einem Simulationsexperiment: Zu einem Zeipunkt t wird der Kurvenverlauf über dem Intervall $[t-\Delta t, t+\Delta t]$ aufgezeichnet. Der Betreiber hat diesem Verlauf subjektiv eine der Klassen $\Omega_{i,j}$ zuzuordnen. Nach Auswertung der Schätzer $\hat{\dot{P}}(t)$, $\hat{\ddot{P}}(t)$ wird dann die Klassifikation

$$(\hat{\dot{P}}(t),\ \hat{\ddot{P}}(t)) \quad \longrightarrow \quad \Omega_{i(t),j(t)}$$

in die Lernstichprobe aufgenommen. Der Vorgang wird in Abbildung 22.5 veranschaulicht.

Ist eine genügend große Anzahl von Beispielen gesammelt, so kann die Lernstichprobe im Merkmalsraum visualisiert werden. Abbildung 22.6 zeigt dies beispielhaft für zwei Klassen. Die Klassifikation abstrahiert damit eine Schar von Kurvenverläufen zu einem Grobschema (Maximum, Minimum, linearer Anstieg, etc.).

22.2.3.4 Heuristischer Klassifikator

Die Abbildung 22.7 zeigt die Gesamtheit aller Lernstichproben aus den 9 Klassen eingezeichnet in den Merkmalsraum. Anhand der Stichprobe muß nun eine numerische Präzisierung der Relationen $\ll$, $\sim$ und $\gg$ erfolgen. Gesucht wird ein möglichst einfach parametrisiertes Schema mit einer geringen Zahl

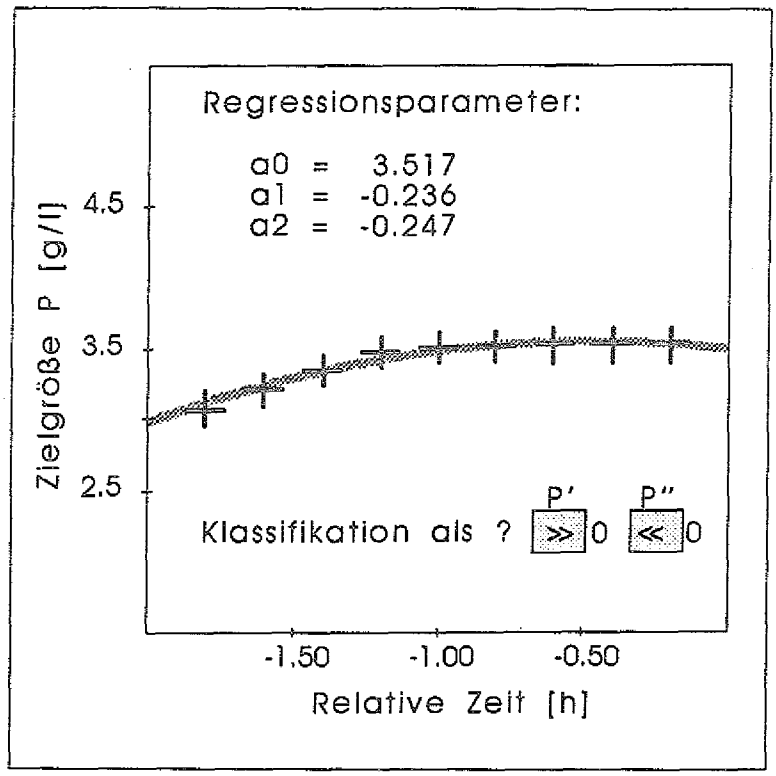


Abbildung 22.5: Interaktive Zuordnung eines Kurvenverlaufs zu einer Klasse

von Fehlklassifikationen. Die einfachste Lösung hat Klassengrenzen, die parallel und symmetrisch um die X- und Y-Achse liegen:

$$\dot{P}(t)\begin{cases} \gg 0 & \text{falls} & & & \beta_0 < \dot{P}(t) \\ \sim 0 & \text{falls} & & -\beta_0 \leq \dot{P}(t) \leq \beta_0 & \\ \ll 0 & \text{falls} & \dot{P}(t) < & -\beta_0 & \end{cases}$$

und

$$\ddot{P}(t)\begin{cases} \gg 0 & \text{falls} & & & \gamma_0 < \ddot{P}(t) \\ \sim 0 & \text{falls} & & -\gamma_0 \leq \ddot{P}(t) \leq \gamma_0 & \\ \ll 0 & \text{falls} & \ddot{P}(t) < & -\gamma_0 & \end{cases}$$

Eine Inspektion der Abbildung 22.7 zeigt, daß dieser Ansatz nur mit einem großen Fehlklassifikationsrisiko zu betreiben wäre. Nach Erprobung einiger einfacher Ansätze mit gekrümmten Klassengrenzen kann man aber von den geradlinigen zu parabelförmigen Bereichsgrenzen übergehen und erhält die Präzisierung:

$$\dot{P}(t)\begin{cases} \gg 0 & \text{falls} & & & \beta_0+\beta_1\ddot{P} < \dot{P}(t) \\ \sim 0 & \text{falls} & & -\beta_0-\beta_1\ddot{P} \leq \dot{P}(t) \leq \beta_0+\beta_1\ddot{P} & \\ \ll 0 & \text{falls} & \dot{P}(t) < & -\beta_0-\beta_1\ddot{P} & \end{cases}$$

und

$$\ddot{P}(t)\begin{cases} \gg 0 & \text{falls} & & & \gamma_0+\gamma_1\dot{P} < \ddot{P}(t) \\ \sim 0 & \text{falls} & & -\gamma_0-\gamma_1\dot{P} \leq \ddot{P}(t) \leq \gamma_0+\gamma_1\dot{P} & \\ \ll 0 & \text{falls} & \ddot{P}(t) < & -\gamma_0-\gamma_1\dot{P} & \end{cases}$$

Zwar kommt auch dieser Ansatz nicht ganz ohne Fehlklassifikationen aus, was jedoch unproblematisch ist, da die Klassengrenzen ohnehin einen unscharfen Verlauf haben. Auch die subjektive Empfindung ist in strittigen Fällen nicht zu einer eindeutigen Klassifikation in der Lage. Die Erfahrung mit dem Simulationsbeispiel hat gezeigt, daß ein und derselbe Merkmalsvektor vom Betreiber zu unterschiedlichen Zeiten unterschiedlichen Klassen zugeordnet wurde.

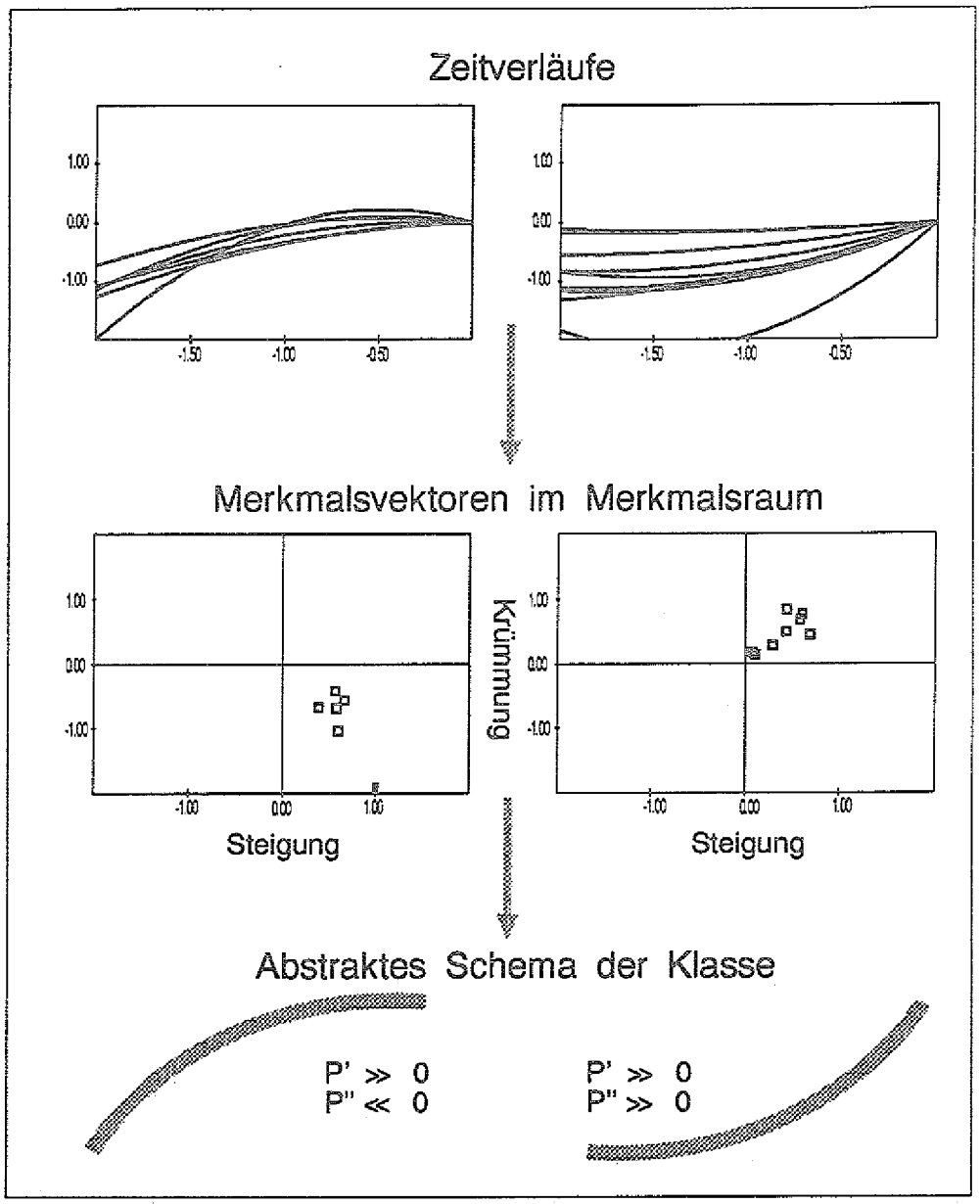


Abbildung 22.6: Auftragung der Objekte zweier Klassen im Merkmalsraum

22.2.3.5 Durchlauf

Die Abbildung 22.8 veranschaulicht die Anwendung des Klassifikators auf einen Kurvenverlauf. Der Zeitverlauf wird in Abschnitte zerlegt, die jeweils einer Klasse entsprechen. Es entsteht eine Sequenz

$$\Omega_{i_1,j_1}, \Omega_{i_2,j_2}, \Omega_{i_3,j_3}, \ldots ,$$

die einer syntaktischen Klassifikation als Eingabe dienen kann (vgl. 8.4).

22.2.4 Ausblick über weitere Entwicklungen

22.2.4.1 Weitere Probleme bei der On-Line-Optimierung

Mit der lokalen Mustererkennung im Kurvenverlauf ist ein wichtiges Teilproblem der On-Line-Optimierung gelöst. Der Klassifikator kann während eines manuell gesteuerten Experiments angelernt werden.

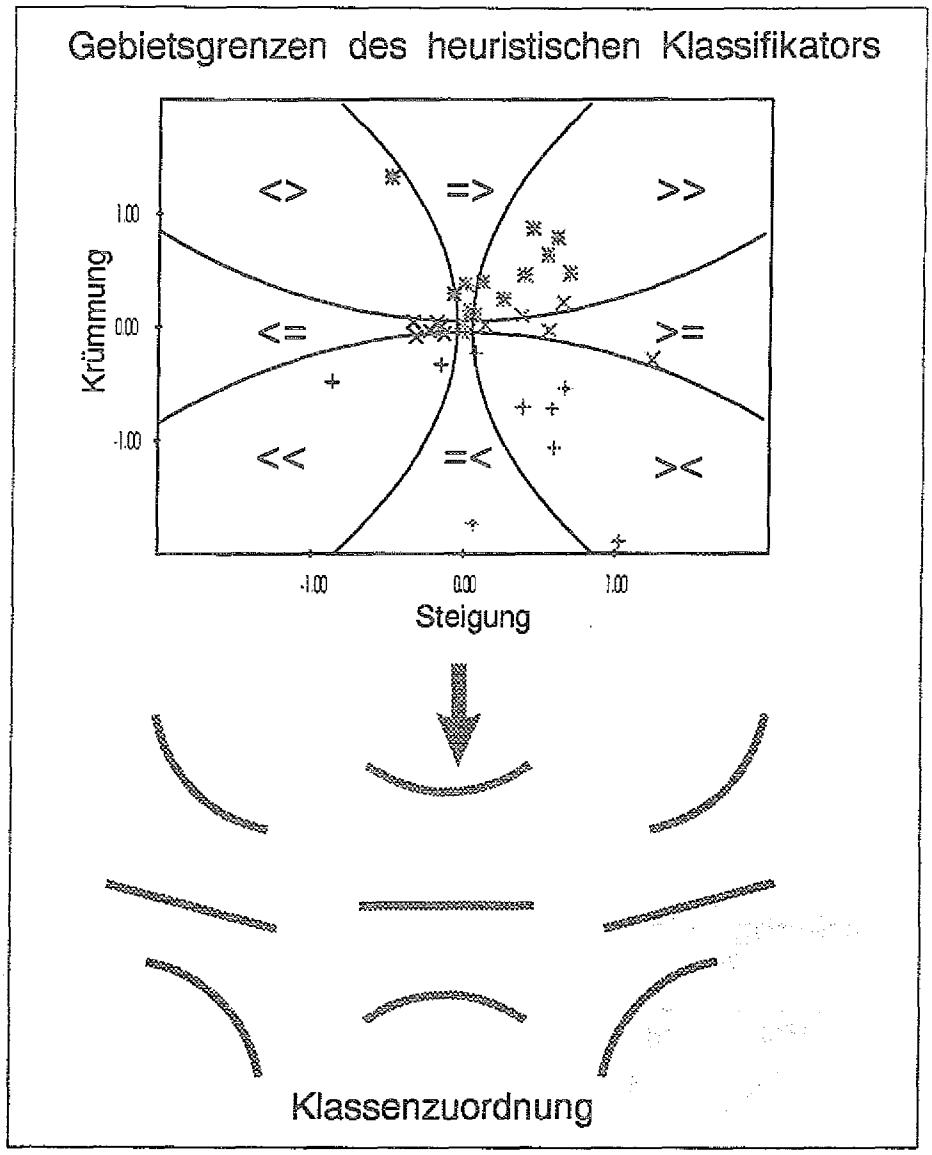


Abbildung 22.7: Heuristischer Klassifikator

Eine interaktive Realisierung der Lernphase trägt hierbei wesentlich zur Verringerung des Zeitaufwands bei. Der heuristische Ansatz kann bei den verschiedensten Prozessen verfolgt werden.

Dennoch ist bisher nur ein Aspekt der On-Line-Optimierung abgedeckt. Weitere Probleme sollen kurz angeschnitten werden:

- Der Regressionsschätzer für den Merkmalsvektor ist ggfs. durch ein anderes Verfahren zu ersetzen, das geeignet zu parametrisieren ist (vgl. z.B. [SY90]). Die Störungsempfindlichkeit des Schätzers wäre im Vorfeld zu untersuchen.
- An die lokale heuristische Klassifikation muß sich eine syntaktische Klassifikation anschließen, die Abschnitte in der Klassensequenz (z.B. Aufwärts- und Abwärtsstufen) erkennt. Diese muß nicht nur die für die Optimierung relevanten Strukturen erkennen, sondern auch Störungszustände und kritische Situationen (z.B. schnelle Abwärtstrends).
- Für die eigentliche Optimierung ist ein ableitungsfreies Optimierungsverfahren zu konfigurieren, welches aufgrund der stationären Werte, die P auf den Plateaus annimmt, den nächsten einzustellenden Wert für D berechnet.

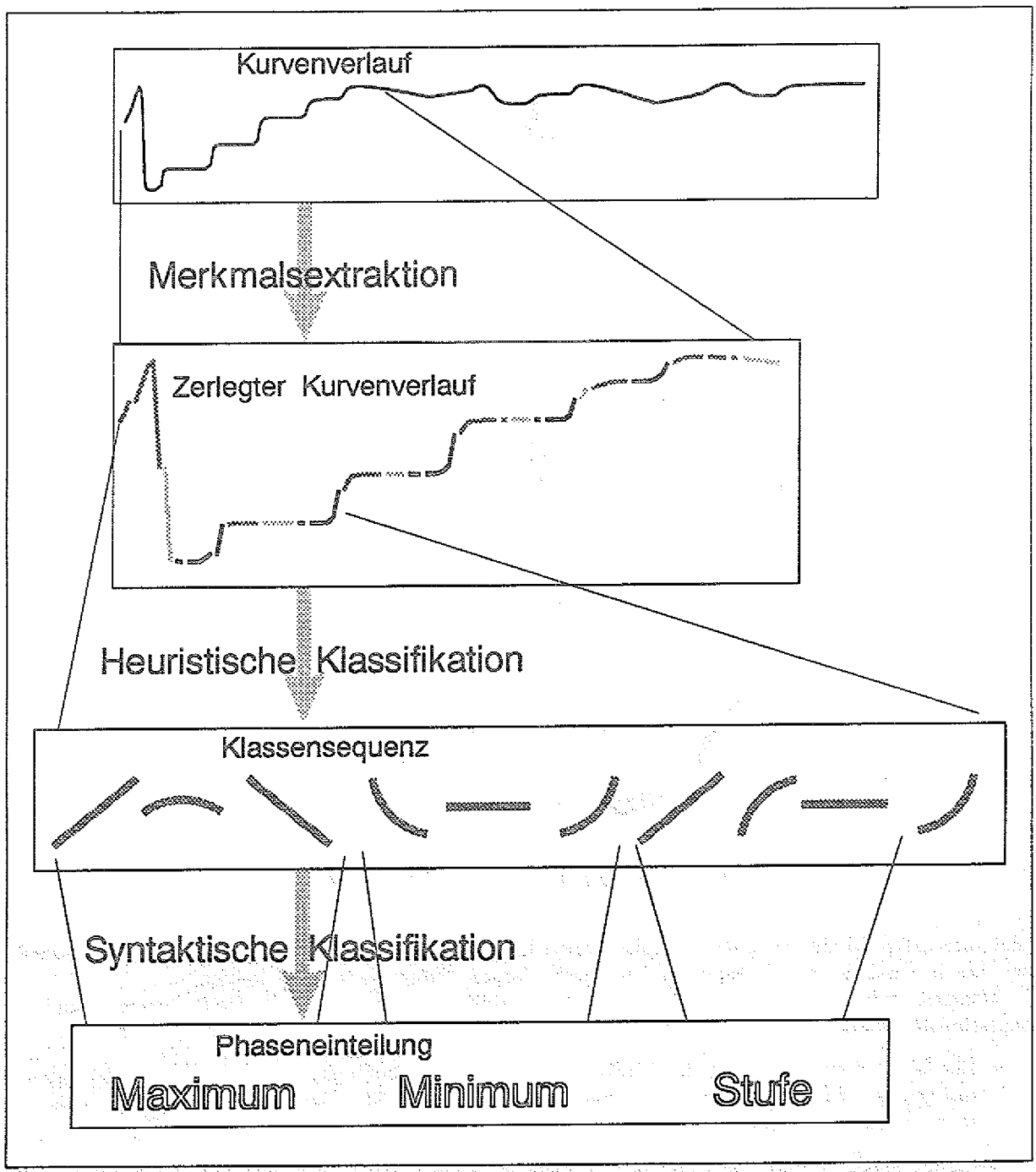


Abbildung 22.8: Durchlauf des heuristischen Klassifikators

- Schließlich muß der Gesamtablauf überwacht und gesteuert werden. Für die Ablaufsteuerung bei störungsfreiem Ablauf bietet sich hierfür eine algorithmische Lösung an.
- Eine weitere heuristische Komponente (z.B. ein Regelsystem) kann in kritischen Situationen die Kontrolle übernehmen. Sie enthält Erfahrungswerte und heuristische Strategien für die Stabilisierung des Prozesses. Ferner könnte eine solche Komponente die Startwertwahl für die Optimierung besorgen.

22.2.4.2 Modularer Aufbau

Die skizzierte Problemlösung zeigt einen modularen Aufbau aus den Komponenten:

- Schätzer für den Merkmalsvektor,
- heuristischer Klassifikator,
- syntaktischer Klassifikator,
- Optimierverfahren,
- Ablaufsteuerung,
- heuristische Störungsbehandlungskomponente.

Sie stellt hohe Anforderungen an alle Bestandteile des Datenanalyse-Konzepts und ist ein hervorragendes Testbeispiel für die zukünftige Entwicklung.

22.2.4.3 Weiterentwicklung

In den letzten Jahren ist die On-Line-Optimierung von vielen Seiten angegangen worden (vgl. z.B. [HLS89] [GBY90] [PW90] [CL90]). Sie stellt auch einen Forschungsschwerpunkt am Institut für Biotechnologie 2 des Forschungszentrums Jülich dar. In den letzten Jahren wurden bereits vielversprechende Lösungen für die Temperatur- und pH-Optimierung entwickelt [FESW89] [EFSW89]. Sie beruhen im Prinzip auf einem Mustererkennungsansatz und berücksichtigen bereits einen Teil der aufgeführten Detailprobleme.

Eine wesentlich schwierigere Aufgabe ist die Nährmedienoptimierung, welche mehrere Größen gleichzeitig berücksichtigen muß. Dieses Feld soll in den nächsten Jahren intensiv bearbeitet werden.

22.3 Dynamische Methode zur Bestimmung des kLa-Werts

22.3.1 Durchführung des Experiments

Die sogenannte *dynamische Methode* zur $k_L a$-Bestimmung [LSV88] beruht auf dem in 5.3.4 vorgestellten Modell

$$\dot{O} = k_L a \cdot (O^* - O) - \omega \cdot X \quad .$$

Sie hat den großen Vorteil, daß sie am laufenden Prozeß durchgeführt werden kann.

22.3.1.1 Grundannahmen

Bei der Durchführung der Methode liegen drei Annahmen zugrunde:

1. Während des (kurzen) Zeitraums des Experiments ist X konstant.
2. Die Sauerstoffkonzentration O bewegt sich während des Experiments nicht in einem für den Organismus limitierenden oder inhibierenden Bereich. Insbesondere ist $\omega = \omega(O) = \text{const}$.
3. Die Trägheit der O_2-Elektrode ist vernachlässigbar gegenüber der Geschwindigkeit der Konzentrationsänderung [2].

[2] Diese Annahme kann entfallen, wenn man das Elektrodenverhalten zusätzlich modelliert (siehe [MYSZ90]). An der Methodik ändert sich dadurch prinzipiell nichts.

22.3.1.2 Ablauf

Unter den genannten Bedingungen kann folgende zweiphasige Strategie zur Bestimmung von k_La gewählt werden (vgl. [SSW90]):

Phase 1: Abschalten der Zuluft, gleichbedeutend mit einem Absinken des k_La-Werts auf 0 (vgl. 5.3.4).

Phase 2: Wiedereinschalten der Belüftung, sobald die O_2-Konzentration eine als limitierend angesehene kritische Grenze unterschreitet.

22.3.2 Theoretischer Kurvenverlauf

Aus der Theorie und den gemachten Annahmen folgt unmittelbar:

$$\dot{O} = \begin{cases} -\omega \cdot X & \text{in Phase 1} \\ k_La \cdot (O^* - O) - \omega \cdot X & \text{in Phase 2} \end{cases}$$

D.h. es liegt zunächst ein linearer Abfall mit Steigung $-\omega \cdot X$ vor, der gefolgt wird von einem umgekehrt exponentiellen Anstieg auf den stationären Wert O_∞ (vgl. Abb. 22.9a).

22.3.3 Auswertung des Experiments

22.3.3.1 Parameteranpassung

Aufgrund des theoretisch zu erwartenden Kurvenverlaufs erscheint eine Auswertung des Experiments mit Hilfe einer Kurvenanpassung sinnvoll. Dabei kann zunächst das Produkt $\omega \cdot X$ (Gesamt-Sauerstoffverbrauch) aufgrund der Daten aus Phase 1 mit Hilfe einer linearen Regression geschätzt werden. Das so geschätzten $\omega \cdot X$ ermöglicht dann eine nichtlineare Regression für Phase 2, mit der die übrigen Parameter k_La und O^* bestimmt werden können. Die Kenntnis von $\omega \cdot X$ aus Phase 1 ist hierzu unbedingt erforderlich, da das nichtlineare Regressionsproblem bzgl. der Parameter k_La, O^* und $\omega \cdot X$ nicht eindeutig bestimmt ist.

22.3.3.2 Problematik der Parameteranpassung

Der Ansatz zur Parameterschätzung erweist sich bei realen Daten als unbrauchbar. Dies ergibt sich erst aus der praktischen Durchführung des Experiments [SSW90]: Die Annahme, daß bei abgeschalteter Zuluft sogleich gilt $k_La = 0$, ist falsch, da der Rührer auch weiterhin Sauerstoff über die Luft-Wasser-Grenzschicht einträgt. Aus diesem Grunde muß zugleich mit der Zuluft auch der Rührer abgeschaltet werden. Die Konsequenz ist eine inhomogene Durchmischung des Mediums zum Zeitpunkt des Wiederanschaltens! Der Kurvenverlauf zeigt vor allem bei niedrigen Drehzahlen Irregularitäten, wie sie in Abbildung 22.9b zu erkennen sind. Erst kurze Zeit nach Wiederanschalten der Belüftung ist die Homogenität des Mediums wiederhergestellt und es stellt sich der vorausgesagte Kurvenverlauf ein.

Es liegt hier ein typisches Beispiel für ein System mit einer nicht modellmäßig beschreibbaren Verzögerungsphase vor. Eine kurze Studie soll zeigen, wie sich die Anwendung des idealen Modells auf das zeitverzögerte System auswirkt: Ausgegangen wird von dem einfachen System

$$O(t) = \begin{cases} 0 & \text{für} \quad 0 \le t \le t_{Lag} \\ 1 - e^{-(t+t_{Lag})} & \text{für} \quad t_{Lag} \le t \le t_{max} \end{cases},$$

das die wesentlichen Aspekte des realen Systems in Phase 2 wiedergibt. Die Parameteranpassung erfolgt nun mit dem fehlerhaften Modell

$$O(t) = \alpha - e^{-\beta t}, \quad 0 \le t \le t_{max}$$

bzgl. der Parameter α und β. Für $t_{Lag} = 0$ verschwindet der Fehler und die korrekten Parameterschätzer sind $\hat{\alpha} = 1$, $\hat{\beta} = 1$. Die Abbildung 22.10 zeigt einige Anpassungsergebnisse für verschiedene Werte von t_{Lag} und t_{max} bei Anwendung der KQ-Methode (vgl. 10.2.2.2). Die Tabelle 22.1 gibt die entsprechenden numerischen Resultate wieder.

Die Studie zeigt, daß im Falle $t_{max} = 10$ (welcher der realen Situation besser entspricht, vgl. Abb. 22.9) bereits kleinste Änderungen in t_{Lag} einen großen Schätzfehler in β (welches den k_La-Wert enthält) nach sich ziehen. Hingegen wird α (entsprechend O_∞) stets zufriedenstellend geschätzt.

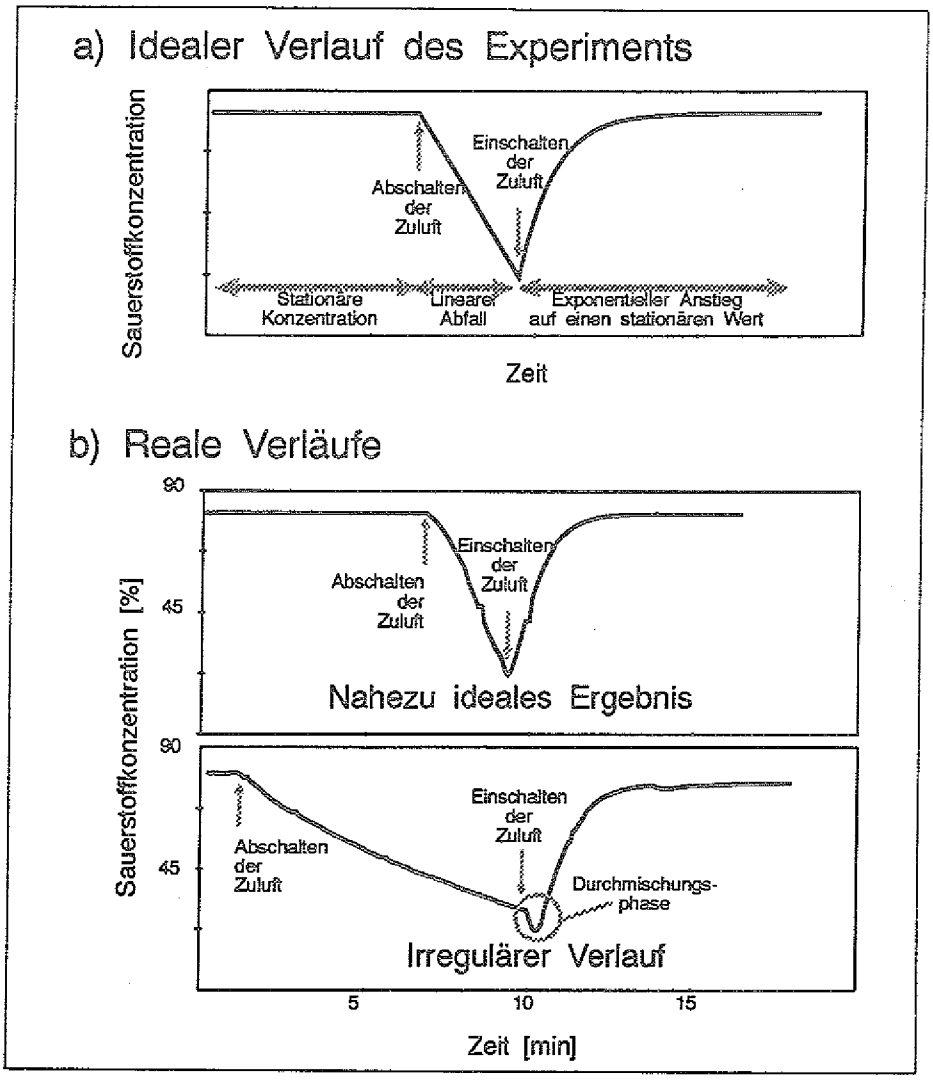


Abbildung 22.9: Idealer und realer Zeitverlauf des dynamischen k_La-Experiments

$\alpha = 1,\ \beta = 1$					
$t_{max} = 3$			$t_{max} = 10$		
t_{Lag}	$\hat{\alpha}$	$\hat{\beta}$	t_{Lag}	$\hat{\alpha}$	$\hat{\beta}$
0.0	1.00	1.00	0.0	1.00	1.00
0.1	1.00	1.00	0.1	1.00	0.91
0.2	1.98	0.88	0.2	1.00	0.82
0.3	0.97	0.82	0.3	1.00	0.76
0.4	0.96	0.77	0.4	1.00	0.70
1.0	0.88	0.55	1.0	1.01	0.46
			2.0	1.00	0.32

Tabelle 22.1: Numerische Ergebnisse für die Schätzer im k_La-Modell bei nicht berücksichtigter Zeitverzögerung

22.3.3.3 Hybrid-Methode

Da eine modelmäßige Beschreibung der Verzögerungsphase nicht sinnvoll ist, kann eine Parameteranpassungsmethode folglich nur dann angewendet werden, wenn die Zeitverzögerung vorher bestimmt und die

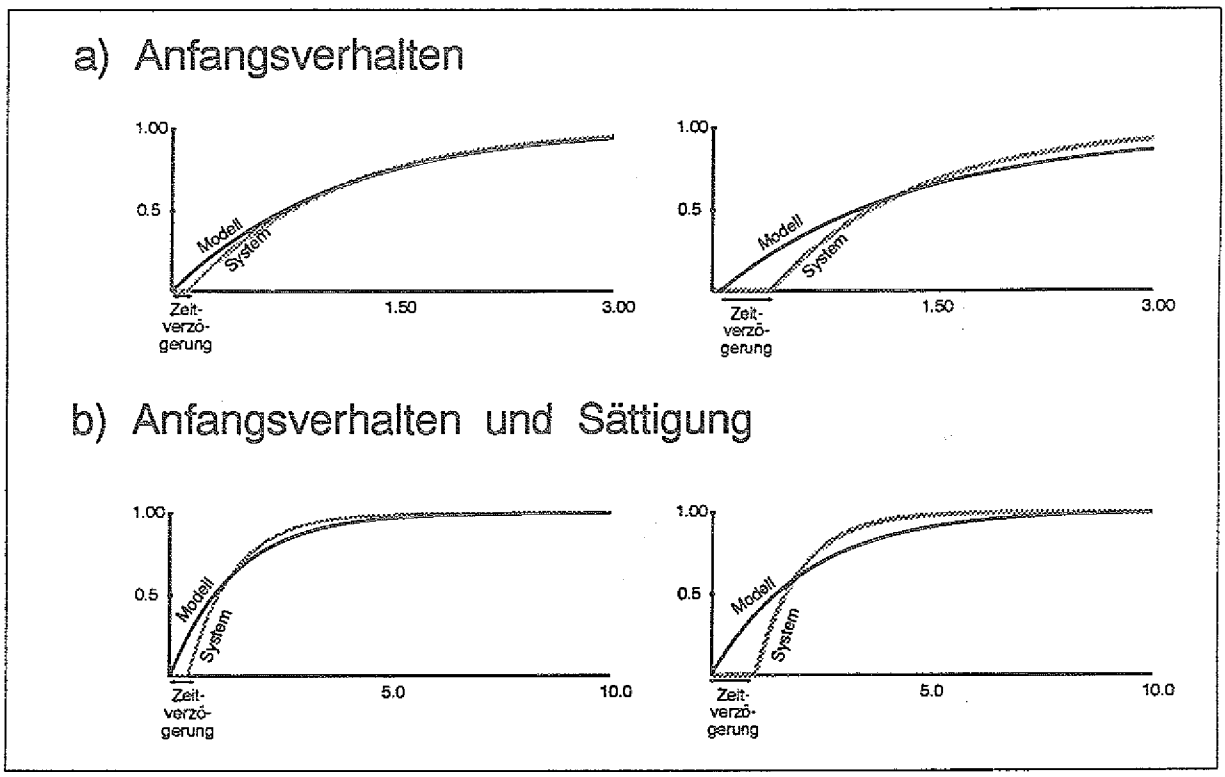


Abbildung 22.10: Parameteranpassung für das $k_L a$-Modell bei nicht berücksichtigter Zeitverzögerung

Methode ausschließlich auf den regulären Kurvenverlauf aufgesetzt wird. Dies führt auf einen hybriden Ansatz, der drei Schritte umfaßt:

1. Bestimmung von $\omega \cdot X$ aus den Daten von Phase 1 mit Hilfe einer linearen Regression.
2. Ermittlung der Verzögerungszeit bis zum Eintreten eines regulären Kurvenverhaltens.
3. Kurvenanpassung auf dem regulären Endstück von Phase 2 zur Bestimmung von $k_L a$ und O^*.

22.3.4 Realisierung

Die skizzierte Methode zur dynamischen $k_L a$-Bestimmung ist ein schönes Beispiel für die Anwendung eines Phasenkonzepts, wie es innerhalb des Datenanalysesystems vorgesehen ist (siehe 10.3.2.2). Wenn die entsprechenden Datenanalysebausteine und die Möglichkeiten zur Ablaufsteuerung gegeben sind, läßt sich in verhältnismäßig kurzer Zeit daraus eine fertige Lösung erstellen.

22.3.4.1 Interaktive Lösung

Eine interaktiv gesteuerte Realisierung der Hybridmethode wurde am Institut für Mikrobiologie und Biotechnologie der Universität Bonn in Zusammenarbeit mit der Abteilung Theoretische Biologie im Rahmen eines Blockpraktikums erstellt [3]. Die Auswertung erfolgt off-line. Eine Festlegung der Phasengrenzen ist mit Hilfe eines graphischen Zeigers leicht vom Benutzer zu bewerkstelligen (vgl. Abb. 22.11). Dabei wird sowohl für das lineare als auch für das exponentielle Teilstück ein geeigneter Zeitbereich zur Anpassung ausgewählt. Auf dem exponentiellen Teilstück erfolgt auf diese Weise zugleich auch die Eingabe des Startwerts O_0 für die Kurvenanpassung [4]. Startwerte für die nichtlineare Regression ergeben sich aus einer Grobschätzung mit Hilfe einer Ad-Hoc-Methode. Bei allen vorliegenden Beispielen konnte so

[3] Ich danke Herrn Michael Seipel für die zur Verfügung gestellte Kopie des Programms.
[4] Der Startwert könnte durchaus auch als dritter Parameter in die Regression aufgenommen werden.

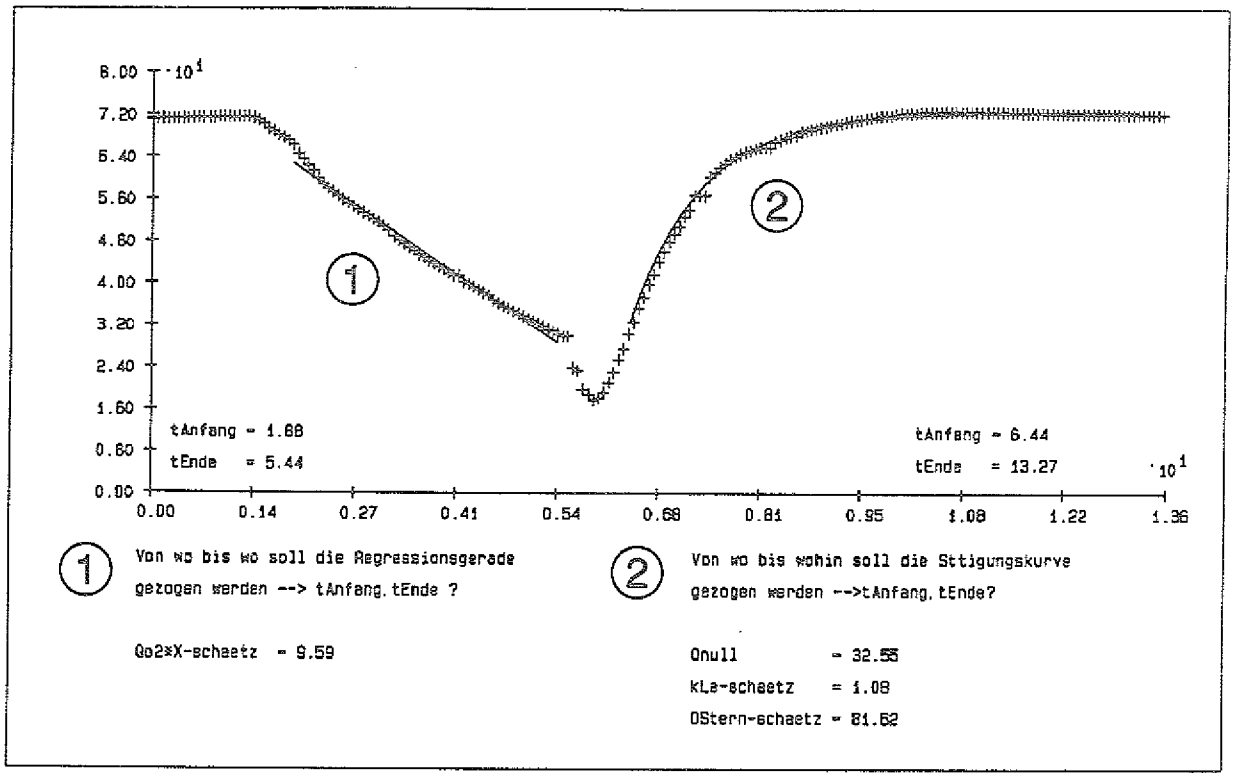


Abbildung 22.11: Durchlaufbeispiel für die interaktiv gesteuerte $k_L a$-Bestimmung

eine Konvergenz des einfachen Newton-Verfahrens zur Minimierung erreicht werden. Der Ablauf des Verfahrens wird durch Neuzeichnen der angepaßten Kurve nach jedem Iterationsschritt visualisiert. Der Benutzer hat damit ein hohes Maß an Kontrolle für die Konvergenz des Verfahrens.

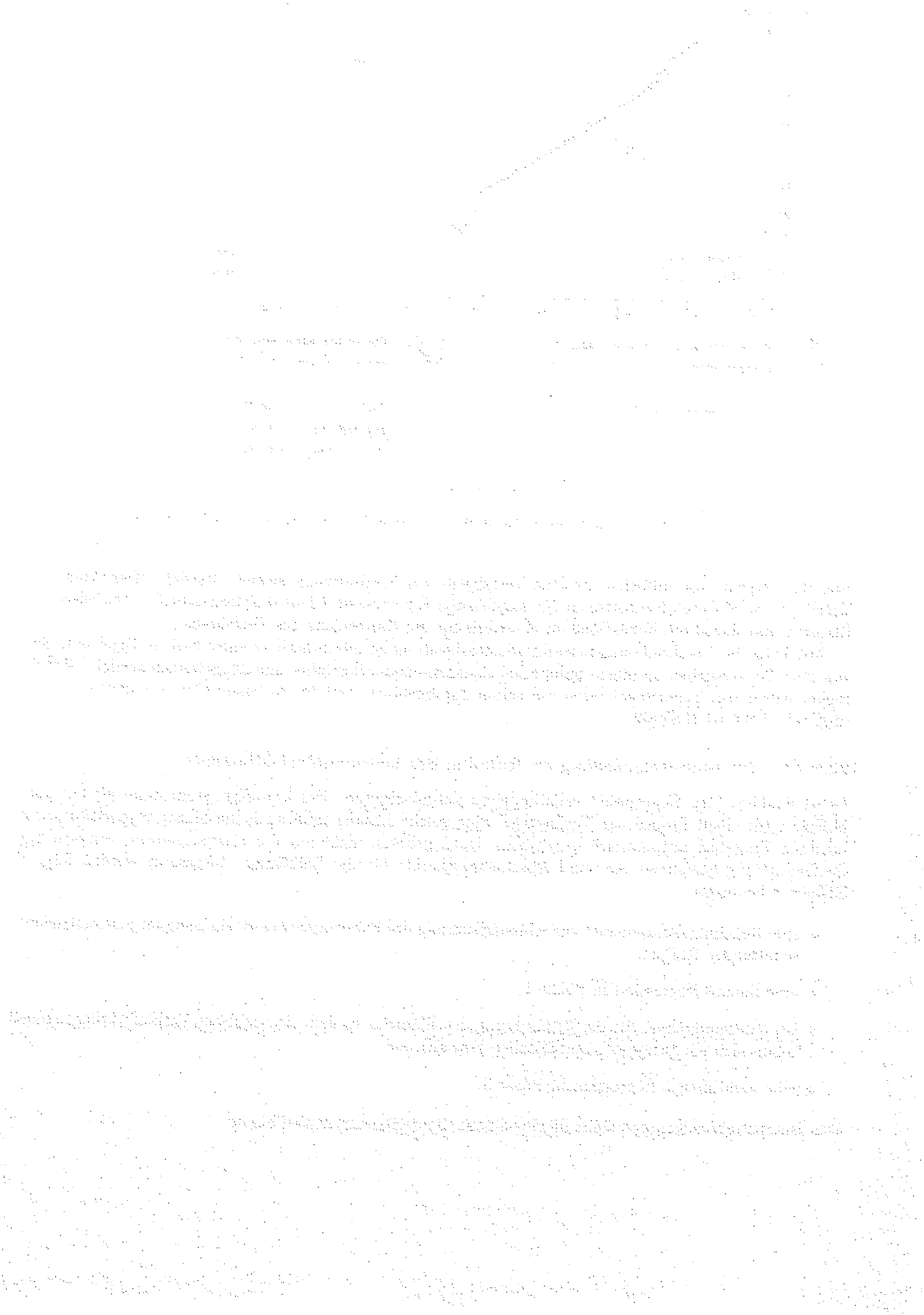
Mit Hilfe der beschriebenen interaktiven Realisierung konnte innerhalb eines halben Tages eine Serie von etwa 20 Versuchen an einem halophilen Organismus durchgeführt und ausgewertet werden [SSW90]. Bisher wurde das Experiment meist nur selten durchgeführt, weil der Aufwand einer manuellen Auswertung sehr hoch ist [LSV88].

22.3.4.2 On-Line-Realisierung im Rahmen des Datenanalyse-Konzepts

Es ist denkbar, das Experiment vollständig zu automatisieren. Der $k_L a$-Wert steht dann als On-Line--Meßwert für einen Prozeß zur Verfügung! Eine solche Lösung muß auch die bisher vom Benutzer gemachten Eingaben automatisch berechnen. Dazu gehören nicht nur die Phasengrenzen, sondern auch die Ermittlung geeigneter An- und Abschaltzeitpunkte für die Belüftung. Insgesamt werden folgende Elemente benötigt:

- eine Regelungskomponente zur Ablaufsteuerung des Experiments (z.B. ein geeignet parametrisierter schaltender Regler),
- eine lineare Regression in Phase 1,
- ein Phasendetektor für die Bestimmung des Bereichs, in dem ein regulärer Verlauf vorliegt (hierzu bietet sich ein geeigneter statistischer Test an) und
- eine nichtlineare Regression in Phase 2.

Das Datenanalyse-Konzept stellt hierfür erneut die geeigneten Mittel bereit.



Teil VI

Anhang

Anhang A

Technische Anforderungen für eine Implementierung

Die bei der Erstellung der Prototypen gemachten Erfahrungen sollen in Hinblick auf eine spätere Fortsetzung des Projekts kurz ausgewertet werden. Dazu werden zunächst mögliche Soft- und Hardwarekonstellationen überblicksartig vorgestellt und verglichen.

A.1 Software

Die Softwaregrundlage sollte so beschaffen sein, daß rasch eine lauffähige Version erstellt werden kann ("Rapid Prototyping"). Gleichzeitig müssen wesentliche Teile auf einen Front-End-Rechner (z.B. einen PC) portierbar sein (vgl. 4.5.2).

A.1.1 Programmiersprachen

Die Vorzüge der objektorientierten Sprachen wurden bereits ausführlich diskutiert (vgl. Kapitel 14). Damit eine gewisse Portabilität auf andere Systeme gewährleistet bleibt, kommen hier nur Standard-Programmiersprachen mit einer entsprechend weiten Verbreitung in Frage [Tuc85]. Diese sind u.a.:

- C++,
- Pascal (mit objektorientierten Erweiterungen) und
- Modula (mit objektorientierten Erweiterungen).

Für spezielle Anwendungen muß jedoch auch auf maschinennähere Codierungsarten zurückgegriffen werden. Zu nennen sind:

- der Maschinencode des jeweiligen Rechners oder ein maschinennaher Code zur zeiteffizienten Formelauswertung,
- FORTRAN als de facto Standard für die Numerik-Programmierung.

Programme, die in verschiedenen Sprachen formuliert worden sind, können heute auf den meisten Rechnern miteinander verbunden werden (*Mixed Language Programming*, vgl. auch Abschnitt A.0.7.4).

A.1.2 Programmierung interaktiver und heuristischer Komponenten

Für die zeitaufwendige Programierung der interaktiven Komponenten (vor allem der Benutzeroberflächen) wurden bereits die Vorteile von

- SmallTalk

hervorgehoben (vgl. Kapitel 16). Die interaktiven Komponenten sind vom Downloading auf den PC ausgenommen, da sie nur zum Zeitpunkt der Baustein-Konfiguration benötigt werden.

Bedingt gilt dies auch für die Formulierung von heuristischen Komponenten. Auf diesem Bereich haben

- PROLOG und
- LISP

eine weite Verbreitung erlangt. In der Regel sind die genannten Programmiersprachen herstellerseitig bereits mit C-Software-Schnittstellen ausgerüstet.

A.1.3 Benutzeroberfläche

Die Verwendung einer modernen graphischen Oberfläche ist unumgänglich (vgl. Kapitel 16). Hier gibt es abhängig vom Betriebssystem die Möglichkeiten [HB89]:

- X-Windows unter UNIX und VMS,
- Presentation Manager unter OS/2,
- MS Windows unter DOS.

Inzwischen gibt es Bibliotheken, die eine Software-Portierung zwischen den verschiedenen Oberflächen erlauben [Sch90a].

A.1.4 Graphik

Besonders problematisch ist die Portierung von Visualisierungskomponenten (vgl. Kapitel 17). Daher sollte als Grundlage ein allgemein verfügbarer Graphik-Standard, wie z.B.

- GKS
- oder PHIGS

gewählt werden [GM89]. Programme auf der Basis von Elementargraphik-Paketen sind im Gegensatz dazu aufwendig zu portieren, da für jede Hardware-Basis eine neue Schnittstelle geschaffen werden muß.

A.1.5 Numerik

Auch im Bereich der Numerik-Programmierung sollte auf Standards gesetzt werden. Hierzu gehören u.a. die Pakete bzw. Bibliotheken:

- BLAS
- LINPACK [Don79],
- EISPACK [Smi76],
- IMSL [IMS87] und
- NAG [K˙c90].

Hier läßt sich in der Programmiersprache C eine einfache Standardisierung erreichen, indem mit Hilfe des C-Präprozessors [KR83] eine einheitliche Namensgebung über die verschiedenen Pakete gelegt wird (vgl. die Ausführungen zum MATHPAK87-Paket auf dem PC in Abschnitt 19.1).

A.1.6 Betriebssystem

Beim Betriebssystem muß vor allem dann, wenn eine direkte Ankopplung des Datenanalysesystems an den Prozeß geplant ist, auf die Multitasking-Fähigkeit geachtet werden. Ferner soll es in der Lage sein, sehr speicherplatzintensive Anwendungen zu unterstützen (was die PC-Betriebssysteme MS DOS und MS Windows von vorn herein ausschließt). Die erforderliche Flexibilität wird im Prinzip von den Betriebssystemen

- UNIX,
- VMS und
- OS\2

erbracht. Allerdings unterscheiden sich UNIX und VMS auf der einen Seite und OS/2 auf der anderen Seite vor allem dadurch, daß für die ersteren ein lange gereiftes Instrumentarium an in das Betriebssystem integrierten Software-Hilfsmitteln zu Verfügung steht (so z.B. die schon erwähnten Programme lex und yacc unter UNIX).

A.2 Hardware

Es wurde bereits diskutiert, daß die objektorientierten und deklarativen Programmierstile hohe Speicherplatz- und Rechenzeitanforderungen haben. Diese Nachteile müssen durch eine entsprechend großzügig ausgelegte Hardware kompensiert werden.

A.2.1 Systemarchitektur

Das System sollte eine professionelle graphische Unterstützung bieten. Als Hardwarebasis in Frage kommen somit nur

- PC's mit hochauflösenden Graphikadaptern,
- Workstations und
- graphische Mehrplatzsysteme.

A.2.2 Rechengeschwindigkeit

Das Datenanalysesystem ist interaktiv, d.h. jede Wartezeit beeinträchtigt die Effizienz, aber auch die Akzeptanz. Bei einigen Methoden (wie z.B. Simulation dynamischer Systeme) zerstört ein diskontinuierlicher Ablauf die Intention völlig. Ist der Prozessor zu langsam, so wird eine Sitzung schnell ermüdend. Eingesetzt werden können z.B.

- die 80486- bzw. 68040-Prozessoren von Intel bzw. Motorola,
- verschiedene RISC-Architekturen [HP90], die heute bereits um den Faktor 2-4 schneller sind,
- Graphikprozessoren, die parallel zum Hauptprozessor arbeiten, und diesen von der graphischen Arbeit entlasten [1],
- Numerische Koprozessoren,
- Transputer-Karten mit mehreren parallelen Prozessoren.

[1] Es sollte erwähnt werden, daß das XWindows-System bereits konzeptuell von einer Trennung zwischen Server- und Client-Prozeß ausgeht

A.2.3 Speicherplatz

Graphische Oberflächen, interaktive Programmierumgebungen und RISC-Prozessoren beanspruchen sehr viel Speicherplatz! Nimmt man den Platzbedarf für die Ablage der Prozeßdaten noch hinzu so kommt man auf eine Anforderung von etwa

- 24 MB Arbeitsspeicher,
- 600 MB Festspeicher (allein ein großes Betriebssystem benötigt über 200 KB) und
- einer Bandstation oder optischen Platte zur Datenarchivierung.

A.3 Entscheidung für eine Konfiguration

A.3.1 Auswahlkriterien

Bei der geplanten Fortsetzung des Projekts am Institut für Biotechnologie 2 des Forschungszentrums Jülich sind folgende Rahmenbedingungen zu beachten:

- Im Mittelpunkt steht die Entwicklung und Erprobung der Konzepte im biotechnischen Labor und nicht die Erstellung eines fertigen Systems.
- Vor allem die personellen Mittel sind beschränkt. Zur Kompensation müssen die modernen Programmiertechniken für das Rapid Prototyping voll ausgereizt werden.
- Die Investition muß mögliche zukünftige Weiterentwicklungen berücksichtigen, um eine Weiterverwendbarkeit der Hardware sicherzustellen.
- Unter dem Gesichtspunkt des Downloadings auf einen als Prozeßrechner eingesetzten PC ist die Programmierung der nicht interaktiven Komponenten in einer prozeduralen Standard-Programmiersprache unumgänglich.
- Das Betriebssystem sollte die Software-Entwicklung optimal unterstützen und hierzu Hilfsmittel (z.B. CASE-Tools) bereitstellen.

A.3.2 Auswahl der Hardware

Ein PC ist mit dem Anforderungskatalog vollkommen überlastet (Speicherplatzproblem, Betriebssystem). Aus diesem Grunde erscheint eine Workstation als ideale Lösung. Zum aktuellen Zeitpunkt ist eine RISC-Architektur [HP90] mit Graphik-Koprozessor die unter einem Kosten-Nutzen-Aspekt effektivste Lösung. Die Prozessoren 80486 und 86040 scheinen im Gegensatz zur RISC-Technologie derzeit an einem gewissen Endpunkt der Entwicklung angelangt zu sein; für parallele Architekturen ist die Software derzeit noch nicht weit genug entwickelt. Dies begründet die Entscheidung für eine

- RISC-Workstation mit,
- Graphik-Prozessor,
- 24 MB Speicherplatz,
- 600 MB Festplatte und
- optischer Platte oder Magnetband.

A.3.3 Auswahl der Software

Ausnahmslos alle heutigen RISC-Workstations laufen unter dem Betriebssystem UNIX. UNIX ist gleichzeitig das Betriebssystem mit der ausgereiftesten Unterstützung für die Software-Entwicklung. In Hinsicht auf das Rapid Prototyping, die schon erstellte Software und die allgemeine Verfügbarkeit ist C++ die Sprache der Wahl. Für die stets äußerst zeitaufwendige Erstellung von Benutzerschnittstellen wurde bereits SmallTalk als hervorragende Lösung beschrieben. Schließlich wird am Jülicher Institut PROLOG als KI-Sprache favorisiert. Zusammengefaßt wird benötigt:

- UNIX als Betriebssystem,
- verbunden mit X Windows als Graphik-Protokoll.
- Der Maschinencode oder ein maschinennaher Code für die Formelauswertung,
- FORTRAN für die Numerik,
- C++ für alle nicht interaktiven Komponenten und die Graphik-Programmierung.
- Ein GKS-System zur Erstellung von Graphiken und ein portabler Menü-Generator zur Erstellung von einfachen Benutzeroberflächen.
- SmallTalk für die Erstellung von komplexen Benutzeroberflächen (hochinteraktive Komponenten zur Baustein- und Netzwerk-Konfiguration) und
- PROLOG für die Formulierung von Regelsystemen.

A.3.4 Mixed Language Programming

Das gleichzeitige Arbeiten mit mehreren Programmiersprachen ist in technisches Problem. Es zeigt sich, daß die einzelnen Elemente durchaus miteinander verbunden werden können:

- Die Auswertung von Formeln auf Maschinencode-Ebene ist für das Downloading nicht erforderlich, da die On-Line-Anwendung nicht zeitkritisch ist. Es ist daher sinnvoll eine maschinenunabhängige Version und eine Workstation-spezifische Version erstellen.
- Die Verwendung von FORTRAN-Numerik-Bibliotheken in einem auf C basierenden System ist unproblematisch, denn FORTRAN und C sind wegen des verwandten Parameterübergabeschemas beim Prozeduraufruf und der einheitlichen Datenrepräsentation kompatibel. Sofern der FORTRAN-Quellcode vorliegt, kann auch an eine direkte Übersetzung des FORTRAN-Codes nach C gedacht werden.
- SmallTalk-80 wird von der Firma Parc Place entwickelt und vermarktet. Im selben Hause ist auch eine mit SmallTalk koppelbare C++-Umgebung verfügbar [RNG90].
- PROLOG und SmallTalk harmonieren gut zusammen, so daß einige SmallTalk-Systeme bereits ein integriertes PROLOG enthalten [Sch90c]. Des weiteren verfügt heute jedes PROLOG-System über eine C-Schnittstelle.

Anhang B

Notationen

Die global verwendeten Notationen seien an dieser Stelle noch einmal nach Sinnzusammenhängen geordnet zusammengefaßt. Man beachte, daß einige Bezeichnungen (wie z.B. X für Biomasse, Konzentrationsvektor und Zufallsvariable) eine Mehrfach-Bedeutung haben, die aus dem jeweiligen Kontext zu erschließen ist. Dies ist darauf zurückzuführen, daß in den verschiedenen Disziplinen Standardbezeichnungen existieren, die sich oftmals überschneiden. Sie werden hier — zur besseren Lesbarkeit für Leser mit Vorkenntnissen auf den einzelnen Gebieten — beibehalten.

B.1 Reaktionssysteme und Kinetik

B.1.1 Verallgemeinerte Reaktionssysteme

Ein (verallgemeinertes) Reaktionssystem wird formuliert in der Form

$$\begin{aligned} \nu_{11} \cdot X_1 + \cdots + \nu_{1n} \cdot X_n &\xrightarrow{R_1} \\ &\vdots \\ \nu_{m1} \cdot X_1 + \cdots + \nu_{mn} \cdot X_n &\xrightarrow{R_m} . \end{aligned}$$

In Kurznotation:

$$\dot{X} = \nu \cdot R(X, Y)$$

Hier ist:

$X = (X_1, \ldots, X_n)^T$	Vektor der Stoffmengen bzw. konzentrationen.
$Y = (Y_1, \ldots, Y_k)^T$	Menge bzw. Konzentration weiterer Stoffe, die bei der Reaktion nicht verbraucht, bzw. produziert werden.
$R = (R_1, \ldots, R_m)^T$	Vektor der Reaktionsgeschwindigkeiten.
$\nu = ((\nu_{ij}))$	Matrix der stöchiometrischen Koeffizienten.
im ν	Theoretischer Reaktionsraum.

B.1.2 Enzymkinetik

Bezogen auf die enzymatische Umsetzung

$$S + E \underset{\alpha}{\overset{\alpha}{\rightleftharpoons}} ES \xrightarrow{\beta} E + P$$

ist:

E	Enzym.
S	Substrat.
ES	Enzym-Substrat-Komplex.
P	Produkt.
$\alpha_\pm, \beta$	Reaktionsgeschwindigkeiten (α_+, α_- beziehen sich auf die Geschwindigkeiten von Hin- und Rückreaktion).
$k_\pm, k_\beta$	Zugehörige Reaktionsgeschwindigkeitskonstanten bei angenommenen Reaktionen erster bzw. zweiter Ordnung.

B.1.3 Kinetik-Funktionen

Die wichtigsten Kinetiken sind die Sättigungskinetik

$$R(X) = R_X \cdot \frac{X}{X + K_X}$$

und die Inhibierungskinetik

$$R(X) = R_X \cdot \frac{X}{\frac{1}{I_X}X^2 + X + K_X} \quad .$$

mit den Parametern:

R_X	Maximale Reaktionsgeschwindigkeit bzgl. der Komponente X bei einer Sättigungskinetik (sinngemäß für Inhibierungskinetiken).
K_X	Konzentration mit halbmaximaler Reaktionsgeschwindigkeit bzgl. X bei Sättigungskinetiken (sinngemäß für Inhibierungskinetiken).
I_X	Konzentration mit halbmaximaler Inhibierung bei Inhibierungskinetiken.

B.2 Stoffwechsel-Modelle

B.2.1 Soffwechselwege

Ein System von Stoffwechselwegen bei einem Mikroorganismus ist beschreibbar durch:

$$\begin{aligned} \nu_{11} \cdot X_1 + \cdots + \nu_{1n} \cdot X_n &\xrightarrow{r_1 X_1} \\ &\vdots \\ \nu_{m1} \cdot X_1 + \cdots + \nu_{mn} \cdot X_n &\xrightarrow{r_m X_1} \quad . \end{aligned}$$

Hier ist:

$X = (X_1, \ldots, X_n)^T$	Vektor der Stoffmengen bzw. Konzentrationen.
X_1	Biomasse (als erster Eintrag in X).
$r = (r_1, \ldots, r_m)^T$	Vektor der spezifischen Reaktionsgeschwindigkeiten (Raten).
$\nu = ((\nu_{ij}))$	Matrix der stöchiometrischen Koeffizienten.
im ν	Theoretischer Reaktionsraum.
$\nu_{\alpha,B}$	Stöchiometrischer Koeffizient bzgl. des Stoffwechselwegs α und der Komponente B (Bei Indizierung mit den Bezeichnern für die Stoffwechselwege und Komponenten anstelle von i, j).

B.2.2 Stoffkonzentrationen bzw. -mengen

Durchgängig verwendete Bezeichnungen sind:

X	Biomasse.
S	Substrat.
P	Produkt.
E	Energie.
T	Zellinternes Intermediat.
Z	Enzym.
A	Äthanol.
O	Sauerstoff.
N	Stickstoffquelle.
C	Kohlendioxid.

B.2.3 Stoffwechselwege

Es handelt sich jeweils um spezifische Größen, bezogen auf die Biomasse:

μ	Wachstumsrate.
σ	Substrataufnahmerate.
π	Produktbildungsrate.
ρ	Energiegewinnung (speziell: Atmung).
m_E	Erhaltungsstoffwechsel (bezogen auf die Energie).
m_S	Erhaltungsstoffwechsel (bezogen auf das Substrat).
η	Äthanolaufnahmerate.
ω	Sauerstoffaufnahmerate.

B.2.4 Wachstumskinetik

Völlig analog zur Enzymkinetik bezeichnet:

μ_S	Maximale spezifische Reaktionsrate bzgl. des Substrats S bei einer Sättigungskinetik (sinngemäß für Inhibierungskinetiken).
K_S	Konzentration mit halbmaximaler Wachstumsrate bzgl. S bei Sättigungskinetiken (sinngemäß für Inhibierungskinetiken).
I_S	Konzentration mit halbmaximaler Inhibierung bei Inhibierungskinetiken.
$m_{S,1}, m_{S,0}$	Koeffizienten eines linearen Ansatzes für den Erhaltungsstoffwechsel.

B.2.5 Stöchiometrische Koeffizienten

$Y_{X/S}$ Yield-Koeffizient für Biomasse und Substrat.

$Y^*_{X/S}$ Scheinbarer Yield-Koeffizient ohne Berücksichtigung eines Erhaltungsstoffwechsels.

$Y_{P/S}$ Yield-Koeffizient für Produkt und Substrat.

$Y_{A/B}$ Analog für beliebige Komponenten A, B.

B.2.6 Allgemeines Stoffwechselmodell

Das Modell ist

$$\dot{X} = N \cdot X_1 \cdot r(X) \quad ,$$

wobei

X Konzentrationsvektor.

X_1 Biomasse (als erster Eintrag in X).

N Stöchiometrische Matrix.

$r(X)$ Kinetik-Vektor.

α_β Koeffizient des linear abhängigen Stoffwechselwegs α bzgl. des linear unabhängigen Wegs β.

$\operatorname{im} N$ Theoretischer Reaktionsraum.

B.2.7 Bilanzierung

Die allgemeine Erhaltungsgleichung ist:

$$E \cdot \dot{X} = 0 \quad \Longrightarrow \quad E \cdot N = 0$$

Hier ist:

$e(\dot{X})$ Erhaltungsgröße (d.h. $e(\dot{X}) = 0$).

$E = (e_1, e_2, \ldots)$ Erhaltungsmatrix zu gegebenen Erhaltungsgrößen.

$m_C, m_H, \ldots$ Massenerhaltungsgröße zum chemischen Element $C, H, \ldots$.

$\ker E$ Stöchiometrischer Reaktionsraum.

B.2.8 Empirischer Reaktionsraum

$\hat{N}$ Schätzung der stöchiometrischen Matrix.

$\operatorname{im} \hat{N}$ Empirischer Reaktionsraum.

B.3 Spezielle Modelle

B.3.1 Backhefe-Modell

Das Backhefe-Modell von Sonnleitner und Käppeli hat die Gleichungen:

$$\begin{aligned}
\sigma &= \sigma_S \tfrac{S}{S+K_S} \\
\sigma_{ox} &= \min(\,\sigma,\; Y_{S/O}\cdot\rho_{max}\,) \\
\sigma_{red} &= \sigma - \sigma_{ox} \\
\eta' &= \eta_A \tfrac{A}{A+K_A}\tfrac{I_S}{S+I_S} \\
\eta_{ox} &= \min(\,\eta',\; (\rho_{max} - Y_{O/S}\cdot\sigma_{ox})\cdot Y_{A/O}\,) \\
\pi &= Y_{A/S}\cdot\sigma_{red} \\
\mu &= Y^{ox}_{X/S}\cdot\sigma_{ox} + Y^{red}_{X/S}\cdot\sigma_{red} + Y_{X/A}\cdot\eta_{ox}
\end{aligned}$$

mit:

ρ_{max}	Maximale Oxidationskapazität.
σ_{red}	Reduktiver Glucose-Abbau.
σ_{ox}	Oxidativer Glucose-Abbau.
η'	Äthanolaufnahmerate bei nicht ausgelastetem Flaschenhals.
η_{ox}	Äthanolaufnahme bei limitierter Aufnahmekapazität des Flaschenhalses.
$Y^{red}_{X/S}, Y^{ox}_{X/S}, Y_{O/S}$	Zugehörige Ertragskoeffizienten ($Y_{O/S} = Y^{-1}_{S/O}$)

B.3.2 Bioreaktoren

Das allgemeine Modell für die flüssige Phase eines homogen gerührten Kessels hat die Form:

$$\dot{X} = N\cdot X_1\cdot r(X) + \frac{1}{V}(F_{in}\cdot X_{in} - F_{out}\cdot X)$$

mit:

X	Stoffkonzentrationsvektor.
F_{in}, F_{out}	Volumenströme im Zu- bzw. Ablauf.
X_{in}	Stoffkonzentrationen im Zulauf.
V	Volumen.
F	Volumenstrom in der kontinuierlichen Kultur ($F_{in} = F_{out}$), bzw. Zulauf in der Fed-Batch-Kultur ($F = F_{in}, F_{out} = 0$).
D	Durchflußrate ($D = \frac{F}{V}$).

B.3.3 Gasübergang

Der Sauerstoffübergang in das Medium wird beschrieben durch:

$$\dot{O} = \mathrm{k_L a}\cdot(O^* - O) - \omega\cdot X \quad ,$$

wobei:

$\mathrm{k_L a}$	Sauerstoff-Transport-Koeffizient.
O^*	Sauerstoff-Sättigungskonzentration.
ω	Spezifische Sauerstoff-Aufnahmerate.
X	Biomasse.

B.4 Wahrscheinlichkeitstheorie

B.4.1 Grundbegriffe

X	Zufallsvariable bzw. Zufallsvektor.
$\mathrm{E}[X] = \mu_X$	Erwartungswert bzw. Erwartungsvektor von X.
$\mathrm{Cov}[X] = \Sigma_X$	Kovarianzmatrix des Zufallsvektors X.
$\mathrm{Var}[X] = \sigma_X^2$	Varianz der Zufallsvariablen X.
σ_X	Standardabweichung von X.

B.4.2 Verteilungen

$f_X(\xi)$	Wahrscheinlichkeitsdichtefunktion von X.
$f_{[a,b]}(\xi)$	Gleichverteilungsdichte auf dem Intervall $[a, b]$.
$f_{\mu,\Sigma}(\xi)$	Normalverteilungsdichte mit Erwartungswert μ und Varianz σ^2.
$f_\lambda(\xi)$	Exponentialverteilungsdichte mit Erwartungswert λ.
$\delta_x(\xi)$	Delta-Distribution über dem Punkt x.
$f_V(\xi)$	Dichte einer Verteilervariablen V ($f_V(\xi) = \sum \lambda_i \delta_{x_i}(\xi)$).
λ_i	Gewichtungsfaktoren mit $\sum \lambda_i = 1$.
$T(X)$	Transformation einer Zufallsvariablen X.
$f_X * f_Y(\xi)$	Faltung zweier Dichtefunktionen.

B.4.3 Hauptachsentransformation

Σ ist eine symmetrische positiv-definite Matrix:

σ_i	Eigenwerte von Σ.
v_i	Hauptachsen bzgl. Σ.
$\sqrt{\Sigma}$	Wurzel aus Σ.

B.5 Statistik

B.5.1 Schätztheorie

$\hat{\beta}$	Schätzer für den Parametervektor β.
$\hat{X}$	Schätzer für die Zufallsvariable X.
$\kappa(\beta, \hat{\beta})$	Gütekriterium für einen Schätzer.

B.5.2 Lineare und nichtlineare Modelle

Man kennt das lineare Modell

$$Y = X \cdot \beta + \epsilon \ ,$$

das Modell mit Fehlern in den Variablen

$$Y = (X + \delta) \cdot \beta + \epsilon \ ,$$

und das Faktoren-Modell

$$Y = X \cdot \beta + \epsilon \ .$$

Im nichtlinearen Fall schreibt man analog:

$$Y = f(X, \beta) + \epsilon \ .$$

Hierbei sind:

ϵ, δ	Störterme.
X	Matrix der unabhängigen Meßgrößen.
$f(X, \beta)$	(Nichtlineare) Funktionsbeziehung.
β	Parametervektor.

B.5.3 Testtheorie

H_0, H_1	Hypothese und Alternativhypothese.
$t(X)$	Teststatistik.
α	Signifikanzniveau.
$L(\Theta; X)$	Likelihood-Funktion.
$\frac{L(\Theta_1; X)}{L(\Theta_0; X)}$	Likelihood-Quotient.
$\lambda_\alpha^\star$	Signifikanzgrenze zum Niveau α.

B.5.4 Sprungdetektion

Das Sprungmodell für abschnittsweise konstante Funktionen hat die Gestalt:

$$f(\Delta\alpha, t_{step}, t) = \begin{cases} \alpha_< & ,0 < t \leq t_{step} \\ \alpha_< + \Delta\alpha & ,t_{step} < t \leq \infty \end{cases} + \epsilon(t)$$

t_{step}	Sprungzeitpunkt.
$\alpha_<$	Funktionswert vor dem Sprung.
$\Delta\alpha$	Sprunghöhe.
$\epsilon(t)$	Störterm.
$\hat{t}_{step}$	Schätzer für den Sprungzeitpunkt.
$\hat{\Delta\alpha}$	Schätzer für die Sprunghöhe.

B.5.5 Klassifikation

Das Mischverteilungsmodell für die statistische Klassifikation lautet:

$$X = X|\Omega_V$$

Es ist:

Ω	Objektraum.
$\omega \in \Omega$	Zu klassifizierendes Objekt.
$\Omega_1, \ldots, \Omega_n$	Klassen in Ω.
V	Verteilervariable, die mit Wahrscheinlichkeit λ_i den Wert i annimmt.
$X(\omega)$	Merkmalsvektor.
$\hat{k}(X)$	Klassifikator.

B.6 Deterministische und stochastische Prozesse

B.6.1 Stochastische Prozesse

$X(t)$	Stochastischer Prozeß.
$x_i(t)$	Realisierungen des Prozesses.
$f(t,\xi)$	Zeitabhängige Wahrscheinlichkeitsdichte.
$\mathrm{E}[X(t)[$	Zeitabhängiger Erwartungswert.
$\mathrm{Cov}[X(t)[= P(t)$	Zeitabhängige Kovarianzmatrix.
$\mathrm{Var}[X(t)[= p(t)$	Zeitabhängige Varianz.
p_∞	Asymptotische Varianz.

B.6.2 Stochastische Differentialgleichungen

Eine stochastische Differentialgleichung wird alternativ formuliert als

$$\dot{x} = a(t,x) + \sqrt{b(t,x)} \cdot e(t)$$

oder differentiell als

$$dX = a(t,X)\,dt + \sqrt{b(t,X)}\,dW(t) \quad .$$

mit

x	Zustandsvektor.
$a(t,X)$	Deterministischer Anteil (Driftkomponente).
$b(t,X)$	Kovarianz des stochastischen Anteils (Diffusionskomonente).
$e(t)$	Standard-normalverteilte Zufallsvariable.
$W(t)$	Wiener-Prozeß.

B.7 Zustandsraum-Modelle, Beobachter und Filter

B.7.1 Zustandsraum-Modelle

Das allgemeine stochastische Zustandsraummodell hat die Form:

$$\begin{aligned} \dot{x} &= f(t,\alpha,x,u) + v_x \\ y &= g(t,\beta,x) + w_y \end{aligned}$$

Es ist:

x	Zustandsvektor.
y	Meßvektor.
u	Steuergröße.
α	Systemparametervektor.
β	Meßparametervektor.
v_x	Systemrauschen.
v_y	Meßrauschen.
f	Systemfunktion.
g	Meßfunktion.

Durch Weglassung der Rauschterme entsteht aus den stochastischen Modellen das entsprechende deterministische Modell.

B.7.2 Lineare Zustandsraum-Modelle

Das lineare Zustandsraummodell ist dementsprechend:

$$\begin{aligned} \dot{x} &= Ax + Bu + \sqrt{R}\cdot e_x \\ y &= Cx + \sqrt{S}\cdot e_y \end{aligned}$$

Hier sind:

A, B, C	Matrizen.
R, S	Symmetrische positiv definite Matrizen (Kovarianzmatrizen der Störterme).
e_x, e_y	Standard-normalverteilte unabhängige Rauschterme.

B.7.3 Filter und Beobachter

Ein linearer Beobachter bzw. ein lineares Filter hat die Struktur:

$$\begin{aligned} \dot{\hat{x}} &= A\hat{x} + Bu + K(y - \hat{y}) \\ \hat{y} &= C\hat{x} \end{aligned}$$

Mit:

$\hat{x}$	Zustandschätzer.
$\hat{y}$	Meßwertschätzer.
K	Verstärkungsmatrix.
P	Kovarianzmatrix von $\hat{x}$.
Q	Kovarianzmatrix von $\hat{y}$.
P_∞	Asymptotische Kovarianzmatrix.

Literaturverzeichnis

[AE888] Adabas Entire - Concept and Facilities. Software AG, Documentation, Uhlandstr. 12, 61 Darmstadt, 1988.

[AL84] P. Agrawal and H.C. Lim. Analysis of various Control Schemes for Continuous Bioreactors. *Advances in Biochemical Engineering/Biotechnology*, 30:62–90, 1984.

[ALS77] G. Adam, P. Läuger, and G. Stark. *Physikalische Chemie und Biophysik.* Springer, 1977.

[ANR+86] N. Ahlmann, A. Niehoff, U. Rinas, T. Scheper, and K. Schügerl. Continuous On-Line-Monitoring of intracellular Enzyme Activity. *Analytica Chimica Acta*, 190:221–226, 1986.

[ASU88] A.V. Aho, R. Sethi, and J.D. Ullman. *Compilerbau*, volume I+II of *Internationale Computer-Bibliothek.* Addison Wesley, 1988.

[Aik80] J.S. Aikins. Representation of Control Knowledge in Expert Systems. In *Proceedings AAAI80*, pages 121–123, 1980.

[All81] J.F. Allen. An Interval-Based Representation of Temporal Knowledge. In IJCAI81 [IJC81], pages 221–226.

[And89] G. Andrews. Estimating Cell and Product Yields. *Biotechnology & Bioengineering*, 33:256–265, 1989.

[Arn74] L. Arnold. *Stochastic Differential Equations : Theory and Application.* Wiley Interscience. Wiley, 1974.

[Åst83] K.L. Åström. Theory and Applications of Adaptive Control — A Survey. *Automatica*, 19(5):471–486, 1983.

[BAB+89] K.-H. Bellgardt, A. Afschar, U. Bartzke, J. Nothnagel, and K. Schaller. Pulsed Control of Substrate Feed for Fed-Batch and Chemostat Processes by using a logical Process Model. In Behrens and Driesel [BD89], pages 461–467.

[BB86] J. Bechlars and R. Buhtz. *GKS in der Praxis.* Springer Compass. Springer, 1986.

[BB86] M. Basseville and A. Benveniste, editors. *Detection of abrupt Changes in Signals and dynamical Systems.* Springer, 86.

[BC84] R.A. Becker and J.M. Chambers. *S: An interactive Environment for Data Analysis and Graphics.* Wadsworth, 1984.

[BC85] S. Barnett and R.G. Cameron. *Introduction to Mathematical Control Theory.* Oxford Applied Mathematics and Computing Science Series. Clarendon, 1985.

[BC88] R.A. Becker and W.S. Cleveland. Brushing Scatterplots. In Cleveland and McGill [CM88], chapter 8, pages 201–224.

[BD81] G. Bader and P. Deuflhard. A semi-implicit Mid-Point Rule for Stiff Systems of Ordinary Differential Equations. Tech. Rep. 114, SFB 123, Heidelberg, 1981.

[BD86] A.H. Borning and R.A. Duisberg. Constraint-Based Tools for Building User Interfaces. *ACM Transactions on Graphics*, 4(5), 1986.

[BD89] D. Behrens and A.J. Driesel, editors. *7th DECHEMA Annual Meeting of Biotechnologists, 30/31 May 1989, Frankfurt am Main*, volume 3 of *DECHEMA Biotechnology Conferences*. DECHEMA, Verlag Chemie Weilheim, 1989.

[BDFB+87] A.H. Borning, R.A. Duisberg, B. Freeman-Benson, A. Kramer, and M. Woolf. Constraint Hierarchies. In *Proceedings of the 1987 ACM Conference on Object-Oriented Programming Systems, Languages and Applications*, pages 48–60, 1987.

[BG89] H. Bandemer and S. Gottwald. *Einführung in FUZZY-Methoden*. Number 305 in Wissenschaftliche Taschenbücher Mathematik/Physik. Akademie Verlag Berlin, 1989.

[BH77] A.H.E. Bijkerk and R.J. Hall. A Mechanistic Model of the Aerobic Growth of Saccharomyces cerevisiae. *Biotechnology & Bioengineering*, 19:267–296, 1977.

[BH85] A.E. Bryson and Y.-C. Ho. *Applied Optimal Control*. Halsted Press, Wiley. Hemisphere, 1985.

[BHLD89] K.-H. Bellgardt, N. Hopf, R. Luttmann, and W.-D. Deckwer. A new Approach for Development of structured Growth Models. In Fish and Fox [FF89].

[BHM+88] A. Borning, R. Henry, J.A. McDonald, F. O'Sullivan, and W. Stuetzle. Visualization, Scientific Computing and Data Analysis. Research Paper, Departments of Computer Science and Statistics, University of Washington, 1988.

[BK72] F. Bergter and W.A. Knorre. Computersimulation von Wachstum und Produktbildung bei Saccharomyces cerevisiae. *Zeitschrift für Allgemeine Mikrobiologie*, 12(8):613–629, 1972.

[BK84] J.L. Bentley and B.W. Kernighan. GRAP - A Language for Typesetting Graphs, Tutorial and User Manual. Computing Science Technical Report No.114, AT&T Bell Laboratories, 600 Mountain Avenue, Murray Hill, NJ 07974, 1984.

[BN77] E. Bergmann and H. Noll. *Mathematische Logik mit Informatik-Anwendungen*. Heidelberger Taschenbücher. Springer, 1977.

[BO86] J.E. Bailey and D.F. Ollis. *Biochemical Engineering Fundamentals*. McGraw-Hill Chemical Engineering Series. McGraw Hill, Second edition, 1986.

[BOHR81] S.W. Brown, S.G. Oliver, D.E.F. Harrison, and R.C. Righelato. Ethanol Inhibition of Yeast Growth and Fermentation: Differences in the Magnitude and Complexity of the Effect. *European Journal of Applied Microbiology and Biotechnolgy*, 11:151–155, 1981.

[BP88a] M.J. Bazin and J.I. Prosser, editors. *Physiological Models in Microbiology*, volume 1+2 of *CRC Series in Mathematical Models in Microbiology*. CRC Press, 1988.

[BP88b] M. Brombacher and M. Polke. Perspektiven der Prozeßleittechnik. *Chemie Ingenieur Technik*, 60(7):539–546, 1988.

[BR86] C.K. Bayne and I.B. Rubin. *Practical Experimental Designs and Optimization Methods for Chemists*. Verlag Chemie, 1986.

[BS75] K. Brammer and G. Siffling. *Stochastische Grundlagen des Kalman-Bucy-Filters - Wahrscheinlichkeitsrechnung und Zufallsprozesse*. Methoden der Regelungstechnik. Oldenbourg, 1975.

[BS85] B. Bieker and G. Schmidt. Fuzzy Regelungen und Linguistische Regelalgorithmen — eine kritische Bestandsaufnahme. *Automatisierungstechnik*, 33(2):45–52, 1985.

[BS86a] H.G. Bock and J.P. Schlöder. Fit, Fitter, the Fittest. Methods for Modeling and Identification of Dynamical Systems. In D.F.F. Müller, editor, *Systemanalysis of Biological Processes*. Vieweg Verlag, Wiesbaden, 1986.

[BS86b] H.G. Bock and J.P. Schlöder. Recent Progress in the Development of Algorithms and Software for Large Scale Parameter Estimation Problems in Chemical Reaction Systems. In *IFAC Workshop on Automatic Control in Petrochemical and Desalination Industries, Kuwait 1986*, 1986.

[BSS84] D.R. Barstow, H.E. Shrobe, and E. Sandewall. *Interactive Programming Environments*. McGraw-Hill, 1984.

[Ban88] J.D. Banfield. *Constraint Cluster Analysis and Image Understanding*. PhD thesis, University of Washington, 1988.

[Bar90] J.P. Barford. A General Model for Aerobic Yeast Growth: Batch Growth. *Biotechnology & Bioengineering*, 35:907–920, 1990.

[Bau78] H. Bauer. *Wahrscheinlichkeitstheorie und Grundzüge der Maßtheorie*. de Gruyter, 1978.

[Bau81] R. Baumann, editor. *Fachtagung Prozeßrechner 1981, München, 10. und 11. März 1981*, number 39 in Informatik Fachberichte. Springer, 1981.

[Baz83] M. Bazin, editor. *Mathematics in Biotechnology*. Academic Press, 1983.

[BdKR80] J.A.B.A.F. Bonnet, H.E. de Kok, and J.A. Roels. The Growth of Saccharomyces cerevisiae CBS 426 on Mixtures of Glucose and Ethanol: a Model. *Antonie van Leeuwenhoek*, 46:565–576, 1980.

[Ber83] F. Bergter. *Wachstum von Mikroorganismen*. Verlag Chemie, Second edition, 1983.

[Beu88] K. Beutler. Anwendung wissensbasierter Systeme in der Automatisierungstechnik. *Automatisierungstechnische Praxis*, 30(8):379–387, 1988.

[Bie81] J. Biewald. Rechnergestützte Erzeugung der Dokumentation für den Funktions- und Software-Entwurf in EPOS. In Baumann [Bau81], pages 98–106.

[Bla81] H.W. Blanch. Invited Review Microbial Growth Kinetics. *Chemical Engineering Communications*, 8:181–211, 1981.

[Blu72] E.K. Blum. *Numerical Analysis and Computation - Theory and Practice*. Addison Wesley, 1972.

[Boc80] H.H. Bock. Explorative Datenanalyse. In Victor et al. [VLvE80], pages 6–37.

[Boc83] H.G. Bock. Recent Advances in Parameter Identification Techniques for O.D.E.-Systems. In *Progress in Scientific Computing*, volume 2, pages 95–121. Birkhäuser, Boston, 1983.

[Bor88] E.J. Bormann. On the Entropy Balance of Microbial Product Formation. *Journal of Theoretical Biology*, 133:215–221, 1988.

[Bre83] F. Breitenecker. The Concept of Supermacros in Today's and Future Simulation Languages. *Mathematics and Computers in Simulation*, 25:279–289, 1983.

[Bri83] E. Brieskorn. *Lineare Algebra und analytische Geometrie*. Vieweg, 1983.

[Büh85] H. Bühler. *Messen in der Biotechnologie*. ABC der Meß- und Analysentechnik. Dr. Alfred Hüthig Verlag Heidelberg, 1985.

[But83] D.K. Button. Differences between the Kinetics of Nutrient Uptake by Micro-Organisms, Growth and Enzyme Kinetics. *TIBS*, 4:121–124, 1983.

[BvdHH90] H.H. Beeftink, R.T.J.M. van der Heijden, and J.J. Heijnen. Maintenance Requirements: Energy Supply from the simultaneous endogenous Respiration and Substrate Consumption. *FEMS Microbiology and Ecology*, 73:203–210, 1990.

[CB76] A. Cornish-Bowden. *Principles of Enzyme Kinetics.* Butterworths, 1976.

[CC86] C.K. Chui and G. Chen. *Kalman Filtering with Real Time Applications.* Number 17 in Series in information sciences. Springer, 1986.

[CCKT83] J.M. Chambers, W.S. Cleveland, B. Kleiner, and P.A. Tukey. *Graphical Methods for Data Analysis.* Wadsworth, 1983.

[CERS88] L.N. Cooper, C. Elbaum, D.L. Reilly, and C.L. Scofield. Parallel, multi-unit, adaptive, nonlinear Pattern Class Separator and Identifier. US Patent Number 4 760 604, July 26, 1988, 1988.

[CFB90] A. Cheruy, H. Farza, and M.P. Bernier. Computer Aided Modelling in Biotechnology. In ECB90 [ECB90], page 339.

[CG87] J.H. Crawford and P.P. Gelsinger. *Programming the 80386.* Sybex, 1987.

[CIF89] CIF - Computer-integrierte Fermentation. Braun Diessel Biotech GmbH, Postfach 120, 3508 Melsungen, 1989.

[CL90] Y.K. Chang and H.C. Lim. Fast Inferential Adaptive Optimization of a Continuous Yeast Culture Based on Carbon Dioxide Evolution Rate. *Biotechnology & Bioengineering*, 35:8–14, 1990.

[CLT76] B.J. Chen, H.C. Lim, and G.T. Tsao. A Model for Bacterial Growth on Methanol. *Biotechnology & Bioengineering*, 18:1629–1633, 1976.

[CM84] W.S. Cleveland and R. McGill. Graphical Perception: Theory, Experimentation, and Application to the Development of Graphical Methods. *Journal of the American Statistical Association*, 79(387):531–554, 1984.

[CM87] W.F. Clocksin and C.S. Mellish. *Programming in Prolog.* Springer, Third edition, 1987.

[CM88] W.S. Cleveland and M.E. McGill, editors. *Dynamic Graphics for Statistics.* Statistics/Probability Series. Wadsworth & Brooks/Cole, 1988.

[CS89] T. Chattaway and G.N. Stephanopoulos. An Adaptive State Estimator for Detecting Contaminants in Bioreactors. *Biotechnology & Bioengineering*, 34:647–659, 1989.

[CWW77] C.L. Cooney, H.Y. Wang, and D.I.C. Wang. Computer-Aided Material Balancing for Prediction of Fermentation Parameters. *Biotechnology & Bioengineering*, 19:55–67, 1977.

[Car89] S.W. Carleysmith. Data Handling for Fermentation Development — an industrial Approach. In Fish and Fox [FF89].

[Cho56] N. Chomsky. Three Models for the Description of Language. *IRE Transactions on Information Theory*, 2(3):113–124, 1956.

[Cle85] W.S. Cleveland. *The Elements of Graphing Data.* Wadsworth, 1985.

[Coc68] W.G. Cochran. Errors of Measurement in Statistics. *Technometrics*, 10:637–666, 1968.

[Con59] D.E. Contois. Kinetics of Bacterial Growth: Relationship between Population Density and Specific Growth Rate of Continuous Cultures. *Journal of General Microbiology*, 21:40–50, 1959.

[Cox86] B.J. Cox. *Object-Oriented Programming - An Evolutionary Approach.* Addison Wesley, 1986.

[DB86] D. Dochain and G. Bastin. Stable Adaptive Algorithms for Estimation and Control of Fermentation Processes. In Johnson [Joh86], pages 1–6.

[DE78] A. Dold and B. Eckmann, editors. *Numerical Analysis, Proceedings of the Biennal Conference, Held at Dundee, June 28-July 1, 1977.* Number 630 in Lecture Notes in Mathematics. Springer, 1978.

[DP80] D. Dubois and H. Prade. *Fuzzy Sets and Systems, Theory and Applications.* Academic Press, 1980.

[DT82] J. Durett and J. Trezona. How to Use Color Displays Effectively. *Byte*, 4:50–53, 1982.

[Del87] H. Dellweg. *Biotechnologie - Grundlagen und Verfahren.* Verlag Chemie Weinheim, 1987.

[Der89] Derive — Der Mathematik-Assistent. 3615 Harding Avenue, Suite 505, Honululu, Hawaii 96816-3735, USA, 1989.

[Doh85] M. Dohnal. Fuzzy Bioengineering Models. *Biotechnology & Bioengineering*, 27:1146–1151, 1985.

[Don79] J.J. Dongarra. *LINPACK, User's guide.* SIAM, Philadelphia, 1979.

[ECB90] *European Congress on Biotechnology, Copenhagen, July 8-13, 1990.* Munksgaard, 1990.

[EFSW89] R. Eberhardt, S. Freyer, D. Schürbüscher, and C. Wandrey. Rechnergesteuerte interaktive Optimierung von biotechnischen Prozessen am Beispiel eines phänotypischen Optimierverfahrens. In *GVC Jahrestreffen 1989 der Verfahrensingenieure, Kurzfassung der Vortäge und Poster, Berlin 27-29.9.89*, page 306. GVC, 1989.

[EHL75] W.H. Enright, T.E. Hull, and B. Lindberg. Comparing Numerical Methods for stiff Systems of O.D.E.'s. *BIT*, 15:10–48, 1975.

[EKB72] V.H. Edwards, R.C. Ko, and S.A. Balogh. Dynamics and Control of Continuous Microbial Propagators to Subject Substrate Inhibition. *Biotechnology & Bioengineering*, 14:939–974, 1972.

[EMR90] G. Engeln-Müllges and F. Reutter. *Formelsammlung zur Numerischen Mathematik mit C-Programmen.* BI Wissenschaftsverlag, 2nd edition, 1990.

[ESF85] A. Einsele, W. Sambacher, and R.K. Finn. *Mikrobiologische und chemische Verfahrenstechnik.* Verlag Chemie Weinheim, 1985.

[EUL+76] V.K. Eroshin, I.S. Utkin, S.V. Ladynichev, V.V. Samoylov, V.D. Kuvshinnikov, and G.K. Skryabin. Influence of pH and Temperature on the Substrate Yield Coefficient of Yeast Growth in a Chemostat. *Biotechnology & Bioengineering*, 18:289–295, 1976.

[Eng81] P.C. Engel. *Enzyme Kinetics.* Chapman and Hall, 1981.

[Eyk90] P. Eykhoff. A Bird's Eye View on Parameter Estimation and System Identification. *The Annals of Statistics*, 7(1):139–149, 1990.

[FESW89] S. Freyer, R. Eberhardt, D. Schürbüscher, and C. Wandrey. Computer-Aided interactive Optimization of Biotechnical Processes using the Example of anaerobic Sewage Purification. In Behrens and Driesel [BD89], pages 661–664.

[FF89] N.M. Fish and R.I. Fox, editors. *Proceedings of the 4th international Congress on Computer Applications in Fermentation Technology: Modelling and Control of biotechnical Processes, Cambridge, September 25-29, 1988.* Society of Chemical Industry, Pergamon Press, 1989.

[FH84] L. Fahrmeir and A. Hamerle. *Multivariate statistische Verfahren.* de Gruyter Berlin, 1984.

[FM82] G. Fischer and C. Mayr. Zur Anwendung unterschiedlicher Simulationsstrategien auf dasselbe Modellobjekt. In Goller [Gol82], pages 89–96.

[FPS85] G. Färber, M. Polke, and H. Steusloff. Mensch-Prozeß-Kommunikation. *Chemie Ingenieur Technik*, 57(4):307–317, 1985.

[Fei77] M. Feinberg. Mathematical Aspects of Mass Action Kinetics. In L. Lapidus and N.R. Amundson, editors, *Chemical Reactor Theory - A Review*, chapter 1, pages 1–78. Prentice Hall, 1977.

[Fel88] W.-D. Fellner. *Computer Graphik*. Number 58 in Reihe Informatik. Bibliographisches Institut Mannheim, 1988.

[Fie84] A. Fiechter. Physical and Chemical Parameters in Microbial Growth. *Advances in Biochemical Engineering/Biotechnology*, 30:7–60, 1984.

[Fis75] G. Fischer. *Lineare Algebra*. Vieweg, 1975.

[Fle87] R. Fletcher. *Practical Methods of Optimization (Volume 1+2)*. Wiley-Interscience. Wiley, Second edition, 1987.

[Fly83] D.S. Flynn. Instrumentation for Fermentation Processes. In Halme [Hal83], pages 5–12.

[For81] K.D. Forbus. Qualitative Reasoning about Physical Processes. In IJCAI81 [IJC81], pages 326–330.

[GBY90] H. Guterman and S. Ben-Yaakov. On-Line Optimization of Biotechnological Processes: I. Application to Open Algal Pond. *Biotechnology & Bioengineering*, 35:417–426, 1990.

[GDW88] Greenleaf Data windows, User's Manual. 16479 Dallas Parkway, BentTree Tower Two, Suite 570 Dallas, TX 75248, 1988.

[GKS90] GSS* GKS Kernel System for DOS. Wiener Weg 5b, D 5 Köln 40, 1990.

[GM84] T. Gasser and H.-G. Müller. Estimating Regression Functions and their Derivatives by the Kernel Method. *Scandinavian Journal of Statististics*, 11:171–185, 1984.

[GM89] M. Göbel and M. Mehl. *Standards der graphischen Datenverarbeitung*. Number 272 in Kontakt & Studium. Expert Verlag, 1989.

[GMM84] T. Gasser, H.-G. Müller, and V. Mammitzsch. Kernels for nonparametric Curve Estimation. *Journal of the Royal Statistical Society, Series B*, 47(2):238–252, 1984.

[GR89] A. Goldberg and D. Robson. *Smalltalk-80, The Language*. Addison Wesley, 1989.

[GS81] E.D. Gilles and H. Schuler. Zur frühzeitigen Erkennung gefährlicher Reaktionszustände in chemischen Reaktoren. *Chemie Ingenieur Technik*, 53(9):673–682, 1981.

[Gar85] C.W. Gardiner. *Handbook of Stochastic Methods*. Number 13 in Springer Series in Synergetics. Springer, Second edition, 1985.

[Gea71] C.W. Gear. *Numerical Initial Value Problems in Ordinary Differential Equations*. Prentice Hall, Englewood Cliffs, 1971.

[Gel72] R.J. Gelinas. Stiff Systems of Kinetic Equations — A Practitioner's View. *Journal of Computational Physics*, 9:222–236, 1972.

[Ges85] C. Geschke. *PostScript Language Reference Manual*. Addison Wesley, 1985.

[Gie88] C. Giersch. Control Analysis of Biochemical Pathways: A Novel Procedure for calculating Control Coefficients, and an additional Theorem for Branched Pathways. *Journal of Theoretical Biology*, 134:451–462, 1988.

[Gil91] S. Gillich. UIL - Unknown Interface? *UNIX Magazin*, 1:52–64, 1991.

[Gol70] S.W. Golomb. Excerpts from: Mathematical Models — Uses and Limitations. *Simulation*, 4:197–198, 1970.

[Gol82] M. Goller, editor. *Proceedings 1. Sympposium Simulationstechnik, 26.-28. April 1982, Erlangen*, number 56 in Informatik Fachberichte. Springer, 1982.

[Gol83] A. Goldberg. *SmallTalk-80, The interactive Programming Environment.* Addison Wesley, 1983.

[Göl87] K. Göldner. *Mathematische Grundlagen der Systemanalyse*, volume I+II of *Mathematik für Ingenieure.* VEB Fachbuchverlag Leipzig, 1987.

[Gol89] B. Goldschmidt. Modellbank Biotechnologie. Münzer + Diehl Electronic GmBH, Zum alten Wasserwerk 9, 5063 Overath 6, 1989.

[Gre89] R. Grehan. Making a Case for CASE. *Byte*, 12:154, 1989.

[Gre90] R. Grehan. Stroke-Character Graphics. *Byte*, 1:351, 1990.

[Gro81] J. Grosche. Methoden der Mensch-Maschine-Kommunikation durch Einsatz grafischer Ein/Ausgaben. In Baumann [Bau81], pages 321–329.

[GvL79] G.H. Golub and C.F. van Loan. Total Least Squares. In A. Dold and B. Eckmann, editors, *Smoothing techniques for curve estimation*, volume 757 of *Lecture Notes in Mathematics.* Springer, 1979.

[HB89] F. Hayes and N. Baran. A Guide to GUIs. *Byte*, 7:250, 1989.

[HDF+85] F.R.A. Hopgood, D.A. Duce, E.V.C. Fielding, K. Robinson, and A.S. Williams, editors. *Methodology of Window Management, Proc. of an Alvey Workshop at Cosener's House Abingdon, UK, April 1985.* Springer, 1985.

[HKK86] A. Halme, R. Kuismin, and M. Korteniemi. A Method to Consider Delayed Laboratory Analysis in State and Parameter Estimation of Bioreactors. In Johnson [Joh86], pages 179–184.

[HLS89] J.L. Harmon, G. Lyberatos, and S.A. Svoronos. A New Method for the Adaptive Determination of Optimum pH and Temperature. *Biotechnology & Bioengineering*, 33:1419–1424, 1989.

[HMN+90] A. Halme, S. Myllylä, J. Nykänen, M. Pokkinen, T. Siimes, A. Holmberg, and R. Oinas. A Computer Workstation supporting Development of Fermentation Processes. In ECB90 [ECB90], page 339.

[HNW87] E. Hairer, S.P. Norsett, and G. Wanner. *Solving Ordinary Differential Equations - Nonstiff Problems.* Series in Computational Mathematics. Springer, 1987.

[HP90] J. Hennessy and D. Patterson. *Computer Architecture: A Quantitative Approach.* Morgan Kaufmann Publishers, 1990.

[HR82] A. Harder and J.A. Roels. Application of Simple Structured Models in Bioengineering. *Advances in Biochemical Engineering/Biotechnology*, 21:55–108, 1982.

[HRWL83] F. Hayes-Roth, D.A. Waterman, and D.B. Lenat. *Building Expert Systems.* Addison-Wesley, 1983.

[HS86] P. Henley and A. Sadana. Deactivation Theory. *Biotechnology & Bioengineering*, 28:1277–1285, 1986.

[HS89] T. Hastie and W. Stützle. Principal Curves. AT&T Bell Laboratories, 1989.

[HSRD78] W. Held, G. Schlanderer, J. Reimann, and H. Dellweg. Continuous Culture of Candida boidinii Variant 60. *European Journal of Applied Microbiology*, 6:127–132, 1978.

[HU88] J.E. Hopcroft and J.D. Ullman. *Einführung in die Automatentheorie, Formale Sprachen und Komplexitätstheorie.* Internationale Computer-Bibliothek. Addison Wesley, 1988.

[HW88] W. Herden and E. Welfonder. Herstellerunabhängige "on-line"-Dokumentation digitaler Prozeßleitsysteme. *Automatisierungstechnische Praxis*, 30(11):558–570, 1988.

[Hab81] A.N. Habermann. *Entwurf von Betriebssystemen - Eine Einführung.* Springer, 1981.

[Hal83] A. Halme, editor. *Modelling and Control of Biotechnical Processes. Proceedings of the First IFAC Workshop, Helsinki, Finland, August 17-19, 1982.* Pergamon Press, 1983.

[Hal89] A. Halme. Expert System Approach to recognize the State of Fermentation and to diagnose Faults in Bioreactors. In Fish and Fox [FF89].

[Haz88] M. Hazewinkel. Nongaussian linear Filtering, Identification of linear Systems, and the sympletic Group. In A. Blaquiere, editor, *Modelling and Control of Systems*, number 121 in Lecture Notes in Control and Information Science, pages 299–308. Springer, 1988.

[Her78] H. Hermes. *Aufzählbarkeit, Entscheidbarkeit, Berechenbarkeit.* Heidelberger Taschenbücher. Springer, 3rd edition, 1978.

[Him78] D.M. Himmelblau. *Fault Detection in Chemical and Petrochemical Processes.* Number 8 in Chemical Engineering Monographs. Elsevier Scientific Publishing Company, 1978.

[Hub81] P.J. Huber. *Robust Statistics.* Wiley, 1981.

[Hub89] P.J. Huber. Strategy and language Issues in Computer assisted interactive Data Analysis. Research Paper, MIT, 1989.

[IJC81] *International Joint Conference on Artificial Intelligence IJCAI81, Vancouver, British Columbia, Canada*, 1981.

[IMS87] IMSL MATH/Library, User's Manual. 2500 ParkWest Tower One, Houston, Texas 77042-3020, USA, 1987.

[ISP88] PC-ISP, Data Exploration and more ... Datavision AG, Postfach 471, 7250 Klosters, Switzerland, 1988.

[Ihm80] P. Ihm. Explorative und Konfirmatorische Datenanalyse — Gegensatz oder Ergänzung? In Victor et al. [VLvE80], pages 38–53.

[Ise87] R. Isermann. *Digitale Regelsysteme*, volume I+II. Springer, 1987.

[JJM86] S.S. Jang, B. Joseph, and H. Mukai. Comparison of two Approaches to On-Line Parameter and State Estimation of Nonlinear Systems. *Ind.Eng.Chem.Process Des.Dev.*, 25:809–814, 1986.

[JMG89] A.B. Jarzebski, J.J. Malinowski, and G. Goma. Modeling of Ethanol Fermentation at High Yeast Concentrations. *Biotechnology & Bioengineering*, 34:1225–1230, 1989.

[JV87] E. Jessen and R. Valk. *Rechensysteme.* Springer, 1987.

[Jaz80] A.H. Jazwinski. *Stochastic Processes and Filtering Theory.* Academic Press, 1980.

[Jef81] R.P. Jefferis. Fault Analysis Methods for Production Fermentation Control. In R.P. Jefferis, editor, *Computer Application in Fermentation Technology*, pages 199–206. Elsevier, 1981.

[Joh86] A. Johnson, editor. *Modelling and Control of Biotechnological Processes. 1st IFAC Symposium, Noordwijkerhoud, The Netherlands, 11-13 December, 1985.* Pergamon Press, 1986.

[Joh87] G. Johannsen. Neue Entwicklungen bei Mensch-Maschine-Systemen. *Automatisierungstechnik*, 35(10):385–395, 1987.

[KD78] R. Keller and I.J. Dunn. Fed-batch Microbial Culture: Models, Errors and Applications. *Journal of Applied Chemistry and Biotechnology*, 28:508–514, 1978.

[KH82] M. Köhne and E. Hasenjäger. Simulation von Zustandsbeobachtern. In Goller [Gol82], pages 413–426.

[KH89] M.N. Karim and A. Halme. Reconciliation of Measurement Data in Fermentation using On-Line Expert System. In Fish and Fox [FF89].

[KR83] B.W. Kernighan and D.M. Ritchie. *Programmieren in C.* PC professionell. Hanser, 1983.

[KW85] H. Küffner and R. Wittenberg. *Datenanalysesysteme für statistische Auswertungen.* Fischer, 1985.

[KW86] D.B. Kell and H.V. Westerhoff. Metabolic Control Theory: Its Role in Microbiology and Biotechnology. *FEMS Microbiology Reviews*, 39:305–320, 1986.

[Kac88] H. Kacser. Regulation and Control of Metabolic Pathways. In Bazin and Prosser [BP88a], chapter 1, pages 1–23.

[Kai88] P. Kail. *FORTH - Einfürung und vollständiger Programmierkurs.* Oldenbourg, 1988.

[Kal60] R.E. Kalman. A New Approach to Linear Filtering and Prediction Problems. *Journal of Basic Engineering. Transactions of the ASME*, 3:35–45, 1960.

[Knu73] D.E. Knuth. *The Art of Computer Programming*, volume I-III. Addison Wesley, 1973.

[Knu84] D.E. Knuth. Literate Programming. *The Computer Journal*, 27(2):97–111, 1984.

[Koc82] A.L. Koch. Multistep Kinetics: Choice of Models for the Growth of Bacteria. *Journal of Theoretical Biology*, 98:401–417, 1982.

[Köc90] N. Köckler. *Numerische Algorithmen in Softwaresystemen.* Teubner, 1990.

[Köh77] M. Köhne. *Zustandsbeobachter fr Systeme mit verteilten Parametern — Thorie und Anwendung*, volume 26 of *Fortschrittsberichte der VDI Zeitschriften, Reihe 8.* VDI-Verlag, Düsseldorf, 1977.

[Koh81] P. Kohlhepp. Ein System für Entwurf, Implementierung und Echtzeitsimulation von Prozefhrungssoftware. In G. Goos, editor, *Werkzeuge der Programmiertechnik, Proceedings GI-Arbeitstagung, Karlsruhe, 16.-17. März 1981*, number 43 in Informatik Fachberichte, pages 65–75. Springer, 1981.

[Kre80] V. Krebs. *Nichtlineare Filterung.* Methoden der Regelungstechnik. Oldenbourg, 1980.

[Kui86] B. Kuipers. Qualitative Simulation. *Artificial Intelligence*, 29:289–338, 1986.

[LEY84] H.Y. Lee, L.E. Erickson, and S.S. Yang. Estimation of True Growth Yield and Maintenance Parameters for Methanol Utilizing Organisms. *Journal of Fermentation Technology*, 62(4):341–351, 1984.

[LG87] J.J. Leary and P.J. Gawthrop. Process Fault Detection using Constraint Suspension. *IEE Proceedings*, 134(4):264–271, 1987.

[LHKH+89] A. Lübbert, B. Hitzmann, A. Kracke-Helm, K. Früh, and K. Schügerl. On Experiences with Expert Systems in the Control of Bioreactors. In Fish and Fox [FF89].

[LL88] J.C. Liao and E.N. Lightfoot. Characteristic Reaction Paths of Biochemical Reaction Systems with Time Scale Separation. *Biotechnology & Bioengineering*, 31:847–854, 1988.

[LMT85] R. Luttmann, A. Munack, and M. Thoma. Mathematical Modelling, Parameter Identification and Adaptive Control of Single Cell Protein Processes in Tower Loop Bioreactors. *Advances in Biochemical Engineering/Biotechnology*, 32:95–206, 1985.

[LS83] L. Ljung and T. Söderström. *Theory and Practice of Recursive Identification*. Number 4 in MIT Series in Signal Processing, Optimization and Control. MIT Press, 1983.

[LSF90] G. Locher, B. Sonnleitner, and A. Fiechter. Pattern Recognition: a useful Tool in technological Processes. *Bioprocess Engineering*, 5:181–187, 1990.

[LSV88] V. Linek, J. Sinkule, and V. Vacek. Dissolved Oxygen Probes. In M. Moo-Young, editor, *Comprehensive Biotechnology*, volume 1-4, pages 363–394. Pergamon, 1988.

[Lap90] J.E. Lapin. *Portable C- und UNIX-Programmierung*. Hanser, 1990.

[Leh82] A.H. Lehninger. *Bioenergetik*. Thieme, Third edition, 1982.

[Lev72] O. Levenspiel. *Chemical Reaction Engineering*. Wiley, 2nd edition, 1972.

[Lev80] O. Levenspiel. The Monod Equation: A Revisit and a Generalization to Product Inhibition Situations. *Biotechnology & Bioengineering*, 22:1671–1687, 1980.

[Lew86] F.L. Lewis. *Optimal Estimation*. Wiley, 1986.

[Lia89] J.C. Liao. Fermentation Data Analysis and State Estimation in the Presence of Incomplete Mass Balance. *Biotechnology & Bioengineering*, 33:613–622, 1989.

[Lju87] L. Ljung. *System Identification - Theory for the User*. Prentice Hall, 1987.

[Lue64] D.G. Luenberger. Observing the State of a Linear System. *IEEE Transactions on Military Electronics*, 4:74–80, 1964.

[Lun88] R. Lunderstädt. Zur Kompensation systematischer Sensorfehler. *Automatisierungstechnik*, 36(8):282–289, 1988.

[Luo87] J.H.T. Luong. Generalization of Monod Kinetics for Analysis of Growth Data with Substrate Inhibition. *Biotechnology & Bioengineering*, 19:242–248, 1987.

[MB89] T. Mason and D. Brown. *lex & yacc*. Nutshell Handbook. O'Reilly & Assoc., 1989.

[MKE89] I.G. Minkevich, A.Y. Krynitskaya, and V.K. Eroshin. Bistat — a Novel Method of Continuous Cultivation. *Biotechnology & Bioengineering*, 33:1157–1161, 1989.

[MOR88] T.A. McMeekin, J. Olley, and D.A. Ratkowsky. Temperature Effects on Bacterial Growth Rates. In Bazin and Prosser [BP88a], chapter 4, pages 75–89.

[MP89] A. Munack and C. Posten. Optimale Versuchsplanung zur Modellierung des Wachstums von Pflanzenzellen in Submerskultur. *Automatisierungstechnik*, 37:55–65, 1989.

[MP886] Mathpak 87, Users Guide and Reference Manual. 1239 Sir David Drive, Oakville, Ontario, Canada L6J 6Y9, 1986.

[MS82] J. Macki and A. Strauss. *Introduction to Optimal Control Theory*. Springer, 1982.

[MS88] M.L. Mavrovouniotis and G. Stephanopoulos. Formal Order-of-Magnitude Reasoning in Process Engineering. *Computers in Chemical Engineering*, 12(9/10):867–880, 1988.

[MSS89] M.L. Mavrovouniotis, Ge. Stephanopoulos, and Gr. Stephanopoulos. Formal Modeling of Approximate Relations in Biochemical Systems. *Biotechnology & Bioengineering*, 34:196–206, 1989.

[MW886] Metawindow Reference Manual. 4575 Scotts Valley Drive, P.O. Box 66779, Scotts Valley, CA 95066, 1986.

[MYSZ90] J.C. Merchuk, S. Yona, M.H. Siegel, and A. Ben Zvi. On the First-Order Approximation to the Response of Dissolved Oxygen Electrodes for Dynamic kLa Estimation. *Biotechnology & Bioengineering*, 35:1161–1163, 1990.

[Mar90] W. Martin. Meta CASE Environments: How to integrate various Information Models. In *Proc. CASE World, Hamburg, October 29-30, 1990.* Digital Consulting Inc., Andover Massachusettes, 1990. in Preparation.

[May82] O. Mayer. *Syntaxanalyse.* Number 27 in Reihe Informatik. Bibliographisches Institut Mannheim, 1982.

[McC87] B.H. McCormick. Visualization in Scientific Computing — A Synopsis. *IEEE CG&A*, 7:61–70, 1987.

[Mel91] H. Melenk. REDUCE Support for Reaction Equation Systems. Konrad-Zuse-Zentrum für Informationstechnik, Heibronner Str. 10, D 1 Berlin 31, 1991.

[Mil78] G.N. Milshtein. A Method of Second-order Accuracy Integration of Stochastic Differential Equations. *Theory of Probability and Applications*, 23:396–401, 1978.

[Mor78] J.J. Moré. The Levenberg-Marquardt Algorithm: Implementation and Theory. In Dold and Eckmann [DE78], pages 105–116.

[Mos81] A. Moser. *Bioprozeßtechnik.* Springer Verlag, 1981.

[Mun89] A. Munack. Optimal Feeding Strategy for Identification of Monod-Type Models by Fed-Batch Experiments. In Fish and Fox [FF89].

[NS89] S.C. Niranjan and Ka-Yiu San. Analysis of a Framework using Material Balances in Metabolic Pathways to Elucidate Cellular Metabolism. *Biotechnology & Bioengineering*, 34:496–501, 1989.

[Neg85] C.V. Negoita. *Expert Systems and Fuzzy Systems.* Benjamin/Cummings, 1985.

[Nie83] H. Niemann. *Klassifikation von Mustern.* Springer, 1983.

[Nie90] W. Niederkrüger. Die Rechnung bitte - Mathematische Toolboxen für Turbo-Pascal. *c't*, 12:204–210, 1990.

[PBCL74] P. Peringer, H. Blachere, G. Corrieu, and A. G. Lane. A Generalized Mathematical Model for the Growth Kinetics of Saccharomyces cerevisiae with Experimental Determination of Parameters. *Biotechnology & Bioengineering*, 16:431–454, 1974.

[PFTV88] W.H. Press, B.P. Flannery, S.A. Teukolsky, and W.T. Vetterling. *Numerical Recipes in C - The Art of Scientific Computing.* Cambridge University Press, 1988.

[PHB78] N.B. Pamment, R.J. Hall, and J.P. Barford. Mathematical Modeling of Lag Phases in Microbial Growth. *Biotechnology & Bioengineering*, 20:349–381, 1978.

[PKGB77] T.A. Petrova, W.A. Knorre, R. Guthke, and F. Bergter. A mathematical Model of Microbial Growth including an Intermediate. I. Growth in Batch Cultures. *Zeitschrift für Allgemeine Mikrobiologie*, 17(7):531–542, 1977.

[PM87] G.K.H. Pang and A.G.J. MacFarlane. *An Expert System Approach to Computer-Aided Design of Multivariable Systems.* Number 89 in Lecture Notes in Control and Information Sciences. Springer, 1987.

[PP76] P. Pilàt and A. Prokop. Oxidation of Methanol, Formaldehyde and Formic Acid by Methanol-utilizing Yeast. *Folia Microbiologica*, 21:306–314, 1976.

[PT85] E. Pardoux and D. Talay. Discretization and Simulation of Stochastic Differential Equations. *Acta Applicandae Mathematicae*, 3:23–47, 1985.

[PUB83] V.G. Popov, G.A. Ugodchikov, and I.N. Blokhina. Logical and Mathematical Models of Growth and Development of Microbial Population Processes. In Halme [Hal83], pages 185–190.

[PW90] J.N. Petersen and G.A. Whyatt. Dynamic On-Line Optimization of a Bioreactor. *Biotechnology & Bioengineering*, 35:712–718, 1990.

[Pea84] J. Pearl. *Heuristics*. Addison Wesley, 1984.

[Pir75] S.J. Pirt. *Principles of Microbe and Cell Cultivation*. Blackwell Scientific Publications, 1975.

[Poh90] M. Pohl. Grafik von Format zu Format, Teil 1: DXF-Format. *PC-Plus*, 4:82, 1990.

[Pop71] K.R. Popper. Conjectural Knowledge: My Solution to the Problem of Induction. *Revue Internationale de Philosophie*, 25(95-6), 1971.

[Pow78] M.J.D. Powell. A Fast Algorithm for Nonlinearly Constrained Optimization Calculations. In Dold and Eckmann [DE78], pages 144–157.

[Pre87] R.S. Pressman. *Software Engineering - A Practitioner's Approach*. McGraw Hill Series in Software Engineering and Technology. McGraw Hill, 1987.

[Pri81] M.B. Priestley. *Spectral Analysis and Time Series (Volume 1+2)*. Probability and Mathematical Statistics. Academic Press, 1981.

[QSQJC89] Cheng Qi, Wang Shu-Quing, and Wang Ji-Cheng. Application of Expert System to the Operation and Control of industrial Antibiotic Fermentation Process. In Fish and Fox [FF89].

[RGRW75] M. Reuss, J. Gnieser, H.G. Reng, and F. Wagner. Extended Culture of Candida boidinii on Methanol. *European Journal of Applied Microbiology*, 1:295–305, 1975.

[RNG90] M. Rüger, A. Nyhuis, and H. Guhl. Objects at Work, Objectworks fr SmallTalk-80 und C++. *iX*, 1:30–34, 1990.

[Ran86] J. Ranta. Some Properties of Microbial Age Dynamic Models. In Johnson [Joh86], pages 93–100.

[Ray81] W.H. Ray. *Advanced Process Control*. Chemical Engineering Series. McGraw-Hill, 1981.

[Red88] C. Reder. Metabolic Control Theory: A Structural Approach. *Journal of Theoretical Biology*, 135:175–201, 1988.

[Rei82] W. Reisig. *Petrinetze*. Springer, 1982.

[Reu89] M. Reuss. Structured Modelling of Bioreactor Systems. In Fish and Fox [FF89].

[Roe83] J.A. Roels. *Energetics and Kinetics in Biotechnology*. Elsevier Biomedical Press, 1983.

[Rue82] W. Ruemelin. Numerical Treatment of Stochastic Differential Equations. *SIAM Journal of Numerical Analysis*, 19(3):604–613, 1982.

[SB73] J. Stoer and R. Bulirsch. *Einführung in die numerische Mathematik (Band 1+2)*. Heidelberger Taschenbücher. Springer, 1973.

[SCAGGM89] R. Suàrez-Cortez, J. Alvarez-Gallegos, and E. Gonzàlez-Mora. Sliding Controller Design for a Nonlinear Fermentation System. *Biotechnology & Bioengineering*, 33:377–385, 1989.

[SG86] R.W. Scheifler and J. Gettys. The X Window System. *ACM Transactions on Graphics*, 5(2):79–109, 1986.

[SGB+88] A.P.J Sweere, J. Giesselbach, R. Barendse, R. de Krieger, G. Honderd, and K.C.A.M. Luyben. Modelling the dynamic Behaviour of Saccharomyces cerevisiae and its Application in Control Experiments. *Applied Microbiology and Biotechnology*, 28:116–127, 1988.

[SGD71] A. Sols, C. Gancedo, and G. DelaFuente. Energy-Yielding Metabolism in Yeasts. In A.H. Rose and J. S. Harrison, editors, *The Yeasts. Vol. 2: Physiology and Biochemistry of Yeasts*, chapter 7, pages 271–307. Academic Press, London, 1971.

[SHRD78] G. Schlanderer, W. Held, J. Reimann, and H. Dellweg. Metabolic Changes during Substrate Shifts in Continuous Cultures of the Yeast Candida Boidinii Variant 60 growing on Methanol. *European Journal of Applied Microbiology*, 6:133–144, 1978.

[SK86] B. Sonnleitner and O. Käppeli. Growth of Saccharomyces cerevisiae is controlled by its limited Respiratory Capacity: Formulation and Verificaton of a Hypothesis. *Biotechnology & Bioengineering*, 28:927–937, 1986.

[SMZ+88] A.P.J. Sweere, Y.A. Matla, J. Zandvliet, K.Ch.A.M. Luyben, and N.W.F. Kossen. Experimental simulation of glucose fluctuations: The influence of continually changing glucose concentrations on the fed-batch baker's yeast production. *Applied Microbiology and Biotechnology*, 28:109–115, 1988.

[SS78] S. Schach and T. Schäfer. *Regressions- und Varianzanalyse*. Springer Berlin, 1978.

[SS84] G. Stephanopoulos and Ka-Yiu San. Studies an On-Line Bioreactor Identification. I. Theory. *Biotechnology & Bioengineering*, 26:1176–1188, 1984.

[SSW90] M. Seipel, J. Severin, and W. Wiechert. Programm zur interaktiven kLa-Wert Bestimmung mit Hilfe der dynamischen Methode. Blockpraktikumsprotokoll, Institut für Mikrobiologie und Biotechnologie, Universität Bonn, 1990.

[ST89] G. Stephanopoulos and C. Tsiveriotis. Toward a systematic Method for the Generalization of Fermentation Data. In Fish and Fox [FF89].

[STSS89] H. Shimizu, T. Takamatsu, S. Shioya, and K.-I. Suga. An Algorithmic Approach to Constructing the On-Line Estimation System for the Specific Growth Rate. *Biotechnology & Bioengineering*, 33:354–364, 1989.

[SW88] J. Severin and W. Wiechert. Computersimulation und Durchführung der Hefefermentation im Batch- und Fed-Batch-Verfahren. Praktikumsskriptum des Blockpraktikums Biotechnologie, Institut für Mikrobiologie und Biotechnologie, Universität Bonn, 1988.

[SW89] J. Severin and W. Wiechert. Mathematische Modelle in Mikrobiologie und Biotechnologie. Vorlesungsskriptum und Lehrsoftware, Institut für Mikrobiologie und Biotechnologie, Universität Bonn, 1989.

[SY90] E. Schuster and S. Yakowitz. Contributions o the Theory of Nonparametric Regression, with Application to System Identification. *The Annals of Statistics*, 7(1):139–149, 1990.

[Sac84] L. Sachs. *Angewandte Statistik*. Springer, 1984.

[Sch71] H. Schneeweiß. *Ökonometrie*. Physica-Verlag Würzburg, 1971.

[Sch75] K.-W. Schrick, editor. *Anwendungen der Kalman-Filter-Technik - Anleitung und Beispiele*. Methoden der Regelungstechnik. Oldenbourg, 1975.

[Sch82a] W. Schaback. Ein interaktives System zur Simulation, Parameterschtzung und graphischen Auswertung diskreter und kontinuierlicher Modelle. In Goller [Gol82], pages 166–172.

[Sch82b] B. Schmidt. Systemanalyse und Modellbildung. In Goller [Gol82], pages 106–115.

[Sch82c] T. Schmieden. Zum Entwurf von Beobachtern bei der optimalen Regelung von Rührkesselreaktoren. *Regelungstechnik*, 30(9):313–318, 1982.

[Sch85] H.G. Schlegel. *Allgemeine Mikrobiologie.* Thieme, 6th edition, 1985.

[Sch88] L. Schmitz. Ojektorientiert = Gut? *PC Magazin*, 51:46, 1988.

[Sch89] D. Schürbüscher. *Rechnergestützte Optimierung und Prozeßfürung kontinuierlicher biotechnischer Prozesse untersucht am Beispiel der anaeroben Abwasserreinigung.* PhD thesis, T.H. Aachen, 1989.

[Sch90a] A. Schulmann. Glockenspiel puts C++ to Work. *Byte*, 1:213, 1990.

[Sch90b] R. Schulz. Das Obskure — Objektorientierte Programmierung mit der Smalltalk-80 Entwicklungsumgebung. *iX*, 1:42–45, 1990.

[Sch90c] R. Schulz. Kleine Unterhaltung, Smalltalk/V: Einstieg in die Entwicklung. *c't*, 1:39–41, 1990.

[Sch90d] R. Schumann. Regelungstechnische Programmpakete für den IBM-PC und kompatible Personal-Computer unter MS-DOS. Handbuch, VDI-Bildungswerk, Graf-Recke-Str. 84, 4 Düsseldorf, 1990.

[Smi76] B.T. Smith. *EISPACK Guide - Matrix Eigensystem Routines.* Number 6 in Lecture Notes in Computer Siences. Springer, 1976.

[Str87] B. Stroustrup. *Die C++ Programmiersprache.* Internationale Computer-Bibliothek. Addison-Wesley, 1987.

[Str88] A. Strickling. CAE in der Prozeßleittechnik. *Automatisierungstechnische Praxis*, 30(2):61–67, 1988.

[Stu87] W. Stuetzle. Plot Windows. *Journal of the American Statistical Association*, 82(398):466–475, 1987.

[TCP90] Turbo C++-Handbücher. Lindwurmstrr. 88, Postfach 14 03 40, D 8 München 2, 1990.

[TCP91] Turbo Pascal 6.0, Handbücher, Turbo Vision. Lindwurmstrr. 88, Postfach 14 03 40, D 8 München 2, 1991.

[TFRF89] N.F. Thornhill, M.Y.P. Fung, F.W. Roche, and N.M. Fish. Detecting faults in fermentation plant using analytical redundancy. In Fish and Fox [FF89].

[TRJ88] B.G. Turner, D. Ramkrishna, and N.B. Jansen. Cybernetic Modeling of Bacterial Cultures at Low Growth Rates: Mixed-Substrate Systems. *Biotechnology & Bioengineering*, 32:46–54, 1988.

[TRJ89] B.G. Turner, D. Ramkrishna, and N.B. Jansen. Cybernetic Modeling of Bacterial Cultures at low Growth Rates: Single-Substrate Systems. *Biotechnology & Bioengineering*, 34:252–261, 1989.

[TY76] G.T. Tsao and C.M. Yang. Extended Monod Equation. *Biotechnology & Bioengineering*, 18:1827–1831, 1976.

[Tuc85] A.B. Tucker. *Programming Languages.* Mc Graw Hill Computer Science Series. Mc Graw Hill, 1985.

[Tuk77] J. Tukey. *Exploratory Data Analysis.* Addison Wesley, 1977.

[Übe71] K. Überla. *Faktorenanalyse.* Springer, 1971.

[Ude90] J. Udell. Smalltalk-80 enters the Nineties. *Byte*, 10:138, 1990.

[VH88] W. Vach and P.J. Huber. Analysis of ISP Script Files as an Example for the retrospective Analysis of machine-written Records. Forschungsbericht Nr. 88/3, Universität Dortmund, Fachbereich Statistik, 1988.

[VLvE80] N. Victor, W. Lehmacher, and W. van Eimeren, editors. *Explorative Datenanalyse. Frühjahrstagung der GMDS, München, 21.-22.März 1980*, number 26 in Medizinische Informatik und Statistik. Springer, 1980.

[VS86] V.B. Veljkovic and D.U. Skala. Power Consumption and the volumetric Oxygen Mass Transfer Coefficient in an aerated stirred Vessel. *Journal of the Serbian Chemical Society*, 51(9-10):497–505, 1986.

[Vic80] N. Victor. Stellung der explorativen Datenanalyse (EDA) im Rahmen der Statistik. In Victor et al. [VLvE80], pages 2–5.

[Vog70] W. Vogel. *Wahrscheinlichkeitstheorie.* Studia Mathematica. Vandenhoek & Ruprecht, 1970.

[Vog80] W. Vogel. Klassische Mathematische Statistik. Vorlesungsscriptum Universität Bonn, 1980.

[WA86] H. Werner and H. Arndt. *Gewöhnliche Differentialgleichungen, Eine Einfhrung in Theorie und Praxis.* Springer, 1986.

[WBLL89] Li-Wai Wan, S. Bröring, B. Larson, and A. Lübbert. Neuere Ergebnisse bei der Meßtechnik in der Bioreaktor-Fluiddynamik. In *Interne Arbeitssitzung des Fachausschusses Bioverfahrenstechnik der GVC, 2./3. Mai 1989, Jülich.* VDI-Gesellschaft Verfahrenstechnik und Chemieingenieurwesen, 1989.

[WJ76] A.S. Willsky and H.L. Jones. A generalized Likelihood Ratio Approach to the Detection and Estimation of Jumps in Linear Systems. *IEEE Transactions on Automatic Control*, AC-21:228–231, 1976.

[WNSW90] M. Wedy, M. Neumann, D. Schürbüscher, and C. Wandrey. Rechnergestützte Optimierung der Fed-Batch-Fermentation von Candida boidinii. In *8th Annual Meeting of Biotechnologists, 1990, Frankfurt am Main*, DECHEMA Biotechnology Conferences. DECHEMA, Verlag Chemie Weilheim, 1990.

[WS79] H. Werner and R. Schaback. *Praktische Mathematik (Band I+II).* Springer, Second edition, 1979.

[WS83] N.S. Wang and G. Stephanopoulos. Application of Macroscopic Balances to the Identification of Gross Measurement Errors. *Biotechnology & Bioengineering*, 25:2177–2208, 1983.

[WSE84] N.S. Wang, G.N. Stephanopoulos, and L.E. Erickson. *Computer Applications to Fermentation ProcessesComputer Applications to Fermentation Processes.* Number 2, Issue 1 in CRC Critical Reviews in Biotechnology. CRC Press, 1984.

[WSS88] W. Wiechert, J. Severin, and D. Schürbüscher. Mathematische Grundlagen der Prozeßsimulation in der Biotechnologie und chemischen Verfahrenstechnik. Seminarskriptum, Haus der Technik, Hollestr. 1, 43 Essen, 1988.

[WSW89] M. Wedy, D. Schürbüscher, and C. Wandrey. Automatic Start-Up of a continuous Fermentation with Candida boidinii. In Behrens and Driesel [BD89], pages 657–660.

[Wal76] W. Walter, editor. *Gewöhnliche Differentialgleichungen.* Heidelberger Taschenbücher. Springer, 1976.

[Wal87] E. Walter. *Identifiability of Parametric Models.* Pergamon Press, 1987.

[Wan88] C. Wandrey. Reaktionstechnik (Technische Reaktionsführung). Vorlesungsscriptum, Universität Bonn, 1988.

[Wan89] C. Wandrey. Reaktionstechnik enzymkatalysierter Umsetzungen, Seminarskriptum. Institut für Biotechnologie 2, Forschungszentrum Jülich, 1989.

[Wat72] T.G. Watson. The Present Status and Future Prospects of the Turbidostat. *Journal of Applied Chemistry and Biotechnology*, 22:229–243, 1972.

[Wed81] H. Wedekind. *Datenbanksysteme I+II.* Number 16+18 in Reihe Informatik. Bibliographisches Institut Mannheim, 1981.

[Wie87] W. Wiechert, 1987. Unveröffentlichte Meßergebnisse, zur Eichung von OD-Meßgeräten.

[Wie89] W. Wiechert. Standardization Concepts for the Implementation of High Performance Signal Analysis Components into Rule Based Process Supervisory Systems. In Behrens and Driesel [BD89], pages 783–785.

[Wil76] A.S. Willsky. A Survey of Design Methods for Failure Detection in Dynamic Systems. *Automatica*, 12:601–611, 1976.

[Wil77] D. Williamson. Observation of Bilinear Systems with Application to Biological Control. *Automatica*, 13:243–254, 1977.

[Win79] P.H. Winston. *Artificial Intelligence.* Addison-Wesley Series in Computer Science. Addison-Wesley, 1979.

[Wir78] N. Wirth. *Systematisches Programmieren.* Number 17 in Leitfäden der angewandten Mathematik und Mechanik. Teubner, 1978.

[Wol88] S. Wolfram. *Mathematica - A System for doing Mathematics by Computer.* Addison wesley, 1988.

[WvD87] H.V. Westerhoff and K. van Dam. *Mosaic Nonequilibrium Thermodynamics and Control of biological Free-Energy Transduction.* Elsevier Amsterdam, 1987.

[YK69] T. Yano and S. Koga. Dynamic Behavior of the Chemostat Subject to Substrate Inhibition. *Biotechnology & Bioengineering*, 11:139–153, 1969.

[YS84] T. Yamané and S. Shimizu. Fed-batch Techniques in Microbial Processes. *Advances in Biochemical Engineering/Biotechnology*, 30:147–194, 1984.

[Yav85] Y. Yavin. *Numerical Studies in Nonlinear Filtering.* Number 65 in Lecture Notes in Control and Information Sciences. Springer, 1985.

[ZH88] D. Zöbel and H.Hogenkamp. *Konzepte der parallelen Programmierung.* Teubner, 1988.

[ZN80] R. Zentgraf and H. Nowak. Voraussetzungen und Grenzen der Explorativen Datenanalyse. In Victor et al. [VLvE80], pages 54–62.

[Zei77] M. Zeitz. *Nichtlineare Beobachter für chemische Reaktoren*, volume 27 of *Fortschrittsberichte der VDI Zeitschriften, Reihe 8.* VDI-Verlag, Düsseldorf, 1977.

[Zim80] H. Zima. *Betriebssysteme. Parallele Prozesse.* Number 20 in Reihe Informatik. Bibliographisches Institut Mannheim, 1980.

[nPo84] K.R. nPopper. *Objektive Erkenntnis - Ein evolutionärer Entwurf.* Hoffmann und Campe, 4th edition, 1984.

[vDMdMW88] K. van Dam, M.M. Mulder, T. de Mattos, and H.V. Westernhoff. A Thermodynamic View of Bacterial Growth. In M.J. Bazin, editor, *Physiological Models in Microbiology*, volume I+II of *CRC Series in Mathematical Models in Microbiology*. CRC Press, 1988.

[vK89] B.H.A. van Kleef. Enthalpy Balances in the Control of Fermentation Processes. In Fish and Fox [FF89].

[vSI89] U. von Stockar and I.W.Marison. The Use of Calorimetry in Biotechnology. *Advances in Biochemical Engineering/Biotechnology*, 40:93, 1989.



Jül-2457
März 1991
ISSN 0366-0885