Entwicklung von Software-Systemen zur Planung, Datenaufnahme und –auswertung von Bioprozessen

Von der Naturwissenschaftlichen
Fakultät der
Gottfried Wilhelm Leibniz
Universität Hannover

zur Erlangung des Grades

Doktor der Naturwissenschaften
- Dr. rer. nat. -

genehmigte Dissertation

von

Dipl.-Chem. Patrick F. O. Lindner
Geboren am 11.07.1977 in Altdorf bei Nürnberg

2006

Referent :	**Prof. Dr. Bernd Hitzmann**
Korreferent :	**Prof. Dr. Thomas Scheper**
Tag der Promotion:	**15.12.2006**

Hiermit versichere ich an Eides statt, die vorliegende Dissertation selbständig und nur unter Verwendung der angegebenen Hilfsmittel angefertigt zu haben. Ich versichere ferner, dass die Dissertation nicht schon als Diplomarbeit oder ähnliche Prüfungsarbeit verwendet worden ist.

_____

Patrick Lindner

Hannover, September 2006

Danksagung

Herrn Prof. Dr. Bernd Hitzmann, unter dessen Leitung diese Arbeit durchgeführt wurde, gilt mein besonderer Dank. Durch sein Engagement, geduldige fachliche Betreuung und viele anregende und interessante Diskussionen hat er maßgeblich zur Entstehung dieser Arbeit beigetragen.

Herrn Prof. Dr. Thomas Scheper danke ich für die Übernahme des Korreferats.

Für die finanzielle Unterstützung während des Prozessanalyse-Projektes gilt mein Dank der Firma Chr. Hansen. Auf die Reisen zum Standort Nienburg habe ich mich immer sehr gefreut und daher möchte ich mich bei den dortigen Mitarbeitern und insbesondere bei Dr. Karsten Hellmuth für die freundliche Aufnahme und die große Unterstützung bedanken.

Bei Guido Rudolph und Arne Bluma möchte ich mich für das Testen der In-situ Software und für viele konstruktive Vorschläge und Anregungen bedanken.

Den Mitarbeitern des Institutes für Technische Chemie in Hannover danke ich für die angenehme Atmosphäre, die anregenden Diskussionen und die gute Zusammenarbeit. Mein besonderer Dank geht an den Arbeitskreis Hitzmann für die Hilfsbereitschaft und das hervorragende Arbeitsklima.

Meiner Familie und Stephanie Wolf danke ich für ihre Liebe und Unterstützung. Ohne Euch wäre mein Studium und diese Arbeit nicht möglich gewesen.

Kurzfassung

Um die kinetischen Parameter einer Reaktion bestimmen zu können, ist die Durchführung von Experimenten notwendig. Für eine Reaktion, die durch die Michaelis-Menten Kinetik beschrieben werden kann, wurden durch Analyse der Fisher-Informations-Matrix optimale experimentelle Bedingungen ermittelt, die eine möglichst genaue Bestimmung der Parameter erlauben. Als Optimierungsmethode wurde ein genetischer Algorithmus eingesetzt. Es konnte gezeigt werden, dass bei einem Fed-Batch-Prozess die Cramer-Rao-Lower-Bound um 40 % für K_m und um 18 % für v_{max} niedriger liegt, als bei einem vergleichbaren Batch-Prozess.
Im zweiten Teil der Arbeit wurde ein industrieller Fermentationsprozess untersucht, mit der Zielsetzung die wichtigsten Einflussgrößen auf die Ausbeute zu ermitteln und eine Prozessoptimierung durchzuführen. Mit einer Korrelations- und einer Haupt-komponentenanalyse konnten die Viskosität, die Zitronensäurekonzentration und die Menge des Substrat-Feeds als wichtigste Einflussgrößen identifiziert werden. Durch Einsatz neuronaler Netze wurde eine Echtzeit-Simulation des Prozesses durchgeführt. Das Netz konnte die zu erwartende Produktkonzentration 12 Stunden im Voraus sowie die Endausbeute mit einem durchschnittlichen Fehler von 6 % vorhersagen. Durch Eingabe modifizierter Datensätze in das neuronale Netz konnte gezeigt werden, dass bei einer Erhöhung der Rührgeschwindigkeit und Veränderung der Substratzufütterung eine Steigerung der Prozessausbeute zu erwarten ist.
Der dritte Schwerpunkt dieser Arbeit war die Entwicklung einer Steuerungs- und Auswertesoftware für ein In-situ-Mikroskop. Das Programm *In-situ-Control* bietet über eine grafische Benutzeroberfläche eine Vielzahl an Einstellungsmöglichkeiten für das Mikroskop und kann automatisch Bilder aus einem Bioreaktor während einer Kultivierung aufzeichnen. Zur automatisierten Auswertung der Bilder wurde die Software *In-situ-Analysis* erstellt. Sie verfügt ebenfalls über eine grafische Benutzeroberfläche und ist modular aufgebaut, sodass für die Auswertung der vorliegenden Bilder geeignete Auswertealgorithmen in die Anwendung eingebunden werden können. Zur Bestimmung des Bewuchsgrades von mit Tierzellen bewachsener Microcarrier wurde ein entsprechender Algorithmus erstellt. Die Fehler der Ergebnisse hängen stark von der Bildqualität ab und lagen im Durchschnitt bei etwa 10 %.

Schlagworte: Experimental Design, Korrelationsanalyse, Neuronale Netze, Digitale Bildverarbeitung, Prozessdatenanalyse**,** Prozessoptimierung

Abstract

To identify the kinetic parameters of a reaction, measurements are necessary. For a reaction, that can be described by a Michaelis-Menten kinetic, the Fisher-Information-Matrix has been analyzed to determine optimal experimental conditions which allow the most accurate parameter estimation. A genetic algorithm has been applied for optimization. The results have shown, that for a fed-batch process the Cramer-Rao-Lower-Bound is 40 % lower for K_m and 18 % lower for v_{max} compared to a batch process.

In the second part of the thesis an industrial fermentative process has been analyzed, with the objective to identify the most important influencing variables for the yield and perform a process optimization. With a correlation and pricipal component analysis the viscosity, the citric acid concentration and the amount of substrate fed to the process could be found as influencing variables. By using artificial neural networks an on-line simulation of the process could be accomplished. The net was able to predict the product concentration 12 hours forward and the yield at end of fermentation with an error of 6 % in average. With the input of modified data sets into the network it could be shown that by increasing the stirrer speed and using a different substrate feeding strategy a gain of product yield can be expected.

The third topic of this work is the development of a control software for an in-situ microscope and a software for image analysis. The program *In-situ-Control* offers a grafical user interface to set a variety of microscope options and can be used to automatically acquire pictures from a bioreactor during a cultivation. To automatize image analysis the program *In-situ-Analysis* has been developed. It also has a grafical user interface and comes with a modular structure, so that algorithms suitable for the available image data can be linked to the application. For the evaluation of the degree of population of microcarriers, that are overgrown by animal cells, an algorithm has been developed. The calculation errors depend on the image quality and were 10 % in average.

Keywords: Experimental Design, Analysis of correlations, Neural networks, Digital image processing, Analysis of process data, Process Optimization

1 Einleitung

1.1 Experimental Design zur optimalen Parameterbestimmung bei einer enzymatischen Reaktion

Die Genauigkeit, mit der man Parameter einer Reaktionskinetik bestimmen kann, hängt von der Durchführung der Reaktion und der Messungen ab. Durch Experimental-Design-Rechnungen können optimale Reaktions- und Messbedingungen erhalten werden, bei denen der Fehler der Parameterbestimmung minimal ist. In dieser Arbeit wurde der Experimental- Design-Ansatz auf eine enzymatische Reaktion, die mit einer Michaelis-Menten-Kinetik beschrieben werden kann, angewendet. Dabei wurde insbesondere untersucht, ob sich zur möglichst genauen Bestimmung der Parameter eher die Durchführung eines Batch oder eines Fed-Batch-Prozesses eignet. Mit Hilfe eines Optimierungsverfahrens (genetischer Algorithmus) wurden optimale Werte für die Substratanfangskonzentration, die Zeitpunkte der Substratzugabe und optimale Messzeitpunkte ermittelt.

Bei Bioprozessen, egal ob sie großtechnisch oder im Labormaßstab durchgeführt werden, ist es von entscheidender Bedeutung prozessrelevante Messgrößen zu bestimmen und auszuwerten. Durch Aufzeichnung und sinnvolle Analyse dieser Messgrößen ist man in der Lage den Prozess besser zu verstehen, schließlich optimieren zu können und dann optimal zu führen. Im Rahmen dieser Doktorarbeit wurden zwei Software-Systeme erstellt, die zur Aufnahme und Auswertung von Prozessdaten verwendet werden.

1.2 Prozessanalyse mittels statistischer Verfahren und neuronaler Netze

Bei dem betrachteten Prozess handelt es sich um eine großtechnische Fermentation mit *Aspergillus niger var awamori* zur Herstellung von Chymosin. Von diesem Prozess lagen über einen Zeitraum von mehreren Jahren aufgezeichnete Online- und Offline-Daten von über 100 Prozessdurchläufen vor. Darunter sind Messungen der

Abgaskonzentration (O_2 und CO_2), Substrat- und Produktkonzentration, Temperatur, Druck, Energieeintrag (z.B. durch das Rührwerk) und andere Prozessgrößen.
In einem ersten Schritt wurden mit statistischen Methoden (Korrelationsanalyse, Hauptkomponentenanalyse) Zusammenhänge und Verbindungen zwischen den Prozessgrößen dieses Datensatzes analysiert. Darauf aufbauend wurde der Prozess mit Hilfe neuronaler Netze weiter untersucht und eine Software erstellt, die in der Lage ist die Produktkonzentration aus den Prozessgrößen vorherzusagen. Dazu wurden eine Vielzahl von unterschiedlichen Netzen mit dem vorliegenden Datenmaterial für diese Aufgabe trainiert. Die Netze unterschieden sich hinsichtlich ihrer Struktur (Schichtanzahl, Neuronenanzahl, etc.) oder, als weiteres Beispiel, durch die verwendete Trainingsfunktion. Das erfolgreichste Modell wurde schließlich für die Vorhersage der Produktkonzentration (während des laufenden Prozesses) verwendet.
Durch Eingabe von artifiziell erzeugten Eingabedatensätzen wurde ermittelt, wie sich die Vorhersage der Produktkonzentration durch das neuronale Netz verändert. Ziel war es dabei zu untersuchen bei welchen Prozessparametern Veränderungen zu einer Steigerung der Prozessausbeute führen können.

1.3 In-situ-Mikroskopie und digitale Bildverarbeitung

Bei der In-situ-Mikroskopie werden Organismen während einer Kultivierung direkt im Bioreaktor beobachtet. Das Ziel ist die einfache und automatisierte Überwachung bzw. Aufzeichnung wichtiger Parameter bei Kultivierungen – der Zelldichte und der Morphologie der Zellen. Da bei dieser Messmethode das Entnehmen von Proben entfällt ist die Messung weniger zeitintensiv und außerdem wird die Gefahr einer Kontaminierung des Kulturmediums durch Fremdorganismen ausgeschlossen. Das Bild aus der Messzone des Mikroskops wird nicht wie bei der traditionellen Mikroskopie auf ein Okular sondern auf eine Digitalkamera abgebildet, die die Bilddaten an einen Rechner weiterleitet. Im Rahmen dieser Doktorarbeit ist die Planung und Erstellung einer Software durchgeführt worden, mit der das Mikroskop benutzerfreundlich bedient werden kann und die das Abspeichern von Bildern erlaubt. Des weiteren ist ein Auswertungsprogramm erstellt worden, das die Auswertung der aufgezeichneten Bilder mittels Algorithmen der digitalen Bildverarbeitung gestattet. Im Vordergrund stand bei

der Planung dieses Programms, dass es eine flexible Struktur hat und Auswertealgorithmen für verschiedene Zelltypen nachgerüstet werden können. Bereits vorhandene Algorithmen wurden so angepasst, dass sie im Auswertungsprogramm nutzbar sind. Für die Auswertung mit Zellen bewachsener Microcarrier wurde ein Algorithmus erstellt, welcher aus einem Bild den Bewuchsgrad des Microcarriers ermitteln kann.

2 Theorie

2.1 Experimental Design für die optimale Bestimmung von Modellparametern

Ein wichtiger Ansatz zum Verständnis chemischer Prozesse ist deren Beschreibung durch theoretische Modelle. Für die sinnvolle Anwendung des Modells ist es notwendig die Modellparameter durch Messungen zu bestimmen. Das Ziel beim Experimental Design ist das Ermitteln einer optimalen Messstrategie, mittels der die Parameter des Modells mit der größtmöglichen Genauigkeit gemessen werden können. Dazu ist neben dem Modell die Kenntnis grober Schätzwerte P_0 für die Parameter notwendig. Es wird der erwartete Verlauf des Gütefunktionals $E<F^2>$ bei kleinen Veränderungen der Parameter ΔP berechnet. Durch Variation der Messstrategie (z. B. der Messzeitpunkte t_i) wird ein besonders sensitiver Verlauf des Gütefunktionals gesucht.

Kann ein Prozess durch ein Modell der Form $\dot{x} = f(x,t,P)$ beschrieben werden, lassen sich damit die zu erwartenden Messwerte mit folgendem Messmodell errechnen: $\hat{y}_i = g(x,t_i,P)$. Hierbei ist t die Prozesszeit und P ist ein Vektor mit Parametern des Prozesses. Die Messwerte y_i^M liegen meist diskret an den Messzeitpunkten t_i vor. Gegenüber den anhand des Modells zu erwartenden Werten $\hat{y}_i$ ergibt sich die Differenz e_i:

$$e_i = y_i^M - \hat{y}_i \qquad \{1\}$$

Die Messfehler werden im Gütefunktional F durch die inverse Matrix der Messfehlerkovarianz Q berücksichtigt (N ist hierbei die Anzahl der Messungen):

$$F^2 = \sum_i^N e_i^T Q_i e_i \qquad \{2\}$$

Um den Einfluss kleiner Parameteränderungen auf den Verlauf des Funktionals zu untersuchen wird der zu erwartende Messwert in eine Taylor-Reihe 1. Ordnung entwickelt. Der Entwicklungspunkt ist der Schätzwert für die Modellparameter P_0.

$$\hat{y}_i = g(x,t_i,P) \approx g\,(x,t_i,P)\Big|_{x_0,t_i,P_0} + \left.\frac{dg}{dP}\right|_{x_0,t_i,P_0} (P-P_0) \qquad \{3\}$$

Führt man als Abweichung von den optimalen Parametern $\Delta P = P - P_0$ ein folgt:

$$\hat{y}_i = g(x,t_i,P_0+\Delta P) \approx g(x,t_i,P)\Big|_{x_0,t_i,P_0} + \left.\frac{dg}{dP}\right|_{x_0,t_i,P_0} \Delta P \qquad \{4\}$$

Der erste Summand auf der rechten Seite ist der zu erwartende Messwert unter Verwendung der optimalen Parameter und sollte daher nahe dem wahren Messwert y_i^W liegen. Der Messwert setzt sich aus dem wahren Wert sowie dem Messrauschen zusammen:

$$y_i^M = y_i^W + \varepsilon_i \qquad \{5\}$$

Unter der Annahme, dass sich das Messrauschen durch weißes Rauschen beschreiben lässt, folgt für den Erwartungswert von ε_i:

$$E\left\langle \varepsilon_i \right\rangle = \overline{\varepsilon} = \frac{1}{N}\sum_i^N \varepsilon_i = 0 \qquad \{6\}$$

Einsetzen der Gleichungen {1}, {4} und {5} in Gleichung {2} ergibt folgenden Ausdruck für das Gütefunktional:

$$F^2 = \sum_i^N (y_i^w + \varepsilon_i - g(x,t_i,P)\Big|_{x_0,t_i,P_0} - \left.\frac{dg}{dP}\right|_{x_0,t_i,P_0} \Delta P)^T Q_i$$
$$(y_i^w + \varepsilon_i - g(x,t_i,P)\Big|_{x_0,t_i,P_0} - \left.\frac{dg}{dP}\right|_{x_0,t_i,P_0} \Delta P) \qquad \{7\}$$

Da der Erwartungswert $E\left\langle g(x,t_i,P)\Big|_{x_0,t_i,P_0}\right\rangle$ nahezu gleich dem wahren Wert y_i^W ist, vereinfacht sich der Erwartungswert des Gütefunktionals zu:

$$E\left\langle F^2 \right\rangle = E\left\langle \sum_i^N (\varepsilon_i - \left.\frac{dg}{dP}\right|_{x_0,t_i,P_0} \Delta P)^T Q_i (\varepsilon_i - \left.\frac{dg}{dP}\right|_{x_0,t_i,P_0} \Delta P) \right\rangle$$
$$= E\left\langle \sum_i^N \varepsilon_i{}^T Q_i \varepsilon_i \right\rangle + E\left\langle \Delta P^T \left[\sum_i^N \left[\left.\frac{dg}{dP}\right|_{x_0,t_i,P_0} \right]^T Q_i \left[\left.\frac{dg}{dP}\right|_{x_0,t_i,P_0} \right] \right] \Delta P \right\rangle \quad \{8\}$$

Der Ausdruck in der großen eckigen Klammer des zweiten Terms ist die Fisher-Informations-Matrix FIM:

$$FIM = \left[\sum_i^N \left[\left.\frac{dg}{dP}\right|_{x_0,t_i,P_0} \right]^T Q_i \left[\left.\frac{dg}{dP}\right|_{x_0,t_i,P_0} \right] \right] \quad \{9\}$$

Gesucht ist diejenige Messstrategie, die zu einer starken Krümmung des Gütefunktionals um das Minimum führt. Je stärker die Krümmung ist, desto besser ist das Minimum festgelegt und desto genauer ist die Parameterbestimmung durch die Messung. Die Berechnung der FIM ist dafür der entscheidende Schritt, denn die Krümmung des Funktionals K hängt mit den Eigenwerten der FIM zusammen:

$$K = 2\lambda \quad \{10\}$$

Die Bestimmung einer optimalen Messstrategie ist folglich ein Optimierungsproblem, das sich mit geeigneten Verfahren (genetischer Algorithmus, Simplex-Verfahren) lösen lässt. Veränderliche Größen sind dabei die Messzeitpunkte t_i und evtl. andere Parameter der Messung. Ist der Zielwert, den es zu maximieren gilt, der kleinste Eigenwert der FIM spricht man vom E-Kriterium. Andere Kriterien, die zur Lösung des Optimierungsproblems angewendet werden können, sowie detaillierte Informationen über Experimental Design sind in der Literatur zu finden [PUKELSHEIM], [GOODWIN].

2.2 Korrelationsanalyse

Mit Hilfe der Korrelationsanalyse kann der Zusammenhang von Variablenpaaren in einer Datenmenge untersucht werden. Dazu wird pro Variablenpaar der Korrelationskoeffizient berechnet, der eine Aussage über die Stärke der Streuung der Variablen um eine lineare Verknüpfung bietet [EHRENBERG], [ESBENSEN], [DANZER], [ANDERSON].

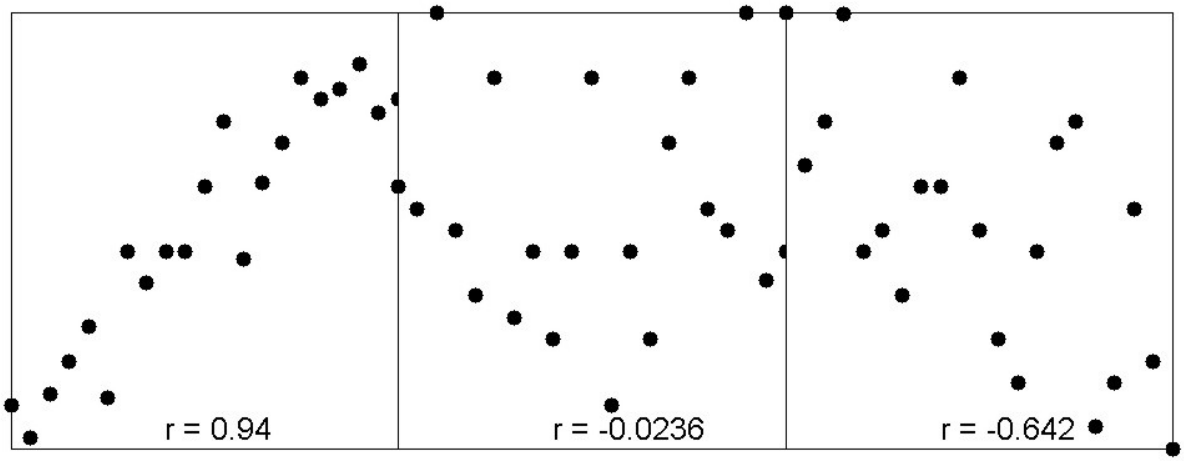

Abbildung 1. Beispiele für stark positive (links), keine (mitte) und negative (rechts) Korrelationen.

Der Korrelationskoeffizient liegt im Wertebereich von –1 bis +1, wobei ein Wert nahe an +1 bedeutet, dass zwischen den betrachteten Variablen ein positiver Zusammenhang mit geringer Streuung vorliegt. Beide Variablen treten also sehr häufig gemeinsam mit hohen Werten oder beide gemeinsam mit geringen Werten auf. Anders ausgedrückt bedeutet dies, dass sich bei einer Auftragung dieser Variablen annähernd eine Gerade ergibt (siehe Abbildung 1, links). Ein Korrelationskoeffizient, der sehr nahe an 0 liegt bedeutet, dass zwischen den jeweiligen Variablen kein linearer Zusammenhang besteht. In einem Diagramm ergibt sich in diesem Fall eine Punktwolke ohne eindeutige Vorzugsrichtung (siehe Abbildung 1, mitte). Liegt der Korrelationskoeffizient nahe bei –1 treten besonders häufig hohe Werte der einen Variable zusammen mit niedrigen Werten der anderen Variable auf und umgekehrt. Bei einer Auftragung der Variablen entsteht ungefähr eine Linie (siehe Abbildung 1, rechts).

$$r_{xy} = \frac{\mathrm{cov}(x, y)}{std(x)std(y)} \quad \{11\}$$

$$\mathrm{cov}(x, y) = \frac{\sum_{i=1}^{n}(x_i - \overline{x})(y_i - \overline{y})}{(n-1)} \quad \{12\}$$

$$std(x) = \sqrt{\frac{\sum_{i=1}^{n}\left(x_i - \overline{x}\right)^2}{n-1}} \quad \{13\}$$

Die Berechnung des Korrelationskoeffizienten, der nach dem Statistiker Karl Pearson in der Literatur oft auch als „Pearsonscher Korrelationskoeffizient“ bezeichnet wird, erfolgt durch Division der Kovarianz der betreffenden Variablen x und y durch die Standardabweichungen beider Variablen (siehe Gleichung {11}).

Die Korrelationsanalyse ist besonders geeignet, um sich in größeren Datensätzen einen Überblick über die Zusammenhänge der Variablen zu verschaffen. Man muss aber bei der Interpretation der Korrelationskoeffizienten sehr vorsichtig sein aufgrund der errechneten Werte quantitative Aussagen zu treffen. Ein Korrelationskoeffizient von z.B. 0,9 sagt aus, dass zwischen den betrachteten Variablen mit hoher Wahrscheinlichkeit ein linearer Zusammenhang besteht, aber er sagt nichts darüber aus, um wie viel die Variable y bei jeder Erhöhung von x anwächst bzw. sinkt. D.h. die Steigung geht aus dem Wert des Korrelationskoeffizienten nicht hervor und kann sowohl sehr flach als auch sehr steil sein. So können zwei Variablenpaare einen sehr ähnlichen Korrelationskoeffizient aber sehr unterschiedliche zugrundeliegende Zusammenhänge haben. Auch das umgekehrte Phänomen kann auftauchen: Zwei Variablenpaare mit dem gleichen zugrundeliegenden Zusammenhang haben völlig unterschiedliche Korrelationskoeffizienten. Dies liegt daran, dass die Streuung der Variablen relativ, also im Verhältnis zur Variation der Variablen selbst, in die Berechnung eingeht [EHRENBERG].

Eine weitere Tücke bei der Interpretation der Korrelationskoeffizienten ist, dass man von den berechneten Werten nicht auf eine ursächliche Verbindung, einen kausalen Zusammenhang, schließen kann. Haben zwei Variablen x und y einen relativ hohen

Korrelationskoeffizienten von z.B. 0,9 bedeutet dies nicht, dass der Grund für eine Zu- oder Abnahme von x bei der Variable y zu finden ist bzw. umgekehrt. Korrelation ist also nicht gleichzusetzen mit Verursachung (Anm.: Allerdings kann es keine Verursachung ohne Korrelation geben). Es ist häufig entscheidend bei der Beurteilung der Ergebnisse einer Korrelationsanalyse Wissen um die Herkunft und die Art der Erzeugung des verwendeten Datensatzes mit einzubeziehen, um Fehl- bzw. Überinterpretationen zu vermeiden [ESBENSEN], [EHRENBERG].
Trotz der dargestellten Nachteile und Tücken, die die Korrelationsanalyse mit sich bringt, ist sie ein wertvolles Hilfsmittel, um Erkenntnisse über die in einem Datensatz enthaltenen Zusammenhänge zu gewinnen.

2.3 Hauptkomponentenanalyse

Das Ziel bei der Hauptkomponentenanalyse (PCA, Principal Component Analyses) ist es, eine Datenmatrix in ein relevanteres Koordinatensystem zu transformieren, die Dimensionalität der Daten zu verringern und Information von Rauschen zu trennen [ESBENSEN], [OTTO], [MASSART].
Den Inhalt einer Datenmatrix, die n Variablen von m Proben, Messungen oder allgemein Objekten enthält, kann man sich als Punktwolke in einem n-dimensionalen Raum vorstellen. Dies ist in Abbildung 2A beispielhaft an Daten in einem dreidimensionalen Koordinatenraum dargestellt. Das im folgenden an diesem Beispiel erläuterte Prinzip der PCA ist aber genauso auch für höherdimensionale Datenmengen gültig. Im ersten Schritt wird das Koordinatensystem in das Zentrum der Daten bewegt. Diejenige Raumrichtung in der die Daten die größte Varianz aufweisen ist in Abbildung 2B durch einen blauen Pfeil angedeutet. Durch Rotation des Koordinatensystems wird die x-Achse mit der Raumrichtung dieses Vektors zur Deckung gebracht. Diese neue Achse stellt die erste Hauptkomponente dar. In den folgenden Schritten wird jeweils die Richtung der nächstgrößten Varianz bestimmt, die zudem orthogonal zu allen vorherigen Hauptkomponenten sein muss (gelber Pfeil in Abbildung 2D), und daran anschließend wird die nächste Achse des Koordinatensystems in diese Richtung gedreht. Auf diese Weise werden die Daten in ein neues Koordinatensystem

transformiert, in dem die Achsen bzw. die Hauptkomponenten aufeinander senkrecht stehen und in Reihenfolge absteigender Varianz sortiert sind. Bei der Hauptkomponentenanalyse wird also eine sukzessive varianzmaximierende orthogonale Rotationstransformation vorgenommen. Mathematisch ausgedrückt werden die Daten in der ursprünglichen Datenmatrix X durch Score- und Loadingvektoren dargestellt. Dabei bezeichnet t_n den Scorevektor der n-ten Hauptkomponente und p_n den entsprechenden Loadingvektor. Der durch die Hauptkomponenten nicht erklärte Anteil der Daten verbleibt in e. Dies ist in Gleichung {14} aufgeführt.

$$X = \vec{t}_1\vec{p}_1^{'} + \vec{t}_2\vec{p}_2^{'} + \vec{t}_3\vec{p}_3^{'} + \ldots + \vec{t}_n\vec{p}_n^{'} + e \qquad \{14\}$$

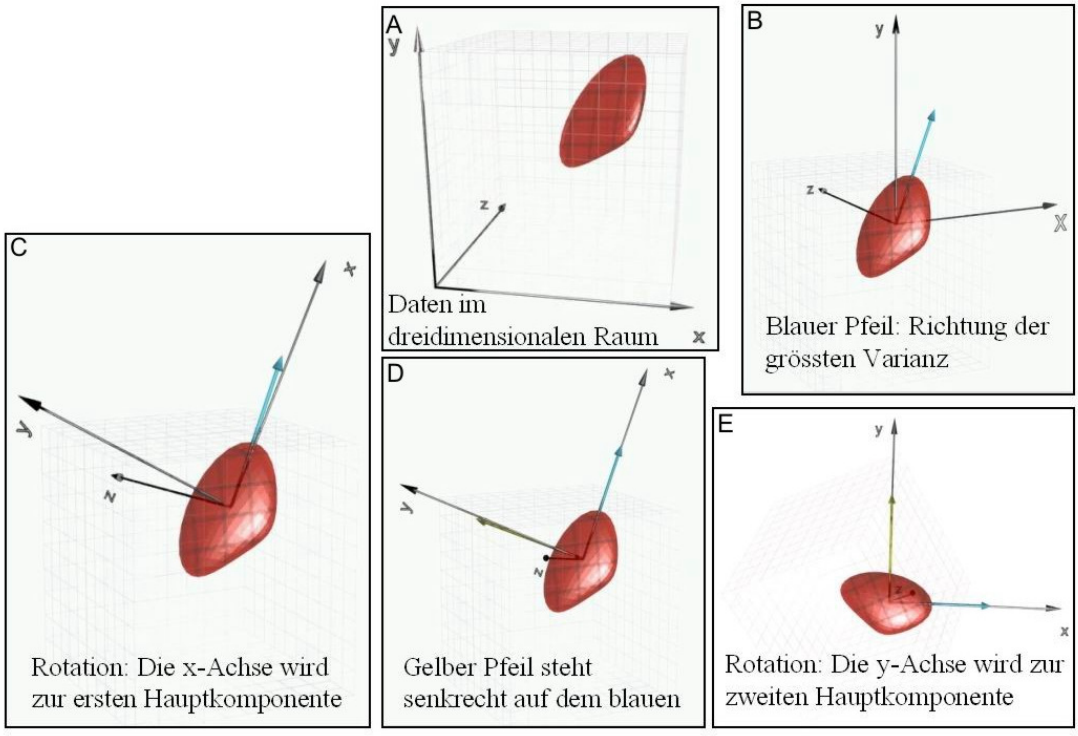

Abbildung 2. Prinzip der Hauptkomponentenanalyse. [Quelle: www.digi-musik.de/PCA/bsp3d.html, *Eduard Hein, Christian Becker*]

Ein Loadingvektor enthält die Cosinuswerte der Winkel zwischen der Hauptkomponente und den Achsen im ursprünglichen Raum. Die Anzahl seiner Komponenten entspricht der Anzahl an Variablen in X. Ein Scorevektor enthält die Abstände zwischen Koordinatenursprung, bzw. Hauptkomponentenmittelpunkt und den Projektionspunkten einer jeden Probe auf die Hauptkomponente. Ein Scorevektor hat

genau so viele Komponenten wie Proben im Datensatz vorhanden sind und stellt die Hauptkomponente, bzw. die neue unabhängige Variable dar.

Ein gängiger Algorithmus, mit dem die Hauptkomponenten berechnet werden können, ist der NIPALS-Algorithmus (Non-linear Iterative Projection by Alternating Least-Squares), der auf Herman Wold zurückgeht [VANDEGINSTE], [ESBENSEN]. Dieser Algorithmus berechnet zuerst die Score- und Loadingvektoren der ersten Hauptkomponente (t_1 und p_1). Die Matrix t_1p_1' wird dann von der ursprünglichen Datenmatrix X abgezogen, ein Rechenschritt, der oft als Deflation bezeichnet wird. Mit der resultierenden Matrix wird dann auf analoge Weise die zweite und alle folgenden Hauptkomponenten berechnet, wobei als Nebenbedingung gilt, dass die neu berechnete Hauptkomponente senkrecht auf den vorherigen stehen muss. Eine ausführliche Darstellung des NIPALS-Algorithmus ist in der Literatur zu finden [ESBENSEN].

Da der Informationsgehalt der Hauptkomponenten (bzw. die beschriebene Varianz in den Originaldaten) nach und nach abnimmt, können die höheren Scores für die Analyse des Datensatzes oft vernachlässigt werden. Eine allgemeingültige Regel bei welcher Hauptkomponentenanzahl die Beschreibung des Datensatzes ausreichend ist gibt es nicht, oft wird die Anzahl der verwendeten Hauptkomponenten aber so gewählt, dass sie ca. 90 % der Varianz beschreiben.

Wichtige Werkzeuge bei der Auswertung eines Datensatzes mit Hilfe der Hauptkomponentenanalyse sind der Scoreplot und der Loadingplot. In einem Scoreplot werden die Werte aus zwei (oder auch drei) Scorevektoren graphisch dargestellt. Ein Plot z.B. der Scorevektoren der ersten beiden Hauptkomponenten wird t_1t_2-Plot genannt. Da die Scores die „Fingerabdrücke" der Objekte (bzw. Proben, Messungen) des Datensatzes projiziert in den Raum der Hauptkomponenten sind, kann ein Scoreplot als „Karte der Objekte" aufgefasst werden. Objekte die im Scoreplot nahe beieinander auftauchen sind sehr ähnlich, während weit entfernte Objekte keine Gemeinsamkeiten haben. Sind im Datensatz also Gruppen mehrerer untereinander ähnlicher Objekte vorhanden, kann im Scoreplot eine Aufspaltung in mehrere Punktgruppen (Cluster) beobachtet werden. Auch eine Erkennung von Ausreißern ist im Scoreplot leicht möglich, da ein Ausreißer sich stark vom Rest des Datensatzes unterscheidet und sich somit weitab von den anderen Objekten im Scoreplot befinden wird.

Bei einem Loadingplot werden die Loadingvektoren von zwei (oder auch drei) Hauptkomponenten gegeneinander geplottet. Während man den Scoreplot als Karte der

Objekte bezeichnen kann, passt bei Loadingplots die Bezeichnung „Karte der Variablen“. Variablen, die ein hohes Maß an systematischer Variation aufweisen haben hohe Absolutwerte bei den Loadings und befinden sich weit vom Ursprung entfernt. Dagegen sind unwichtige Größen nahe am Ursprung bzw. haben geringe Loadings. Dabei ist es entscheidend, bei der Beurteilung wie wichtig eine Variable ist auch die erklärte Varianz der jeweiligen Hauptkomponente mit einzubeziehen. Erklärt z.B. die erste Hauptkomponenten 75 % der Varianz aber die zweite nur noch 5 % ist ein hohes Loading auf der ersten Hauptkomponenten „15 mal so wichtig“ wie eines auf der zweiten. Eine weitere sehr nützliche Information, die aus dem Loadingplot entnommen werden kann, ist die Information über Zusammenhänge (Korrelationen, siehe Kapitel 2.2) der Variablen. Liegen zwei Variablen dicht zusammen auf derselben Seite des Koordinatensystems besteht zwischen ihnen eine positive Korrelation, liegen sie auf entgegengesetzten Seiten (ungefähr auf einer gedachten Linie durch den Ursprung) besteht eine negative Korrelation. Variablen, die ungefähr einen Winkel von 90° bezüglich des Ursprungs einschließen, sind unkorreliert. Eine weiterführende Darstellung zur Interpretation von Score- und Loadingplots ist in der Literatur zu finden [ESBENSEN].

2.4 Neuronale Netze

Neuronale Netze, die oft auch künstliche neuronale Netze (KNN) bzw. im Englischen artificial neural networks (ANN) genannt werden, sind informations- oder signalverarbeitende Systeme, die aus einer großen Anzahl von einfachen Verarbeitungseinheiten (Neuronen, Zellen) bestehen. Sie sind, zumindest ansatzweise, Nachbildungen der Struktur und Funktion von Nervensystemen in lebenden Organismen. Je nach Fachrichtung und Betrachtungsweise des Anwenders kann die Motivation des Studiums und der Implementation von künstlichen neuronalen Netzen ganz verschieden sein: Bei (Neuro-) Biologen, Psychologen und Medizinern steht im Vordergrund durch die computergestützte Simulation neuronaler Netze neue Erkenntnisse zu den Abläufen und den Eigenschaften der biologischen Systeme zu gewinnen. In der (Bio-) Informatik, Physik und Mathematik stehen bei neuronalen Netzen eher ihre Eigenschaften als parallel operierende, lernfähige und sehr effiziente Algorithmen im Vordergrund.

Neuronale Netze (im biologischen Sinn) bestehen aus einer sehr großen Anzahl von Nervenzellen. Die Anzahl von Nervenzellen z.B. im menschlichen Gehirn wird auf ca. 100 Milliarden geschätzt. Der typische Aufbau einer Nervenzelle ist in Abbildung 3 zu sehen: Sie bestehen aus einem Zellkörper (Soma), der einen Kern (Nukleus) und viele Organellen, wie z.B. Mitochondrien oder das endoplasmatische Retikulum enthalten, die u.A. die Energieversorgung der Zelle sicherstellen und Enzyme und andere notwendige Zellbestandteile herstellen. Die Dendriten sind hochverzweigte Nervenfasern, über die die Zelle Eingangssignale anderer Neuronen aufnimmt. Die Weiterleitung eines Nervenimpulses geschieht mit dem Axon, das eine Länge von wenigen Millimetern bis zu fast einem Meter besitzen kann. Auch das Axon ist stark verzweigt. An seinen Endpunkten befinden sich Verdickungen, die Synapsen, welche die Kontaktstellen zwischen dem Axon einer Zelle und den Dendriten anderer Nervenzellen darstellen[1].

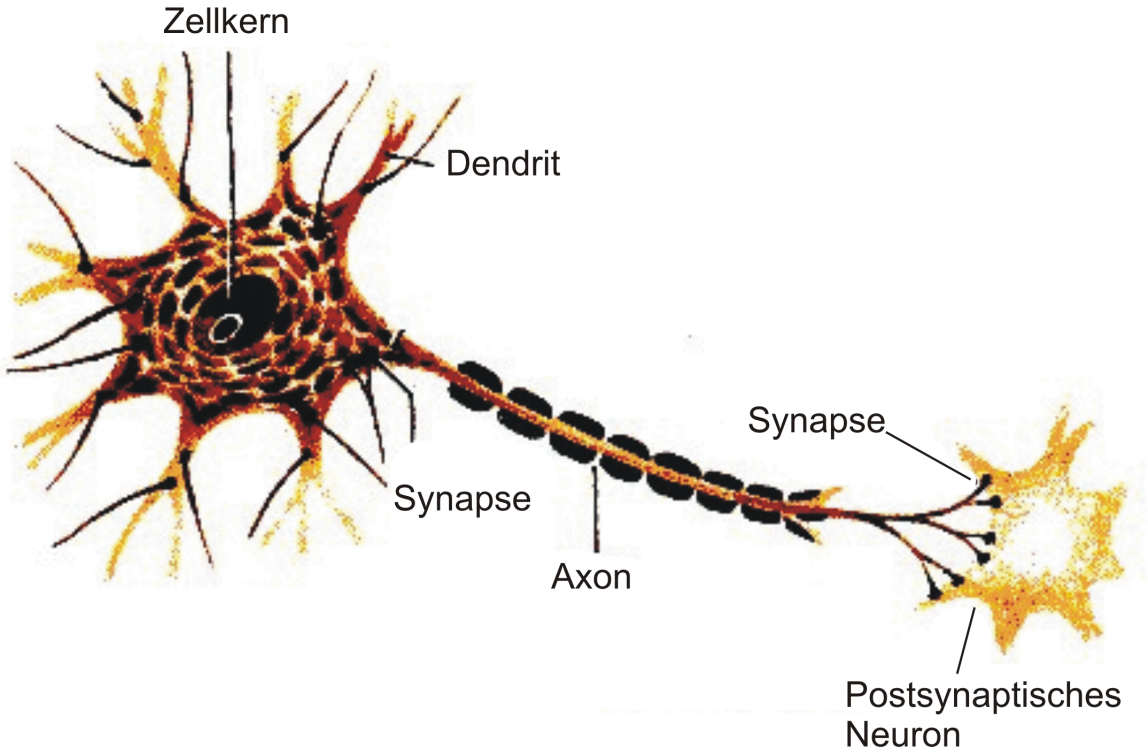

Abbildung 3. Schematischer Aufbau einer Nervenzelle (Neuron). Quelle: [SITTER]

[1] Es gibt auch Kontaktstellen zwischen Dendriten verschiedener Zellen bzw. auch direkt zwischen den Axonen.

Menschliche Zellen besitzen ca. 1000 bis 10000 Synapsen, mit denen Signale an nachfolgende Neuronen weitergeleitet werden. Synapsen können exzitatorisch oder inhibitorisch wirken, also das Signal verstärken oder hemmen. Die Information steckt hauptsächlich in der Frequenz und der Dauer mit der die Signale gesendet werden. Für die Unterscheidung, ob ein Signal durch ein Neuron weitergeleitet wird oder nicht, ist nicht die Amplitude oder Form des Signals entscheidend, sondern die Frequenz und die Dauer mit der Signale an einer Nervenzelle ankommen. Eine genauere Erläuterung zu biologischen neuronalen Netzen, Neuronen und zur Informations- und Signalverarbeitung in neuronalen Netzen ist u. A. bei Zell und Stryer zu finden [ZELL], [STRYER].

Die Komplexität der Struktur und Funktion neuronaler Netze in der Natur, die hier nur stark vereinfachend angesprochen wurde, wird in künstlichen neuronalen Netzen zwar als Vorbild angenommen aber häufig nur stark idealisiert umgesetzt. Eigenschaften des biologischen Vorbildes, die dabei oft erhalten bleiben sind die massive Parallelität einer großen Anzahl einfach aufgebauter Verarbeitungseinheiten (Neuronen), die Existenz gerichteter Verbindungen zwischen den Neuronen sowie eine hohe Plastizität (Verbindungsgewichte zwischen den Neuronen können verstärkt oder verringert werden durch Lernen). Die wichtigsten Unterschiede zwischen biologischen und künstlichen neuronalen Netzen sind die viel geringere Anzahl an Neuronen und Verbindungen und, dass die genauen zeitlichen und chemischen Abläufe in den Axonen, Synapsen etc. nicht modelliert werden. Des weiteren werden chemische Interaktionen zwischen räumlich benachbarten Neuronen sehr häufig nicht berücksichtigt, durch die Neuronen beispielweise als Gruppe angeregt oder gehemmt werden können. Auch werden oft biologisch nicht plausible Lernregeln zur Festlegung der Verbindungsgewichte verwendet. Vom Standpunkt der Anwendung neuronaler Netze betrachtet ist eine exakte Modellierung des biologischen Vorbildes zur Lösung technischer oder naturwissenschaftlicher Problemstellungen auch nicht erforderlich [ZELL], [CICHOCKI]. Die Anwendungsmöglichkeiten neuronaler Netze sind sehr breit gefächert: Hochleistungs-Autopiloten in Flugzeugen, Automatische Verarbeitung von Dokumenten (z.B. Schecks/Überweisungen in Bankautomaten), Finanzanalyse, Robotik (u. A. zur Steuerung) und Sprach- sowie Gesichtserkennung. In der Chemie bzw. Verfahrenstechnik werden sie zur Planung und Management von Prozessabläufen, Qualitätskontrolle/-sicherung und zur dynamischen Vorhersage, Simulation und

Optimierung von Prozessen eingesetzt [ZUPAN]. So verwendeten z. B. Becker et al. [BECKER] neuronale Netze zur Prozesssimulation und -optimierung einer Bierfermentation. Als Ergebnis dieser Optimierung konnte die Prozesszeit um 20 % gesenkt werden und somit die Auslastung der Anlage erhöht werden. In einer Arbeit von Valdez-Castro et al. [VALDEZ-CASTRO] konnte mit einem neuronalen Netz eine Prozesssimulation einer Fed-batch-Fermentation von *Bacillus thurigiensis* durchgeführt werden. Der Fehler zwischen dem von dem Netz vorhergesagten Werten und den Messwerten betrug dabei nur ca. 2 %.

2.4.1 Modell eines Neurons

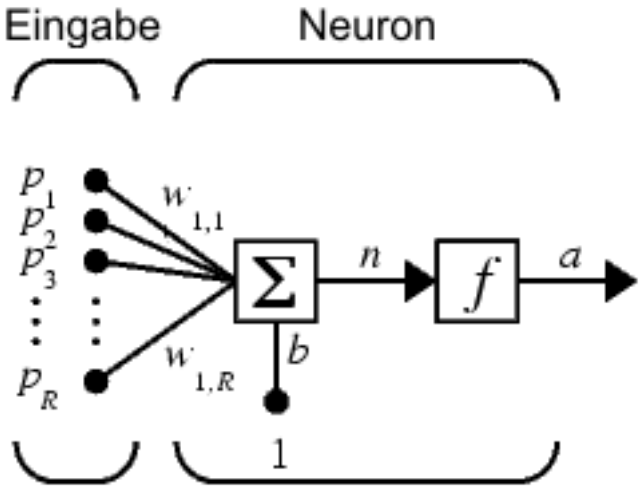

Abbildung 4. Ein Neuron mit Vektoreingabe. p ist der Eingangsvektor mit R Elementen, $w_{1,1}$ bis $w_{1,R}$ sind die Gewichte des Neurons bezüglich der ersten bis R-ten Eingabe, b ist das Bias (Verschiebung, hier auf eins gesetzt). Die Nettoeingabe (engl. net input) n wird durch die Transferfunktion f in die Ausgabe des Neurons a umgewandelt. Quelle: [DOKUNNT]

Ein einfaches Modell eines Neurons ist in Abbildung 4 dargestellt. Bei der Simulation neuronaler Netze im Rechner dient ein Neuron als einfache Recheneinheit, die einen Vektor p als Eingabe hat. Die R Elemente des Vektors werden mit der Wichtungsmatrix des Neurons (die in dem Fall nur *eines* Neurons nur ein Zeilenvektor ist, s.u.) und dem Bias in der Nettoeingabe n umgewandelt (Gleichung {15}). Die Nettoeingabe wird über eine Transferfunktion f in die Ausgabe des Neurons a umgerechnet: $a = f(n)$. Zur Bestimmung der Ausgabe eines Neurons wird also jeder Eingabewert entsprechend der Gewichtung in w berücksichtigt (ist ein Element aus w gleich null fließt die

entsprechende Eingabe nicht mit in die Ausgabe ein), durch das Bias kann der Wertebereich der Ausgabe verändert werden und schließlich kann durch eine entsprechende Transferfunktion die Nettoeingabe linear oder nichtlinear abgebildet werden.

$$n = w_{1,1} p_1 + w_{1,2} p_2 + ... + w_{1,R} p_R + b \qquad \{15\}$$

2.4.2 Aufbau neuronaler Netze

Ein neuronales Netz ist in Schichten aufgebaut, wie in Abbildung 5 schematisch am Beispiel eines dreischichtigen Netzes dargestellt ist. Die Neuronen in der Eingabeschicht leiten die Eingaben in das Netz weiter und werden Eingabeneuronen genannt. Jedes dieser Neuronen ist mit jedem anderen Neuron der folgenden Schicht, der sogenannten verdeckten Schicht, verbunden. Die Stärke der Verbindung ist durch die Wichtungsparameter festgelegt. Die dritte Schicht, die Ausgabeschicht, besteht in diesem Beispiel nur aus einem Neuron und hat hauptsächlich die Aufgabe die Ausgaben der verdeckten Schicht zur Gesamtausgabe des Netzes zusammenzufassen.

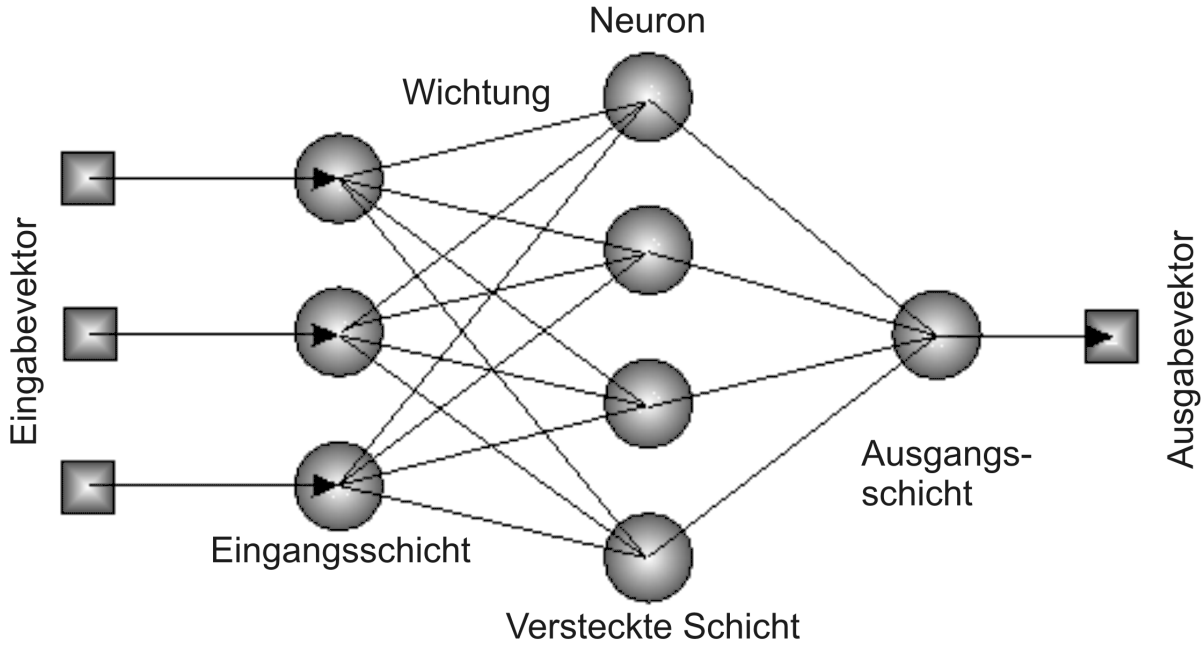

Abbildung 5. Schematischer Aufbau eines neuronalen Netzes. Quelle: [MALMGREN]

Bei diesem Netz handelt es sich um ein Feed-Forward-Netz, da es keinen Pfad gibt, der von einem Neuron direkt oder über zwischengeschaltete Neuronen wieder zu diesem Neuron zurückführt, d.h. es gibt keine Schleifen (Feedback-Loops). Ein detailliertere Darstellung eines dreischichtigen Feed-Forward-Netzes ist in Abbildung 6 gegeben. Diese Abbildung schließt sich an die in Abbildung 4 verwendete Nomenklatur[2] an. Bei dieser Nomenklatur wird durch hochgestellte Indizes die Nummer der Schicht angegeben. Des weiteren wird in dieser Nomenklatur unterschieden zwischen Neuronen die mit Eingaben des Netzes verbunden sind und Neuronen, die mit anderen Neuronen verbunden sind. Bei ersteren wird die Wichtungsmatrix als IW (*input weights)*, bei letzteren als LW (*layer weights*) bezeichnet. Anhand dieser Darstellung lassen sich die Dimensionen der beteiligten Vektoren und Matrizen sowie die mathematischen Zusammenhänge, die zur Berechnung der Netzausgabe führen, sehr gut ablesen. Die Formel, die zur Berechnung der Netzausgabe dient, ist in Gleichung {16} gezeigt. Bei der Rechenoperation $IW^{1,1}p+b^1$ wird der Eingabevektor analog zu Gleichung {15} gewichtet. Es entsteht der net input-Spaltenvektor n^1 der aus S^1-Elementen besteht, wobei S^1 die Neuronenanzahl der ersten Schicht ist. Die Nettoeingabe wird durch die Transferfunktion der ersten Schicht in den Ausgabevektor a^1 umgewandelt, der ebenfalls S^1 Elemente hat. Die Dimension des Ausgabevektors einer Schicht entspricht also der Neuronenanzahl und nicht der Dimension des Eingabevektors.

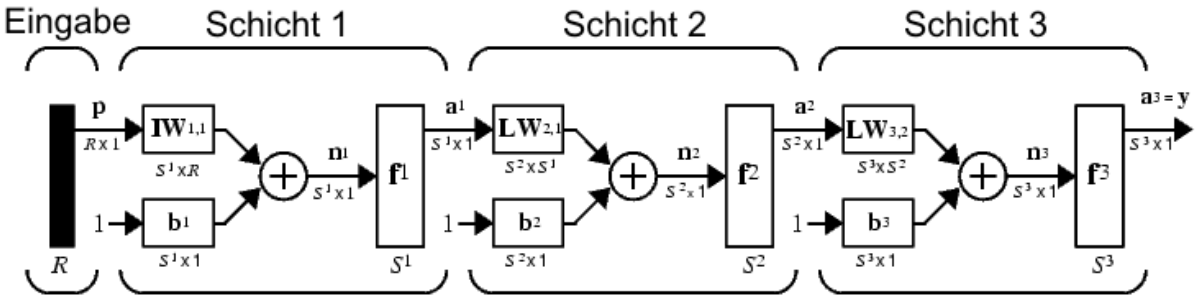

Abbildung 6. Darstellung eines dreischichtigen neuronalen Netzes in Kurzschreibweise. Zur Erläuterung der Abkürzungen siehe Abbildung 4 oder Anhang 8.1. Die Dimensionen der Vektoren und Matrizen stehen jeweils unter den Pfeilen bzw. Kästen. Quelle: [DOKUNNT]

[2] Die Nomenklatur zur Bezeichnung der verschiedenen Bestandteile neuronaler Netze wird auch in der Neural Network Toolbox in Matlab 6.5 angewandt.

$$y = a^3 = f^3(LW^{3,2} f^2(LW^{2,1} f^1(IW^{1,1} p + b^1) + b^2)) + b^3) \quad \{16\}$$

Die Ausgabe der ersten Schicht ist die Eingabe der zweiten Schicht, deren Ausgabe von der dritten Schicht letztlich in die Netzausgabe y umgerechnet wird. Die Ausgaben des neuronalen Netzes hängt also wesentlich von den Wichtungsmatrizen und den Bias-Vektoren ab. Diese sind somit die Parameter des Netzes, die durch *Lernen* an die Problemstellung angepasst werden müssen (s.u.).

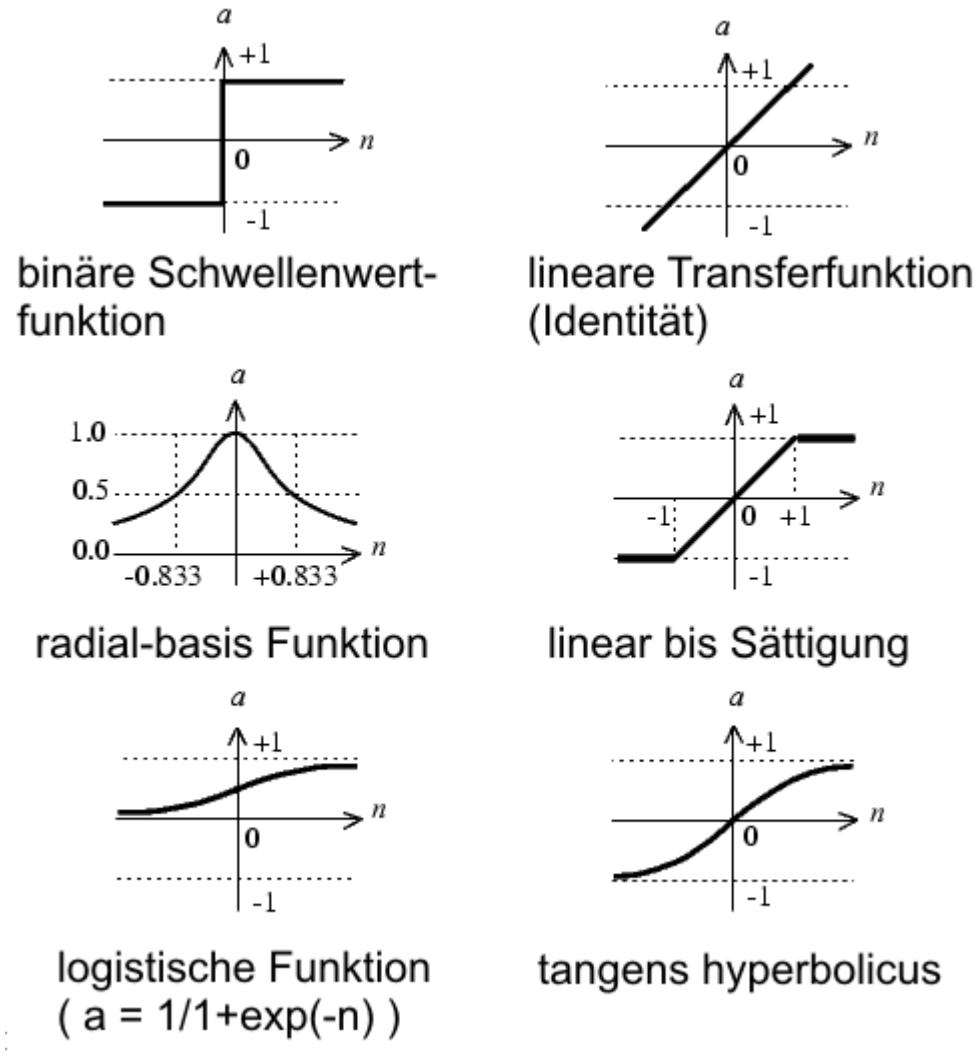

Abbildung 7. Häufig verwendete Transferfunktionen.

Durch mehrschichtige Feed-Forward-Netze können beliebig komplizierte Abbildungen von Eingabevektoren p auf Ausgabevektoren y erreicht werden. Häufig werden bei Feed-Forward-Netzen sigmoidale Transferfunktionen in der Eingabe- und in der/den

versteckten Schicht/en eingesetzt und in der Ausgabeschicht eine lineare Transferfunktion (die Identität). Diese und einige andere gängige Transferfunktionen sind in Abbildung 7 dargestellt. Wie man an den Funktionen erkennt, wird die Ausgabe durch sie auf ein Intervall von 0 bis 1 bzw. auf ein Intervall von –1 bis +1 begrenzt. Eine Limitierung auf diskrete Ausgabewerte (nur -1 oder +1) ist durch die binäre Schwellenwertfunktion möglich. Diese wird oft in Netzen, die zu Klassifizierungsaufgaben verwendet werden sollen, implementiert. Die sigmoiden Transferfunktionen (Abbildung 7, unten links und unten rechts) werden sehr häufig in den verdeckten Schichten von Feed-Forward-Netzen verwendet. Sie bieten ihre höchste Sensitivität auch im Bereich sehr kleiner Eingabewerte ($n \approx 0$), sind aber im Gegensatz zur binären Schwellenwertfunktion stetig und überall differenzierbar. Besonders diese Eigenschaft ist zum Training neuronaler Netze entscheidend, da dazu die Ableitung der Transferfunktion nötig ist (s.u.).

2.4.3 Training neuronaler Netze

Um ein neuronales Netz auf eine gegebene Aufgabe anwenden zu können, muss das Netz durch Lernregeln anhand von Beispielen (Eingangsmustern) angepasst werden. Eine Übersicht über den Lernvorgang, der auch als *Training* bezeichnet wird, ist in Abbildung 8 gegeben. Dabei gibt es eine Vielzahl an Möglichkeiten ein neuronales Netz während des Lernvorgangs zu verändern. Zum Einen können neue Verbindungen zwischen Neuronen erschaffen werden bzw. existierende Verbindungen können gelöscht werden. Zum Anderen kann die Stärke bestehender Verbindungen modifiziert werden. Diese Fälle können durch Anpassung der Werte in den Wichtungsmatrizen realisiert werden. Das Löschen bzw. Erschaffen von Verbindungen sind letztlich nur Grenzfälle der Veränderung von Verbindungsstärken: Zum Erschaffen einer neuen Verbindung wird ein anfangs auf null gesetzter Wichtungsfaktor auf einen Wert ungleich null gesetzt und zum Löschen einer Verbindung wird der entsprechende Wert auf null gesetzt. Weitere Möglichkeiten der Anpassung neuronaler Netze sind u. A. die Veränderung der Transferfunktion und das Löschen bzw. Erschaffen neuer Neuronen. In der Praxis wird die Anpassung neuronaler Netze aber meist durch Modifikation der Gewichte vorgenommen.

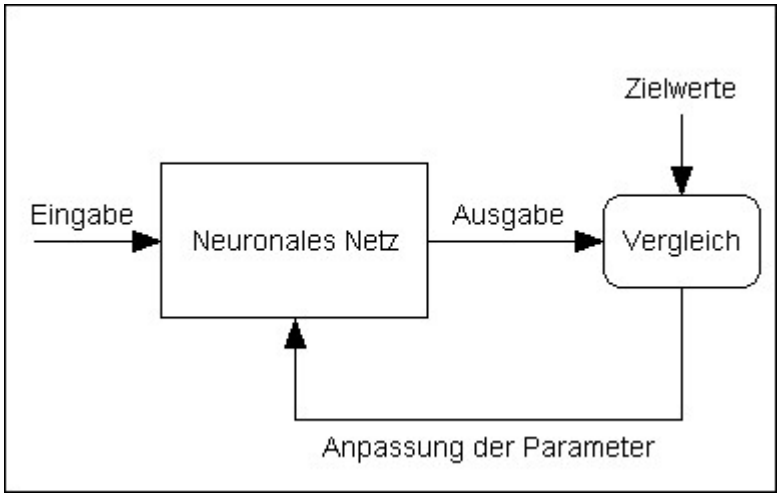

Abbildung 8. Schematischer Ablauf des Trainings neuronaler Netze

Beim Lernen werden drei Verfahren unterschieden: überwachtes, bestärkendes und unüberwachtes Lernen. Beim überwachten Lernen wird dem Netz zu jedem Eingabemuster auch das erwartete, „korrekte“ Ausgabemuster vorgegeben. Ziel des Lernalgorithmus (siehe Kapitel 2.4.4) ist es nun, während wiederholter Präsentation der Eingabe- und Ausgabemuster die Parameter des Netzes so anzupassen, dass es die Assoziation zwischen Eingabe und Ausgabe selbst vornehmen kann. Des weiteren wird bei einigen Lernalgorithmen der Lernvorgang so gestaltet, dass das trainierte Netze auch die Ausgabemuster neuer Eingabemuster (die nicht zum Lernen verwendet wurden) so genau wie möglich wiedergeben kann. Dieses Übertragen auf neue Eingabemuster wird als Generalisierung bezeichnet. Diese Art des Lernens führt im Allgemeinen am schnellsten und sichersten zu Lernerfolgen. Eine biologisch plausiblere Art des Lernens ist das bestärkende Lernen. Hierbei wird dem Netz nicht das korrekte Ausgabemuster vorgelegt, sondern nur, ob die Ausgabe richtig oder falsch war. Die korrekte und/oder „beste“ Ausgabe muss das Netz selbst finden. Diese Art des Lernens ist oft deutlich langsamer als das überwachte Lernen. Bei unüberwachten Lernen schließlich wird während des Lernvorgangs keine Rückmeldung gegeben, sondern dem Netz werden nur Eingabemuster präsentiert. Durch dieses Lernverfahren wird oft erreicht, dass das Netz ähnliche Eingaben in Klassen einordnet. Ein bekanntes Beispiel hierfür sind die selbstorganisierenden Karten von Kohonen [ZELL], [DOKUNNT], [RITTER]. Diese Art des Lernens ist allerdings nicht für alle Problemstellungen sinnvoll anwendbar.

2.4.4 Lernregeln und -algorithmen

Für den Schritt der Parameteranpassung (s. Abbildung 8) während des Lernvorgangs sind Lernregeln nötig, die die Gewichte des Netzes sinnvoll verändern. Eine sehr bekannte Lernregel ist die von Donald O. Hebb bereits 1949 formulierte *Hebbsche Lernregel*: „Wenn Neuron j eine Eingabe von Neuron i erhält und beide gleichzeitig stark aktiviert sind, dann erhöhe das Gewicht w_{ij}." [HEBB]. Die mathematische Form dieser Regel ist in Gleichung {17} dargestellt. Dabei ist Δw_{ij} die Veränderung des Gewichts, η die Lernrate (eine Konstante) und p_i und a_j sind die Eingabe bzw. die Ausgabe des Neurons.

$$\Delta w_{ij} = \eta p_i a_j \qquad \{17\}$$

Diese Lernregel wird häufig bei Neuronen mit binären bzw. diskreten Aktivierungszuständen eingesetzt. Die Delta-Regel bzw. Widrow-Hoff-Regel, sowie die Backpropagation-Regel sind Spezialisierungen der Hebbschen Lernregel. Da der Hauptunterschied zwischen der Widrow-Hoff-Regel und der Backpropagation-Regel darin besteht, dass letztere die Verallgemeinerung der Widrow-Hoff-Regel auf mehrschichtige Netze mit nichtlinearen Aktivierungsfunktionen darstellt, wird hier nur die Backpropagation-Regel vorgestellt.

Das neuronale Netz macht bei gegebenen Gewichten W einen Fehler E bezüglich der Netzausgaben und den erwarteten, korrekten Ausgaben. Mit einem Gradientenverfahren (Methode des steilsten Abstiegs) wird versucht möglichst schnell ein globales Minimum der Fehlerfunktion E zu finden. Für ein einzelnes Gewicht w_{ij} in der k-ten Schicht ergibt sich somit Gleichung {18}. η ist hierbei wieder die Lernrate. Als Fehlerfunktion E wird oft die Summe der quadratischen Abstände zwischen den Zielwerten t_{pj} und der Netzausgabe a_{pj} über alle Eingabemuster p herangezogen (Gleichung {19}). Mit der Kettenregel, Gleichung {20}, dem lokalen Fehler δ_j (Gleichung {21}) und unter Berücksichtigung von {22} folgt für die Veränderung des Gewichts w_{ij} Gleichung {23}. Dies gilt aber nur für die Neuronen in der Ausgabeschicht (z.B. bei einem Netz mit drei Schichten also für k=3). Für die versteckten Schichten muss der Fehler δ_j^k aus dem der

jeweils höheren Schicht berechnet werden (daher der Name des Verfahrens: Der Fehler wird durch das Netz zurückpropagiert). Diese Berechnung kann mit Gleichung {24} vorgenommen werden.

$$\Delta w^k{}_{ij} = -\eta \frac{\partial E}{\partial w^k{}_{ij}} \qquad \{18\}$$

$$E = \sum_p E_p \;\; mit \;\; E_p = \frac{1}{2}\sum_j \left(t_{pj} - a_{pj}\right)^2 = \frac{1}{2}\sum_j e_{pj}^2 \qquad \{19\}$$

$$\frac{\partial E_p}{\partial w^k{}_{ij}} = \frac{\partial E_p}{\partial n^k{}_j}\frac{\partial n^k{}_j}{\partial w^k{}_{ij}} \qquad \{20\}$$

$$\delta^k{}_j = -\frac{\partial E_p}{\partial n^k{}_j} = e_{pj}\frac{\partial f^k}{\partial n^k{}_j} \qquad \{21\}$$

$$n^k{}_j = \sum_i w^k{}_{ij} a_i^{k-1} \qquad \{22\}$$

$$\Delta w_{ij}^k = \eta \delta_j^k a_i^{k-1} \qquad \{23\}$$

$$\delta_j^k = \frac{\partial f^k}{\partial n_j^k}\sum_i \delta_i^{k+1} w_{ij}^{k+1} \qquad \{24\}$$

Ein typischer Backpropagation-Algorithmus umfasst im Wesentlichen folgende 5 Schritte:

1) Initialisieren der Gewichte w_{ij} mit Zufallswerten
2) Eingabemuster vorgeben und Netzausgabe berechnen
3) Mit Hilfe der gewünschten Ausgabe nacheinander die lokalen Fehler der Neuronen berechnen
4) Die Wichtungsparameter des Netzes mit Gleichung {23} anpassen
5) Mit dem nächsten Eingabemuster bei Schritt 2 fortfahren

Diese Vorgehensweise wird als *Online Learning* oder auch *Incremental Training* bezeichnet. Im Gegensatz dazu wird beim *Batch Training* die Änderung der Wichtungsparameter erst durchführt nachdem alle Eingabemuster durch das Netz geschickt wurden. Schritt 4 wird also zunächst ausgelassen, der Wert für Δw_{ij}^k des entsprechenden Eingabemusters aber gespeichert. Nachdem alle Δw_{ij}^k bekannt sind, wird das Verbindungsgewicht mit der Summe aller Δw_{ij}^k modifiziert. Eine detailliertere Darstellung zur Herleitung und Anwendung des Backpropagation-Algorithmus findet sich in der Literatur: [ALEKSANDER], [CICHOCKI], [HECHT-NIELSEN], [ZELL] und [HAYKIN].

Bei der praktischen Anwendung von Backpropagation-Algorithmen zum Training neuronaler Netze ergeben sich ein Reihe von Problemen. Das Verfahren hat, wie alle anderen Gradientenverfahren auch, das grundsätzliche Problem, dass nur die lokale Umgebung der Fehlerfläche bekannt ist. D. h. der Algorithmus kann in lokalen Minima hängen bleiben, die im Gegensatz zu dem globalen Minimum natürlich nur suboptimale Lösungen darstellen. Da die Fehlerfläche, besonders bei großen Netzen mit vielen freien Parametern, stark zerklüftet sein kann und diese auf komplizierte Art und Weise von der Struktur und der Vorverarbeitung der Eingabedaten und Zielwerte abhängt, gibt es für dieses Problem keine allgemeingültigen Lösungsansätze. Weitere Probleme bei der Anwendung von Backpropagation-Algorithmen sind flache Plateaus, Oszillationen in steilen Schluchten und das Verlassen guter Minima. Hat die Fehlerfunktion ausgedehnte Bereiche mit sehr geringem „Gefälle“, oder anders ausgedrückt, Bereiche in denen sehr geringe Gradienten auftreten, sind die Gewichtsänderungen pro Iterationsschritt so gering, dass der Algorithmus unter Umständen auf flachen Plateaus stagniert. Das liegt daran, dass die Gewichtsänderung proportional zum Gradienten vorgenommen wird (siehe Gleichung {18}). Zur Lösung dieses Problems ist eine Modifikation des Backpropagation-Verfahrens entwickelt worden, in dem in die Gleichung zur Gewichtsänderung ein sogenannter Momentum-Term eingeführt wird. Dadurch wird die im vorangegangenen Iterationsschritt vorgenommene Änderung über einen Faktor α mit berücksichtigt (Gleichung {25}).

$$\Delta w^k{}_{ij}(t+1) = -\eta \delta^k_j a^{k-1}_i + \alpha \Delta w^k{}_{ij}(t) \qquad \{25\}$$

Durch diese Modifikation des Backpropagation-Algorithmus können flache Plateaus schneller überwunden werden und Oszillationen in steilen Schluchten können gedämpft oder ganz vermieden werden. Unglücklicherweise wird das dritte erwähnte Problem, das Verlassen guter Minima, durch diese Modifikation verstärkt. Dabei hängt es allerdings sehr stark von der konkreten Aufgabenstellung ab, ob sich dieser letztgenannte Punkt wirklich stark negativ auf das Training auswirkt. Überaus häufig hat hingegen die Wahl der Lernrate η einen entscheidenden Einfluss auf das Training. Wird sie sehr groß gewählt, treten sehr große Sprünge auf der Fehlerfläche auf und enge Täler können nicht gefunden werden oder es kann zu Oszillationen kommen. Zu kleine Werte führen dazu, dass der Zeitaufwand des Training sehr stark ansteigt. Bei Zell wird darauf hingewiesen, dass es leider keine allgemeingültigen Leitlinien zur geeigneten Wahl der Lernrate gibt [ZELL]. Die Wahl einer optimalen Lernrate hängt von der Problemstellung, den Trainingsdaten und auch von der Größe und Topologie des Netzes ab.

Neben der Verbesserung des Backpropagation-Verfahrens durch Einführen des Momentum-Terms wurden noch eine Reihe von ähnlichen Algorithmen entwickelt, die das Training schneller und effizienter machen sollen. Einer davon ist der Levenberg-Marquardt-Algorithmus der auf dem Newton-Verfahren basiert. Dort wird angenommen, dass die Fehlerfunktion E lokal durch eine quadratische Funktion angenähert werden kann. Dies geschieht mit einer Taylor-Reihe. Diese quadratische Funktion wird dann exakt minimiert. Von diesem Ausgangspunkt lässt sich Gleichung {26} für die Anpassung der Wichtungsparameter ableiten [CICHOKI], [DOKUNNT]. Hierbei ist w ein Vektor der alle Parameter des Netzes (inklusive evtl. vorhandene bias-Werte) zusammenfasst und H ist die Hessematrix, die die partiellen, zweiten Ableitungen der Fehlerfunktion E enthält. Diese wird im Levenberg-Marquardt-Verfahren durch Verwendung der Jacobimatrix J angenähert, welche durch Backpropagation berechnet werden kann. Dies ist wesentlich weniger aufwendig als die Berechnung der Hessematrix. μ ist ein positiver Faktor, I die Einheitsmatrix, e ein Vektor, der die Fehler des Netzes enthält und t eine Iterationslaufzahl.

$$w(t+1) = w(t) - (H + \mu I)^{-1} J^T e$$
$$\approx w(t) - (J^T J + \mu I)^{-1} J^T e \qquad \{26\}$$

Ist der Faktor μ gleich 0 geht das Levenberg-Marquardt-Verfahren in das Newton-Verfahren über (mit dem Unterschied, dass für die Hessematrix eine Näherungslösung verwendet wird). Bei großen μ verhält sich das Verfahren wie ein Gradientenabstiegsverfahren mit kleiner Schrittweite. Nach einem erfolgreichen Schritt (bei dem der Fehler geringer geworden ist) wird μ verkleinert, sonst erhöht. Das führt dazu, dass in der Nähe des Minimums das Levenberg-Marquardt-Verfahren in das Newton-Verfahren übergeht ($\mu \to 0$), welches in der Nähe eines Minimums schneller konvergiert. Durch Anwendung dieses Verfahrens können Netze schnell und effizient trainiert werden.

$$E = \gamma \frac{1}{N} \sum_{i=1}^{N} e_i^2 + (1-\gamma) \frac{1}{n} \sum_{j=1}^{n} w_j^2 \qquad \{27\}$$

Ein weiteres Problem beim Training neuronaler Netze ist das Overfitting. Damit ist gemeint, dass das Netz während des Trainingsvorgang die Assoziation zwischen präsentierten Eingabe- und Ausgabemuster perfekt erlernt, bei neuen Eingabemustern aber versagt und stark fehlerhafte Ausgabemuster ausgibt. Die Fähigkeit des Netzes zu *Generalisieren* ist also nur sehr eingeschränkt vorhanden. Ein Ansatz schon während des Trainingsvorgangs darauf hinzuwirken, dass Overfitting möglichst nicht auftritt, ist ein *Regularisierung* genanntes Verfahren. Dabei wird eine modifizierte Fehlerfunktion verwendet (siehe Gleichung {27}). Hier werden im Gegensatz zu Gleichung {19}, bei der nur die quadratischen Fehler der Netzausgaben eingehen, auch die quadratischen Wichtungsfaktoren des Netzes berücksichtigt. Wie stark dies geschieht, wird durch den Faktor γ, die sogenannte *performance ratio*, festgelegt. Mit dieser Fehlerfunktion entstehen Netze, die kleinere Verbindungsgewichte und somit „weichere“ Ausgaben haben. Diese Netze neigen mit geringerer Wahrscheinlichkeit zum *Overfitting*.Eine detailliertere Beschreibung zu Regularisierungsverfahren sowie deren Implementation ist der Literatur zu entnehmen [DOKUNNT], [CICHOCKI]. Auch dieser Ansatz beseitigt aber nicht das schon angesprochene generelle Problem bei Gradientenabstiegsverfahren,

dass sie in lokalen Minima hängen bleiben können. Die Fehlerfläche neuronaler Netze ist oft sehr stark zerklüftet und weist u. U. viele schmale Täler auf, sodass die Wahrscheinlichkeit, dass der Trainingsalgorithmus nur suboptimale Netzparameter findet, relativ hoch ist. Dieser Nachteil der bisher angesprochenen Trainingsverfahren lässt sich nur durch den Einsatz stochastischer Methoden, wie z.B. einem genetischen Algorithmus (siehe Kapitel 2.5.2), beheben. Nach der von Zell vertretenden Meinung sind aber genetische Algorithmen bei zunehmender Netzgröße den anderen Trainingsmethoden unterlegen, da der Zeitaufwand bis zur Ermittlung verwendbarer Parameterkombinationen sehr hoch ist [ZELL].

2.5 Optimierungsverfahren

2.5.1 Simplex-Algorithmus

Das Simplex-Verfahren, das auf Nelder und Mead [NELDER] zurückgeht, dient zur Auffindung des Optimums (meistens des Minimums) einer Funktion bei der die Parameter voneinander abhängig sind. Unter einem Simplex versteht man ein N-dimensionales geometrisches Gebilde mit N+1 Ecken, wobei N die Anzahl der Parameter ist. Bei einem Optimierungsproblem mit einem unbekannten Parameter ist der Simplex eine Linie, bei zwei und drei Parametern ein Dreieck bzw. ein Tetraeder und bei mehr als drei Parametern spricht man von einem Hypertetraeder.

Im Folgenden wird von einem Minimierungsproblem mit 2 unbekannten Parametern ausgegangen, da sich so die anschaulichste Darstellung des Simplex-Verfahrens ergibt. Ausgehend von einem Startpunkt auf der Fläche der zu minimierenden Funktion, der vorgegeben werden muss, wird ein Startsimplex erstellt. Dazu werden zwei weitere Punkte generiert, indem jeweils einer der Parameter um einen definierten Betrag variiert wird. An diesen drei Punkten des Startsimplexes wird der Funktionswert berechnet. Nun werden die Ergebnisse bewertet: der beste Punkt mit dem niedrigsten Funktionswert (**b** in Grafik Abbildung 9), liegt dem Minimum am nächsten, **s** ist der schlechteste Punkt mit dem höchsten Funktionswert und **n** liegt zwischen diesen beiden.

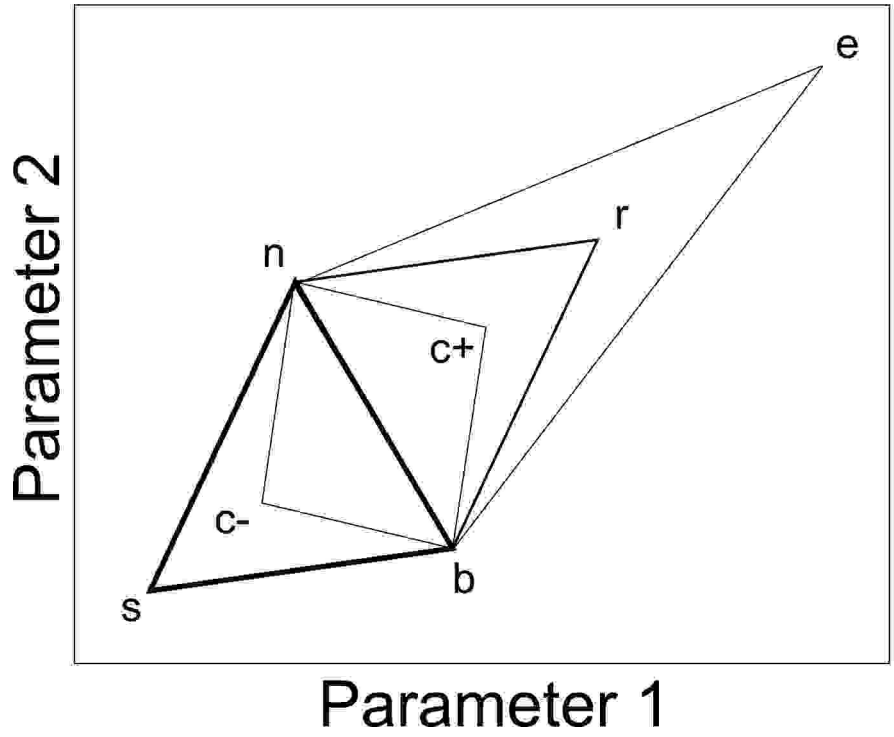

Abbildung 9. Simplexverfahren mit variabler Schrittweite

Nun wird der schlechteste Punkt (**s**) am Schwerpunkt des Simplex (**p**) gespiegelt. Für einen N-dimensionalen Simplex gilt allgemein :

$$p = \frac{1}{N}\sum_{i \neq j}^{N} v_i \qquad \{28\}$$

$$r = p + (p - v_j) \qquad \{29\}$$

Hierbei sind v_i Vektoren, die die Parameter enthalten und somit die Punkte beschreiben, und j der Index des schlechtesten Vektors (bzw. Punktes). In Abbildung 9 liegt **p** auf der Mitte der Strecke $\overline{nb}$ und Spiegelung von **s** führt zum Punkt **r**. Nun werden die Punkte **n**,**b** und **r** bewertet und der nun schlechteste am neu berechneten Schwerpunkt gespiegelt. In einem Simplexverfahren mit konstanter Schrittweite wird dieser Schritt solange wiederholt bis ein vorgegebenes Abbruchkriterium zutrifft. Die Nachteile einer konstanten Schrittweite liegen darin, dass bei zu hoher Schrittweite das Optimum verfehlt werden kann und bei zu kleiner sehr viele Iterationsschritte erforderlich sind, um das Optimum zu finden.

Abhilfe schafft das Simplexverfahren mit variabler Schrittweite. Hier schliesst sich an jede Reflexion noch eine Bewertung des neuen Punktes **r** an und führt ggf. zu einer

Expansion oder Kontraktion des Simplex. Ist **r** besser als der beste, alte Punkt **b**, wird der Simplex um einen vorgegebenen Faktor a expandiert und es ergibt sich der Punkt **e** :

$$e = p + a(p - v_j) \quad \{30\}$$

Der Simplex wird auf den Punkt $\mathbf{c}_+$ kontrahiert, wenn der Funktionswert von **r** zwischen dem von **n** und **s** liegt, oder auf den Punkt $\mathbf{c}_-$, wenn **r** noch schlechter ist als **s**. b ist wie a eine vorgegebene Konstante.

$$c_+ = p + b(p - v_j) \quad \{31\}$$

$$c_- = p - b(p - v_j) \quad \{32\}$$

Für den Fall, dass der Funktionswert von **r** zwischen denen von **n** und **b** liegt, wird weder Expandiert noch Kontrahiert und der Simplex **bnr** wird unverändert für die nächste Reflexion verwendet.

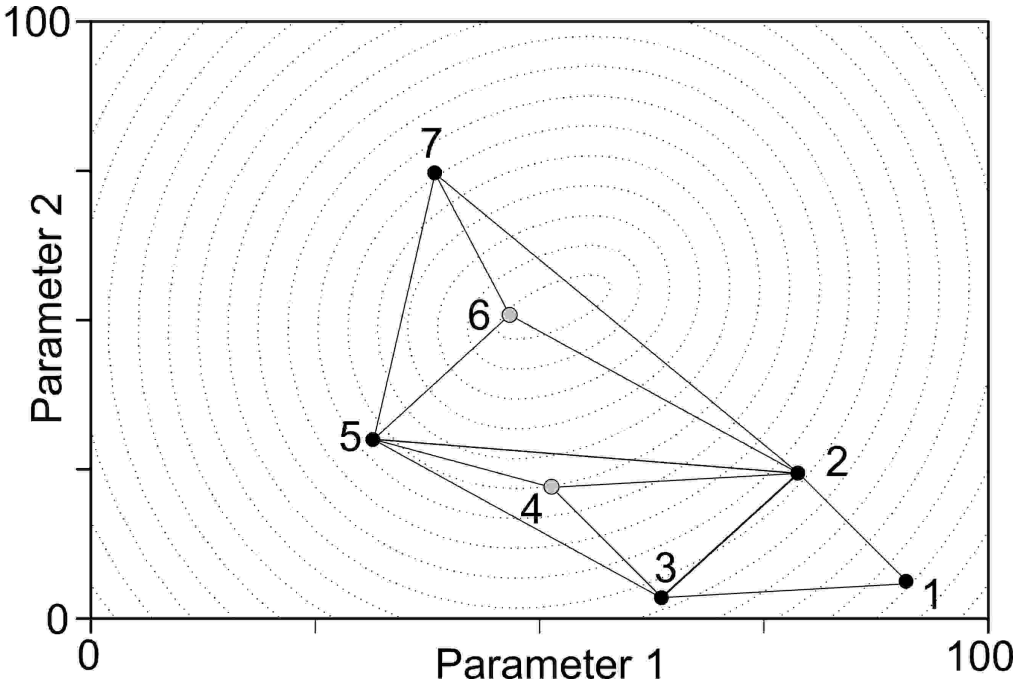

Abbildung 10. Simplexverfahren mit variabler Schrittweite – die Ellipsen stellen von innen nach außen steigende Funktionswerte dar, das Optimum (hier Minimum) liegt innerhalb der innersten Ellipse

Das bedeutet, dass sich der Simplex durch Expandieren schnell auf das Optimum zubewegen kann, es aber auch nicht verfehlt, da er in der Nähe des Optimums kontrahieren kann. Abbildung 10 verdeutlicht dies: Ausgehend vom Startsimplex 123 wird Punkt 4 durch Spiegelung von 1 (dem schlechtesten Punkt) erhalten. Da 4 sowohl

besser als 2 als auch als 3 ist, wird der Simplex auf Punkt 5 expandiert. Der neue schlechteste Punkt ist nun 3. Im nächsten Schritt wird er auf Punkt 6 gespiegelt und da dieser besser als alle anderen ist, wird auf Punkt 7 expandiert. Das Optimum liegt nun innerhalb des von den Punkten 257 gebildeten Simplex. In den nächsten Iterationsschritten (nicht mehr gezeigt) wird der Simplex kontrahieren und den Ort des Optimums immer mehr eingrenzen.

Die Optimierung ist beendet, wenn ein vorgegebenes Abbruchkriterium zutrifft: Anzahl an Iterationsschritten ist erreicht, Unterschreitung einer minimalen Schrittweite oder Erreichen eines bestimmten Funktionswertes.

Ein generelles Problem des Simplexverfahrens (bei konstanter und auch bei variabler Schrittweite) tritt auf, wenn die zu optimierende Funktion neben einem globalen Minimum (Maximum) noch ein oder mehrere lokale Minima (Maxima) aufweist. In solch einem Fall ist es nur von dem gewählten Startpunkt abhängig welches Optimum gefunden wird, da der Simplex immer zum nächstliegenden Optimum konvergiert. Weitere Ausführungen zum Simplex-Verfahren finden sich in der Literatur [OTTO], [DANZER].

2.5.2 Genetischer Algorithmus

Ein genetischer Algorithmus (GA) ist ein globales Suchverfahren, in dem die Prinzipien der Evolution, Vererbung, Mutation und Selektion, implementiert sind. Bei den Lebewesen oder Individuen, die den Evolutionsprozess durchlaufen, handelt es sich hier um Parameterkombinationen der zu optimierenden Funktion. Das genetische Material des Individuums, die Chromosomen, sind (meist binär) kodierte Parameter. Durch Selektion pflanzen sich diejenigen Parameterkombinationen d.h. Individuen mit höherer Wahrscheinlichkeit fort, die näher am gewünschten Ergebnis liegen, als diejenigen, die davon weit entfernt liegen. Durch Crossover und Mutation liegt einem GA auch eine gewisse Zufälligkeit zugrunde. Dadurch wird verhindert, dass sich die Gesamtheit der Individuen, die Population, nur auf ein lokales Minimum zu bewegt und dort verharrt. Das Abbruchkriterium eines GA ist oft eine festgelegte Anzahl an Fortpflanzungsschritten (Generationen). Der Aufbau eines GA ist aus Abbildung 11 ersichtlich.

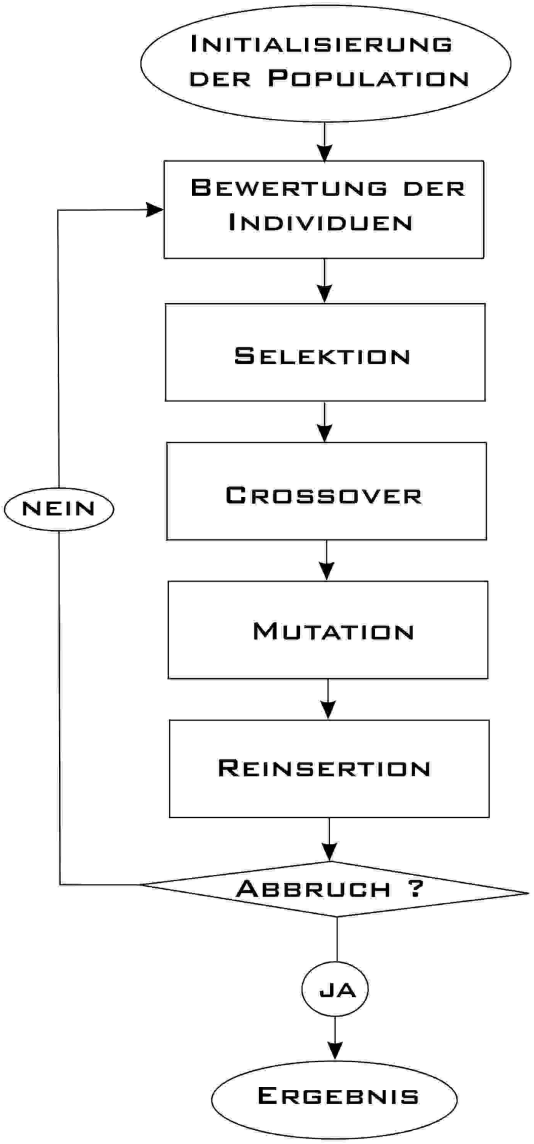

Abbildung 11. Schematischer Ablauf eines GA

Initialisierung

Vor dem Beginn der Optimierung wird der Chromosomensatz der Population erzeugt. Dazu werden für jeden unbekannten Parameter der zu optimierenden Funktion F Grenzen festgelegt, innerhalb derer sich der Parameter bewegen darf. Für jedes Individuum der Population wird nun per Zufallszahl ein Parametersatz generiert, wobei die Werte innerhalb der entsprechenden Grenze liegen. Im nächsten Schritt werden alle generierten Werte aller Individuen kodiert. Das gängigste Kodierungsverfahren ist die binäre Kodierung. Hierbei wird jeder Parameter durch eine Abfolge aus 0 und 1

dargestellt. Bei drei Parametern im Bereich von 0 bis 255 könnte ein generiertes Individuum somit folgende Chromosomen besitzen :

00101001 01000111 11001110

Andere mögliche Kodierungen sind Integer- und Real-Kodierungen. Abhängig von F und dem Wertebereich der Parameter lässt sich durch eines dieser Kodierungsverfahren der Speicherbedarf eines GA senken oder die Rechengeschwindigkeit erhöhen. Im Folgenden wird im Wesentlichen auf binäre Kodierungsverfahren eingegangen.

Bewertung der Individuen

Die Chromosomen jedes Individuums werden decodiert, um die Parametersätze zu erhalten. Diese werden auf F angewendet und man erhält pro Parametersatz, sprich Individuum, einen Funktionswert aus F. Jetzt werden die Individuen hinsichtlich ihrer Güte („*fitness*“) bewertet. Für den Fall, dass das Minimum von F gesucht ist, ist das beste Individuum dasjenige, dass dem kleinsten Funktionswert zuzuordnen ist und das schlechteste das mit dem größten usw. Oft wird, nachdem die Population nach ihrer Güte sortiert wurde, der Funktionswert von F (das „Gütekriterium“) in einen relativen Fitnesswert umgerechnet. Das bedeutet z. B., dass das schlechteste Individuum einem Wert von 0 zugeordnet wird und das beste einem Wert von 1.

Selektion

Durch Selektion pflanzen sich die guten Individuen (Fitnesswert nahe an 1) mit höherer Wahrscheinlichkeit fort als die schlechten (Fitnesswert bei 0). Um die natürliche Selektion in einem GA nachzuempfinden gibt es eine Vielzahl von Rechenverfahren. Das bekannteste, das Roulette-Verfahren, wird hier vorgestellt. Die Population wird als Kreis dargestellt und die einzelnen Individuen werden Kreissegmenten zugeordnet, deren Größe von der Güte des Individuums abhängt. Für eine Population mit 6 Individuen, wobei eines besonders gut (2) und eines besonders schlecht (6) ist, ergibt sich z.B. ein Bild wie in Abbildung 12.

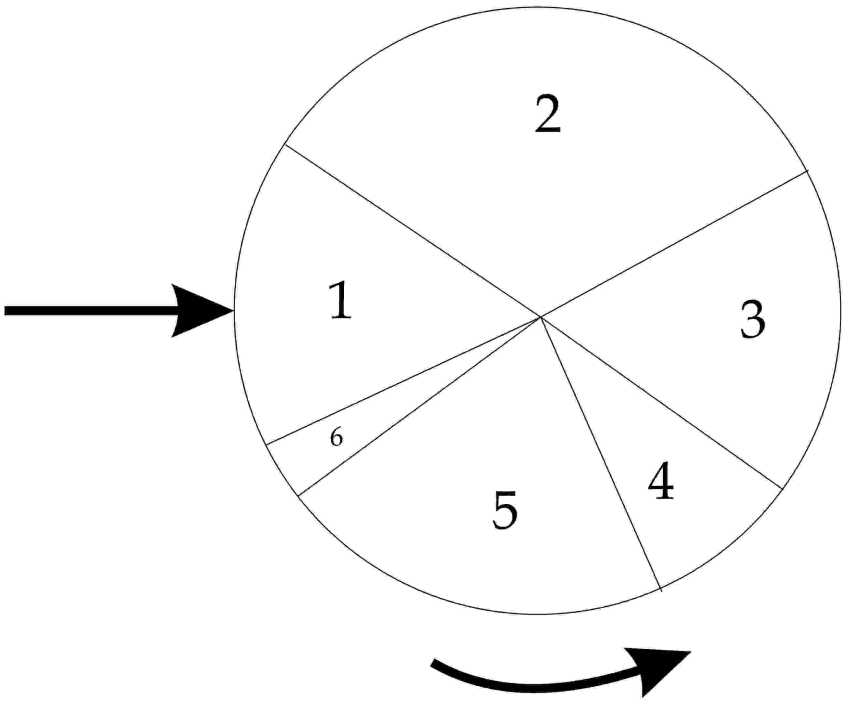

Abbildung 12. Roulette-Selektion

Der Umfang des Kreises entspricht der Gesamtfitness der Population. Mittels einer Zufallszahl aus dem Intervall 0 bis Gesamtfitness wird das Individuum ausgewählt, das sich fortpflanzt. Nach mehrmaliger Selektion je eines Individuums ergibt sich die Gesamtheit der Individuen, die sich fortpflanzen werden, wobei einige Individuen dort mehrfach, andere nicht vertreten sein können. Das bedeutet, dass mit sehr geringer Wahrscheinlichkeit das schlechteste Individuum (6 in Abbildung 12) per Zufall jedes Mal ausgewählt wird und sich alle anderen (und besseren) Individuen überhaupt nicht fortpflanzen. Bei einer Variante des Roulette-Verfahrens wird jedes Mal nach erfolgter Selektion eines Individuums das entsprechende Kreissegment verkleinert, sodass oben genannter Fall noch unwahrscheinlicher wird und die Güte eines Individuums noch stärker in den Selektionsprozess eingeht.

Crossover

Aus den zur Fortpflanzung ausgewählten Individuen werden Paare gebildet und per Zufall ermittelt, bei wie vielen und bei welchen Paaren Crossover stattfindet. Beim Simple-Point-Crossover wird bei jedem dieser Paare per Zufall eine Stelle in den Chromosomen ausgewählt und die Bits rechts davon mit dem Partner getauscht :

Crossover-Punkt ▼
Individuum 1 0 0 1 1 1 0 1 1
Individuum 2 1 0 0 1 0 0 1 0
Nachkomme 1 0 0 1 1 1 0 1 0
Nachkomme 2 1 0 0 1 0 0 1 1

Andere Crossover-Verfahren sind Multi-Point- und Uniform-Crossover bei denen die Chromosomen an mehreren Stellen zerschnitten und die Bits getauscht werden. Wurden die Chromosomen nicht binär kodiert findet das Crossover auch statt, ist aber weniger anschaulich und wird hier nicht dargestellt.

Mutation

Außer durch Crossover entstehen in der Natur auch durch Mutation neue Chromosomen. Auch dieser Vorgang kann in einen GA mit eingebunden werden. Es werden zufällig Individuen ausgewählt, bei denen Mutation auftritt. Bei jedem wird nun wiederum per Zufall eines der Bits invertiert :

Mutation an Bit 5
vorher 1 0 0 1 0 0 1 1
nachher 1 0 0 1 1 0 1 1

Reinsertion

Nachdem die Nachkommen der alten Generation aus Selektion, Crossover und Mutation hervorgegangen sind, werden sie in die alte Generation reinsertiert. Hierfür gibt es eine Vielzahl von gängigen Verfahrensweisen.

Die Nachkommen können die alte Generation komplett ersetzen ohne die Güte der Nachkommen dabei zu berücksichtigen. Es kann aber auch von Vorteil sein, erst die Güte der Nachkommen zu bewerten und die schlechtesten Individuen der alten Generation mit den besten Nachkommen zu ersetzen – man spricht von *elitist strategy*. Eine weitere Möglichkeit besteht darin, die ältesten Individuen durch die besten Nachkommen zu ersetzen.

Abbruch des GA

Am Häufigsten wird der Optimierungsvorgang nach einer bestimmten Anzahl von Generationen gestoppt. Ein Konstantbleiben der durchschnittlichen Fitness der Population als Abbruchkriterium zu verwenden kann problematisch sein, da dies recht

häufig vorkommt und es alleine dadurch nicht auszuschließen ist, dass noch ein stark überlegenes Individuum, d.h. eine viel bessere Lösung, gefunden wird.

Ein GA sucht im Gegensatz z.B. zum Simplex-Algorithmus von mehreren Startpunkten aus gleichzeitig nach der optimalen Lösung. Durch den Selektionsvorgang entwickelt sich die Population in Richtung auf die beste, bisher bekannte Lösung zu. Ein frühzeitiges Konvergieren des Verfahrens (wohlmöglich in einem unerwünschten lokalen Optimum) wird durch die zufälligen Prozesse Crossover und Mutation verhindert. Durch diese entstehen immer wieder Individuen, die sich in ganz anderen Bereichen des Lösungsraumes befinden als der Grossteil der Population. Die Chance das globale Optimum zu finden erhöht sich durch diese Fähigkeit des GA praktisch den gesamten Lösungsraum zu durchkämmen sehr stark. Bei sehr komplizierten Problemstellungen und/oder sehr sensitiven Parametern stößt allerdings auch der GA an seine Grenzen, sodass es nicht immer gelingt in einer angemessenen Rechenzeit ein befriedigendes Ergebnis zu erhalten.
Weitere Informationen über den GA finden sich in der Literatur [OTTO], [DANZER], [GATUTORIAL], [GOLDBERG].

2.6 Digitale Bildverarbeitung

2.6.1 Filter, Kantenerkennung und Konvolution

Bei der digitalen Bildverarbeitung geht es u. A. darum aus Bildern durch Anwendung geeigneter Filter und Operatoren Merkmale zu extrahieren und somit die Information, die in einem Bild verborgen ist einer weiteren Bearbeitung und Auswertung zugänglich zu machen. Weitere Ziele der Bildverarbeitung können z. B. Rauschunterdrückung und Kontrastverbesserung sein. Die Operationen, die auf das Originalbild angewendet werden, lassen sich in drei Klassen einteilen: Punktoperatoren, lokale und globale Operatoren. Bei den Punktoperatoren wird zur Berechnung des Farbwertes eines Pixels im Ergebnisbild nur der Farbwert des entsprechenden Pixels im Originalbild benötigt. Beispiele für Punktoperatoren sind Helligkeitsveränderungen, bei der jeder Farbwert im Originalbild mit einer Konstante multipliziert und somit entweder verkleinert oder vergrößert wird, oder auch Kontrastveränderungen. Lokale Operatoren benutzen zur Berechnung eines Ergebnispixels auch die Pixel der näheren Umgebung (z. B.

Gaußscher Glättungsfilter, Sobel-Operator, etc.) während globale Operatoren alle Pixel des Originalbildes verwenden (z. B. Fourier-Transformation).

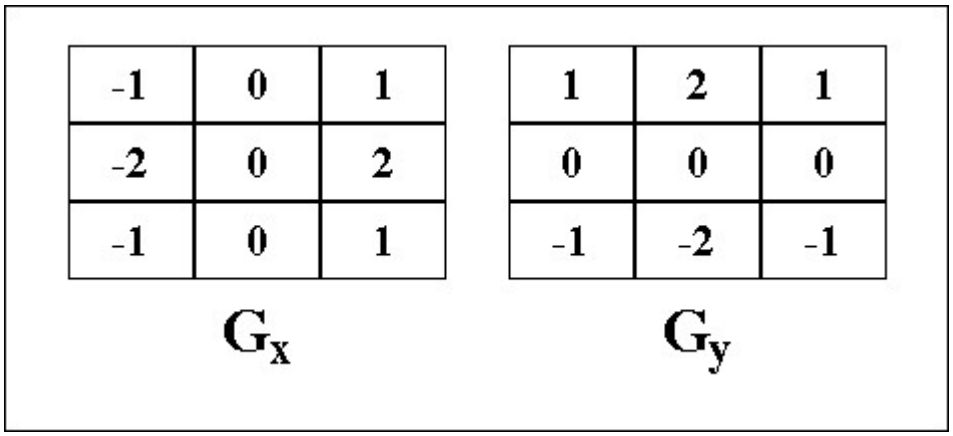

Abbildung 13. Sobel-Konvolutionskerne für x- und y-Gradienten

$$G_x(x,y) = -g(x-1,y-1) - 2g(x-1,y) - g(x-1,y+1) + g(x+1,y-1) + 2g(x+1,y) + g(x+1,y+1) \quad \{33\}$$

$$G_y(x,y) = +g(x-1,y-1) + 2g(x,y-1) + g(x+1,y+1) - g(x-1,y+1) - 2g(x,y+1) - g(x+1,y+1) \quad \{34\}$$

$$|G(x,y)| = \sqrt{G_x^2(x,y) + G_y^2(x,y)} \quad \{35\}$$

$$|G(x,y)| = |G_x(x,y)| + |G_y(x,y)| \quad \{36\}$$

$$\theta(x,y) = \arctan\left(G_y(x,y) / G_x(x,y)\right) \quad \{37\}$$

Eine häufig auftretende Aufgabenstellung bei der digitalen Bildverarbeitung ist die Erkennung von Kanten. Kanten sind Stellen, an denen im Bild große Kontrastunterschiede vorliegen. Fasst man ein Graustufenbild als Funktion g(x,y) auf,

sind Kanten Wendepunkte der Funktion g. Um diese zu finden kann der Sobel-Operator eingesetzt werden. Die dabei eingesetzten Konvolutionskerne sind in Abbildung 13 gezeigt. Diese beiden Matrizen, werden auf jeden Pixel des Originalbildes angewandt und aus den Pixeln in der Umgebung des betreffenden Pixels werden die Gradienten der Grauwerte in x- und y-Richtung bestimmt. Die in den Konvolutionskernen enthaltenen Werte sind dabei die Multiplikatoren für den Grauwert des Pixels an der entsprechenden Stelle relativ zur Position des Zielpixels (siehe Gleichungen {33} und {34}). Die Gradienten für Pixel am Rand des Bildes können nicht berechnet werden, da für sie nicht alle Pixel in der Umgebung vorhanden sind, und werden meist auf Null gesetzt. Die Berechnung des Gesamtgradienten geschieht mit Gleichung {35}, oft wird aber zwecks Rechenzeitersparnis die Näherungslösung aus Gleichung {36} angewendet. Zusätzlich dazu ist die Richtung des Gradienten an der Stelle x,y mit Hilfe des Arcustangens (Gleichung {37}) zugänglich. $\theta = 0$ bedeutet dabei, dass der Kontrastunterschied von schwarz auf weiß von links nach rechts auf dem Bild verläuft. Die Winkel sind dann von dieser Position aus gegen den Uhrzeigersinn angegeben.

Durch Anwendung des Sobel-Operators wird die erste Ableitung der Grauwerte mit einer gleichzeitigen Glättung berechnet. Stellt man den richtungsunabhängigen Gradienten G in einem neuen Bild dar, wobei hohe Werte von G weiß und niedrige Werte schwarz dargestellt sind, ergibt sich ein Bild, in dem die Kanten des Originalbildes als weiße Linien erscheinen. Dies ist in Abbildung 14 an einem Beispielbild dargestellt. Maximale Sensitivität zeigt der Sobel-Operator bei Kanten, die horizontal oder vertikal im Bild verlaufen. An die Kantenerkennung mit dem Sobel-Operator schließt sich oft die Anwendung eines Schwellenwertfilters an, mit den kleine Gradienten, die oft nur auf Rauschen im Originalbild zurückzuführen sind, herausgefiltert werden. Weitere weit verbreitete Operatoren zur Auffindung von Kanten sind das Roberts-Cross oder der Canny-Operator. Weitergehende Informationen zu diesen Operatoren und zum Sobel-Operator findet sich in der Literatur [GESTEWITZ], [WEB1] und [WEB2].

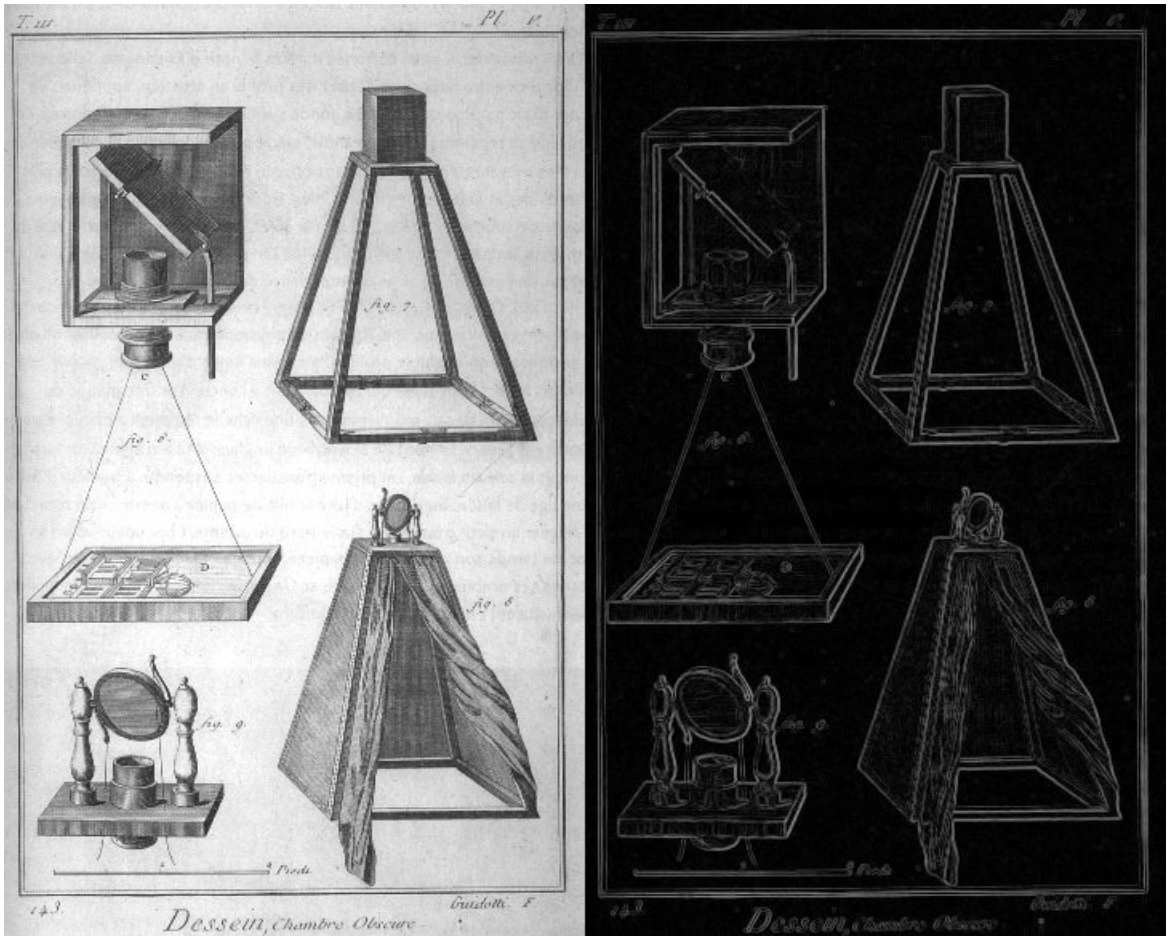

Abbildung 14. Links Originalbild, rechts Ergebnis nach Anwendung des Sobel-Operators. Die Kanten im Originalbild erscheinen im Ergebnisbild als weiße Linien.

2.6.2 Objekterkennung durch Kantenverfolgung

Um ein Objekt vom Hintergrund eines Bildes zu trennen, ist ein eindeutiges Unterscheidungskriterium nötig. Dies kann in einem einfachen Fall z. B. der Helligkeitsunterschied zwischen Objekt und Hintergrund sein. Bei Bildern bei denen eine Unterscheidung aufgrund eines solch leicht zugänglichen Merkmals[3] nicht möglich ist, geht oft die Anwendung eines Kantenerkennungsoperators (siehe Kapitel 2.6.1) oder eines ähnlichen Vorverarbeitungsschrittes voraus. Zur Darstellung der Methode der Kantenverfolgung wird hier davon ausgegangen, dass das zu verarbeitende Bild ein

[3] Zum Beispiel in einem Graustufenbild durch Vergleich der Grauwerte von benachbarten Bildpunkten oder durch Thresholding.

Graustufenbild mit weißen Objekten auf schwarzem Grund ist. Das Bild wird zuerst nach Stellen durchsucht, in denen ein weißer Bildpunkt einen schwarzen Nachbarn hat. Ist eine solche Stelle gefunden lauten die Regeln, um nun das Objekt mit einer geschlossenen Linie zu umranden:

- Drehung nach links, wenn das aktuelle Pixel weiß ist.
- Drehung nach rechts, wenn das aktuelle Pixel schwarz ist.
- Nach drei Drehungen in die gleiche Richtung die beiden ersten Regeln ignorieren und in die andere Richtung drehen.

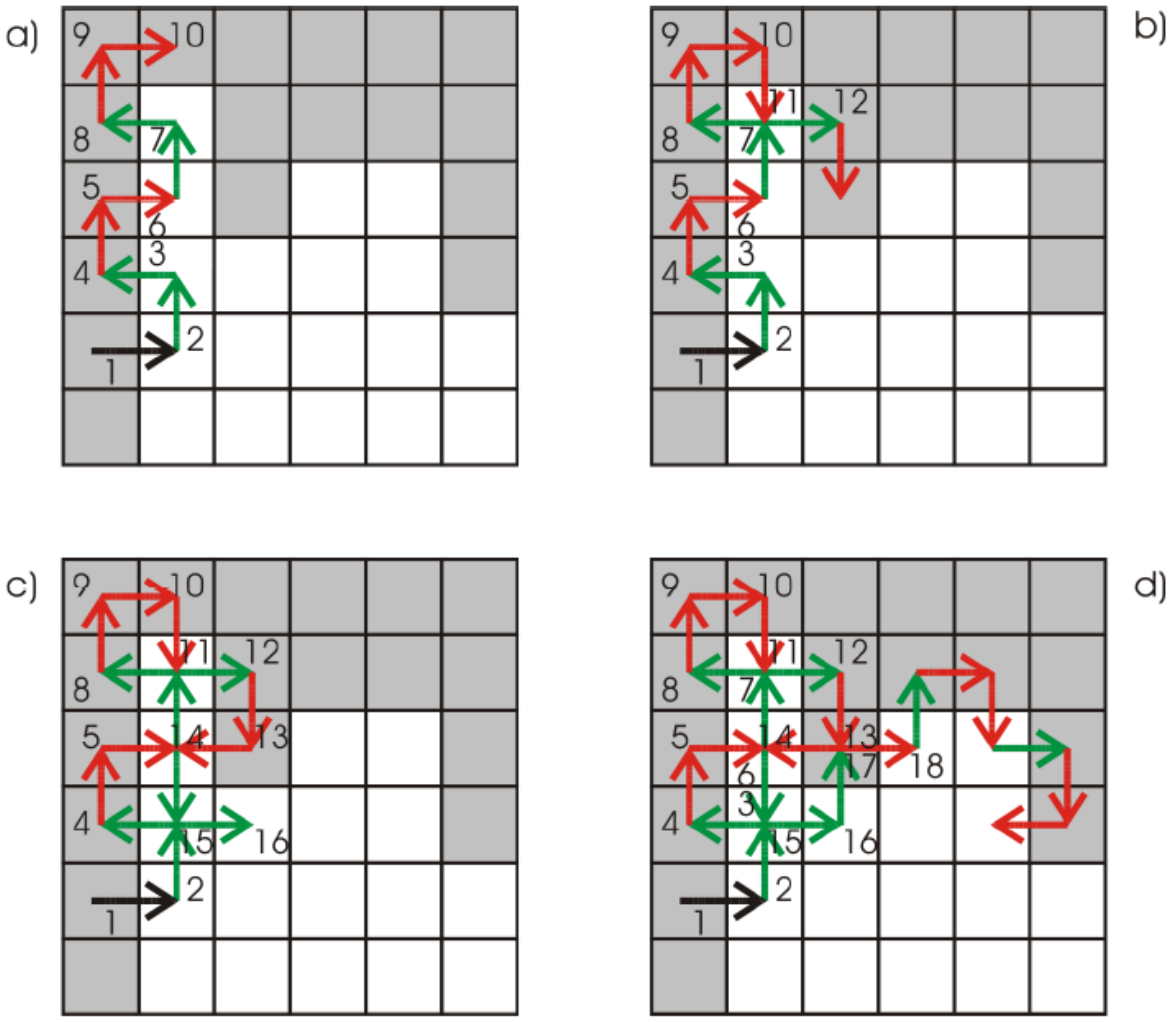

Abbildung 15. Prinzip der Objekterkennung durch Kantenverfolgung. Erläuterungen siehe Text. Quelle: [KRABICHLER]

Die Auswirkungen dieser Regeln sind in Abbildung 15 an einem Beispielobjekt dargestellt. Der schwarze Pfeil bezeichnet dabei die Startposition, an der der Übergang von schwarz nach weiß, d.h. das Vorhandensein eines Objektes, gefunden wurde. Der nächste Bildpunkt ist weiß, daher erfolgt eine Drehung nach links (Pfeil 2). Daran anschließend erfolgt noch eine Drehung nach links (Pfeil 3), dann auf dem schwarzen Bildpunkt eine Drehung nach rechts (Pfeil 4) usw. Der Algorithmus bricht ab, wenn der Ausgangspunkt wieder erreicht wurde. Das Objekt wurde mit einer durchgehenden Linie umrundet und ist nun vom Rest des Bildes separierbar und einer weiteren Verarbeitung bzw. Auswertung zugänglich.

3 Optimale Parameterbestimmung bei einem enzymatischen Prozess durch Analyse der Fisher- Informations-Matrix

Die Kenntnis der Kinetik enzymatischer Reaktionen ist von hoher Bedeutung in der Biochemie, der Biotechnologie und in der Medizin. Sie ist wichtig zum Verständnis intrazellulärer Abläufe und beim Auffinden wirkaktiver Substanzen. Aus diesem Grund ist es wichtig die kinetischen Parameter möglichst genau und mit geringem Zeit- und Ressourcenaufwand bestimmen zu können. Die genaue Planung von Experimenten (*Experimental Design*), die in optimaler Weise zur Bestimmung der kinetischen Parameter führen sollen, ist daher von Vorteil [ATKINSON].

Endrenyi und Chan berechneten ein optimales Design für Experimente zur Bestimmung der Parameter der Michaelis-Menten Kinetik durch Messung der Reaktionsgeschwindigkeit bei verschiedenen Substratkonzentrationen [ENDRENYI]. Zum Auffinden optimaler Messzeitpunkte bei konstanter Messfehlerkovarianz benutzten sind das D-Kriterium (Maximierung der Determinante der Fisher-Informations-Matrix, FIM). Als optimaler Fall ergab sich, dass die Hälfte der Messungen bei Vorliegen der maximal möglichen Substratkonzentration c_{max} erfolgen sollen und die andere Hälfte bei

$$c = \frac{K_m c_{\max}}{2K_m + c_{\max}}.$$

Duggleby und Clarke wendeten Experimental Design an, um die optimale Substratanfangskonzentration sowie das optimale Messintervall zu ermitteln [DUGGLEBY]. Sie analysierten dafür die integrierte Form der Michaelis-Menten Differentialgleichung. Das verwendete Optimierungskriterium ist die Minimierung der Varianz der Parameterschätzung von K_m. In einem Review über Experimental Design-Ansätze zur Bestimmung der Parameter bei Enzymkinetiken von Murphy wird nur von Studien an enzymatischen Batch-Prozessen berichtet [MURPHY]. Zugaben von Substrat oder Enzymlösung während des Prozesses wurden dabei nicht berücksichtigt. Die Tatsache, dass die Änderung der Prozessführung von Batch nach Fed-Batch eine genauere Parameterbestimmung ermöglichen kann, wurde z. B. von Veloso gezeigt [VELOSO]. Bei einem *Escherichia-coli*-Modell wurde durch Experimental-Design-

Rechnungen ein optimales Feed-Profil zur Bestimmung des Ausbeutekoeffizients berechnet. Die Optimierung geschah über die Maximierung der Determinante der FIM mittels eines genetischen Algorithmus.

In der Literatur sind bisher keine Untersuchungen zu Experimental Design an enzymatischen Fed-Batch-Prozessen, die mit einem dynamischen Modell beschrieben werden, zu finden. Daher wurde im Rahmen dieser Arbeit eine entsprechende Studie durchgeführt.

3.1 Berechnung der FIM bei einem enzymatischen Prozess

Bei dem enzymatischen Prozess wird ein Substrat S betrachtet, das von einem Enzym E umgesetzt wird. Die Reaktion läuft in einer Reaktionslösung mit dem Volumen V in einem idealen Rührkesselreaktor ab. Aus den Massen- und Volumenbilanzen für dieses System folgt:

$$\frac{d}{dt}\begin{pmatrix} S \\ E \\ V \end{pmatrix} = \begin{pmatrix} f_1 \\ f_2 \\ f_3 \end{pmatrix} = \begin{pmatrix} \dfrac{(S_0 - S)\ \dot{V}_{Substrat}}{V} - \dfrac{\dot{V}_{Enzym}}{V} S - \dfrac{v_{\max} ES}{K_m + S} \\ \dfrac{(E_0 - E)\dot{V}_{Enzym}}{V} - \dfrac{\dot{V}_{Substrat}}{V} E \\ \dot{V}_{Substrat} + \dot{V}_{Enzym} - \dot{V}_{Sample} \end{pmatrix} \quad \{38\}$$

S_0 ist dabei die Konzentration im Substrat-Feed und $\dot{V}_{Substrate}$ ist dessen Volumenstrom. Die Volumenströme die bei Enzymzugabe bzw. beim Messen auftreten werden durch $\dot{V}_{Enzyme}$ und $\dot{V}_{Sample}$ berücksichtigt. E_0 ist die Enzymkonzentration im Enzym-Feed. Die Parameter der Michaelis-Menten-Kinetik sind v_{max} und K_m. Das Prozessmodell hat drei Zustandsgrößen und zwei freie Parameter. Die Zustandssensitivitätsdifferenzialgleichung ist daher eine 3x2 Matrix mit folgender Form:

$$\begin{pmatrix} \dot{S}_{v_{\max}} & \dot{S}_{K_m} \\ \dot{E}_{v_{\max}} & \dot{E}_{K_m} \\ \dot{V}_{v_{\max}} & \dot{V}_{K_m} \end{pmatrix} = \begin{pmatrix} \frac{\partial f_1}{\partial S} & \frac{\partial f_1}{\partial E} & \frac{\partial f_1}{\partial V} \\ 0 & \frac{\partial f_2}{\partial E} & \frac{\partial f_2}{\partial V} \\ 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} S_{v_{\max}} & S_{K_m} \\ E_{v_{\max}} & E_{K_m} \\ V_{v_{\max}} & V_{K_m} \end{pmatrix} + \begin{pmatrix} \frac{\partial f_1}{\partial v_{\max}} & \frac{\partial f_1}{\partial K_m} \\ 0 & 0 \\ 0 & 0 \end{pmatrix} \quad \{39\}$$

$S_{v\max}$ ist die Sensitivität des Substrats S bezüglich des Parameters v_{max} und $\dot{S}_{v\max}$ ist deren Ableitung nach der Zeit. Für die anderen Größen auf der linken Seite der Gleichung gilt entsprechendes. In Gleichung {39} sind die Ableitungen der Funktionen, die nicht von der Variable nach der sie abgeleitet werden abhängen, schon auf Null gesetzt. Als Anfangsbedingung für dieses Gleichungssystem gilt, dass alle Sensitivitäten zu Anfang des Prozesses Null sind. Daraus folgt, dass die Sensitivitäten des Volumens nie einen von Null verschiedenen Wert annehmen können. Für die Sensitivitäten bezüglich der Enzymkonzentration gilt dies gleichermaßen, da die entsprechenden Gleichungen keine Inhomogenität enthalten. Es müssen folglich nur die beiden Sensitivitäten von S betrachtet werden:

$$\dot{S}_{v_{\max}} = \left(-\frac{\dot{V}_{Substrat} + \dot{V}_{Enzym}}{V} - \frac{\nu_{\max} E}{K_m + S} + \frac{\nu_{\max} ES}{\left(K_m + S\right)^2} \right) S_{v_{\max}} - \frac{ES}{K_m + S} \quad \{40\}$$

$$\dot{S}_{K_M} = \left(-\frac{\dot{V}_{Substrat} + \dot{V}_{Enzym}}{V} - \frac{\nu_{\max} E}{K_m + S} + \frac{\nu_{\max} ES}{\left(K_m + S\right)^2} \right) S_{K_M} + \frac{\nu_{\max} ES}{\left(K_m + S\right)^2} \quad \{41\}$$

Beide Gleichungen sind, abgesehen von dem inhomogenen Anteil, sehr ähnlich. Der inhomogene Anteil bewirkt, dass S_{vmax} ausgehend von dem Startwert bei Null negativ wird. S_{Km} nimmt dagegen positive Werte an. Bei hohen Substratkonzentrationen ($S \gg K_m$) sind der zweite und der dritte Term in den Klammern gleich und der Verlauf beider Sensitivitäten wird gänzlich durch den inhomogenen Anteil bestimmt. Je höher die Anfangskonzentration an Substrat ist, um so länger streben die Sensitivitäten hohen absoluten Werten zu (S_{vmax} zu hohen negativen Werten und S_{Km} zu hohen positiven Werten). Haben die Sensitivitäten einen hohen Wert, bedeutet dies, dass die Parameter mit Messungen an diesen Zeitpunkten mit hoher Genauigkeit bestimmbar sind. Im

Prinzip deckt sich dieses Ergebnis aus der Analyse der Gleichungen mit den Aussagen von Endrenyi und Chan, die als optimalen Messzeitpunkt die Stelle höchstmöglicher Substratkonzentration gleich zu Beginn des Prozesses vorschlugen [ENDRENYI]. Der Unterschied liegt darin, das bei Endrenyi und Chan die Geschwindigkeit der Reaktion betrachtet wurde und in dieser Studie ist der Messwert die Substratkonzentration selbst. Das führt dazu, dass die Sensitivitäten hier zu Beginn des Prozesses Null sind und eine Messung dort keinen Informationsgewinn darstellt. Der optimale Messzeitpunkt muss also durch Berechnung der FIM mit einem Optimierungskriterium bestimmt werden. Zunächst aber kann durch Analyse der Gleichungen noch der Schluss gezogen werden, dass die Betrachtung eines Enzym-Feeds bei diesem Prozess nicht sinnvoll ist. Die Änderung der Sensitivität ist proportional zur Enzymkonzentration E. Zum Erreichen hoher absoluter Werte für die Sensitivität ist eine hohe Enzymkonzentration zu Beginn des Prozess nötig. Am zweiten und dritten Term in den Klammern kann man ablesen, dass gilt:

$$\frac{\nu_{\max} E}{K_m + S} > \frac{\nu_{\max} ES}{\left(K_m + S\right)^2} \qquad \{42\}$$

Der Term in der Klammer ist folglich immer negativ und die Veränderung der Sensitivitäten ist deswegen immer gegenläufig zum aktuellen Wert. Dies führt dazu, dass die Sensitivitäten zum Ende des Prozesses dem Wert Null zustreben. Zum Beibehalten hoher Werte im späteren Verlauf des Prozesses ist wiederum eine niedrige Enzymkonzentration günstig. Beide beschriebenen Auswirkungen auf die Sensitivität, die die Genauigkeit der Messung verbessern würden, lassen sich durch einen Enzym-Feed nicht realisieren.

Aus Gleichung {9} folgt für die FIM:

$$FIM = \sum_i^N \left[\begin{pmatrix} S_{v_{\max}}(t_i) & S_{K_m}(t_i) \end{pmatrix}^T \frac{1}{\sigma_i} \begin{pmatrix} S_{v_{\max}}(t_i) & S_{K_m}(t_i) \end{pmatrix} \right]$$

$$FIM = \begin{pmatrix} \sum_i^N \left(\frac{\left(S_{v_{\max}}(t_i)\right)^2}{\sigma_i} \right) & \sum_i^N \left(\frac{S_{v_{\max}}(t_i) S_{K_M}(t_i)}{\sigma_i} \right) \\ \sum_i^N \left(\frac{S_{K_m}(t_i) S_{v_{\max}}(t_i)}{\sigma_i} \right) & \sum_i^N \left(\frac{\left(S_{K_m}(t_i)\right)^2}{\sigma_i} \right) \end{pmatrix} \quad \{43\}$$

Dabei wurde anstelle von Q die Messfehlervarianz σ eingesetzt. Zum Auffinden optimaler Messzeitpunkte t_i und weiterer Messbedingungen wurde das E-Kriterium verwendet – die Maximierung des kleinsten Eigenwertes der FIM. Die Eigenwerte lassen sich mit Gleichung {44} berechnen.

$$\lambda_{1/2} = \sum_{i=1}^N \frac{\left(S_{v_{\max}}(t_i)\right)^2 + \left(S_{K_m}(t_i)\right)^2}{2\sigma_i} \pm \sqrt{\left[\sum_i^N \frac{\left(S_{v_{\max}}(t_i)\right)^2 - \left(S_{K_m}(t_i)\right)^2}{2\sigma_i} \right]^2 + \left[\sum_i^N \frac{S_{K_m}(t_i) S_{v_{\max}}(t_i)}{\sigma_i} \right]^2} \quad \{44\}$$

Aus der Inversen der FIM lässt sich die Cramer-Rao-Lower-Bound (CRLB) berechnen, die die minimal erreichbare Varianz bei der Parameterbestimmung mit den gewählten Messbedingungen angibt [KAY]. Sie ist neben dem kleinsten Eigenwert der FIM ein weiteres Maß für die Güte der Parameterbestimmung.

3.2 Lösung des Optimierungsproblems

Das Ziel dieser Studie war die Berechnung von Prozess- und Messbedingungen, mit denen die Parameter v_{max} und K_m mit größtmöglicher Genauigkeit bestimmt werden können. Die Berechnung wurde mehrmals unter Annahme verschiedener experimenteller Bedingungen durchgeführt. Zum Einen wurde zwischen der Prozessführung als Batch- und als Fed-Batch-Prozess unterschieden, um die Frage zu

klären welche Art der Prozessführung zur Parameterbestimmung günstiger ist. Zum Anderen wurden die Berechnungen mit und ohne Nebenbedingungen durchgeführt. Im ersten Fall werden die Lösungen der optimalen Messzeitpunkte und der Substratanfangskonzentration auf bestimmte, diskrete Werte eingeschränkt, im zweiten Fall sind beliebige Werte zulässig. Die Einschränkung auf bestimmte Werte wurde deswegen eingeführt, da es z. B. bei der Wahl der Messzeitpunkte in der Realität durchaus vorkommen kann, dass die Durchführung von Messungen mit einem Intervall von kleiner als eine Viertel Stunde nicht möglich ist. Die im Folgenden erläuterten Nebenbedingungen führen also dazu, dass in der Praxis unrealistische und undurchführbare Messstrategien aus dem Lösungsraum ausgeschlossen werden. Gleiches gilt dabei für die Substratzugabe im Falle des Fed-Batch-Prozesses. Die angenommenen experimentellen Bedingungen und Konstanten sind in Tabelle 1 zusammengestellt.

Tabelle 1. Angenommene Bedingungen und Konstanten.

Gesamtmenge Substrat M_S^{total}	10 mmol
Gesamtvolumen Pufferlösung V_{total}	10 mL
Gesamtmenge Enzym M_E^{total}	50 mg
Anzahl Messungen N	10
Probenahmevolumen V_{sample}	0,3 µL
Schätzwert v_{max}	0,12 mol/hL
Schätzwert K_m	0,3 mol/L
Maximale Anzahl Feed-Pulse	5
Pulszeit Δt	0,001 h
Prozesszeit	5 h

Beim Batch-Prozess wird die gesamte Menge an Substrat und Enzym in 5 mL Pufferlösung gelöst, sodass die Startkonzentration des Substrats 2 mol/L betrug. Beim Fed-Batch-Prozess wurde die zur Verfügung stehende Menge an Substrat und Pufferlösung aufgeteilt in den Anteil, der vorgelegt wird und den Anteil, der über die maximal 5 Feed-Pulse zugegeben wird. Konzentration und Volumen der 5 Feed-Pulse waren jeweils identisch. Bei den Rechnungen ohne Nebenbedingungen konnte die Gesamtmenge an Substrat und Volumen in beliebiger Weise zwischen Vorlage und

Pulsen aufgeteilt werden (das minimal zulässige Volumen der Feed-Pulse wurde auf 5 µL gesetzt). Mit Nebenbedingungen waren für die Substratmenge in der Vorlage Werte von $(1\text{-}2^{-i})\,M_S^{total}$ mit i = 1-5 und M_S^{total} zulässig und entsprechend Werte von $2^{-i}\,M_S^{total}$ mit i = 1-5 und 0 für die Substratmenge in den Pulsen. Für das Volumen, das auf die 5 Pulse aufgeteilt wurde, waren Werte von $0{,}25*i*V_{total}$ mit i = 1-3 zulässig. Eine weitere Einschränkung, die die Vergleichbarkeit von Batch- und Fed-Batch-Experimenten ermöglichen sollte, ist, dass Kombinationen von Substratmenge und Volumen, die zu Substratkonzentrationen über 2 mol/L führen würden, ausgeschlossen wurden. Die Probenahme an den 10 Messzeitpunkten t_i wurde in den Differentialgleichung wie folgt realisiert:

$$\dot{V}_{Sample} = V_{Sample}\,\frac{Heaviside(t-t_i)*Heaviside(t_i-t+\Delta t)}{\Delta t} \qquad \{45\}$$

Auf analoge Weise wurde der Substrat-Feed simuliert. Die Messzeitpunkte konnten bei den Rechnungen ohne Nebenbedingungen beliebige Werte im Bereich von 0 h bis 5 h annehmen, einschließlich des Falles, dass mehrere oder alle Messungen zum selben Zeitpunkt erfolgen sollen. Gleiches gilt dabei für die Feed-Pulse. Bei den Rechnungen mit Nebenbedingungen können die Feed-Pulse nur bei i*0,5 h mit i = 1-9 und die Messungen nur bei i*0,25 h mit i = 1-20 auftreten. Mehrere Messungen oder Pulse zur gleichen Zeit sind ebenfalls ausgeschlossen.

Die Rechnungen wurden mit MATLAB (Ver. 6.0.0.88 R12, Simulink 4.0, The MathWorks Inc., Natik, USA) durchgeführt. Die Differentialgleichungen wurden in Simulink mit der Methode 15s gelöst. Als Optimierungsalgorithmus wurde ein genetischer Algorithmus (The Genetic Algorithm Toolbox, Department of Automatic Control and Systems Engineering, University of Sheffield) verwendet. In den Individuen einer Population sind die 5 Zeitpunkte des Substrat-Feeds (nur im Fall des Batch-Prozesses), die 10 Messzeitpunkte und die Aufteilung von Substrat- und Pufferlösung zwischen Vorlage und Pulsen als Realzahlen kodiert. Die Population setzte sich aus 1000 Individuen zusammen. Als Rekombinationsverfahren wurde das Ein-Punkt-Crossover gewählt. Die Mutationsrate betrug 10 %. Die Selektion wurde mit der Roulette-Wheel-Methode durchgeführt. Bei den Optimierungen erfolgte jeweils die Berechnung von 1000 Populationen. Die durch den genetischen Algorithmus zu minimierende Zielgröße ist der Kehrwert des kleinsten Eigenwertes der FIM.

Die Optimierungen wurden für drei verschiedenen Messfehlervarianzen durchgeführt, um zu berücksichtigen, dass der Messfehler vom Wert und vom Bereich der Messgröße abhängt. Eine Varianz ist vom Messbereich und –wert unabhängig und hat einen Wert von 2,5 % der Substratanfangskonzentration:

$$\sigma^1 = \left(0,05 \frac{M_S^{total}}{V_{total}} \right)^2 \qquad \{46\}$$

Die zweite Varianz hängt vom Messwert ab und stellt somit den Fall eines konstanten relativen Fehlers dar:

$$\sigma_i^2 = \left(0,05 S\left(t_i\right) \right)^2 \qquad \{47\}$$

Die dritte Varianz stellt den Fall eines linear mit der Messgröße ansteigenden Fehlers dar, wobei der Fehler immer größer ist als 0,03 mol/L:

$$\sigma_i^3 = \left(0,03 \frac{mol}{L} + 0,04 S(t_i) \right)^2 \qquad \{48\}$$

Diese drei Fälle sollten die meisten real auftretenden Messfehler abdecken.

3.3 Ergebnisse

3.3.1 Batch-Prozess ohne Nebenbedingungen

Für den Batch-Prozess ohne Nebenbedingungen sind die Substratkonzentration und die Quadrate der Sensitivitäten in Abbildung 16 dargestellt. Zu Beginn des Prozesses sind die Sensitivitäten Null und durchlaufen danach ein Maximum. Die Sensitivität bezüglich v_{max} ist immer höher als bei K_m und folglich kann v_{max} mit höherer Genauigkeit als der andere Parameter K_m gemessen werden. Als Optimierungsergebnis bei den Messzeitpunkten wurde eine Zwei-Punkt-Mehrfachmessung erhalten, was sich mit den Resultaten früherer Untersuchungen von enzymatischen Batch-Prozessen deckt [ENDRENYI]. Bei konstantem absolutem Messfehler (σ^1) sind die optimalen Messzeitpunkte bei 0,97 h (4 Messungen) und bei 2,39 h (6 Messungen). Bei konstantem relativen Fehler (σ^2) ergaben sich optimale Messzeitpunkte bei 2,53 h (7

Messungen) und 5 h (3 Messungen). Für den letzten betrachteten Fall (σ^3) sind die optimalen Messzeitpunkte bei 1,32 h (4 Messungen) und bei 2,52 h (6 Messungen). In Tabelle Tabelle 2 sind die Ergebnisse der Optimierung zusammengefasst. Die CRLB des Parameterschätzfehlers ist in Prozent, bezogen auf den Wert des groben Schätzwertes des Parameters P_0, angegeben. Weil die drei angenommenen Messfehler sehr unterschiedlich sind, können diese drei Fälle untereinander nicht direkt verglichen werden. Die CRLB für σ^2 sind viel geringer als die für σ^1 und σ^3, welche annähernd gleich sind. Die Präzision, mit der v_{max} bestimmt werden kann ist viermal so hoch wie die von K_m für die Fälle σ^1 und σ^3 bzw. doppelt so hoch für σ^2 (je niedriger die CRLB, desto höher ist die erreichbare Genauigkeit).

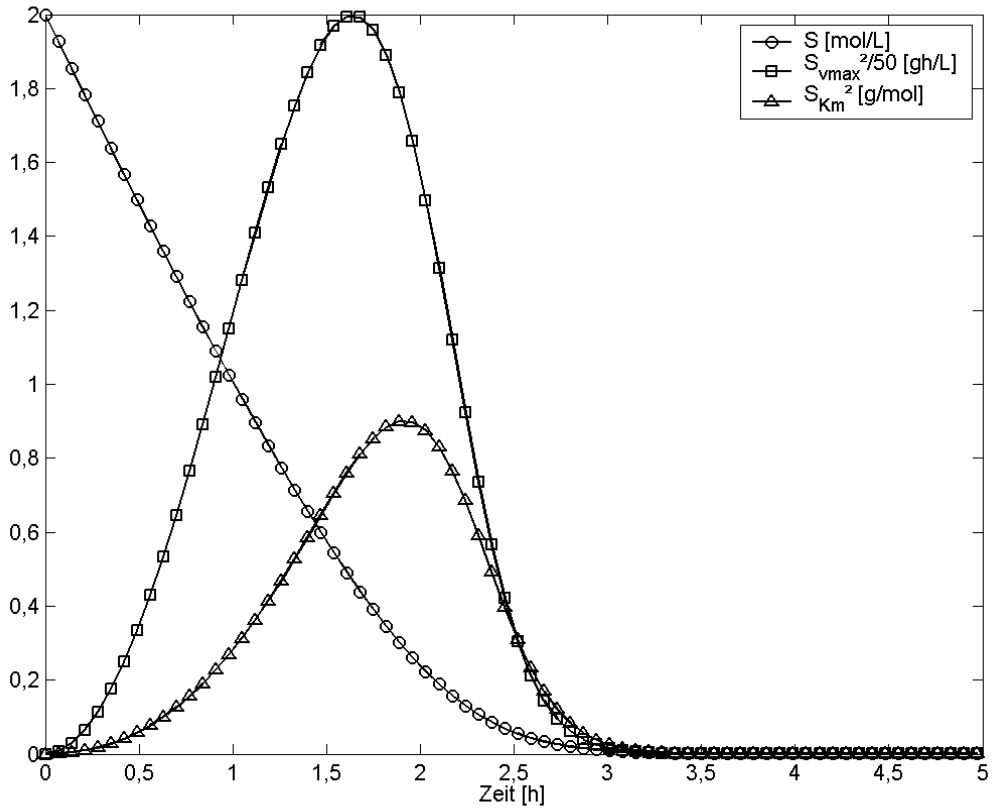

Abbildung 16. Substratkonzentration und quadrierte Sensitivitäten für den Batch-Prozess ohne Nebenbedingungen.

Tabelle 2. Ergebnisse der Optimierung für den Batch-Prozess ohne Nebenbedingungen. Die CRLB ist in Prozent bezogen auf den geschätzten Parameterwert angegeben.

	Messzeitpunkt 1 [h] (Anzahl)	Messzeitpunkt 2 [h] (Anzahl)	kleinster Eigenwert λ_{min}	CRLB v_{max} [%]	CRLB K_m [%]
σ^1	0,97 (4)	2,39 (6)	158	6,64	26,4
σ^2	2,53 (7)	5,0 (3)	144802	0,504	0,854
σ^3	1,32 (4)	2,52 (6)	199	6,62	23,5

3.3.2 Fed-Batch-Prozess ohne Nebenbedingungen

Für den Fall des konstanten absoluten Fehlers sind der Konzentrationsverlauf des Substrats sowie die quadrierten Sensitivitäten des optimalen Designs in Abbildung 17 gezeigt. In Tabelle 3 sind die optimalen Feed- und Messzeitpunkte sowie der kleinste Eigenwert und die CRLB aufgeführt. Der Substratfeed wurde mit dem kleinsten zulässigen Volumen von 0,5 µL durchgeführt. Der Feed hat daher keine sichtbare Auswirkung auf die Sensitivitäten. Als Substratanfangskonzentration wurde nicht die maximal mögliche Konzentration von 2 mol/L gewählt, sondern 1,56 mol/L. Bei den Messzeitpunkten wurde, wie beim Batch-Prozess, eine Zwei-Punkt-Mehrfachmessung erhalten. Ein Vergleich der CRLB von v_{max} und K_m mit denen des Batch-Prozesses zeigt, dass die CRLB von v_{max} um 0,19 % höher und die von K_m um 6,2 % niedriger ist. Somit hat sich die Genauigkeit, mit der der Parameter K_m gemessen werden kann, im Vergleich zum Batch-Prozess verbessert, während der Parameter v_{max} annähernd gleich gut bestimmbar ist.

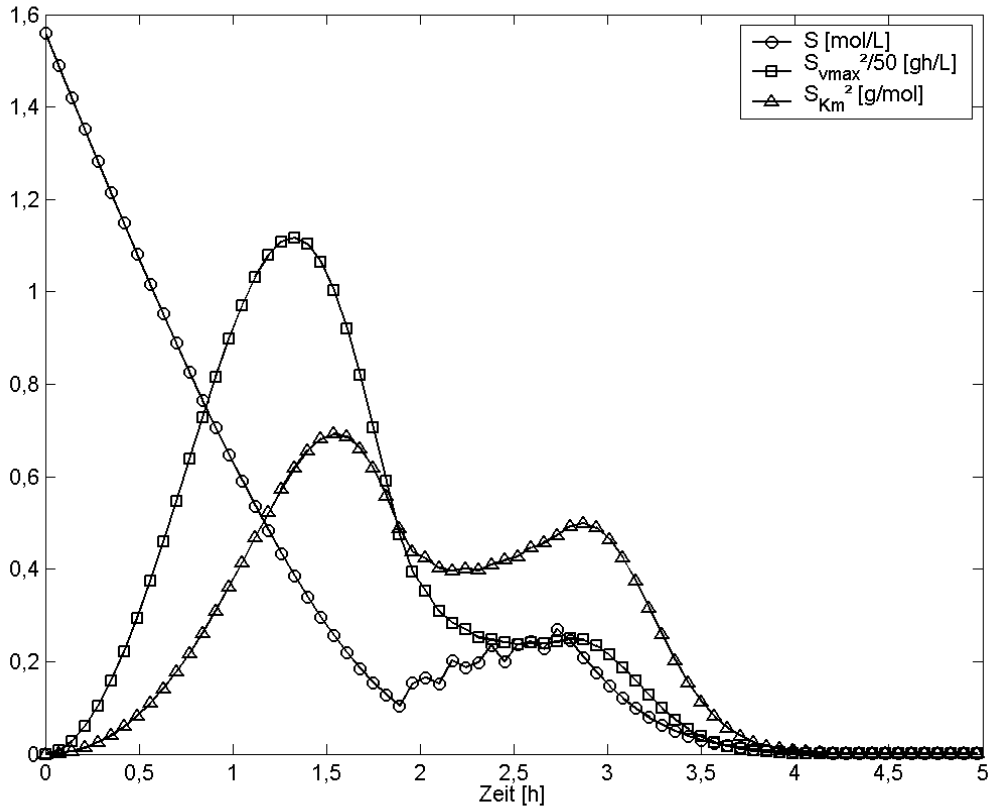

Abbildung 17. Substratkonzentration und quadrierte Sensitivitäten für den Fed-Batch-Prozess ohne Nebenbedingungen (Messfehlermodell σ^1).

Tabelle 3. Ergebnisse der Optimierung für den Fed-Batch-Prozess ohne Nebenbedingungen. Die CRLB ist in Prozent bezogen auf den geschätzten Parameterwert angegeben.

	Zeitpunkte Substrat-zugabe [h] (Feedvo-lumen [µL])	Messzeitpunkte [h] (Anzahl)	kleinster Eigenwert λ_{min}	CRLB v_{max} [%]	CRLB K_m [%]
σ^1	1,9, 2,09, 2,27, 2,46, 2,64 (0,5)	0,84 (4), 2,98 (6)	269	6,83	20,2
σ^2	4,73, 5,0, 5,0, 5,0, 5,0 (0,5)	1,62 (6), 2,12 (2), 4,73 (2)	968195	0,294	0,318
σ^3	2,01, 2,22, 2,42, 2,62, 2,81 (0,5)	1,10 (4), 3,25 (6)	403	6,12	16,4

Für den Fall σ^2 ist die Verbesserung der Parameterbestimmung stärker ausgeprägt, was aber im Wesentlichen darauf zurückzuführen ist, dass die optimierten Messzeitpunkte vor den Zeiten der Substratzugabe liegen. Wie in Abbildung 18 zu sehen ist, wurde als Substratanfangskonzentration die maximal mögliche Konzentration gewählt. Als optimale Messstrategie wurde eine Drei-Punkt-Mehrfachmessung erhalten.

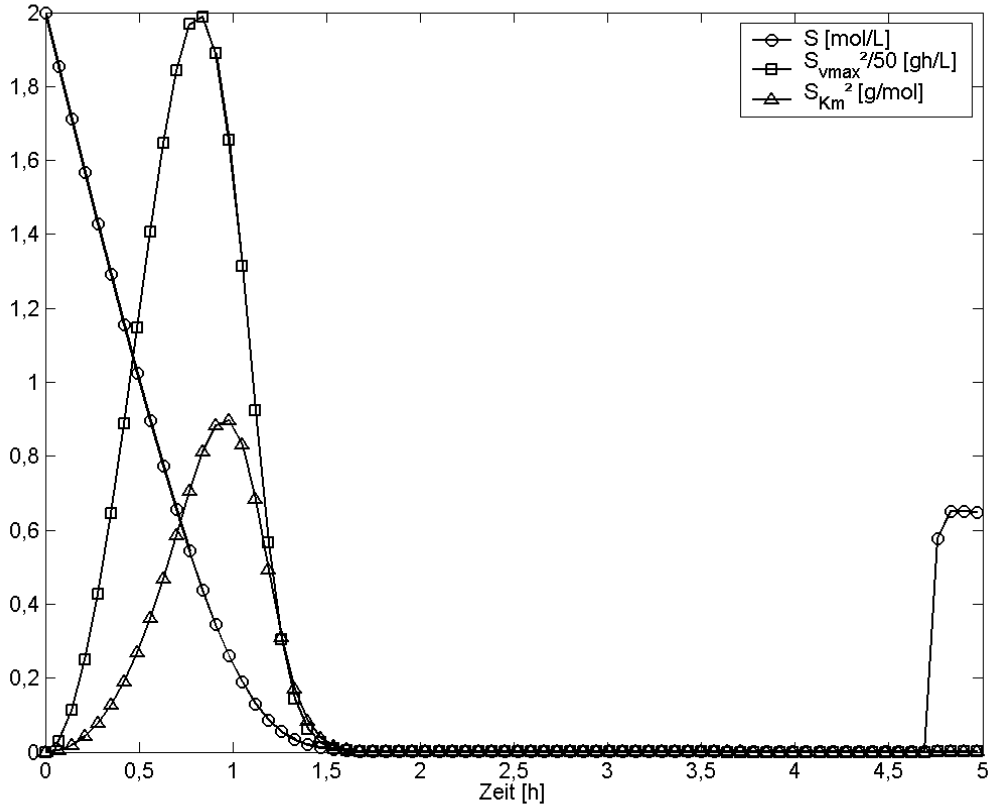

Abbildung 18. Substratkonzentration und quadrierte Sensitivitäten für den Fed-Batch-Prozess ohne Nebenbedingungen (Messfehlermodell σ^2).

Beim Fall σ^3 ist der kleinste Eigenwert mehr als doppelt so hoch im Vergleich zum Batch-Prozess. Während die CRLB von v_{max} annähernd gleich geblieben ist (0,5 % kleiner), ist die CRLB von K_m signifikant gesunken (7,1 %) und somit die Genauigkeit der Parameterbestimmung wesentlich besser als beim Batch-Prozess. Wieder wurde als Feedvolumen der kleinstmögliche Wert gewählt. Wie beim Fall σ^1 ergab die Optimierungsrechnung eine Substratanfangskonzentration von 1,56 mol/L und eine Zwei-Punkt-Mehrfachmessung als optimale Messstrategie. Der Verlauf der

Substratkonzentration und die quadrierten Sensitivitäten sind in Abbildung 19 dargestellt.

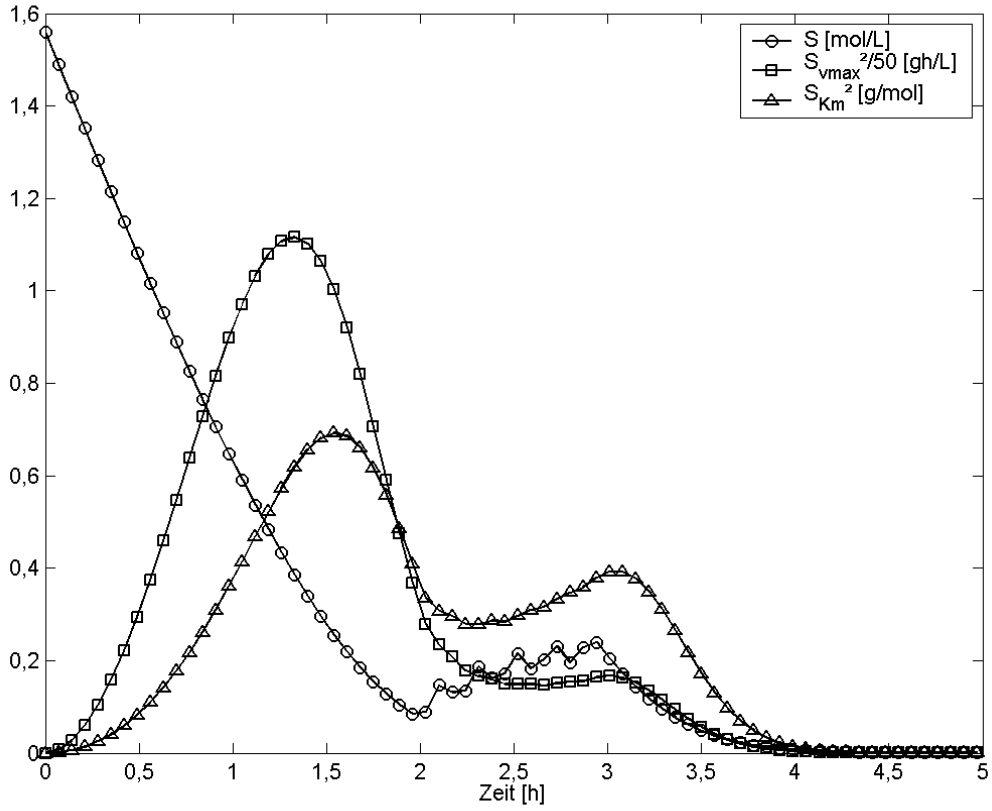

Abbildung 19. Substratkonzentration und quadrierte Sensitivitäten für den Fed-Batch-Prozess ohne Nebenbedingungen (Messfehlermodell σ^3).

3.3.3 Batch-Prozess mit Nebenbedingungen

Bei diesen Optimierungsrechnung wurden die in Abschnitt 3.2 dargestellten Nebenbedingungen hinzugenommen. Die Ergebnisse, zusammen mit einer Auswertung unter Benutzung von äquidistanten Messzeitpunkten, sind in Tabelle 4 zusammengefasst. Verglichen mit der Rechnung ohne Nebenbedingungen liegen die Eigenwerte bei den drei betrachteten Messfehlern um durchschnittlich 60 % niedriger. Der Verlust bei der Genauigkeit der Parameterbestimmung ist etwas geringer, aber dennoch signifikant. Für K_m nimmt die CRLB um 32 % (σ^1), 24 % (σ^2) bzw. 32% (σ^3) zu. Bei v_{max} ergeben sich CRLB, die um 22 % (σ^1), 23 % (σ^2) und um 19 % (σ^3) höher

liegen, als bei der vergleichbaren Rechnung ohne Nebenbedingungen. Erwartungsgemäß sind die hier auf diskrete Werte eingeschränkten Messzeitpunkte um die optimalen Messzeitpunkte der Rechnung ohne Nebenbedingungen angeordnet. Die Verteilung der Messungen auf den frühen und späten Messzeitpunkt ist aber bei σ^1 und bei σ^3 anders. Bei σ^1 finden im optimalen Fall fünf Messungen um den frühen und ebenfalls fünf Messungen um den späten Messzeitpunkt statt. Im Fall σ^3 ist die Verteilung der Messungen im Vergleich zur Rechnung ohne Nebenbedingungen genau umgekehrt (6 früh und 4 spät). Vergleicht man das optimale Messdesign mit dem Fall äquidistanter Messzeitpunkte ergibt sich, dass letzterer eine wesentlich schlechtere Parameterbestimmung erlaubt: die kleinsten Eigenwerte sind bei allen drei Messfehlermodellen niedriger und die CRLB sind immer höher als bei den Rechnungen mit optimierten Messzeitpunkten.

Tabelle 4. Ergebnisse der Optimierung für den Batch-Prozess mit Nebenbedingungen (untere Tabellenhälfte) und Ergebnisse bei äquidistanten Messzeitpunkten (obere Tabellenhälfte).

	Messzeitpunkte [h]										kleinster Eigen-wert	CRLB [%]	
	t_1	t_2	t_3	t_3	t_5	t_6	t_7	t_8	t_9	t_{10}	λ_{min}	v_{max}	K_M
σ^1 äqui.	0,5	1,0	1,5	2,0	2,5	3,0	3,5	4,0	4,5	5,0	47	11	48
σ^2 äqui.	0,5	1,0	1,5	2,0	2,5	3,0	3,5	4,0	4,5	5,0	55539	0,81	1,38
σ^3 äqui.	0,5	1,0	1,5	2,0	2,5	3,0	3,5	4,0	4,5	5,0	60	11	43
σ^1 opt.	0,25	0,5	0,75	1,0	1,25	2,0	2,25	2,5	2,75	3,0	92	8,12	35
σ^2 opt.	1,75	2,0	2,25	2,5	2,75	3,0	3,25	4,5	4,75	5,0	94110	0,62	1,06
σ^3 opt.	0,5	0,75	1,0	1,25	1,5	1,75	2,25	2,5	2,75	3,0	114	7,89	31,1

3.3.4 Fed-Batch-Prozess mit Nebenbedingungen

Die Ergebnisse der Optimierung mit Nebenbedingungen für den Fed-Batch-Prozess sind in Tabelle 5 gezeigt. Im Vergleich zu den Ergebnissen des Batch-Prozesses ist der Parameterschätzfehler signifikant verbessert. Die Verbesserung der Genauigkeit ist für K_m stärker ausgeprägt als für v_{max}. Bei allen drei betrachteten Fällen für den Messfehler wurde mit 2 mol/L die maximal mögliche Substratanfangskonzentration verwendet. Ebenso war bei allen drei Fällen das Anfangsvolumen im Reaktor (2,5 mL), das Feedvolumen (7,5 mL) und die Substratkonzentration im Feed (0,67 mol/L) gleich. In Abbildung 20 ist der Verlauf der Substratkonzentration und die quadrierten Sensitivitäten für den konstanten, absoluten Messfehler (σ^1) dargestellt. Im Vergleich zu dem Konzentrationsverlauf und den Sensitivitäten bei der Rechnung ohne Nebenbedingungen (Abbildung 17) ergeben sich hier andere Verläufe, da besonders die Messzeitpunkte aufgrund der Nebenbedingungen sich gänzlich anders verteilen als bei der Rechnung ohne Nebenbedingungen.

Tabelle 5. Ergebnisse der Optimierung mit Nebenbedingungen für den Fed-Batch-Prozess.

	Anfangs-konzentra-tion [mol/L]	Feed-konzentra-tion [mol/L]	Feed-Zeitpunkte [h]	Messzeitpunkte [h]	Kleinster Eigenwert	CRLB [%]	
			t_1.t_5	t_1-t_{10}	λ_{min}	v_{max}^E	K_M
σ^1	2	0,67	1,0; 1,5; 2,0; 2,5; 2,5	0,5; 0,75; 1,5; 2,0; 2,5; 3,25; 3,5; 3,75; 4,0; 4,25	207	6,6	23,1
σ^2	2	0,67	4,5; 4,5; 4,5; 4,5; 4,5	0,5; 0,75; 1,0; 1,25; 1,5; 1,75; 2,0; 3,25; 4,5; 5,0	305986	0,46	0,57
σ^3	2	0,67	1,5; 2,0; 2,5; 3,0; 3,0	0,5; 0,75; 2,0; 2,5; 3,0; 3,75; 4,0; 4,25; 4,5; 4,75	331	6,4	18

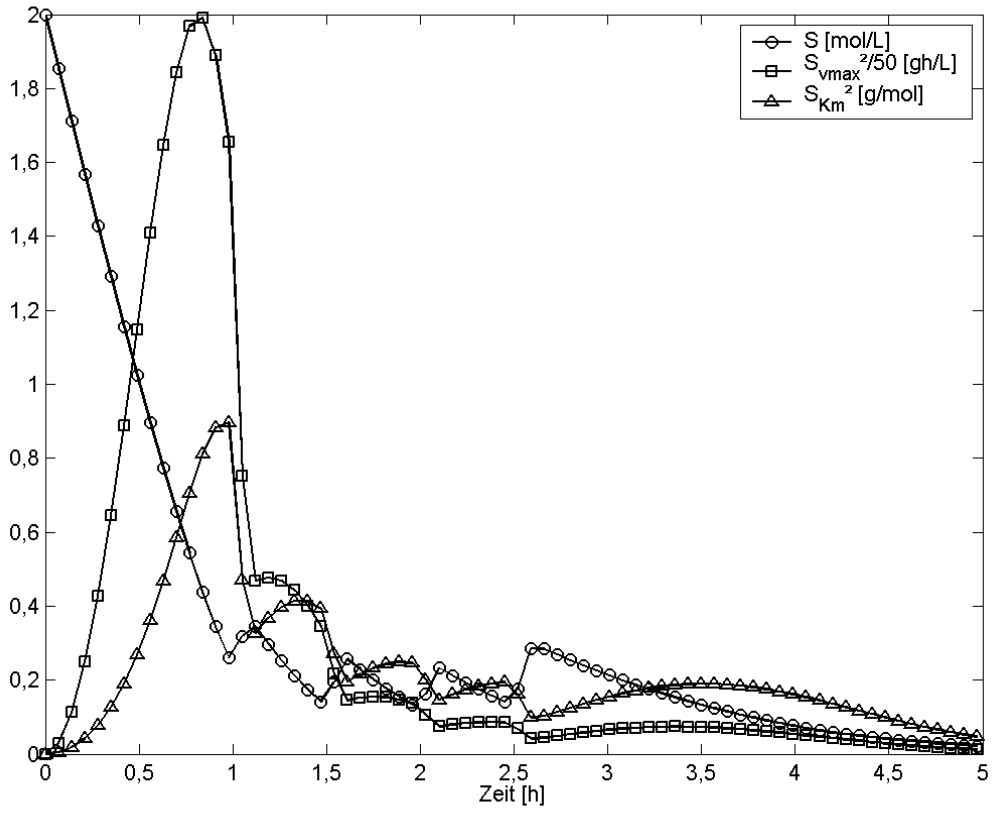

Abbildung 20. Substratkonzentration und quadrierte Sensitivitäten für den Fed-Batch-Prozess mit Nebenbedingungen (Messfehlermodell σ^1).

Die CRLB ist gegenüber der Ergebnis ohne Nebenbedingungen für v_{max} geringfügig kleiner (0,23 %) und für den anderen Parameter, K_m, etwas höher (2,9 %). Im Vergleich zu den Ergebnissen des Batch-Prozesses aus Abschnitt 3.3.3 zeigt sich, das beim Fed-Batch-Prozess die Parameter wesentlich genauer bestimmt werden können: Die CRLB für v_{max} ist um 1,5 % niedriger und die für K_m sogar um 11,9 %.

In Abbildung 21 sind die Substratkonzentration und die quadrierten Sensitivitäten für den Fall des konstanten, relativen Messfehlers (σ^2) gezeigt. Die Verläufe weisen große Ähnlichkeit zu den in Abbildung 18 gezeigten Verläufen auf. Verglichen mit den Ergebnissen der Rechnung ohne Nebenbedingungen ist die Genauigkeit der Parameterbestimmung hier geringer. Im Vergleich zu den Ergebnissen des Batch-Prozesses mit Nebenbedingungen ergeben sich, wie schon für den Fall σ^1, signifikante Verbesserungen in der Genauigkeit.

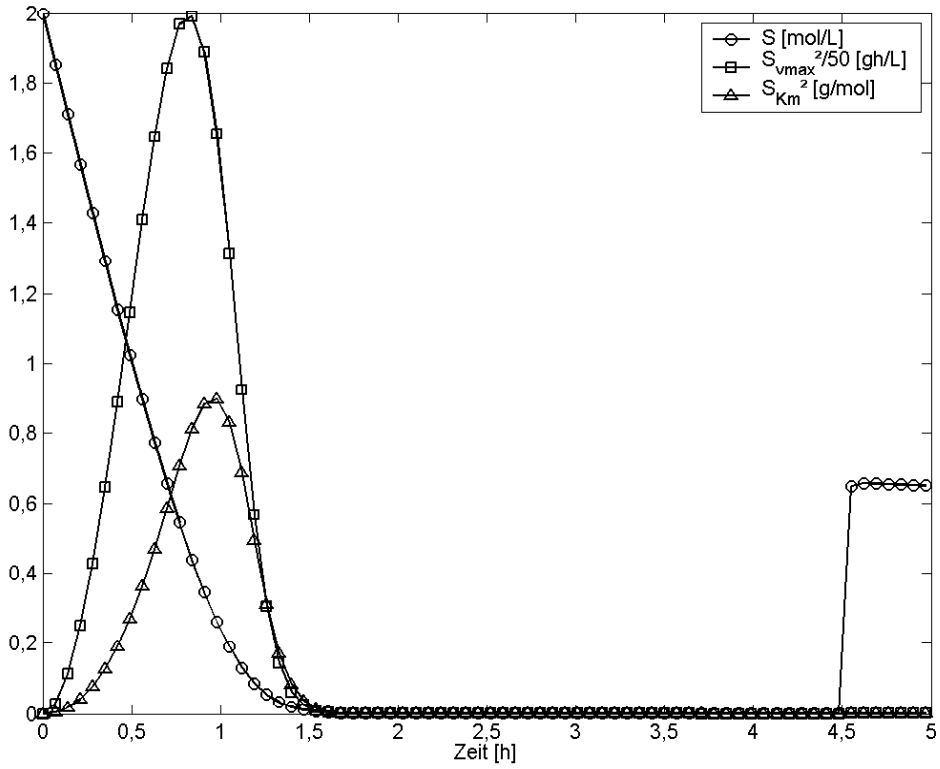

Abbildung 21. Substratkonzentration und quadrierte Sensitivitäten für den Fed-Batch-Prozess mit Nebenbedingungen (Messfehlermodell σ^2).

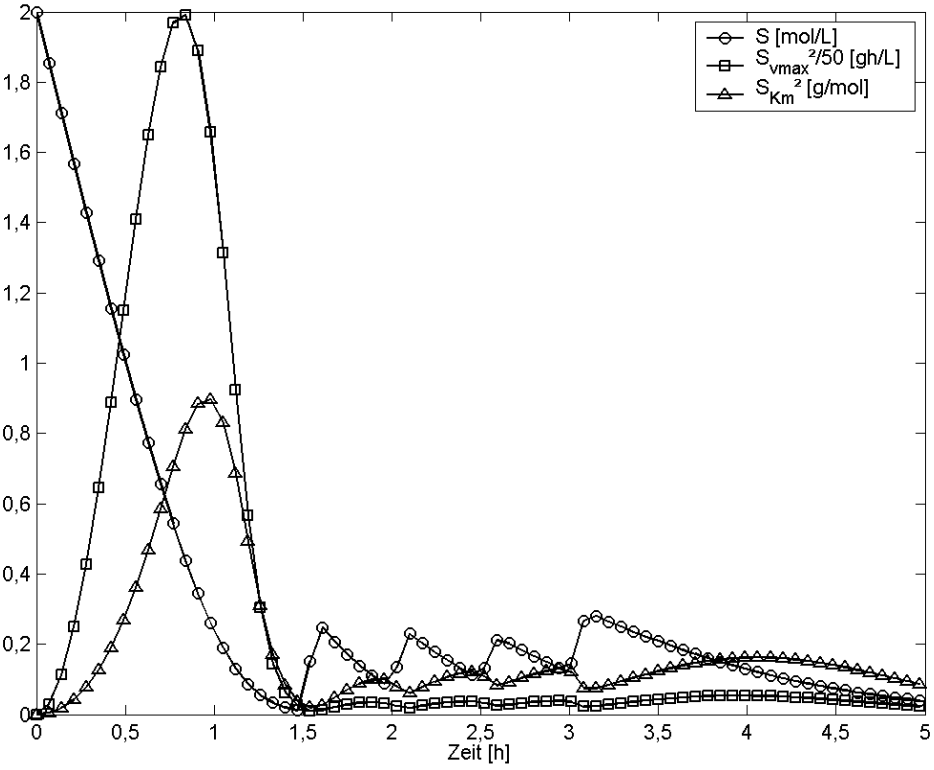

Abbildung 22. Substratkonzentration und quadrierte Sensitivitäten für den Fed-Batch-Prozess mit Nebenbedingungen (Messfehlermodell σ^3).

Für den Fall σ^3 ist der Verlauf der Substratkonzentration sowie die quadrierten Sensitivitäten in Abbildung 22 dargestellt. Auch hier ergeben sich höhere CRLB verglichen mit der entsprechenden Optimierungsrechnung ohne Nebenbedingungen und niedrige CRLB, also verbesserte Genauigkeit der Parameterbestimmung, verglichen mit dem Batch-Prozess mit Nebenbedingungen. Das Absinken der CRLB ist von den drei betrachteten Messfehler hier am stärksten: verglichen mit dem Batch-Prozess sind die CRLB für v_{max} um 1,49 % und die CRLB für K_m und 13,1 % niedriger.

3.4 Zusammenfassung

Die Resultate zeigen, dass die Genauigkeit der Parameterbestimmung durch optimal durchgeführte Experimente signifikant verbessert werden kann. Bei einer enzymatischen Reaktion, die durch die Michaelis-Menten-Kinetik beschrieben werden kann, konnte gezeigt werden, dass Messungen der Substratkonzentration beim Fed-Batch-Prozess wesentlich genauere Werte für die Parameter liefern, als Messungen am Batch-Prozess. Des weiteren ergab sich, dass durch einen Enzym-Feed keine Verbesserung zu erzielen ist. Durch Substratzugabe während des Prozesses dagegen, besonders bei kleinen Volumina der Feed-Pulse, kann die Genauigkeit der Parameterbestimmung verbessert werden.

Zum Auffinden der optimalen Prozessführung und Messstrategie wurde ein genetischer Algorithmus eingesetzt. Die Lösung der Differentialgleichungen wurde numerisch durchgeführt. Die Optimierungsgröße (die in diesem Fall maximiert werden sollte) war der kleinste Eigenwert der FIM (E-Kriterium).

Als optimales Design für den Batch-Prozess ohne Nebenbedingungen wurde, ähnlich wie bei Endrenyi und Chan [ENDRENYI], eine Zwei-Punkt Mehrfachmessung erhalten. Im Gegensatz zu den dort beschriebenen Ergebnissen, lag der erste Messpunkt nicht direkt am Anfang des Prozesses sondern, abhängig vom angenommenen Messfehler, bei späteren Zeitpunkten im Prozess. Die optimalen Messzeitpunkte für die Rechnung mit Nebenbedingungen lagen um die Zeitpunkte der Optimierung ohne Nebenbedingungen. Dies spiegelt einfach die Tatsache wieder, dass wenn das Optimum durch bestimmte Einschränkungen nicht erreichbar ist, sich eine Lösung ergibt, die so nah wie möglich am Optimum liegt. Die Optimierung der Messzeitpunkte selbst, im Vergleich zu

äquidistanten Messungen, verbessert die Genauigkeit der Parameterbestimmung wesentlich. Die CRLB sank dadurch im Mittel um 26 %.

Eine noch größere Verbesserung ist aber dadurch zu erreichen, dass man den Prozess als Fed-Batch-Prozess durchführt. Im Mittel betrug die Steigerung der Genauigkeit der Parameterbestimmung 40 % für K_m und 18 % für v_{max} im Vergleich zum Batch-Prozess.

Wie die Ergebnisse zeigen hat der angenommene Messfehler großen Einfluss auf das optimale Design. Die optimalen experimentellen Bedingungen für die Fälle σ^1 (konstanter, absoluter Fehler) und σ^3 (linear mit der Messgröße ansteigender Fehler) waren sich sehr ähnlich während sich bei σ^2 (konstanter, relativer Fehler) ganz andere optimale Bedingungen ergaben.

Sollten die zu Beginn geschätzten Werte für die Parameter weit von den realen Werten entfernt liegen, kann das berechnete optimale Design ungenügend sein. In diesem Fall wäre es notwendig sowohl die Experimental-Design-Rechnung als auch die Messung mehrfach durchzuführen: Experimental Design, Messung, nochmals Experimental Design mit den neuen Parameterwerten aus der Messung, usw.

Die angewendete Methode zum Ermitteln optimaler Prozess- und Messbedingungen kann leicht auf andere enzymatische Prozesse übertragen werden. Im Prozessmodell muss dafür lediglich die Michaelis-Menten-Kinetik durch eine andere Enzymkinetik ausgetauscht werden. Dies führt u. U. dazu, dass mehr als zwei Parameter auftreten, wodurch sich der Rechenaufwand erhöht. Mit modernen Rechnersystemen ist diese Aufgabe aber dennoch in vertretbarem Zeitrahmen zu bewältigen.

4 Prozessanalyse und -simulation mit neuronalen Netzen

4.1 Beschreibung des Prozesses und Zielsetzung

Bei dem betrachteten Prozess handelt es sich um die großtechnische Herstellung von Chymosin durch eine Pilzfermentation. Chymosin ist ein Enzym, das in der Nahrungsmittelindustrie zur Käseherstellung verbreitete Anwendung findet. Es wird durch aerobe Fed-Batch-Fermentation mit *Aspergillus niger var. awamori* produziert. In einem 50 m³ Fermenter werden dazu ein Komplexmedium und Maltose als Substrate vorgelegt. Nach dem Animpfen wird weitere Maltose nach einem bestimmten Muster zugefüttert. Neben der Substratzufuhr ist die Begasung mit Sauerstoff, das Halten eines bestimmten pH-Niveaus und eine gute Durchmischung für den Fermentationsprozess von entscheidender Bedeutung. Für die Herstellungskosten und den Ressourcenverbrauch ist vor allem das Betreiben des Rührwerks ein wesentlicher Faktor. Die Fermentationsdauer hängt dabei von der Prozessführung ab und kann zwischen 100-160 h variieren. Nach Abschluss der Fermentation wird die Produktkonzentration in der Fermentationsbrühe gemessen und es erfolgen umfangreiche Aufarbeitungsschritte. Nach zwei Aufarbeitungsschritten wird die Produktkonzentration erneut gemessen. Dabei treten in einigen Fällen erhebliche Veränderungen bezüglich der Produktendkonzentration auf, sowohl im negativen als auch im positiven Sinne.

Um den enormen Energieverbrauch während der Fermentation zu reduzieren, ist eine Verkürzung der Fermentationsdauer bzw. eine Erhöhung der Raum-Zeit-Ausbeute wünschenswert. Daher sind Online- und Offlinemessdaten in großem Umfang aufgezeichnet worden. Diese sollen mit statistischen Methoden ausgewertet werden. Darüber hinaus soll durch den Einsatz neuronaler Netze ein Modell des Prozesses erstellt werden, das zur Simulation und zur Optimierung verwendet werden kann.

4.2 Der Datensatz

Der Datensatz besteht aus Online- und Offlinedaten von ca. 100 Fermentationsprozessen, die in den Jahren 2003 und 2004 durchgeführt wurden. Bei den Onlinedaten sind Werte für die Prozesszeit, die Sauerstoff- und Kohlendioxidkonzentration im Abgas, die Zufütterungsmenge des Substrates (Maltose), die Sauerstoffaufnahmerate (OUR) und das Volumen der Fermentationsbrühe vorhanden. Weitere Prozessgrößen, die online gemessen wurden, sind die Zufütterungsmenge und –rate von Ammoniak, die Begasungsrate sowie der Kopfdruck im Fermenter, der Leistungseintrag durch das Rührwerk (MoF), der pH Wert, die Gelöstsauerstoffkonzentration (pO_2), die Temperatur im Fermenter und die Umdrehungsgeschwindigkeit des Rührwerks (in rpm). Die Anzahl der vorhandenen Messwerte für die 15 Prozessvariablen ist dabei innerhalb der Prozesse gleich, schwankt aber für die Prozesse zwischen ca. 1500 und 2000 Werten pro Variable, da die Laufzeiten der Prozesse unterschiedlich waren.

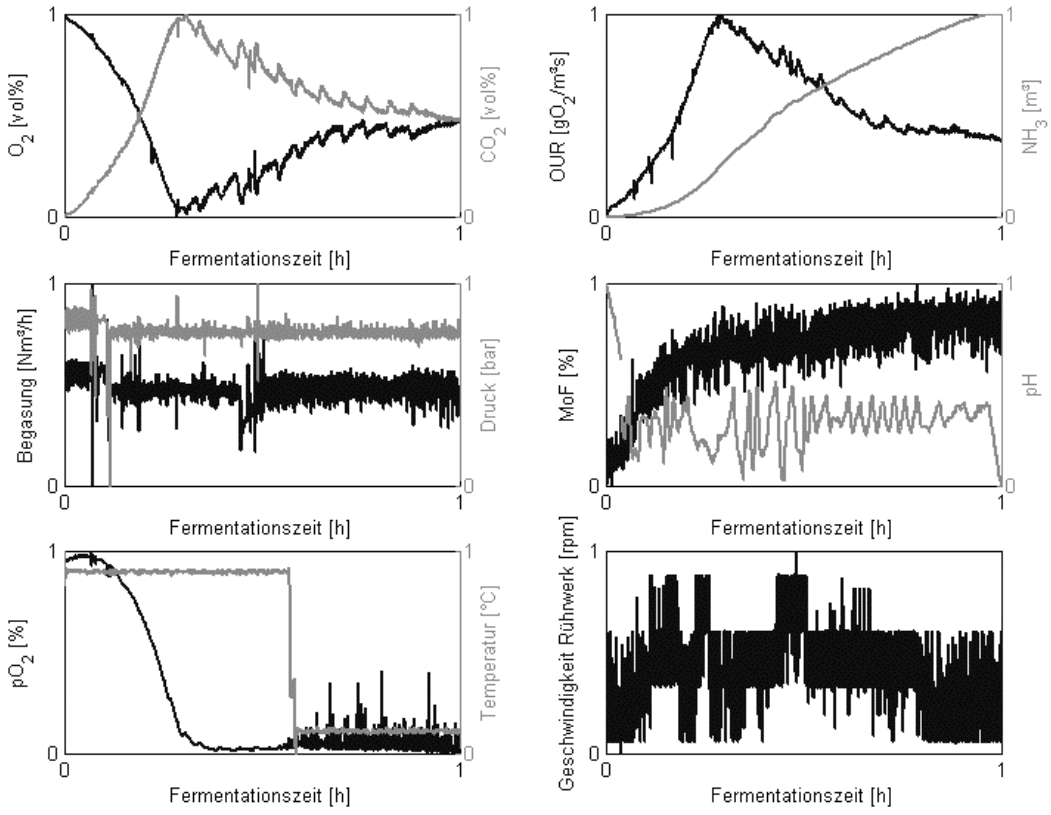

Abbildung 23. Onlinedaten des Fermentationsprozesses. (Die Substratzufütterung kann aus Datenschutzgründen nicht gezeigt werden. Aus dem gleichen Grund können die Daten nur normiert dargestellt werden.)

Das Zeitintervall, mit denen die Onlinemesswerte aufgezeichnet wurden, war bei allen Prozessen und für jede Prozessgröße identisch und betrug etwa 5 Minuten. Die Onlinedaten eines repräsentativen Prozesses sind in Abbildung 23 gezeigt. Nach ca. einem Drittel der Fermentationszeit durchlaufen CO_2 und OUR ein Maximum, O_2 ein Minimum und der pO_2-Wert fällt fast bis auf null ab. Dieser Zeitpunkt markiert den Wechsel von der Phase, in der hauptsächlich Biomasse gebildet wird, zur Produktbildungsphase. Während dieser Phase, die bis zum Ende des Prozesses andauert, befinden sich die Organismen in einer Sauerstofflimitierung. Stickstoff wird der Fermentation im Überfluss zugeführt, wie an der stetig steigenden Ammoniak-Kurve zu erkennen ist. Dies geschieht einerseits zur Stickstoffversorgung und andererseits zur Regulierung des pH-Wertes, der ebenso wie die Begasung, der Druck und die Rührgeschwindigkeit während des gesamten Prozesses konstant gehalten wird. Der Leistungseintrag (MoF) in das System steigt anfänglich, da sich die Viskosität der Fermentationsbrühe verändert, und bleibt dann nach ca. einem Drittel der Fermentation konstant. Die Temperatur wird nach ca. der Hälfte der Fermentationsdauer von einem hohen auf ein niedriges Niveau heruntergefahren.

Für die Produktkonzentration in der Fermentationsbrühe stehen Daten aus Offlinemessungen zur Verfügung. Des weiteren liegen auf diesen Werten basierende Regressionsdaten vor (polynomische Regression). In der ersten Prozesshälfte liegt im Fermentationsmedium kein Produkt vor und erst ungefähr ab dem Zeitpunkt, an dem die O_2-, CO_2- und andere Messgrößen Maxima bzw. Minima aufweisen, beginnt die Produktkonzentration zu steigen. Da die Offlinemessung der Produktkonzentration relativ aufwendig ist, liegen pro Prozessdurchlauf leider nur 3 bis 6 Messpunkte vor. Dies ist für einige der verwendeten Auswertungs- und Simulationsverfahren, besonders dem Training neuronaler Netze, als Datenbasis nicht ausreichend, sodass oft die Daten eines Regressionspolynoms, das zur Interpolation der Daten diente, für weitere Berechnungen verwendet wurde (siehe z. B. die Diskussion in Kapitel 4.4.1). Weitere vorhandene Offlinedaten sind Messungen der Viskosität sowie Messungen der Biotrockenmasse und der Konzentrationen von Glucose, Maltose, Zitronensäure, Oxalsäure, Laktose, Acetat, Ammonium und Glucoamylase. Diese Werte liegen allerdings nur unvollständig vor, d.h. sie sind nicht für jeden der ca. 100 Fermentationsprozesse verfügbar.

Der gesamte Datensatz hat eine Größe von ca. 30 Megabyte.

4.3 Statistische Datenauswertung - Korrelationsanalyse und Hauptkomponentenanalyse

Um sich ein Bild von den Zusammenhängen zwischen den Prozessgrößen zu machen und um Erkenntnisse über diejenigen Prozessparameter zu sammeln, die für die Produktbildung am wichtigsten sind, wurden die Daten mit statistischen Methoden untersucht. Im Vordergrund stand dabei die Frage, welchen Einfluss die Prozessgrößen auf die Produktausbeute am Ende der Fermentation (im Folgenden als EoF = End of Fermentation bezeichnet) bzw. am Ende des Prozesses (Final) und auf die Verluste während des Aufarbeitungsprozesses haben. Einige der Abhängigkeiten der Größen untereinander wurden ebenfalls analysiert.

Datenvorverarbeitung

Wie in Kapitel 4.2 beschrieben wurde, liegen die Daten in Form von zeitabhängigen Messreihen vor, die ca. 1500 bis 2000 (Onlinemessung) bzw. 3 bis 10 (Offlinemessung) Werte umfassen. Um einen Bezug zu zeitunabhängigen Größen, wie z.B. dem EoF-Wert, herzustellen, ist eine Datenvorverarbeitung unerlässlich, die die Rohdaten in eine für die Korrelationsanalyse bzw. die Hauptkomponentenanalyse verwertbare Form bringt. Dazu wurden aus den Messreihen Größen berechnet, welche die wesentlichen Informationen über den Prozess beinhalten. So ist es im Sinne der Korrelationsanalyse unerheblich, welche leichten Schwankungen aufgrund von Sensorstörungen oder Messrauschen in einer Messung, z.B. der CO_2-Konzentration, zu finden sind (siehe Abbildung 23). Der wichtige Informationsgehalt ist bei diesem Beispiel, wie schnell die Konzentration anfangs steigt, welcher Maximalwert erreicht wird und wie stark die Konzentration danach wieder abfällt. Somit ist es nicht sinnvoll mit der kompletten Messreihe von fast 2000 Werten zu rechnen, es genügen wenige Werte, die die charakteristischen Eigenschaften der Messung beinhalten – Steigungen, Maximal- bzw. Minimalwerte, Zeitwerte an besonderen Messpunkten und Mittelwerte.

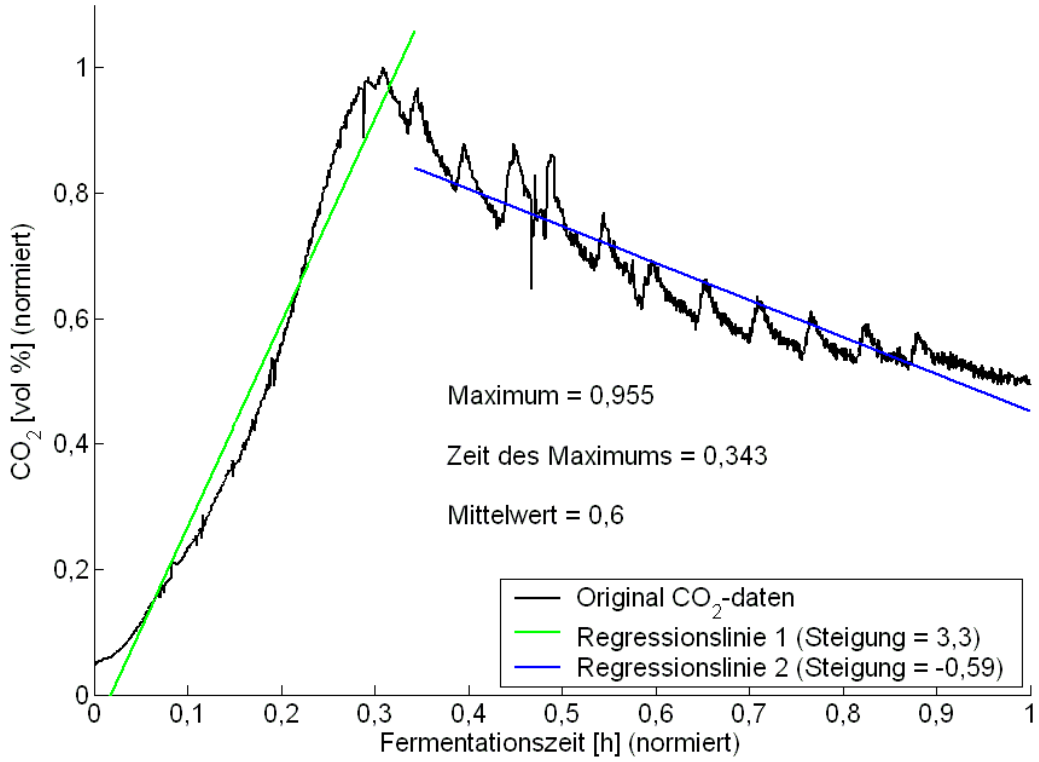

Abbildung 24. Datenvorverarbeitung am Beispiel einer CO_2-Messung

In Abbildung 24 ist die Vorgehensweise bei der Datenvorverarbeitung am Beispiel einer CO_2-Messung veranschaulicht. Aus der Datenreihe wurde zuerst der Mittelwert, der Maximalwert und der Zeitpunkt des Maximums berechnet. Dann wurde die Datenreihe am Maximum in zwei Teile zerschnitten und jeweils eine lineare Regressionsrechnung durchgeführt, aus der zwei weitere Kenngrößen für die CO_2-Messung hervorgehen: Die Steigung vor und nach dem Maximum. Mit dieser einfachen Herangehensweise wurden die CO_2-Messungen aller Prozesse, die ja jeweils eine unterschiedliche Anzahl von Datenpunkten beinhalten (was die weitere Behandlung des Problems erschwert hätte), auf *genau* fünf charakteristische Werte umgerechnet. Mit der Messung der Sauerstoffkonzentration sowie mit den Messwerten für die Sauerstoffaufnahmerate (OUR) wurde in analoger Art und Weise verfahren, da die Messwerte von ihrer Form und ihrem zeitlichen Verlauf denen der CO_2-Messung sehr ähnlich sind (siehe Abbildung 23). Bei Messgrößen wie Druck, Begasung, pH-Wert oder Energieeintrag wurde der Mittelwert als charakteristische Größe berechnet, da diese Messreihen lediglich aus einem konstanten Wert mit überlagertem Rauschen bestehen. Im Fall der Ammoniak- und pO_2-Messungen wurde mit Hilfe der zweiten Ableitung des Reg-

ressionspolynoms vierter Ordnung der Zeitwert des Wendepunktes berechnet. Weitere, als relevant erachtete Größen, die aus dieser Messung berechnet wurden, sind Steigungen sowie Mittelwerte.

Die Messreihe, aus der durch die Datenvorverarbeitung die meisten charakteristischen Größen hervorgingen, ist die Messreihe des Substratfeeds (Zugabe der Maltose). Die Messreihen wurden hinsichtlich ihrer zeitlichen Verläufe und der zugefütterten Substratmenge ausgewertet, sodass sich pro Prozess je 10 Variablen ergeben, die für den betrachteten Prozess das Feedprofil so genau wie möglich wiederspiegeln.

Aus den über 32000 Datenpunkten jedes Prozesses wurden so 63 Werte extrahiert. Die Daten von 66 Fermentationsprozessen wurden verwendet und somit ergab sich eine 66x63 große Datenmatrix M (siehe Abbildung 25), die als Ausgangspunkt für eine Korrelations- bzw. Hauptkomponentenanalyse diente. Die Datenvorverarbeitung wurde mit dem Programm Matlab, Version 6.5.0.180913a (R13), von der Firma The Mathworks Inc. durchgeführt.

	Variable 1	Variable 2	...	...	Variable m
Prozess 1	$M_{1,1}$	$M_{1,2}$	...	...	$M_{1,m}$
Prozess 2	$M_{2,1}$	$M_{2,2}$	...	...	$M_{2,m}$
...		...	...	...	...
...	...	...	...	...	...
Prozess n	$M_{n,1}$	$M_{n,2}$	...	...	$M_{n,m}$

Abbildung 25. In der Datenmatrix M sind die Variablen eines Prozesses zeilenweise angeordnet.

Korrelationsanalyse

Zur Berechnung der Korrelationskoeffizienten wurde der Matlab-Befehl corrcoef eingesetzt, der die Berechnung nach Gleichung {11} ausführt. Der Befehl kann sowohl auf Matrizen als auch auf Vektoren angewendet werden. Durch Anwendung von corrcoef auf die Matrix M entstand die Korrelationsmatrix R, deren Elemente $r_{i,j}$ die Korrelationskoeffizienten der i-ten Variable bezüglich der j-ten Variable darstellen. Diese Art der Anwendung des Befehls führt aber zu einer Matrix R, die zu großen Teilen NaNs enthält. NaN, „Not a number“, wird in der Mathematiksoftware Matlab als Element in Vektoren oder Matrizen verwendet, das fehlende oder nicht berechenbare Werte anzeigt. Diese NaNs entstehen dadurch, dass für einige der 66 verwendeten

Datensätze nicht immer alle Messwerte vollständig vorhanden sind, d.h. das in der Matrix M an der entsprechenden Stelle NaN eingetragen wurde (bzw. werden musste). Einige der in Kapitel 4.2 erwähnten Messgrößen sind davon nicht betroffen wie z.B. die Messung der O_2- und der CO_2-Konzentration. Dies gilt im Wesentlichen auch für die restlichen Onlinemessungen, die ebenfalls immer vorliegen. Die Ergebnisse der Offlinemessungen liegen aber nicht für jeden der 66 Prozesse vor. So liegen z.B. nur von 14 Prozessen Viskositätsdaten vor und nur von 24 Prozessen Messwerte der NH_4^+-Konzentration. Für das Entstehen von NaNs in R ist die Anzahl an fehlenden Messwerten allerdings unerheblich, da bereits bei *einem* nicht vorhandenen Messwert die gesamte Reihe und Spalte der betreffenden Variable in R nicht berechenbar ist. Um das Auftreten von NaNs zu verhindern und für jede Kombination zweier Variablen i und j den Korrelationskoeffizienten $r_{i,j}$ berechnen zu können, wurde das Verfahren zur Berechnung von R modifiziert. Die Korrelationsmatrix R wurde nun nicht mehr in einem Schritt durch Anwendung von corrcoef auf die Matrix M berechnet, sondern die Elemente r_{ij} wurden einzeln berechnet und schließlich zur Matrix R wieder zusammengefügt. Dazu wurde vor jedem Rechenschritt die Matrix M durch Ausschneiden von Zeilen bzw. Spalten, die NaNs bei den entsprechenden Variablen enthalten, verändert. Auf die resultierende Matrix wurde dann der Befehl corrcoef angewendet. Mit dieser Verfahrensweise können alle r_{ij} berechnet werden, allerdings muss beachtet werden, dass die verschiedenen Korrelationskoeffizienten nun aus einer unterschiedlichen Anzahl von Datensätzen (bzw. Prozessen) hervorgegangen sind und dadurch unterschiedlich starke Aussagekraft besitzen.

Die Matrix R wurde tabellarisch dargestellt und zur Vereinfachung der Auswertung eingefärbt. In Tabelle 6 ist ein Ausschnitt der Korrelationsmatrix zu sehen, der die wesentlichen Ergebnisse enthält. Die Hauptfragestellung war, welche Prozessvariablen Zusammenhänge mit der Produktausbeute (EoF) zeigen. Aus den Werten aus Tabelle 6 ist ersichtlich, dass die Ausbeute stark mit der Viskosität zusammenhängt (negative Korrelation). Mit folgenden Parametern ergab sich ein positiver Korrelationskoeffizient: Zitronensäure-Produktion (Korrelation von +0,68 von EoF mit der Variable *Citric acid_max*) und Substratverbrauch (Korrelation von +0,63 von EoF mit der Variable *Maltose Feed_max*). Die Korrelationskoeffizienten der Ausbeute mit den Abgaskonzentrationen (Variable *mtl. O_2* und *mtl. CO_2*) sind sehr schwach, zeigen aber die erwartete Tendenz, dass bei Prozessen mit hoher Ausbeute die

Kohlendioxidkonzentration im Abgas eher hoch ist (hohe Zellaktivität), während die Sauerstoffkonzentration eher niedrig ist. Keine Korrelation ergibt sich mit den Prozessvariablen Temperatur, pH-Wert, Begasung und Ammonium.

Tabelle 6. Auszug aus der Korrelationsmatrix R. (Farbgebung: dunkelgrau wenn 0,7 < |r|, grau wenn 0,6 < |r| < 0,7, hellgrau wenn 0,5 < |r| < 0,6, sonst farblos; m(x) = Steigung von x)

Korrelationsanalyse						
	EoF	mtl. CO2	Feed_max	m(Feed)	Citric_max	mtl. Vis
EoF	1	0,34	0,63	0,35	0,68	-0,68
mtl. O2	-0,27	-0,48	-0,44	-0,38	-0,04	0,26
mtl. CO2	0,34	1	0,67	0,83	-0,06	-0,49
CO2_max	0,21	0,9	0,65	0,71	-0,13	-0,56
NH3_max	0,25	0,55	0,63	0,65	0,18	-0,33
m(NH3)	0,23	0,65	0,6	0,8	0,18	-0,27
mtl. NH3 rate	0,21	0,63	0,58	0,77	0,16	-0,25
mtl. Airflow	-0,09	-0,3	-0,18	-0,05	-0,09	-0,25
mtl. MoF	0,07	-0,21	-0,15	-0,2	0,22	0,14
mtl. pH	0,03	0,17	0,34	0,24	0,23	0,15
mtl. T	0,02	0,39	0,32	0,68	0,22	-0,05
mtl. Stirrer Speed	0,37	0,08	0,26	0,15	0,4	-0,51
mtl. Glucose	-0,54	-0,34	-0,52	-0,49	-0,46	0,42
mtl. Vis	-0,68	-0,49	-0,67	-0,38	-0,39	1
Vis_max	-0,72	-0,48	-0,66	-0,4	-0,54	0,73
DryMass_max	-0,22	-0,4	-0,48	-0,45	-0,13	0,1
Citric acid_max	0,68	-0,06	0,39	0,22	1	-0,39
NH4_max	0,3	-0,08	0,2	0,28	0,61	-0,15
Maltose Feed_max	0,63	0,67	1	0,68	0,39	-0,67
m(Maltose Feed)	0,35	0,83	0,68	1	0,22	-0,38
mtl. SL Maltose	-0,44	-0,78	-0,75	-0,93	-0,3	0,41
mtl. SH Maltose	0,14	0,22	0,08	0,17	0,36	-0,38
Anzahl StMaltose	0,47	0,44	0,82	0,49	0,26	-0,51
t12 StMaltose	-0,31	-0,04	-0,22	0,05	-0,49	0,52

Die Veränderungen der Produktkonzentration, die während des Aufarbeitungsprozesses entstehen (nicht dargestellt in Tabelle 6), korrelieren mit keiner der betrachteten Größen. Somit können die Verluste (bzw. auch Gewinne) an Produkt, die bei der Aufarbeitung entstehen, als vom Prozessverlauf unabhängig betrachtet werden. Eine Auswirkung der Art der Prozessführung auf das Verhalten der Kulturbrühe während der Aufarbeitung ist in den Daten also nicht zu finden.

Weitere Korrelationen, die im Folgenden betrachtet werden, schließen hauptsächlich diejenigen Variablen mit ein, die für die Ausbeute am bedeutsamsten sind. So weisen

z.B. der Substrat-Feed und die Viskosität untereinander hohe Korrelationskoeffizienten auf. Dies zeigt, dass es einen gemeinsamen Faktor gibt, der sich auf diese Größen und auch auf die Ausbeute auswirkt.

Die Gesamtmenge des während der Fermentation zugegebenen Substrats (*Maltose Feed_max*) bzw. die mittlere Geschwindigkeit, mit der das Substrat zugegeben werden konnte (*m(Maltose Feed)*), zeigen Abhängigkeiten zu einigen Parametern des Zufütterungsprofils. Es ergaben sich hohe, negative Korrelationskoeffizienten bezüglich des zeitlichen Abstandes der Pulse (*mtl. SL Maltose*) bzw. positive Korrelationskoeffizienten bezüglich der Anzahl der Pulse (*Anzahl StMaltose*). Die Höhe der Pulse (*mtl. SHMaltose*) hatte dagegen nur einen geringen Einfluss. Größen, die mit der Viskosität korrelieren, sind unter Anderem die CO_2-Konzentration im Abgas, die Rührgeschwindigkeit und zwei Parameter der Substratzufütterung: Zum einen ist dies die Anzahl an Feedpulsen und zum Anderen die Zeit, die zwischen dem ersten und dem zweiten Puls verstreicht. Eine naheliegende Vermutung für das Zustandekommen der letztgenannten Korrelationen ist, dass beim Feedvorgang neben dem Substrat selbst natürlich auch eine große Menge Wasser mit hinzugegeben wird. Dadurch wird die Kulturbrühe verdünnt und die Viskosität sinkt. Insgesamt ergibt sich durch häufiges Feeden mit einer eher großen Anzahl an Pulsen eine günstige Viskositätsentwicklung während des Prozesses. Besonders wichtig scheint dieser Effekt zu Beginn des Prozesses zu sein, wie der Korrelationskoeffizient von 0,52 bei den Variablen *mtl. Vis* und *t12Maltose* zeigt. Bei denjenigen Prozessen, bei denen der Zeitabstand zwischen dem ersten und zweiten Puls besonders lang war, ergab sich also im Mittel eine hohe Viskosität mit ungünstigen Folgen für die Ausbeute.

Hauptkomponentenanalyse

Die Zielsetzung beim Einsatz der Hauptkomponentenanalyse ist identisch mit derjenigen von der Korrelationsanalyse: Auffinden der wichtigsten Einflussgrößen auf die Produktkonzentration am Ende der Fermentation (EoF).

Die in Matlab erstellte Datenmatrix M wurde in die Software *The Unscrambler* (Version 7.6, CAMO ASA, Norwegen) importiert, mit der die Hauptkomponentenanalyse durchgeführt wurde. Als Gewichtung wurde die im

Unscrambler als „Standardization“ bezeichnete Methode gewählt, bei der jede Variable durch ihre Standardabweichung geteilt wird, sodass alle Variablen danach eine Varianz von 1 besitzen und mittelwertzentriert sind. Diese Art der Gewichtung wurde gewählt, da die Variablen stark unterschiedliche Wertebereiche und Einheiten haben. Andere Gewichtungen wie z.B. über einen konstanten Faktor oder auch das Weglassen einer Gewichtung sind bei dieser Struktur der Daten nicht sinnvoll [UNSCRAMBLER].

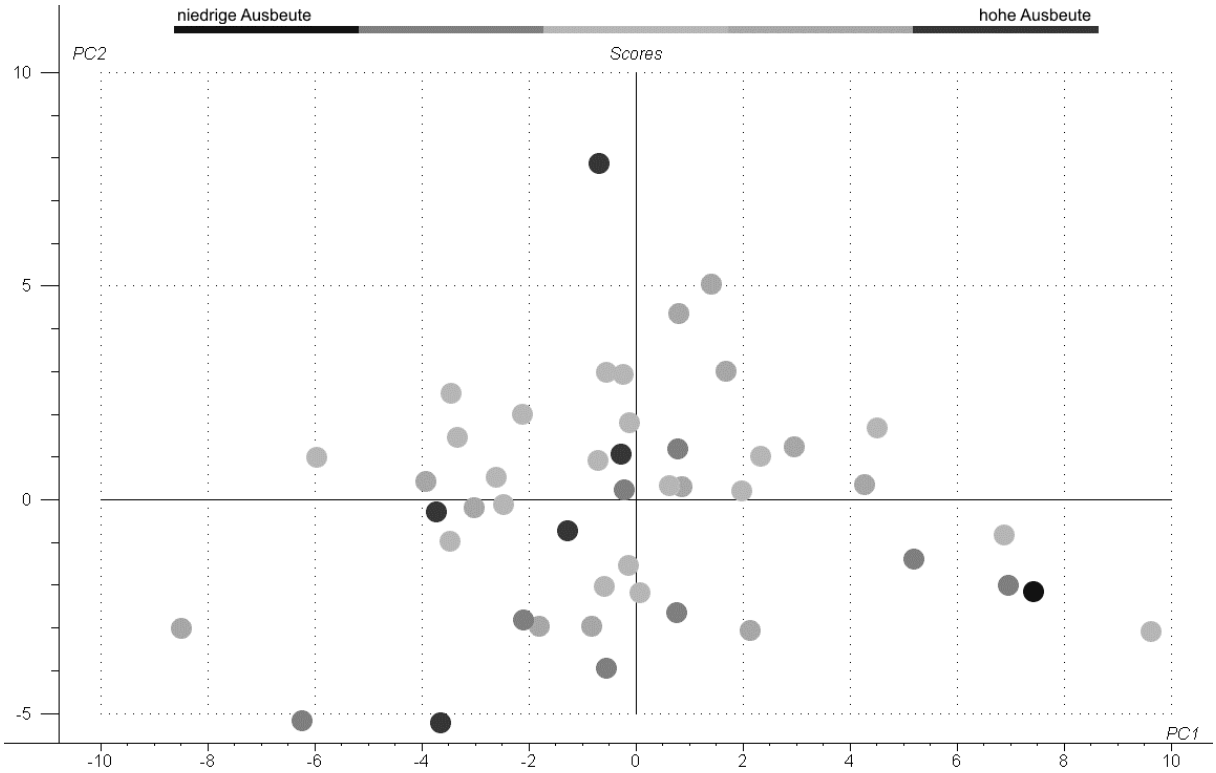

Abbildung 26. Ergebnis der PCA: Scoreplot t_1t_2 der ersten und zweiten Hauptkomponente. Jeder Punkt stellt die Abbildung eines Fermentationsprozesses in den Raum der ersten beiden Hauptkomponenten dar. Die Farbgebung bezieht sich auf die Ausbeute am Ende der Fermentation (EoF).

Wie in Kapitel 2.3 ausgeführt, lassen sich im Scoreplot (insbesondere im Scoreplot t_1t_2, also erste gegen zweite Hauptkomponente) Gemeinsamkeiten und Beziehungen zwischen den Objekten ablesen. Ähnliche Objekte werden im Scoreplot nahe beieinander stehen, während voneinander verschiedene Objekte weit voneinander entfernt sind. Sofern im Datensatz voneinander unterscheidbare Gruppen von Objekten vorhanden sind, lässt sich eine Aufteilung in mehrere Gruppen bzw. Cluster im Scoreplot beobachten. Im Optimalfall würde dies bei dem vorliegenden Datensatz eine Aufspaltung der verschiedenen Prozesse bezüglich ihrer Ausbeute oder einer anderen

wichtigen Prozessgröße bedeuten. D.h. Prozesse mit hoher Ausbeute und Prozesse mit geringer Ausbeute sollten sich im Scoreplot in ihrer räumlichen Anordnung deutlich unterscheiden lassen. Bei der durchgeführten PCA ist ein solches Verhalten nicht erkennbar (siehe Abbildung 26). Zur Verdeutlichung wurden die Scores hinsichtlich der entsprechenden Ausbeute farblich markiert. Diejenigen Prozesse mit ähnlicher Ausbeute sind jeweils relativ gleichmäßig über den Plot verteilt.

Auch durch Auswertung der Scoreplots t_1t_3, t_1t_4 usw. und der Loadingplots konnte kein entscheidender Fortschritt für das Auffinden der wichtigsten Einflussgrößen auf die Ausbeute erzielt werden. Aus diesem Grund wurde der Datensatz erneut ausgewertet, diesmal mit Hilfe der PLS1-Methode. Wie oben angedeutet, bestand das Hauptziel bei Anwendung der Hauptkomponentenanalyse darin, Abhängigkeiten der Ausbeute (EoF) zu den Prozessgrößen zu finden. Allgemein formuliert sollten somit Zusammenhänge zwischen einem Datensatz y (der in diesem Fall nur aus einem Vektor besteht: den Ausbeutewerten der Prozesse) und einem Datensatz x (hier: restliche Daten aus der Matrix M) hergestellt werden. Dazu eignet sich die PLS1-Methode viel besser als die PCA, da hier bei der Berechnung der Hauptkomponenten für die x-Variablen die Varianz der y-Variablen stark mit einfließt. Die Hauptkomponenten werden dabei so gewählt, dass die y-Variablen sich so gut wie möglich aus den x-Variablen beschreiben lassen. Der Vorteil der PLS1-Methode in diesem Anwendungsfall ist, dass sich auch in einem komplexen Datensatz Korrelationen zwischen den x-Variablen und der Zielgröße, hier der Ausbeute, finden lassen. Weitere Eigenschaften der PLS1-Methode, eine ausführlichere Darstellung der Unterschiede zur PCA sowie die Modifikation des NIPALS-Algorithmus für die PLS1-Rechnung lassen sich in der Literatur finden [ESBENSEN].

Die PLS1-Rechnung wurde ebenfalls mit dem Programm *The Unscrambler* durchgeführt. In Abbildung 27 ist der Scoreplot der ersten und zweiten Hauptkomponente dargestellt. Prozesse, die eine hohe Ausbeute hatten, sind orange bzw. rot eingefärbt und Prozesse mit niedriger Ausbeute sind dunkelgrün bzw. blau dargestellt. Es ist leicht zu erkennen, dass sich Prozesse mit hoher Ausbeute im rechten oberen Quadranten des Graphs gruppiert haben, während Prozesse mit niedriger Ausbeute tendenziell eher links unten in der Grafik zu finden sind. Die Auftrennung in „gute und schlechte“ Prozesse ist immer noch nicht optimal, aber gegenüber der PCA wesentlich verbessert.

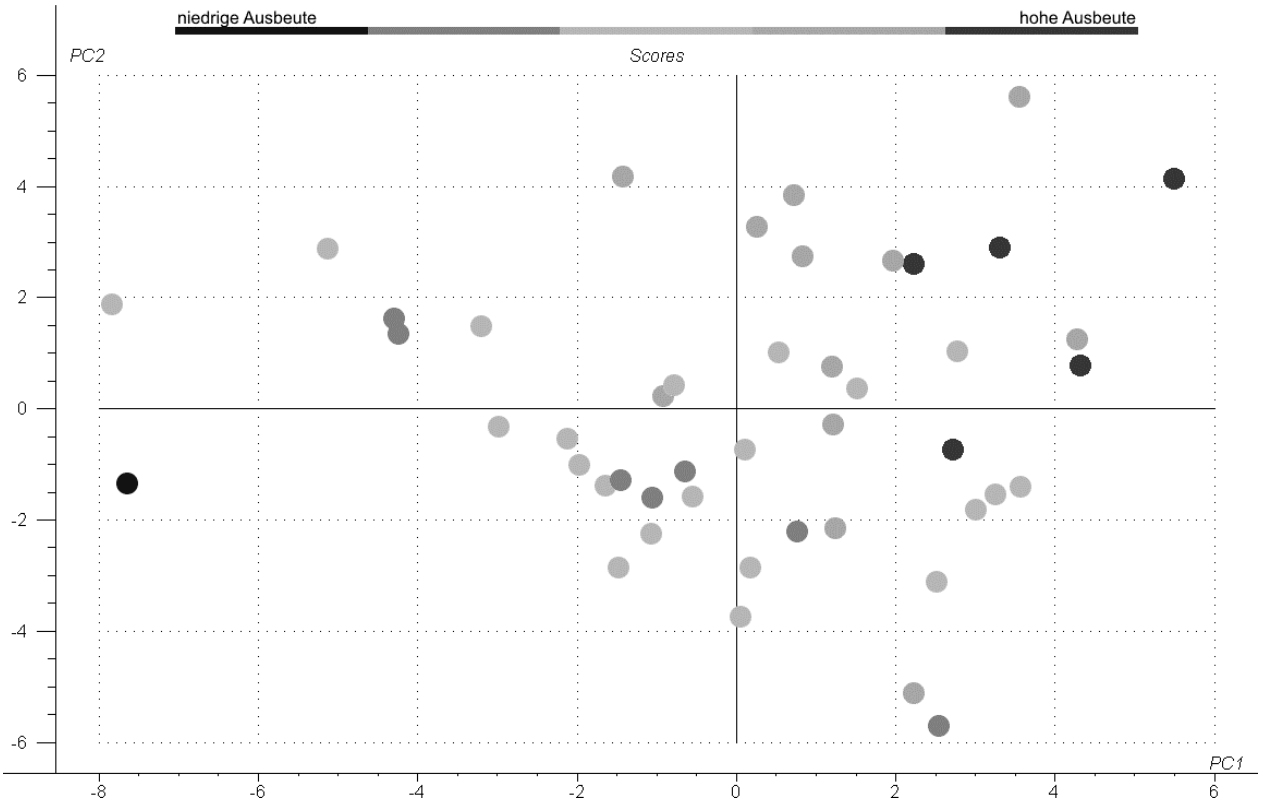

Abbildung 27. Ergebnis der PLS1-Methode: Scoreplot der ersten und zweiten Hauptkomponente. Jeder Punkt stellt die Abbildung eines Fermentationsprozesses in den Raum der ersten beiden Hauptkomponenten dar. Die Farbgebung bezieht sich auf die Ausbeute am Ende der Fermentation (EoF).

Die Variablen, die diese Auftrennung verursachen, bzw. die „in den Hauptkomponenten stecken“, lassen sich mit Hilfe des Loadingplots (siehe Abbildung 28) ermitteln. Eine positive Verbindung (Korrelation) zur Ausbeute besteht mit den Variablen *citric_max* (Maximalwert der Zitronensäurekonzentration), *m(citric)* (Steigung der Zitronensäurekonzentration), *MaltoseFeed_max* (Zugegebene Menge an Substrat) und *Anzahl_StMaltose* (Anzahl Pulse im Substratfeed). Eine negative Verbindung besteht besonders zu den beiden Viskositätsvariablen im linken unteren Quadranten (Variablen, die auf gegenüberliegenden Seiten des Ursprungs und ungefähr auf einer Linie liegen, weisen eine negative Korrelation auf, s. Kap. 2.3). Weitere Größen, die ebenfalls einen negativen, aber schwächeren Einfluss auf die Ausbeute haben, sind der mittlere pO_2 (*mtl. pO_2*), die Zeit zwischen den ersten beiden Substratpulsen (*t12STMaltose*), der Zeitpunkt des CO_2-Maximums (*t(CO_2_max)*) und der Zeitpunkt des NH_3-Maximums (*t(NH_3Rate_max)*). Keine, oder nur geringe Verbindung zur Ausbeute haben alle Variablen, die sich dicht am Ursprung des Loadingplots befinden.

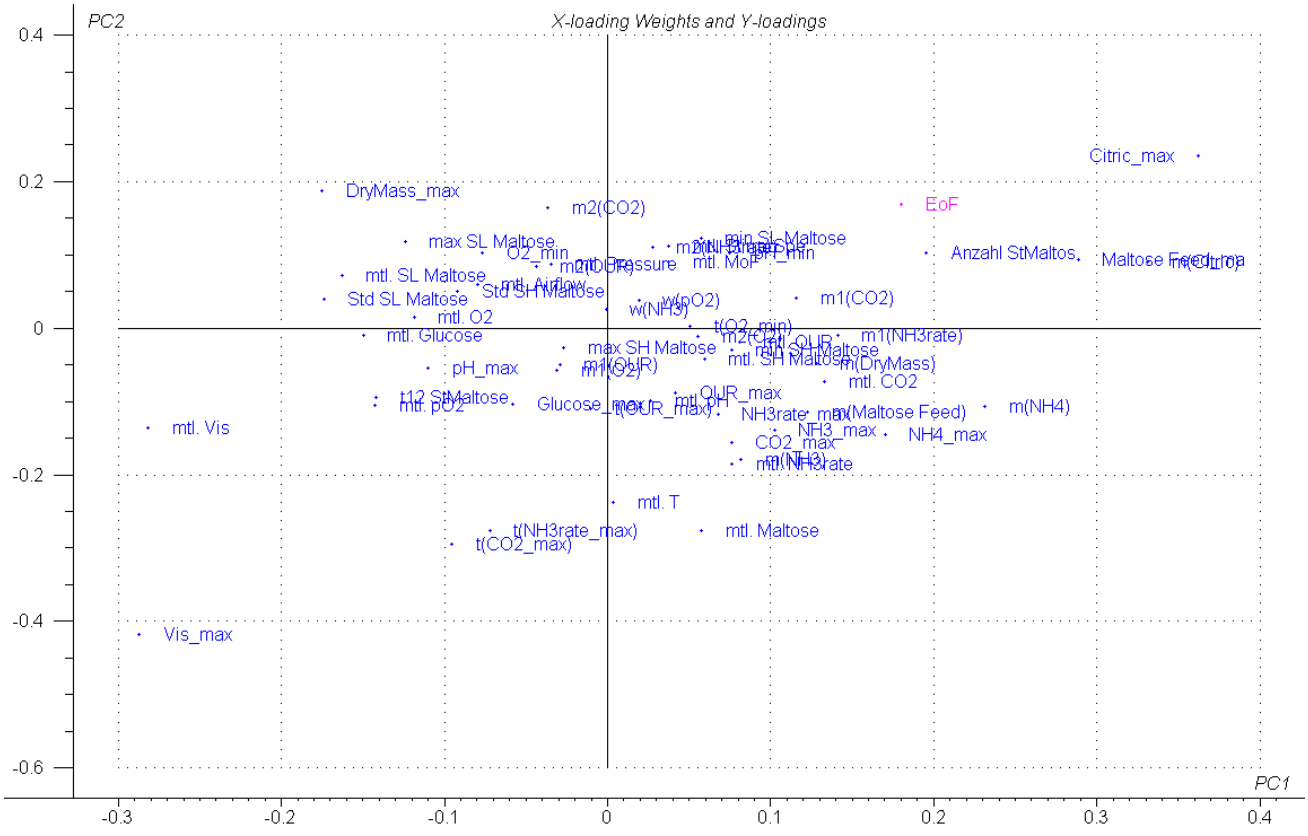

Abbildung 28. Ergebnis der PLS1-Methode: Loadingplot p_1p_2.

Eine Auswertung der Loadingplots höherer Hauptkomponenten ergab keine verwertbaren Informationen über die Ausbeute. Analog zur PLS1-Rechnung mit der Zielgröße EoF wurde die PLS1-Rechnung auch für die Produktverluste während der Aufarbeitung durchgeführt. Ebenso wie bei der Korrelationsanalyse konnten keine Abhängigkeiten der Verluste mit den Prozessvariablen ermittelt werden.

Vergleich der Analysemethoden und Zusammenfassung

Beide Analysemethoden ergeben im Wesentlichen die gleichen Ergebnisse. Die wichtigsten Einflussgrößen, die positiv mit der Ausbeute korrelieren, sind:

- **Substratfeed**
- **Zitronensäure** (Maximalkonzentration und Steigung der Konz.)
- Anzahl Feedpulse

Prozesse, bei denen viel Maltose zugegeben wurde, bei denen viel Zitronensäure produziert wurde und bei denen das Substrat mit vielen Pulsen zugegeben wurde, hatten im Mittel also hohe Ausbeuten. Die wichtigsten Einflussgrößen, die negativ mit der Ausbeute korrelieren, sind:

- **Viskosität** (mittlere und maximale gleichermaßen)
- Zeitspanne zwischen erstem und zweitem Feedpuls
- Zeitpunkte von CO_2 und NH_3-Zugabe Maxima

Prozesse, bei denen die Viskosität hoch war, bei denen zwischen dem ersten und zweiten Feedpuls besonders viel Zeit verstrichen ist, und Prozesse, bei denen die Maxima von CO_2-Konzentration bzw. NH_3-Zugabe besonders spät im Prozessverlauf aufgetreten sind, hatten im Durchschnitt somit eine geringe Ausbeute.
Bei der Auswertung von Korrelationen und Abhängigkeiten zwischen Variablen ist es auch wichtig zu bedenken, ob die gefundenen Einflussgrößen direkt und von sich aus ihre Wirkung ausüben oder ob sie selbst nur Folgen einer übergeordneten Ursache sind. Bei den hier gefundenen Einflussgrößen ist die übergeordnete Verbindung zwischen ihnen mit großer Sicherheit das Zellwachstum bzw. die Aktivität der Organismen. Dies ist eine sehr naheliegende Schlussfolgerung, da bei raschem Zellwachstum besonders viel Substrat zugegeben werden kann, die Zitronensäurekonzentration schnell ansteigt und hohe Maximalwerte erreicht (Produkt- und Zitronensäurebildung sind bei *Aspergillus niger* häufig gekoppelt). Auch die Zeitpunkte der Maxima von CO_2-Konzentration bzw. NH_3-Zugabe sind mit der Zellaktivität gekoppelt. Sind die Organismen besonders aktiv, erfolgt der Wechsel von Wachstums- zu Produktbildungsphase früh im Prozess, mit der Folge, dass der Zeitwert des Maximums der CO_2-Konzentration eher klein ist und die Ausbeute eher hoch ist.
Die anderen drei Größen, die großen Einfluss auf die Ausbeute haben, sind die Viskosität und die Anzahl der Pulse im Substratfeed sowie die Zeitspanne zwischen dem ersten und dem zweiten Feedpuls. Der Einfluss der Viskosität auf Pilzfermentationen ist seit langem bekannt, da sich eine hohe Viskosität nachteilig auf die Vermischung sowie den Gasaustausch in der Fermentationsbrühe auswirkt. Die beiden anderen Variablen, beides Parameter der Substratzufütterung, haben ihrerseits Auswirkungen auf die Viskosität (siehe Tabelle 6, rechts unten), da durch die Substratzugabe immer auch eine Verdünnung der Fermentationsbrühe erfolgt.

Im Hinblick auf eine Optimierung des Prozesses lassen sich die gefundenen Einflüsse Menge des Substratfeeds, Zitronensäureproduktion und Viskosität nicht ausnutzen, da sie keine Größen darstellen, die sich direkt regeln lassen. Sie sind selbst nur Folgen anderer Faktoren und somit nicht direkt beeinflussbar. Wichtiger sind in diesem Zusammenhang die Ergebnisse bezüglich der Parameter der Substratzufütterung, durch die eine Einflussnahme auf die Viskosität und damit auf einen für die Produktbildung entscheidenden Faktor möglich ist. Durch frühere Substratzugabe wird dem Prozess auch Wasser hinzugefügt und eine Verdünnung der Fermentationsbrühe kann erreicht werden, wodurch die Viskosität gleich zu Beginn des Prozesses niedrig gehalten wird, mit voraussichtlich positiven Auswirkungen auf die Ausbeute.

4.4 Einsatz neuronaler Netze

Die Zielsetzung bei der Erstellung und dem Einsatz neuronaler Netze ist es, ein Prozessmodell zu generieren, das in der Lage ist, die Produktkonzentration aus den Online-Messwerten vorherzusagen. Ausgehend von Messdaten vom Fermentationszeitpunkt t soll das Modell die vorrausichtliche Produktkonzentration zum Zeitpunkt t+12 h berechnen können. Des weiteren soll durch das neuronale Netz eine Prognose der Endausbeute erstellt werden. Das neuronale Netz muss dafür auf eine Weise implementiert werden, dass eine Produktvorhersage am laufenden Prozess möglich ist. Es muss also mit dem Prozessleitsystem verbunden werden können, um die Online-Messdaten zu empfangen.

Darüber hinaus soll das erstellte Modell zur Prozessoptimierung und zur Risikoabschätzung der möglichen Veränderungen in der Prozessführung herangezogen werden.

Um diese Aufgabenstellung zu erfüllen, wurde in der Programmiersprache Matlab ein Programmpaket erstellt, mit dem man die einzelnen Schritte von Erzeugung des Trainingsdatensatzes über Erstellung der neuronalen Netze bis hin zur Prozesssimulation durchführen kann. Diese Funktionen werden dem Benutzer durch mehrere GUIs (Graphical User Interface) zur Verfügung gestellt, sodass zum Einsatz

des Programms keine Programmierkenntnisse erforderlich sind. Die Struktur des Programms ist in Abbildung 29 gezeigt.

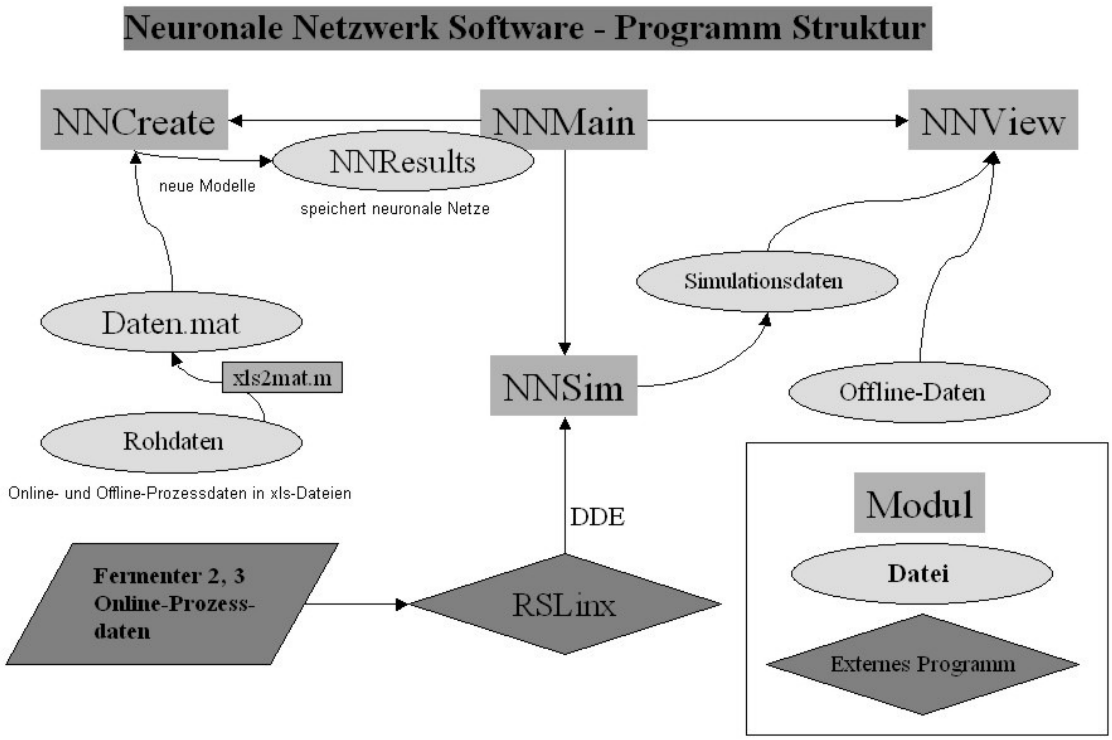

Abbildung 29. Struktur des Software-Systems, das zur Erstellung und Anwendung von neuronalen Netzen in Matlab implementiert wurde.

Es gliedert sich in das Hauptprogramm NNMain, das Modul NNCreate, mit dem neuronale Netze trainiert werden können, das Modul NNSim, mit dem Prozesssimulationen durchgeführt werden können, sowie das Modul NNView, mit dem die Simulationsergebnisse visualisiert und ausgewertet werden können. Das Hauptprogramm NNMain besteht aus einer sortierbaren Liste mit neuronalen Netzen, die zur Simulation zur Verfügung stehen. Unterhalb der Liste befinden sich Pushbuttons, mit denen die anderen Programmmodule NNCreate, NNSim oder NNView aufgerufen werden können oder mit denen weitere Funktionen des Hauptprogramms wie Löschen oder Anzeigen der Trainingsresultate bestehender neuronaler Netze möglich ist. Ein Screenshot des Programms NNmain ist in Abbildung 30 zu sehen. Die einzelnen Module werden in den folgenden Kapiteln detailliert vorgestellt.

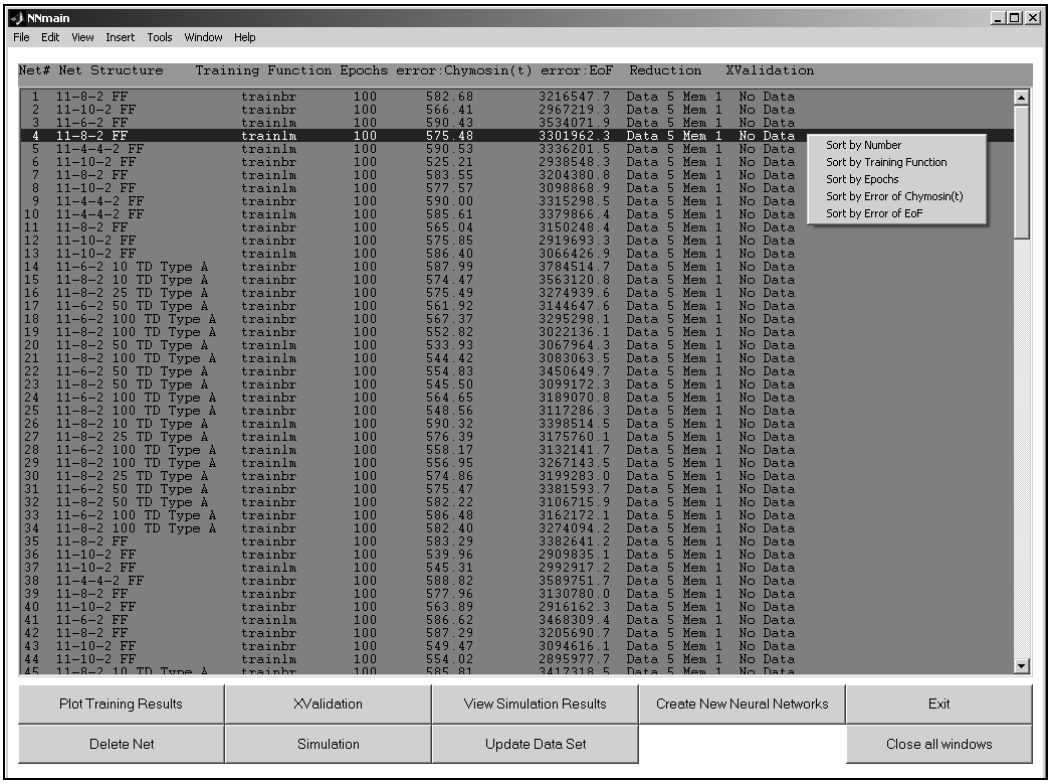

Abbildung 30. Screenshot des Hauptprogramms NNMain der Neuronale Netze Software.

4.4.1 Erstellung des Trainingsdatensatzes

Zur Erstellung neuronaler Netze ist ein Trainingsdatensatz notwendig. Dieser wurde mit Hilfe eines in der Programmiersprache Matlab geschriebenen Konvertierungsalgorithmus[4] aus den vorhandenen Datensätzen des Fermentationsprozesses (siehe Kapitel 4.2) erzeugt. Die als einzelne Exceldateien vorliegende Messreihen wurden so zu einem Datensatz im mat-Format (Datenformat der Programmiersprache Matlab) zusammengefügt. Aus diesem Datensatz wurden dann die Eingabevariable p und die Zielgrößenvariable[5] t gebildet. p enthält sämtliche Eingangsmuster aller betrachteten Prozesse, d.h. die Fermentationszeit, O_2 und CO_2-Konzentration im Abgas, Daten der Substrat- und Ammoniakzufütterung, Begasungsrate, Druck, pH-Wert, Temperatur und Rührgeschwindigkeit. Die Variable t enthält die entsprechenden Zielwerte, die das Netz aus den Eingaben in p errechnen soll:

[4] xls2mat.m in Abbildung 29
[5] Anmerkung: Der Datentyp der Variablen p und t in Matlab ist *cell array*. In diesem Fall sind es Vektoren, deren Elemente Matrizen sind.

Die Produktkonzentration als Funktion der aktuellen Zeit+12 h sowie die Endausbeute. Zur Erzeugung der Eingabe p wurden die Daten auf den Wertebereich –1 bis +1 normiert und in die in Gleichung {49} dargestellte Form gebracht. Die Variable p besteht aus TS 11xQ-Matrizen, wobei TS die Anzahl der Datenpunkte in den Messreihen ist, Q ist die Anzahl der Fermentationsprozesse im gesamten Datensatz und 11 ist die Anzahl der für den Trainingsdatensatz verwendeten Messreihen. Die Elemente innerhalb der 11xQ-Matrizen sind Spaltenvektoren, die aus 11 Werten bestehen: Den Messwerten des entsprechenden Prozesses zum jeweiligen Zeitschritt (siehe Gleichung {50}). Die Zielgrößenvariable t besteht aus TS 2xQ-Matrizen, da das Netz zwei Ausgabegrößen hat. Sie ist in Gleichung {51} gezeigt. $C_n(m)$ ist dabei die Produktkonzentration des n-ten Prozesses zum Zeitschritt m. Dabei ist zu Bedenken, dass das fertige Netz die Produktkonzentration zwölf Stunden im Voraus vorhersagen soll, d.h. in t ist diejenige Produktkonzentration einzusetzen, die der entsprechenden Eingabe in p um zwölf Stunden voraus ist. Daraus ergibt sich, dass die Produktkonzentrationen die in t am Ende eingetragen werden müssten, nicht existieren (ist ein Fermentationsprozess z.B. nach 120 Stunden abgeschlossen, existieren keine Konzentrationswerte bei Stunde 132). Da für das Training neuronaler Netze in Matlab die Variablen p und t die gleiche Länge haben müssen, wurden entsprechende Stellen in t mit dem letzen bekannten Wert der Produktkonzentration bis zum Ende aufgefüllt. Der zweite Zielwert in t ist die Produktausbeute am Ende des Prozesses (EoF). Da dieser Wert zeitunabhängig ist, ergibt sich in jeder Matrix in t der gleiche Zeilenvektor für die EoF-Werte (siehe Gleichung {51}). Auch die Zielwerte sind einer Normierung auf den Wertebereich –1 bis +1 unterzogen worden. Zur Beschleunigung des Trainingsvorgangs bzw. um diesen überhaupt zu ermöglichen, wurde nicht jeder sondern jeder fünfte Wert aus den ursprünglichen Messreihen zum Aufbau der Variablen p und t verwendet. Bei Verwendung von Eingabemustern p, die aus dem kompletten Datensatz hervorgingen, brach der Trainingsvorgang mit einer Fehlermeldung ab. Die Reduktion des Datensatzes ist ohne weiteres möglich, da die große zeitliche Auflösung bei den Onlinedaten nicht benötigt wird.

$$\mathrm{p} = \begin{Bmatrix} [p_1(1), p_2(1), ..., p_Q(1)], [p_1(2), p_2(2), ..., p_Q(2)], ..., \\ [p_1(TS), p_2(TS), ..., p_Q(TS)] \end{Bmatrix} \quad \{49\}$$

$$\mathrm{p}_1(1) = [Zeit_1(1), O_{21}(1), CO_{21}(1), Substrat_1(1), NH_{31}(1), NH_3rate_1(1), Luftdurchfluss_1(1), Druck_1(1), pH_1(1), Temperatur_1(1), Rührgeschwindigkeit_1(1)]' \quad \{50\}$$

$$\mathrm{t} = \begin{Bmatrix} \begin{bmatrix} C_1(1), C_2(1), ..., C_Q(1) \\ EoF_1, EoF_2, ..., EoF_Q \end{bmatrix}, \begin{bmatrix} C_1(2), C_2(2), ..., C_Q(2) \\ EoF_1, EoF_2, ..., EoF_Q \end{bmatrix}, ..., \\ \begin{bmatrix} C_1(TS), C_2(TS), ..., C_Q(TS) \\ EoF_1, EoF_2, ..., EoF_Q \end{bmatrix} \end{Bmatrix} \quad \{51\}$$

Das Erstellen bzw. Aktualisieren des Datensatzes kann von NNMain aus mit „Update Data Set“ ausgelöst werden (siehe Abbildung 30 unten Mitte).

4.4.2 Erstellung und Training neuronaler Netze - NNCreate

Aufgrund der Struktur des Trainingsdatensatzes und der Aufgabenstellung allein ist eine Entscheidung, welche Art von Netz, Transfer- und Trainingsfunktion für die Bewältigung der Aufgabe am Besten geeignet ist, oftmals nicht möglich. Daher wurden vor der Erstellung des Programmmoduls NNCreate, das zum Erstellen und Trainieren neuronaler Netze dient, Voruntersuchungen durchgeführt. Das Ziel bestand darin, aus der Vielzahl an möglichen Netzstrukturen, Transfer- und Trainingsfunktionen diejenigen ausfindig zu machen, die für das gegebene Problem geeignet sind bzw. diejenigen auszuschließen, die ungeeignet sind. Im Laufe dieser Tests wurden folgende Netzparameter variiert:

- Schichtanzahl (drei- oder vierschichtige Netze)

- Neuronenanzahl (zwei bis 12 Neuronen pro Schicht)
- Transferfunktion der versteckten Schicht(en)
- Epochenzahl (Anzahl Iterationsschritte beim Training)
- Trainings- bzw. Lernfunktion
- Netze ohne und mit zeitverzögerten Feedback-Schleifen

Bei allen Netzen handelt es sich um Feed-Forward-Netze, die mit der Matlab-Funktion newff oder newfftd erschaffen wurden und die 11 Neuronen in der Eingabe- und 2 Neuronen in der Ausgabeschicht haben. Weiterhin hat bei allen Netzen die Ausgabeschicht eine lineare Transferfunktion (die Identität). Beim Trainingsverfahren handelte es sich immer um überwachtes Lernen im Batch-Verfahren, d.h. die Änderung der Wichtungsparameter wurde erst durchführt, nachdem alle Eingabemuster dem Netz präsentiert wurden. Die Bewertung der Güte der Vorhersage des Netzes geschah über den mittleren quadratischen Fehler mit der Matlab-Funktion mse (mean square error).
Es wurden eine große Anzahl an Netzen, bei denen die oben genannten Parameter systematisch variiert wurden, mit dem in Kapitel 4.4.1 beschriebenen Datensatz trainiert und bezüglich des Trainingsfehlers ausgewertet. Dabei zeigte sich, dass dreischichtige Netze mit weniger als 6 Neuronen in der versteckten Schicht sowie vierschichtige Netze mit insgesamt weniger als 8 Neuronen in den versteckten Schichten sehr schlechte Trainingsresultate aufwiesen. Das Training mit den Lernalgorithmen trainscg (scaled conjugate gradient algorithm), trainbfg (quasi-Newton Methode nach Broyden, Fletcher und Goldfarb) und trainrp (resilient backpropagation algorithm, RProp) dauerte sehr lange und führte ebenfalls zu schlechten Resultaten. Die Trainingsalgorithmen mit den besten Ergebnissen bezüglich schnellem Konvergenzverhalten und niedrigen Trainingsfehlern waren der traimln (Levenberg-Marquardt-Verfahren) und der trainbr (Bayesian Regularization) Algorithmus[6]. Bei den verwendeten Transferfunktionen zeigte sich, dass die Kombination von sigmoiden Transferfunktionen in den versteckten Schichten (tangens hyperbolicus) und linearer Transferfunktion in der Ausgabeschicht zu den geringsten Trainingsfehlern führte. Der Einsatz anderer Transferfunktionen wie z.B. der radial-basis Funktion (siehe Abbildung 7) erwies sich als nicht erfolgreich.
Basierend auf den Erkenntnissen dieser Voruntersuchung wurden im Modul NNCreate nur Netze mit folgenden Merkmalen berücksichtigt:

[6] Eine detaillierte Ausführung zu den genannten Algorithmen findet sich in der Literatur [DOKUNNT].

- dreischichtige Netze mit mindestens 6 und vierschichtige Netze mit insgesamt mindestens 8 Neuronen
- eine sigmoide Transferfunktion in der/den versteckten Schicht/en und eine lineare Transferfunktion in der Ausgabeschicht
- als Trainingsalgorithmus wurde trainlm und trainbr mit 100 Trainingsepochen verwendet
- es wurden Netze ohne und mit zeitverzögerten Feedback-Schleifen verwendet

Wird das Modul NNCreate aufgerufen, so werden ca. 250 Netze erzeugt, trainiert und der in Abbildung 30 gezeigten Liste der vorhandenen Modelle in NNMain hinzugefügt. Bei diesen Netzen werden die oben genannten Parameter systematisch variiert, d.h. die Hälfte der Netze wird mit der Trainingsfunktion trainlm und die andere Hälfte mit der Trainingsfunktion trainbr trainiert. Die Neuronenzahl wird von 6 bis 12 Neuronen pro Schicht variiert und es werden Netze ohne und mit Feedback-Schleife erzeugt.
Das Hinzunehmen von Netzen mit zeitverzögerten Feedback-Schleifen ist durch folgenden Aspekt der Aufgabenstellung motiviert: Bei der Vorhersage der Produktkonzentration steht zwar die korrekte Vorhersage der Endausbeute im Vordergrund, für eine Einflussnahme auf die Prozessführung ist aber der zeitliche Verlauf, bzw. die Krümmung der Konzentrationskurve von entscheidender Bedeutung. Das liegt daran, dass die Konzentrationskurve bei einigen der bisherigen Prozessdurchgänge gegen Ende der Fermentation stark abgeknickt ist und sehr flach verlaufen ist. Die Steigerung der Produktkonzentration pro Zeit war also sehr gering. Wesentliche Kostenfaktoren des Prozesses wie Strom für das Rührwerk, Wasserverbrauch, Energieaufwand für Heizen/Kühlen etc. liefen aber weiter, sodass es sinnvoll wäre den Prozess frühzeitiger als geplant abzubrechen und damit die zeitliche Ausbeute zu steigern. Um den optimalen Zeitpunkt zum Beenden der Fermentation zu finden, muss der Verlauf der Produktkonzentration durch das Modell korrekt vorhergesagt werden. Daher erscheint es sinnvoll, dem Netz nicht nur Informationen über den momentanen Zustand der Prozessgrößen zuzuführen (wie es bei Netzen ohne Feedback-Schleifen der Fall ist) sondern auch die „Entwicklung“ der Prozessgrößen über die Zeit mit einfließen zu lassen. Bei den zeitverzögerten Feedback-Schleifen, wie sie im Modul NNCreate implementiert wurden, wird ein Eingabevektor, der gegenüber dem aktuellen Eingabevektor X Zeitschritte zurückliegt, dem Netz zusätzlich zur aktuellen Eingabe

präsentiert. Für diese Art von Netzen ergab sich somit auch ein neuer Parameter, der ebenfalls variiert wurde: Die Anzahl der Verzögerungs-Zeitschritte X, für die Werte von 10, 25, 50 und 100 verwendet wurden. Bei den Netzen mit Feedback-Schleifen wurden einerseits Schleifen, die von der versteckten Schicht zur Eingabeschicht zurückführen (Typ A) und Schleifen, die von der Ausgabeschicht zur Eingabeschicht zurückführen verwendet (Typ B). Eine schematische Darstellung der neuronalen Netze ohne Schleifen sowie der Netze vom Typ A und B findet sich in Abbildung 31.

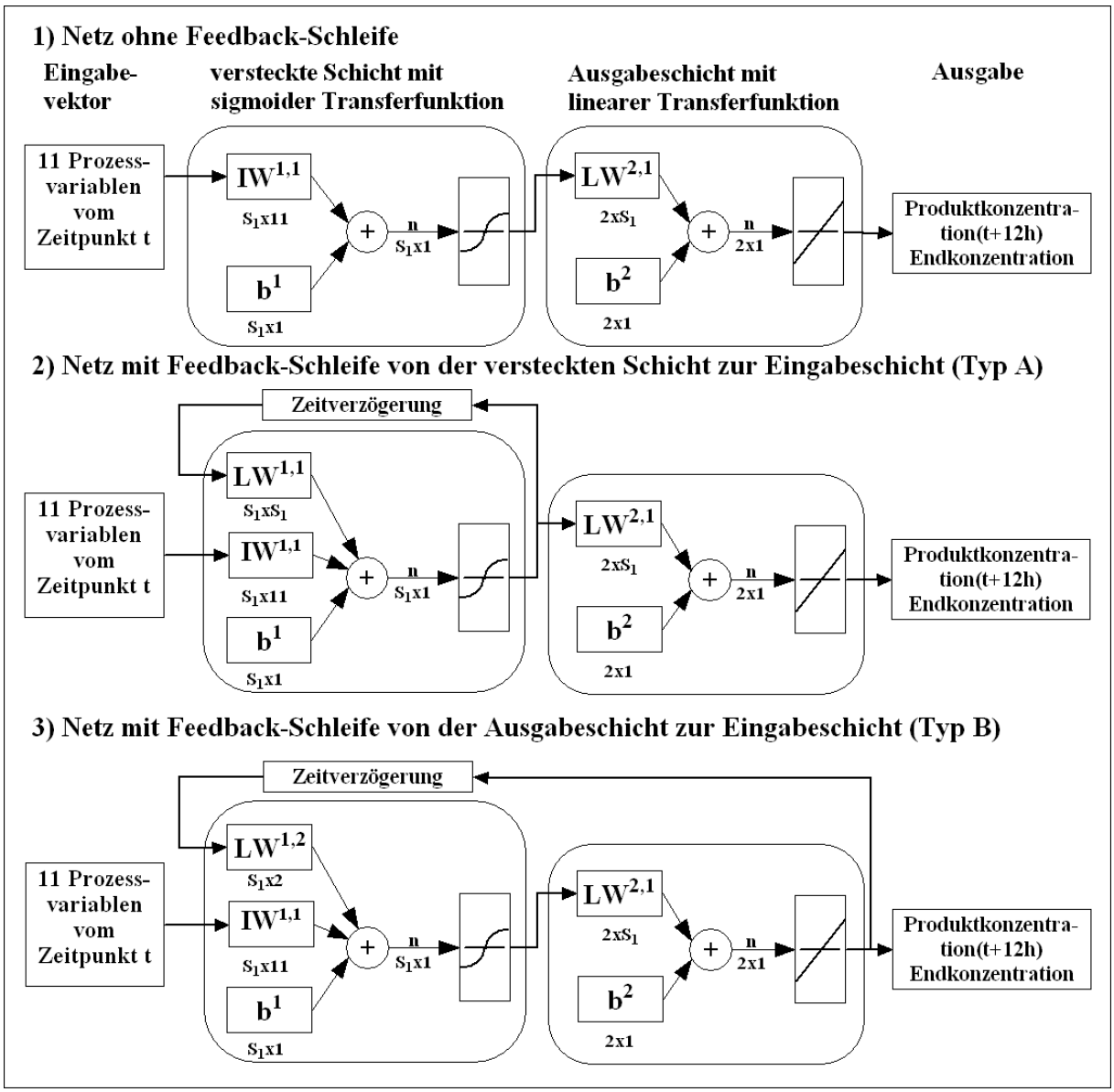

Abbildung 31. Schematische Darstellung der verwendeten Netztypen. Zur Erläuterung der Abkürzungen siehe Anhang 8.1.

Die durch NNCreate erzeugten Netze werden dann in der in Abbildung 30 gezeigten Liste dargestellt und sind nach der Größe ihres Fehlers bezüglich der Vorhersage der Produktkonzentration und der Endausbeute sortierbar. Die Trainingsergebnisse lassen sich dann in NNMain für jedes Netz und für jeden im Trainingsdatensatz verwendeten Prozess auswerten. In Abbildung 32 ist ein Auszug der Trainingsergebnisse eines 11-8-2 Netzes mit Feedback-Schleife (Typ B) gezeigt. In der Grafik oben links sind die Zielwerte (rot) und die Vorhersage des Netzes (blau) der Produktkonzentration für einen der im Trainingsdatensatz vorhandenen Prozesse dargestellt. Sowohl der Verlauf als auch der Endwert der Produktkonzentration wird für diesen Prozess sehr gut wiedergegeben.

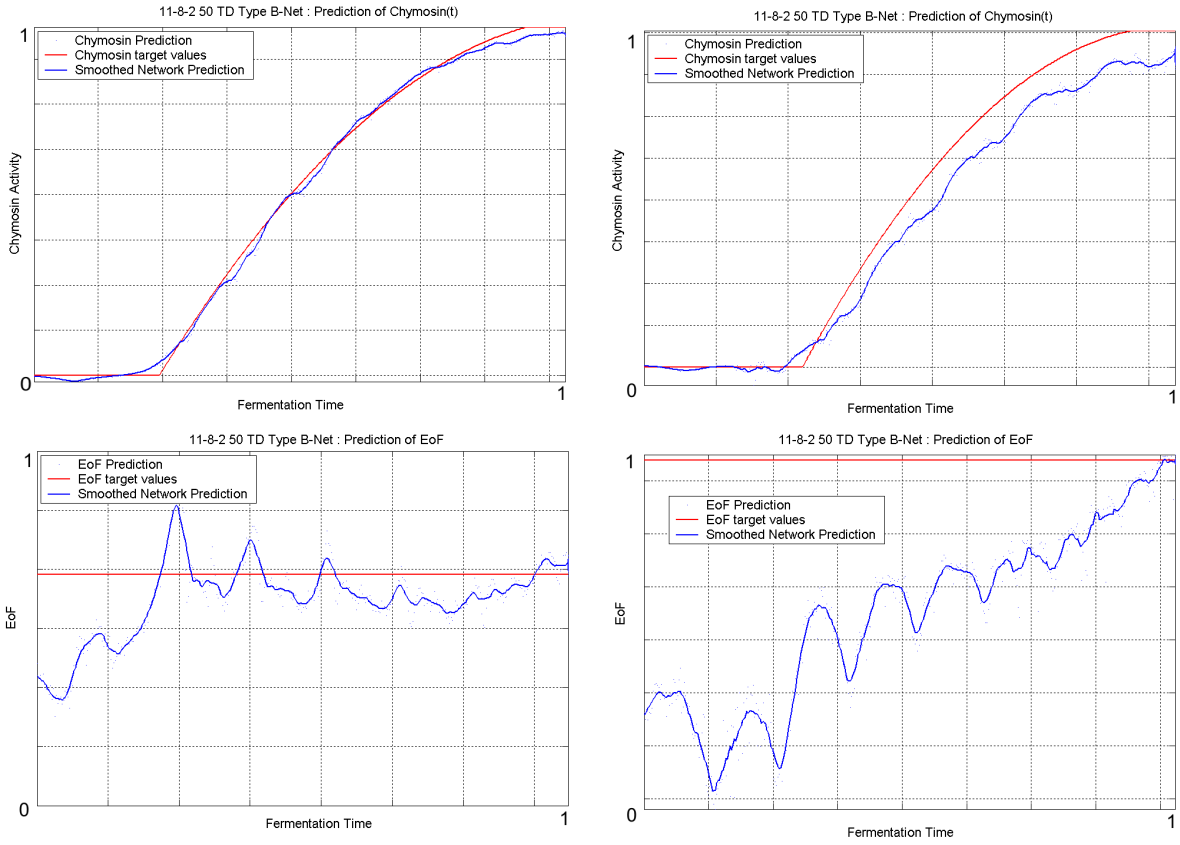

Abbildung 32. Trainingsergebnisse eines 11-8-2 Netzes mit Feedback-Schleife (Typ B). Oben sind die Zielwerte (rot) und die Vorhersage der Chymosinkonzentration (blau) von zwei Prozessen des Trainingssets gezeigt. In den unteren beiden Grafiken sind die Zielwerte der Endausbeute (rot) sowie die Vorhersage des Netzes (blau) dargestellt.

Rechts davon ist ein anderer Datensatz aus dem Trainingsset präsentiert, der mit dem gleichen Netz ausgewertet worden ist. Bei diesem Datensatz ergaben sich deutliche

Abweichungen von den Zielwerten. Bei der Vorhersage der Endausbeute zeigt sich das erwartete Verhalten, dass die Prognose, die aus den Eingabewerten vom Anfang des Prozesses errechnet werden, teilweise sehr deutliche Abweichungen zu den Zielwerten zeigen, sich aber im Verlauf eine Annäherung an den wahren Wert der Endausbeute ergibt. Dies war bei den meisten betrachteten Prozessen sehr schnell der Fall (siehe Grafik unten links), bei einigen der Prozesse dauerte es dagegen bis zum Ende der Fermentation bis die Annäherung an den Zielwert erfolgte (Grafik unten rechts) bzw. es kam keine Annäherung zustande. Die meisten Prozesse ließen sich mit diesem Netz aber sehr gut beschreiben, daher gehörte dieses Netz zu den besten bezüglich der Trainingsfehler.
Bei den 250 mit NNCreate erschaffenen Netzen gab es keine signifikanten Unterschiede in den Fehlerwerten zwischen Netzen mit und ohne Feedback-Schleifen oder zwischen den Trainingsfunktionen trainlm oder trainbr. Weiterhin ist an den Trainingsfehlern nicht zu ersehen, dass Netze mit vielen Neuronen in den verdeckten Schichten Netzen mit wenigen Neuronen über- oder unterlegen sind.

4.4.3 Anwendung der neuronalen Netze zur Simulation - NNSim

Das Programmmodul NNSim dient zur Online-Simulation des Fermentationsprozesses mit einem neuronalen Netz. Dazu wird das entsprechende Netz in der Liste im Hauptprogramm NNMain ausgewählt und die Simulation mit dem Button *Simulation* (siehe Abbildung 30 unten) gestartet. Das Simulationsprogramm NNSim befindet sich zunächst in einem Wartemodus, in dem in regelmäßigen Abständen der Status zweier Fermenter abgefragt wird. Die Simulation wird automatisch gestartet, wenn in einem der Fermenter ein Fermentationsprozess beginnt bzw. wenn beim Start des Programms bereits ein Prozess läuft. Der Fall, dass in beiden Fermentern ein Prozess läuft, kann von NNSim ebenfalls abgehandelt werden. Beide Prozesse werden dann gleichzeitig und getrennt voneinander aber mit dem gleichen Netz simuliert. Im Simulationsmodus empfängt das Programm alle acht Minuten die Onlinedaten vom Prozessleitsystem RSLinx mittels einer DDE-Verbindung (*Dynamic Data Exchange*). Auch die eingangs erwähnte Abfrage des Fermenterstatus wird über diese DDE-Verbindung durchgeführt.
DDE ist ein gängiges Verfahren, das den Datenaustausch zwischen mehreren Anwendung erlaubt. Dazu wird eine Anwendung als S*erver* (hier RSLinx) und die

andere als *Client* (hier NNSim bzw. Matlab) definiert, welches mit dem Matlabbefehl *ddeinit* vorgenommen wird. Nach Festlegen eines *Topics* (Art der auszutauschenden Daten) können Daten mit dem Befehl *ddereq* zwischen den Anwendungen ausgetauscht werden.

Die empfangenen Prozessdaten werden nun auf die gleiche Art und Weise wie die Daten des Trainingsdatensatzes auf den Wertebereich von –1 und +1 normiert und in das neuronale Netz eingegeben. Die Ausgabe des Netzes wird dann wieder denormiert und im Diagrammfenster von NNSim dargestellt (siehe Abbildung 33).

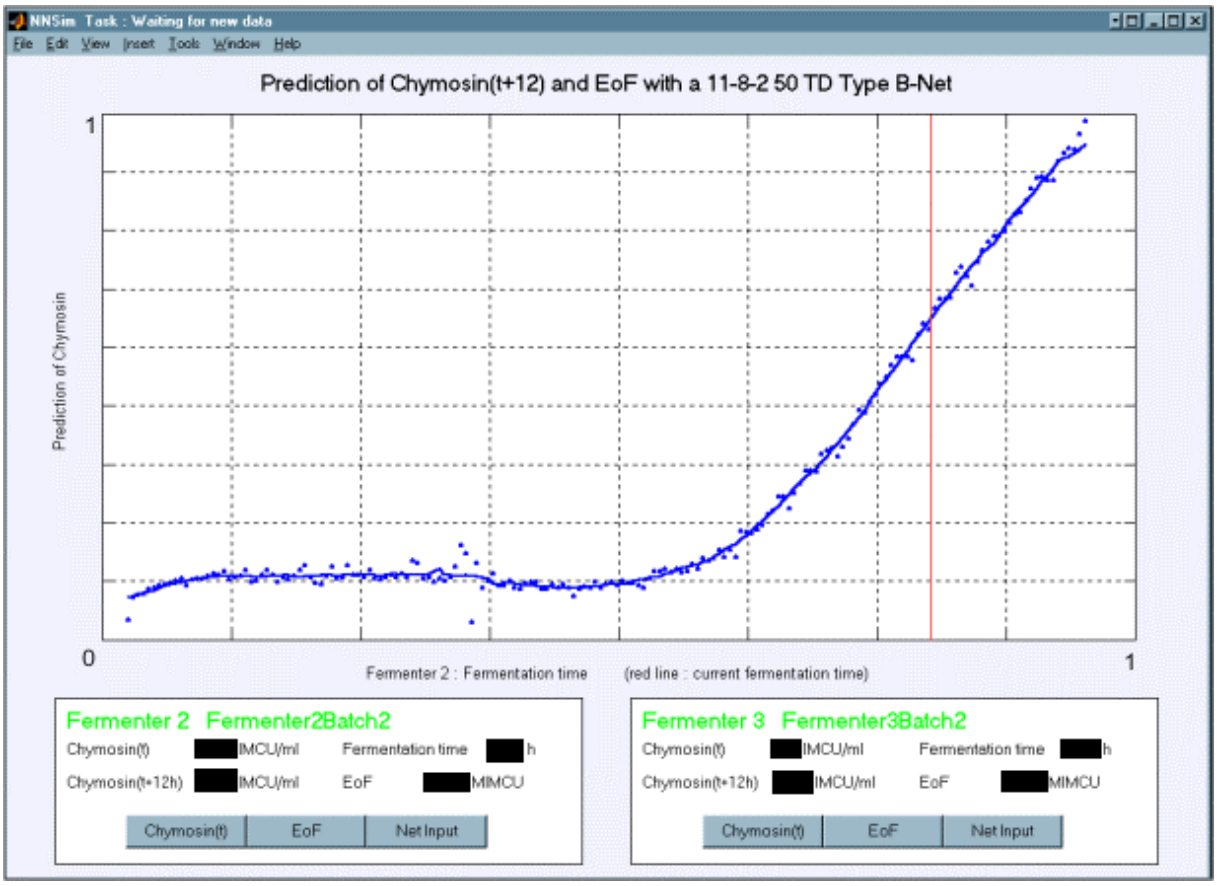

Abbildung 33. Screenshot des Programmmoduls NNSim. Dargestellt ist die Vorhersage der Produktkonzentration mit einem 11-8-2 Netz mit Feedback-Schleife (Typ B). Aus den Ausgaben des Netzes (blaue Punkte) wurde mit einem gleitenden Durchschnittfilter die blaue Kurve errechnet. Die aktuelle Fermentationszeit ist durch die rote Linie gekennzeichnet.

Mit Hilfe von sechs Buttons kann wahlweise die Vorhersage der Produktkonzentration, die Vorhersage der Endausbeute oder die Netzeingaben für beide Fermenter dargestellt werden. Bei Anzeige der Vorhersagen werden sowohl die Ausgaben des Netzes sowie eine daraus berechnete Glättungskurve (gleitender Durchschnittsfilter mit einer

Fensterbreite von 5 Werten) gezeigt. Die aktuelle Fermentationszeit wird durch eine rote Linie gekennzeichnet. Die Prognose der aktuellen sowie der erwarteten Produktkonzentration in zwölf Stunden und die Prognose der Endausbeute wird darüber hinaus in den Kästen unter dem Diagrammfenster als Zahlenwert angezeigt.

Nach jedem Simulationsschritt werden die Eingabedaten und die Vorhersagen des Netzes abgespeichert und das Programm tritt in einen erneuten Wartezyklus von acht Minuten Länge ein. Ist eine Fermentation vollständig abgelaufen, wird dies automatisch erkannt und das Programm geht in den Wartemodus über, bis eine neue Fermentation gestartet wird. Das Programm ist dafür ausgelegt, über lange Zeiträume zu laufen, selbständig Start und Ende der Fermentationen festzustellen, diese Prozesse mit dem in NNMain ausgewählten Netz zu simulieren und die Ergebnisse abzuspeichern.

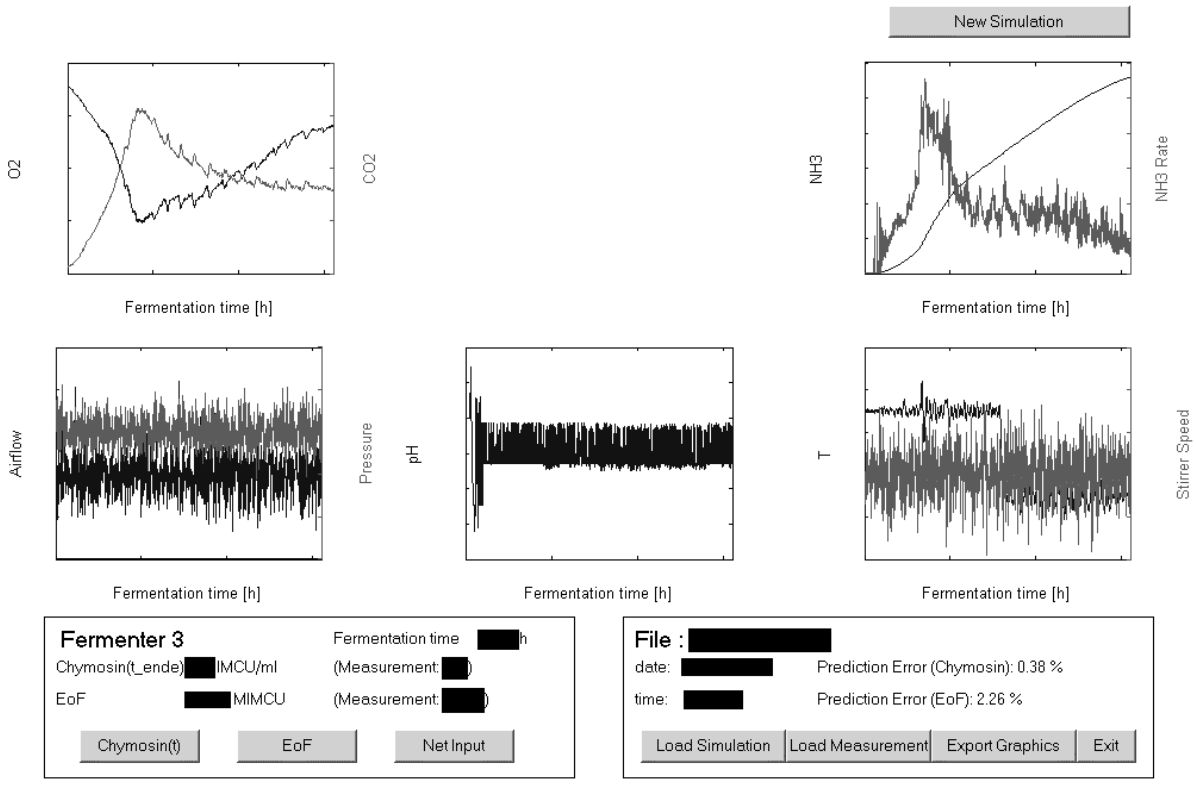

Abbildung 34. Screenshot des Programmmoduls NNView. Gezeigt sind die Online-Messwerte eines Fermentationsprozesses, der in NNSim mit einem 11-8-2 Netz simuliert worden ist. Die Achsenbeschriftungen, die Werte des Maltose Feeds sowie die Angaben zu den Messgrößen im unteren Bereich des Bildschirm können aus Datenschutzgründen nicht gezeigt werden.

Liegen die Offline-Messwerte der Produktkonzentration vor, können sie zusammen mit den Ergebnisdateien von NNSim im Modul NNView angezeigt werden. Dazu können

mit *Load Simulation* die Simulationsdaten von NNSim und mit *Load Measurement* die Messwerte der Produktkonzentration geladen werden (siehe Abbildung 34 unten rechts). Mit den drei Knöpfen unten links können wahlweise die Eingaben des Netzes, die Vorhersage der Produktkonzentration und die Vorhersage der Endausbeute angezeigt werden. Des weiteren werden die Endwerte der Vorhersagen, die Zielwerte, die aus den geladenen Messwerten hervorgehen, sowie die prozentualen Fehler als Zahlenwert angezeigt. Weitere Funktionen, die das Programm bietet, sind das Exportieren der Grafiken als Bitmapdatei (*Export Graphics*) und das Simulieren des momentan geladenen Prozesses mit einem anderen Netz zu Vergleichszwecken (*New Simulation*).

Über einen Zeitraum von mehreren Wochen wurden mit Hilfe des Programmmoduls NNSim Vorhersagen zu den aktuell laufenden Fermentationsprozessen erstellt. Dabei kam ein 11-8-2 Netz (im Folgenden: Netz 1) ohne Feedback-Schleife und ein 11-8-2 Netz mit Feedback-Schleife (Typ B, Verzögerungswert: 100, im Folgenden: Netz 2) zum Einsatz. Es wurden insgesamt 19 Prozesse simuliert und anschließend mit NNView ausgewertet. Beim ersten Prozess traten zwei Probleme auf. Die Simulationssoftware wurde mitten im laufenden Prozess gestartet und somit lagen nur Daten aus der zweiten Prozesshälfte zur Vorhersage vor. Das andere Problem war die Störung in den Druck-, O_2- und CO_2-Werten. Diese und die anderen Messreihen dieses Prozesses sind in Abbildung 35 gezeigt. Ob die Störung aufgrund eines Fehlers in der Messtechnik auftrat oder eine andere Ursache hat, ist nicht bekannt. Interessant war bei diesem Prozess, wie die beiden unterschiedlichen verwendeten Netze auf diese beiden Faktoren, unvollständige Daten und Störung, reagieren.

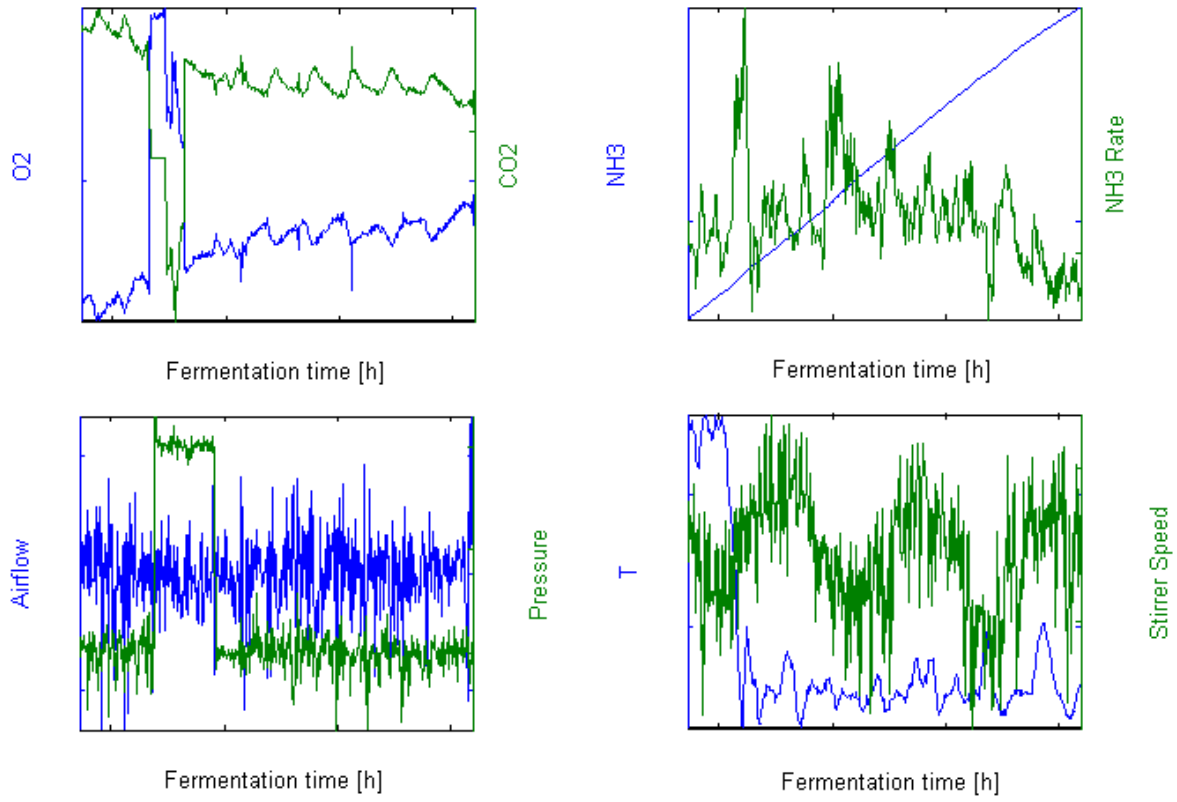

Abbildung 35. Online-Messdaten von Prozess 1, bei dessen Simulation zwei Probleme auftraten: Zum Einen lagen nur Daten aus der zweiten Hälfte des Prozesses vor und zum Anderen trat bei den Messungen für Druck, O_2 und CO_2 eine Störung auf.

In Abbildung 36 sind die Vorhersagen von Netz 1 gezeigt. Es ist deutlich zu erkennen, dass die Störung in den Eingabedaten zu einer Schwingung der errechneten Chymosinkonzentration führt. Die Abweichungen sind dabei auf die Zeitpunkte, an denen die Störung auftrat, begrenzt und wirken sich nicht auf die restliche Simulation aus. Ähnliches gilt für die Vorhersage der Endausbeute (Abbildung 36, unten). Während der Störung fällt die Vorhersage deutlich ab, kehrt danach aber wieder auf das Ausgangsniveau zurück. Dieses Netz zeichnet sich also durch Robustheit gegenüber Störungen in den Eingangsvariablen aus.

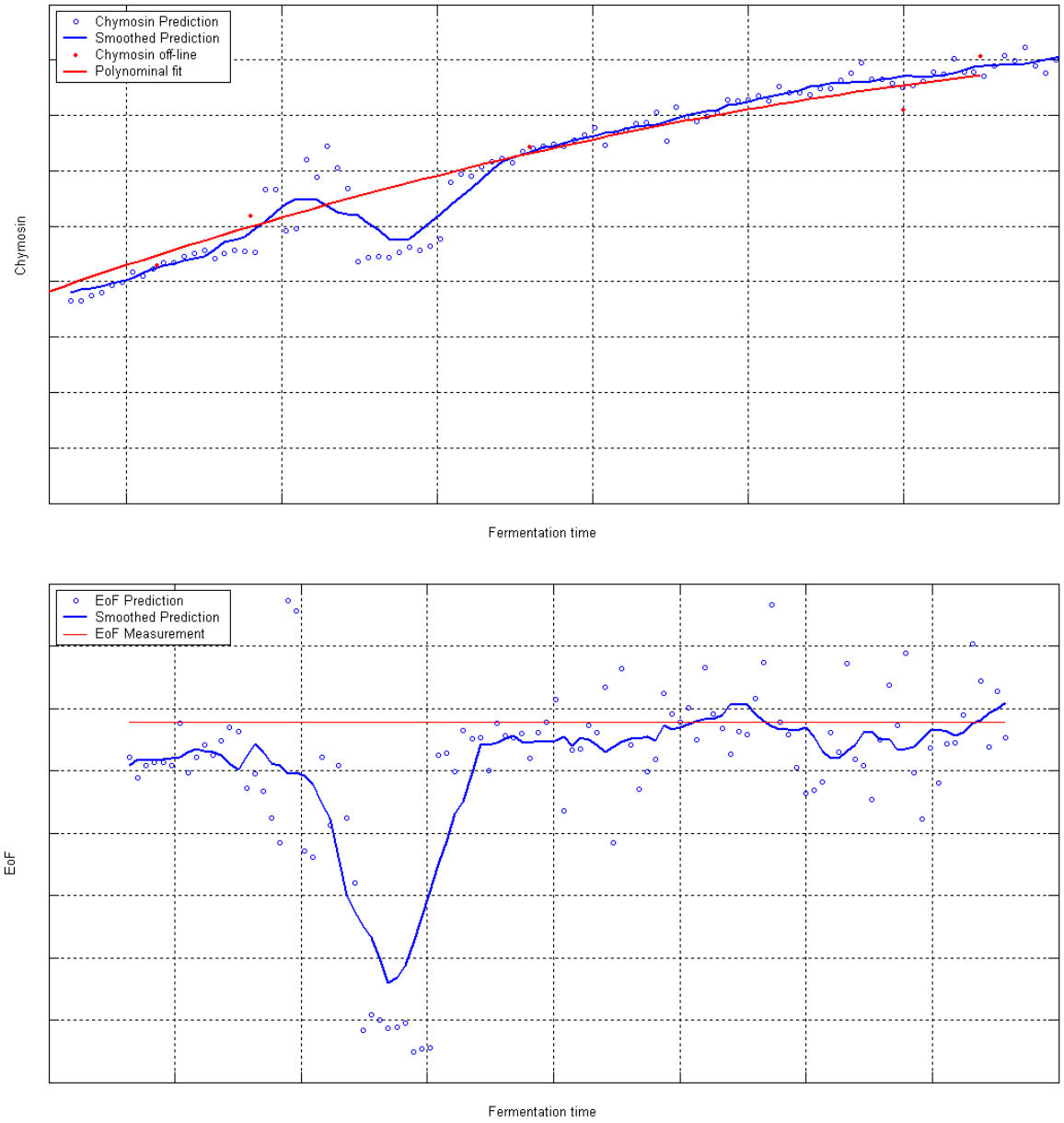

Abbildung 36. Oben: Auftragung der Vorhersage der Chymosinkonzentration (blau) und der Messwerte (rot) gegen die Fermentationszeit. Deutlich ist zum Zeitpunkt der Störung in den Eingabedaten die Auswirkung auf die Vorhersage des neuronalen Netzes zu erkennen. Unten: Auftragung der Vorhersage der Endausbeute (blau) und des Messwertes (rot) gegen die Fermentationszeit. Das verwendete neuronale Netz ist ein 11-8-2 Netz ohne Feedback-Schleife (Netz 1).

Bei den Vorhersagen von Netz 2 ergibt sich ein anderes Bild. Auch hier kommt es während der Störung zu einer Schwingung in der Vorhersage der Chymosinkonzentration (Abbildung 37, oben), danach aber knickt die Kurve ab und Messwerte und Vorhersage weichen deutlich voneinander ab. Auch bei der Vorhersage der Endausbeute hat die Störung in den Eingabedaten Folgen über ihre zeitlichen Dauer

hinaus: Nach dem Absinken der Vorhersage während der Störung wird das ursprüngliche Niveau nicht wieder erreicht.

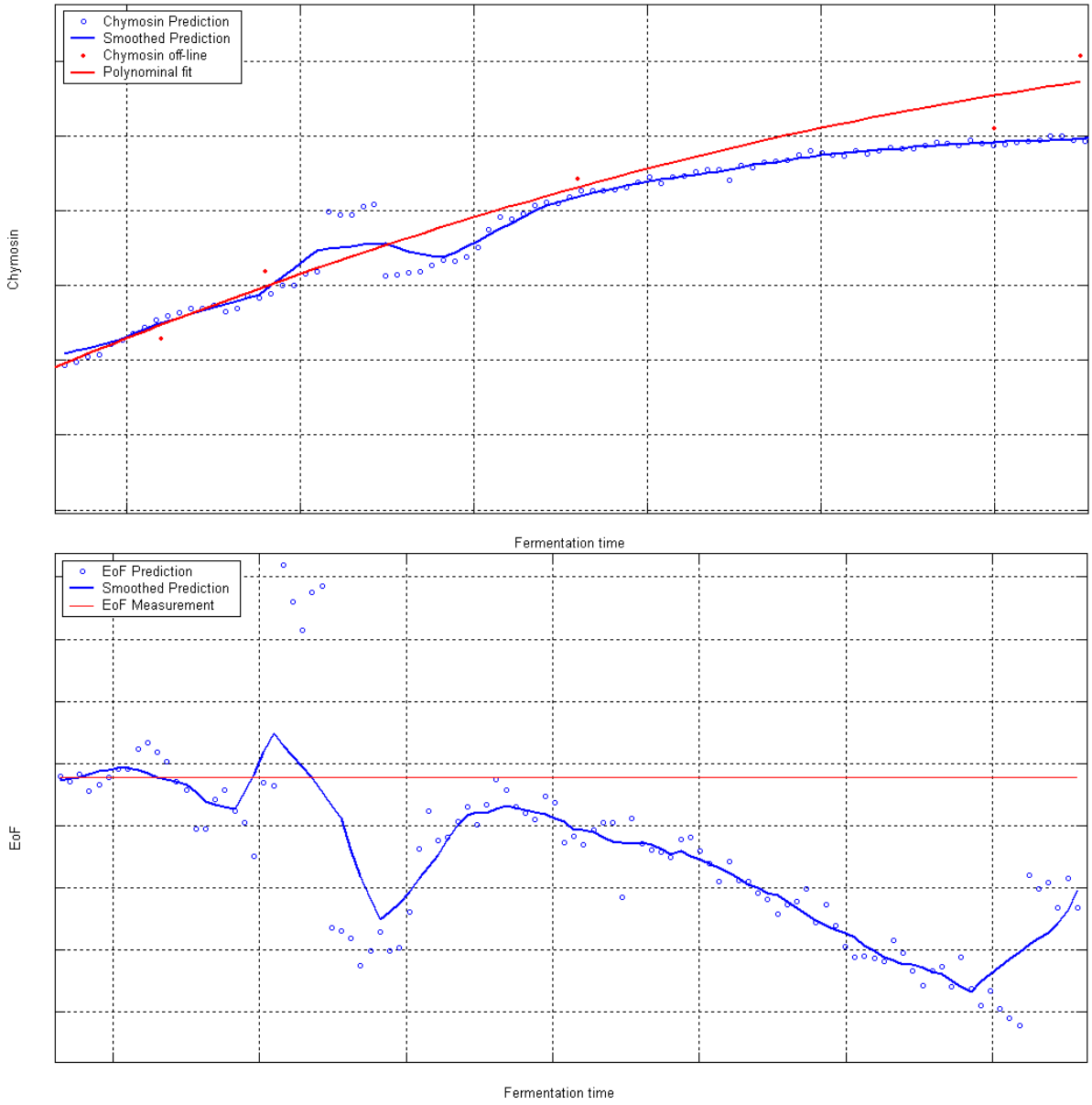

Abbildung 37. Zur Erläuterung der Grafik siehe Abbildung 36. Das hier verwendete neuronale Netz ist ein 11-8-2 Netz mit Feedback-Schleife (Netz 2).

Die Fehler der Vorhersagen, bezogen auf die Abweichung der Endwerte von dem jeweiligen Messwert, beträgt bei diesem Netz ca. 10 % bei der Chymosinkonzentration und 7 % bei der Endausbeute. Die Fehler liegen damit deutlich höher als die Fehler von Netz 1 (ca. 1 % bei der Chymosinkonzentration und 2 % bei der Endausbeute). Betrachtet man die Gesamtheit der 19 simulierten Prozesse und vergleicht die beiden Netze über die Mittelwerte der Vorhersagefehler zeigen sich auch dort beim Netz ohne

Feedback-Schleife geringere Fehler. Der mittlere Fehler der Vorhersagen von Netz 2 liegt bei 9 % für die Chymosinkonzentration gegenüber 5 % bei Netz 1 bzw. bei 14 % für die Endausbeute gegenüber 6 % bei Netz 1.

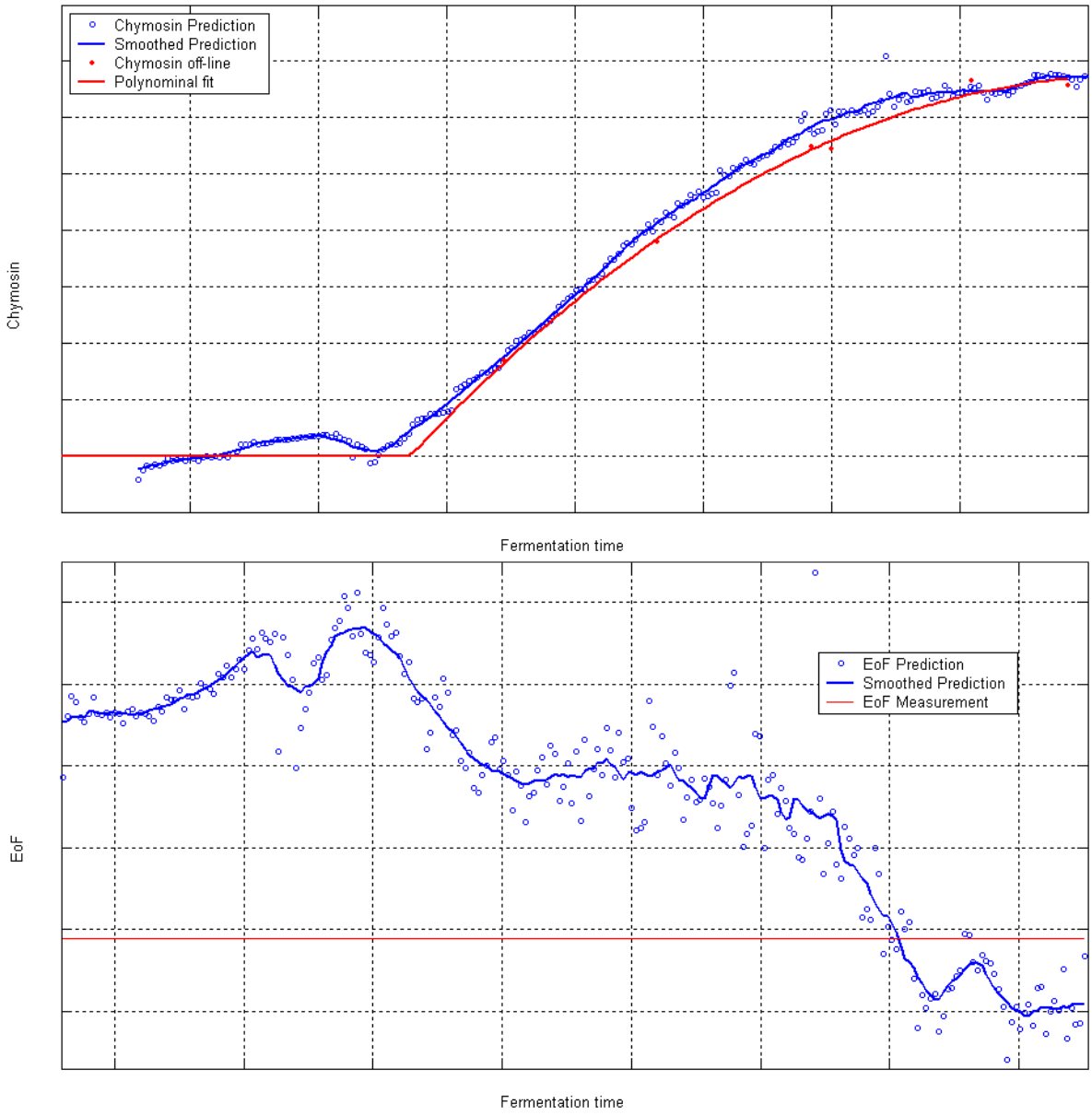

Abbildung 38. Oben: Vorhersage der Chymosinkonzentration mit einem 11-8-2 Netz basierend auf den Eingaben aus Abbildung 34. Die Ausgaben des Netzes sind in blau dargestellt, die Zielwerte in rot. Unten: Vorhersage der Endausbeute des 11-8-2 Netzes an den selben Prozessdaten. Auch hier ist die Vorhersage in blau dargestellt und der Zielwert in rot. Beide Grafiken wurden mit NNView erstellt.

Dieses Ergebnis zeigt, dass die in Kapitel 4.4.2 dargestellte Vermutung, dass Netze mit Feedback-Schleife zur Simulation dieses Prozesses besser geeignet sind als Netze ohne Feedback-Schleife, in der Praxis nicht zutrifft.

Die Robustheit gegenüber schwankenden Eingabedaten oder gar Störungen in den Eingabedaten wird durch eine Feedback-Schleife vermindert, was sich in höheren Vorhersagefehlern niederschlägt. Dies zeigt sich nicht nur beim Vergleich von Netz 1 und Netz 2 sondern auch bei der Auswertung anderer mit NNCreate erzeugter Netze: Netze mit Feedback-Schleife, egal ob vom Typ A oder B, weisen bei der Simulation höhere Fehlerwerte auf als Netze ohne Feedback-Schleife. Das 11-8-2 Netz ohne Feedback-Schleife (Netz 1) zeigte von allen Netzen die geringsten Fehler bei der Simulation. Die Vorhersagen eines störungsfreien Prozesses sind in Abbildung 38 gezeigt. Sie basieren auf den Eingabewerten aus Abbildung 34. Der Verlauf der tatsächlichen Chymosinkonzentration wird durch das Netz gut wiedergegeben, die Übereinstimmung zwischen den Endwerten von Vorhersage und Messwerten ist sogar sehr gut. Der prozentuale Fehler liegt hier bei nur 0,38 %. Die Angabe des prozentualen Fehlers bezieht sich immer auf die prozentuale Abweichung des letzten Vorhersagewertes bezogen auf den letzten Messwert: $Fehler = |Vorhersage - Messwert| * 100 / Messwert$. In der unteren Grafik ist die Vorhersage der Endausbeute dargestellt. Es zeigt sich das erwartete Verhalten, dass zu Beginn ein großer Unterschied zum Zielwert vorliegt und sich mit fortschreitender Prozessdauer eine Annäherung an den Zielwert ergibt. Der prozentuale Fehler bezogen auf die Endausbeute liegt bei 2,26 %.

In Tabelle 7 ist die Auswertung der Vorhersagen aller 19 Fermentationsprozesse aufgeführt. Aus Gründen des Datenschutzes können die Absolutwerte der Konzentrations- und Ausbeutemessungen sowie deren Vorhersagen nicht angegeben werden. Es wurden statt dessen dimensionslose, normierte Werte angegeben. Die Normierung wurde so durchgeführt, dass die Vorhersagewerte des ersten Prozesses 100 betragen.

Tabelle 7. Auswertung der Vorhersagen von 19 Fermentationsprozessen mit einem 11-8-2 Netz ohne Feedback-Schleife (Netz A).

Prozess Nr.	EoF Vorhersage	EoF Messung	EoF Fehler %	Chymosin Vorhersage	Chymosin Messung	Chymosin Fehler %
1	100,00	101,66	1,63	100,00	98,87	1,15
2	98,38	97,50	0,91	104,52	94,29	10,85
3	92,29	96,14	4,00	96,72	92,75	4,29
4	94,28	89,06	5,86	98,61	87,63	12,53
5	97,82	103,90	5,86	103,22	108,41	4,78
6	101,75	96,74	5,18	103,48	99,16	4,36
7	90,80	91,17	0,41	96,88	95,25	1,71
8	103,81	101,93	1,84	107,62	104,35	3,14
9	92,40	95,67	3,42	102,34	96,24	6,34
10	98,75	111,35	11,32	104,07	119,53	12,93
11	98,74	108,78	9,23	106,01	108,84	2,60
12	88,61	100,29	11,65	94,37	99,20	4,87
13	84,55	88,12	4,06	91,51	87,22	4,91
14	76,57	77,62	1,35	88,27	85,08	3,76
15	79,50	77,75	2,26	85,93	85,60	0,38
16	96,07	78,91	21,74	78,73	85,26	7,66
17	83,32	95,96	13,18	94,34	99,68	5,36
18	93,43	87,22	7,12	98,36	93,01	5,75
19	103,05	100,35	2,70	102,45	98,24	4,28
		Mittelwert	**5,98**		Mittelwert	**5,35**

Die prozentualen Fehler der Vorhersage der Ausbeute am Ende der Fermentation (EoF) liegen größtenteils um bzw. sogar unter 5 %, bei drei Prozessen liegt der Fehler über 10 %. Ein Prozess konnte nicht gut simuliert werden und der Fehler lag deutlich über 20 %. Bei der Vorhersage der Chymosinkonzentration ist die Streuung der Fehler etwas geringer: Bei drei Prozessen liegt der Fehler über 10 % und sonst größtenteils um bzw. unter 5 %. Die mittleren Fehler der Vorhersage dieses Netzes liegen bei ca. 6 % bei der Endausbeute und bei ca. 5,4 % bei der Produktkonzentration. Bei der Bewertung dieser Vorhersageergebnisse ist zu bemerken, dass bei einigen der simulierten Prozesse fehlerbehaftete und verrauschte Prozessdaten bzw. mit Störungen überlagerte Messdaten als Eingabewerte für das neuronale Netz verwendet werden mussten. Berücksichtigt man des Weiteren, den Messfehler bei der Bestimmung der Chymosinkonzentration, der typischerweise bei 3-5 % liegt [HELLMUTH], und dass der Fehler der Vorhersage mit 6 % deutlich unter der Varianz der vorherzusagenden Zielwerte liegt, lässt sich festhalten, dass mit dem hier verwendeten 11-8-2 Netz ein sehr gutes Prädiktionsmodell für diesen Prozess erstellt wurde.

4.4.4 Anwendung der neuronalen Netze zur Optimierung

Neben der im vorangegangenen Abschnitt beschriebenen Anwendung in der Prozesssimulation wurde das neuronale Netz auch zur Optimierung der Prozessführung eingesetzt. Die Zielsetzung bestand darin, diejenigen Prozessgrößen zu identifizieren, bei deren Veränderung eine Steigerung der Produktausbeute zu erwarten ist. Diese Ergebnisse sollen zudem eine Risikoabschätzung ermöglichen, welche der betrachteten Prozessgrößen den größten Einfluss auf die Produktbildung haben und nicht verändert werden dürfen.

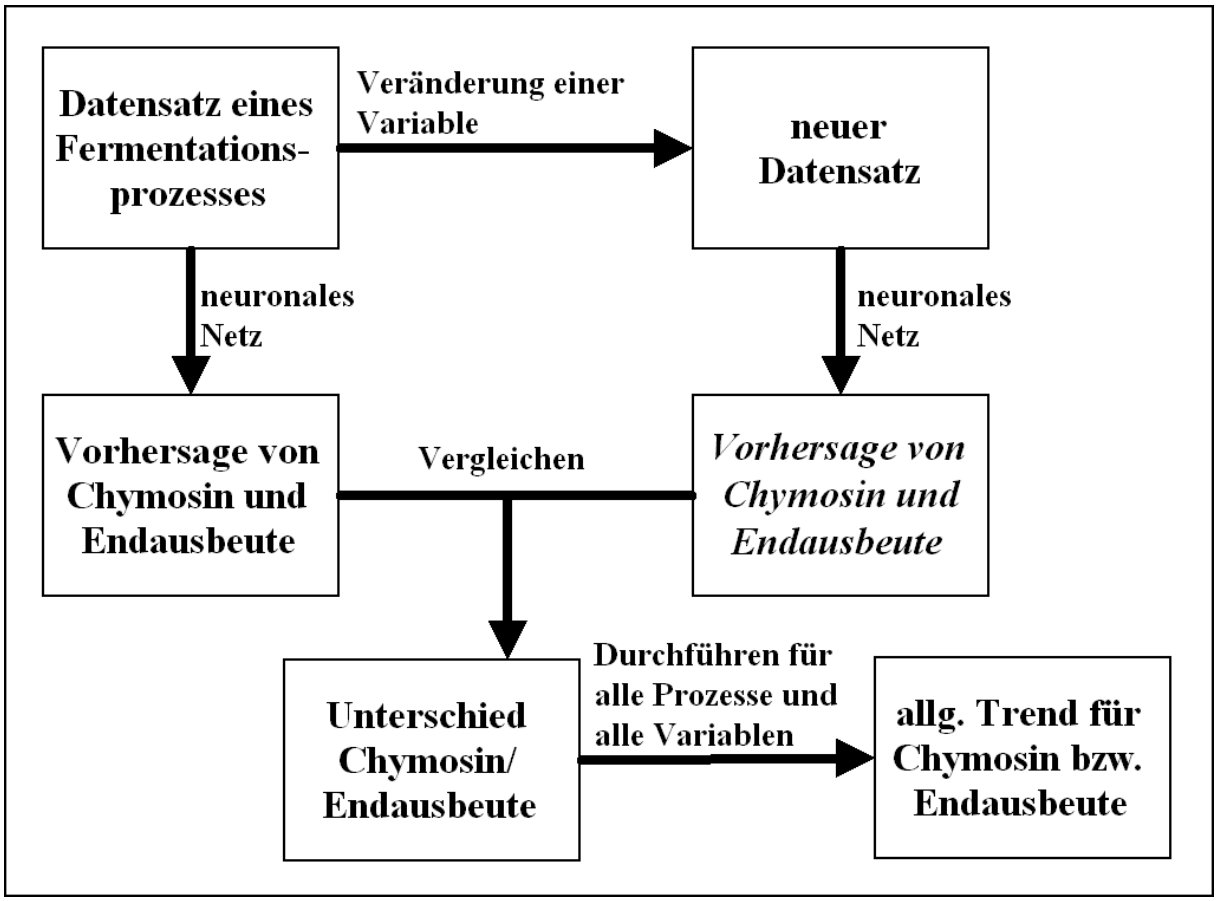

Abbildung 39. Schematische Darstellung des Ablaufs der Prozessoptimierung mit einem neuronalen Netz.

Die Herangehensweise ist in Abbildung 39 dargestellt. Ausgehend von dem Datensatz eines Prozesses wird mit einem neuronalen Netz eine Vorhersage der Chymosinkonzentration und der Endausbeute erstellt. Ausgehend von dem gleichen Datensatz wird durch Veränderung einer Prozessvariable (Details: s.u.) ein neuer

Datensatz erzeugt, mit dem ebenfalls eine Vorhersage der Chymosinkonzentration und der Endausbeute erstellt wird. Durch Vergleich der beiden Vorhersagen und Durchführung dieser Rechnung für alle vorhandenen Datensätze und für alle Variablen ergibt sich der allgemeine Trend für die zu erwartenden Veränderungen in der Ausbeute bezüglich aller betrachteten Prozessvariablen.

Folgende Änderungen an den Prozessvariablen (Modifikationsmethoden) wurden bei der Optimierung betrachtet:

1. Veränderung der Feed-Strategie (Feedpulse früher bzw. später)
2. Veränderung der Feed-Strategie 2 (Höhe und Länge der Feedpulse verändert)
3. Erhöhen oder Absenken der Temperatur insgesamt (± 2,5 °C)
4. Veränderung der Höhe des Temperatur-Shifts (± 2,5 °C)
5. Veränderung des Zeitpunktes des Temperatur-Shifts (± 30 h)
6. Erhöhen oder Absenken der Rührgeschwindigkeit (± 10 rpm)
7. Erhöhen oder Absenken des Drucks (± 0,1 bar)
8. Veränderung der Begasungsstärke
9. Erhöhen oder Absenken des pH-Wertes (± 0,3)
10. Erhöhen oder Absenken der Rührgeschwindigkeit während eines bestimmten Zeitintervalls (± 10 rpm)
11. Erhöhen oder Absenken der Rührgeschwindigkeit in einem treppenförmigen Profil (± 10 rpm)
12. Erhöhen oder Absenken des pH-Wertes in einem treppenförmigen Profil (± 10 rpm)

Zur Berechnung der veränderten Datensätze wurden Algorithmen in der Software Matlab implementiert, die die oben genannten Änderungen der Prozessvariablen vornehmen können. Die Algorithmen wurden so gestaltet, dass sich die Variablen jeweils in 11 Stufen verändern lassen. Dazu ein Beispiel: Beim Erhöhen bzw. Absenken der Temperatur (Modifikationsmethode 3, s.o.) lassen sich Datensätze generieren, bei denen die Temperatur gegenüber den Ursprungsdaten um 2,5 °C, 2,0 °C, 1,5 °C, 1,0 °C und 0,5 °C niedriger ist, gleich geblieben ist oder um 2,5 °C, 2,0 °C, 1,5 °C, 1,0 °C und 0,5 °C erhöht wurde. In Abbildung 40 ist an einem Beispieldatensatz die Veränderung der Temperatur um –1,5 °C dargestellt. In Abbildung 41 ist ein Beispiel für die Modifikation der Rührgeschwindigkeit nach Methode 11 gezeigt.

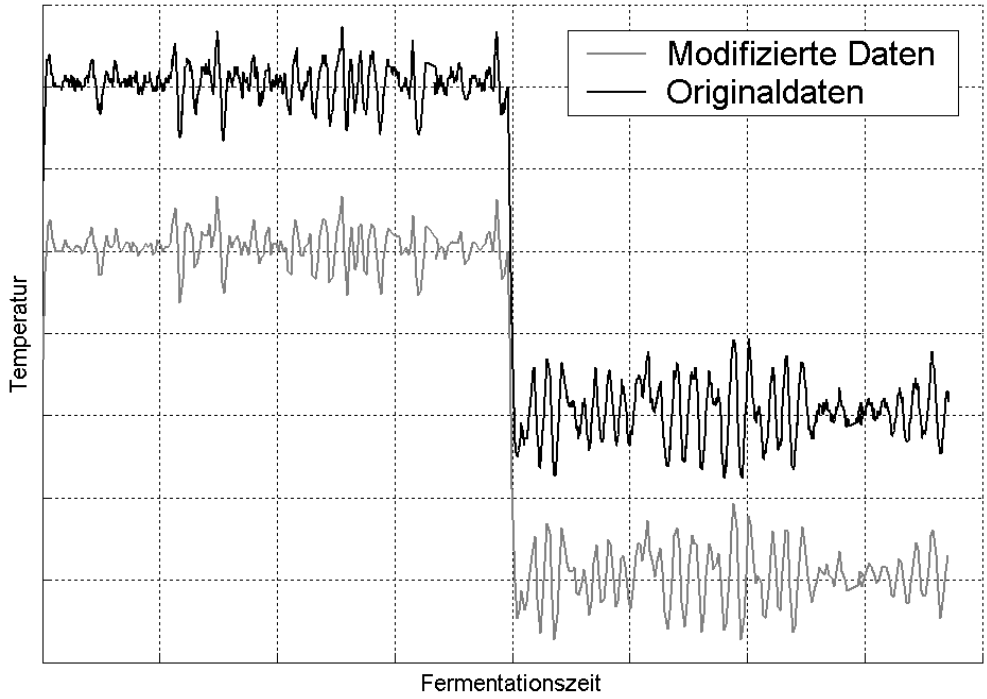

Abbildung 40. Beispiel für die Veränderung einer Prozessvariable im Rahmen der Optimierung. Dargestellt ist hier in schwarz der Verlauf der Temperatur gegen die Fermentationszeit. Der modifizierte Datensatz (grau) enthält eine um 1,5 °C erniedrigte Temperaturkurve.

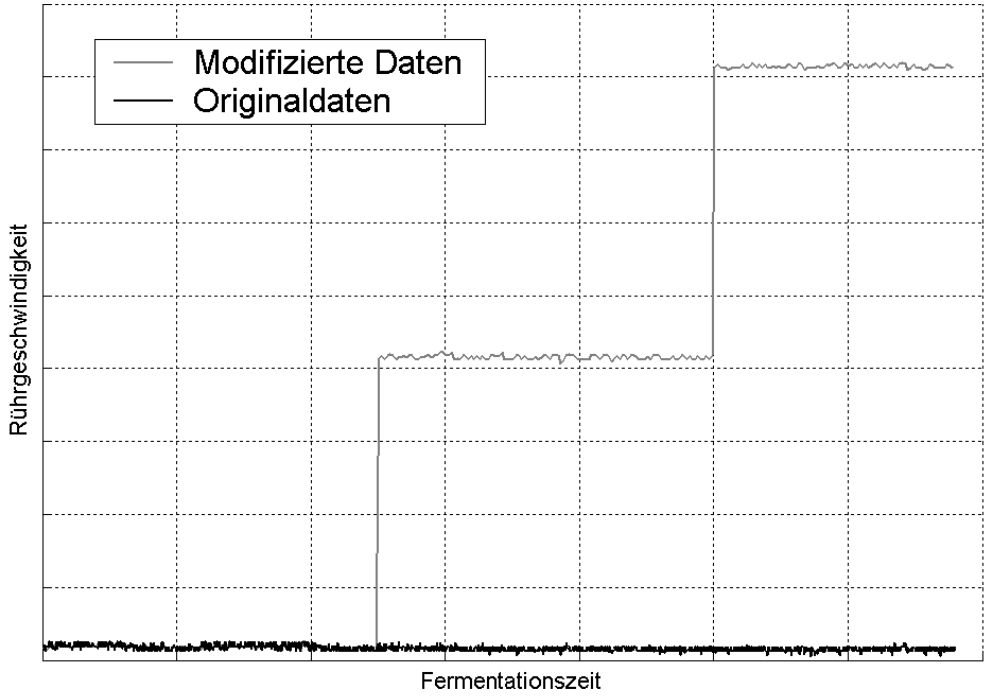

Abbildung 41. Beispiel für die Veränderung einer Prozessvariable im Rahmen der Optimierung. Dargestellt ist hier in schwarz der Verlauf der Rührgeschwindigkeit gegen die Fermentationszeit. Im modifizierten Datensatz (grau) wird die Rührgeschwindigkeit stufenweise erst um 5 und dann um 10 rpm angehoben.

Die Rührgeschwindigkeit wird dabei nach einem Drittel der Prozesszeit um 5 und nach zwei Dritteln der Prozesszeit um 10 rpm erhöht. Die Modifikationsstärken bei Methode 11 reichte von +5 (+10) bis zu –5 (–10) rpm in 11 Abstufungen. Auf analoge Weise wurden auch die 10 anderen, hier nicht detailliert dargestellten Modifikationsmethoden, mit 11 Abstufungen implementiert.

Die modifizierten Datensätze wurden in ein neuronales Netz eingegeben und es wurde eine Vorhersage der Ausbeute erstellt, wie in Abbildung 39 (rechts) dargestellt. Pro Datensatz bzw. pro Prozess erhält man so 11 Vorhersagewerte für jede Modifikationsmethode, da die bei der Optimierung betrachteten Prozessvariablen in 11 Abstufungen modifiziert wurden. Durch Darstellung dieser Werte in einem Vorhersage-Modifikationsstärke-Diagramm erhält man den Trend der Ausbeute gegenüber Veränderungen der betrachteten Prozessvariablen.

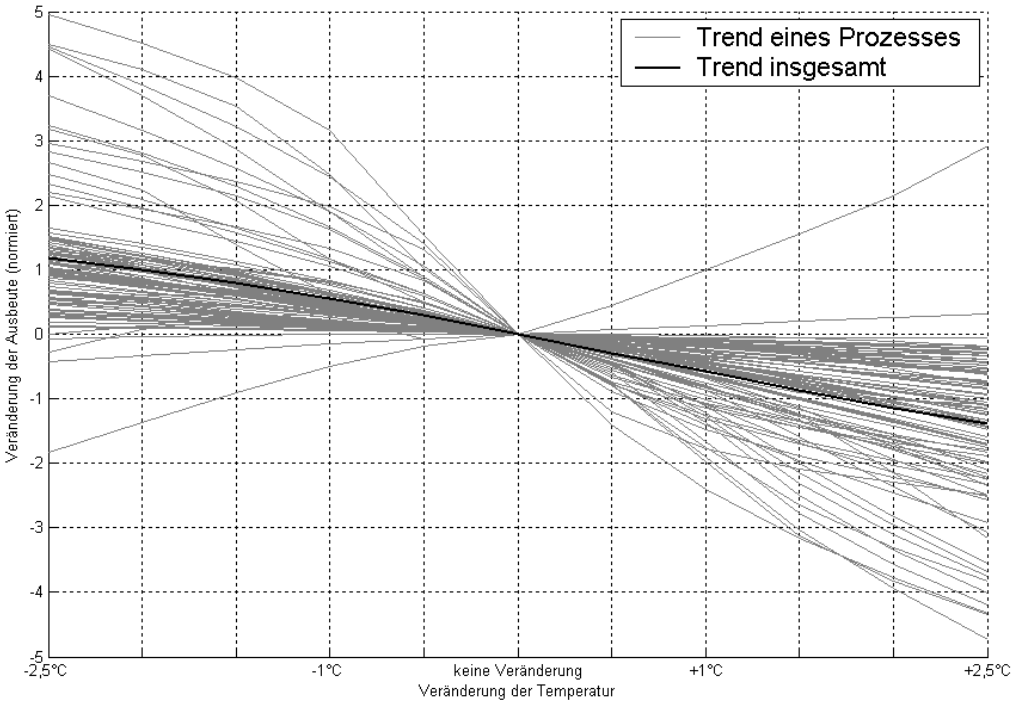

Abbildung 42. Von einem neuronalen Netz prognostizierte Veränderung der Ausbeute bei Veränderung der Temperatur (Modifikationsmethode 3). In grau sind die Prognosen einzelner Prozessdatensätze gezeigt, in schwarz der Gesamttrend.

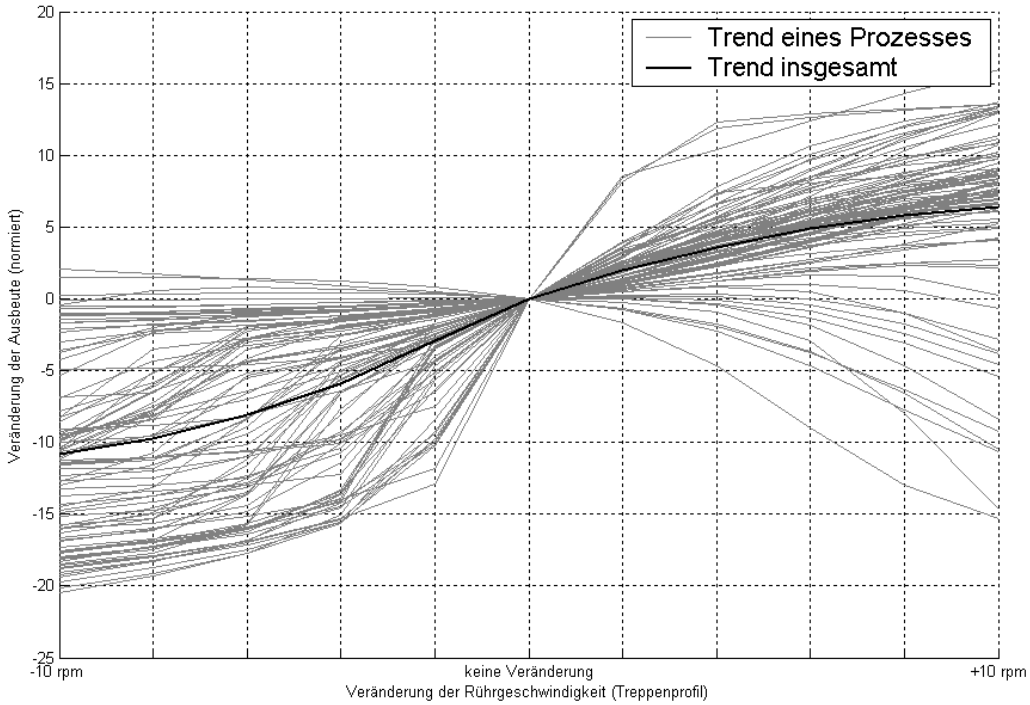

Abbildung 43. Von einem neuronalen Netz prognostizierte Veränderung der Ausbeute bei Veränderung der Rührgeschwindigkeit (Modifikationsmethode 11). In grau sind die Prognosen einzelner Prozessdatensätze gezeigt, in schwarz der Gesamttrend.

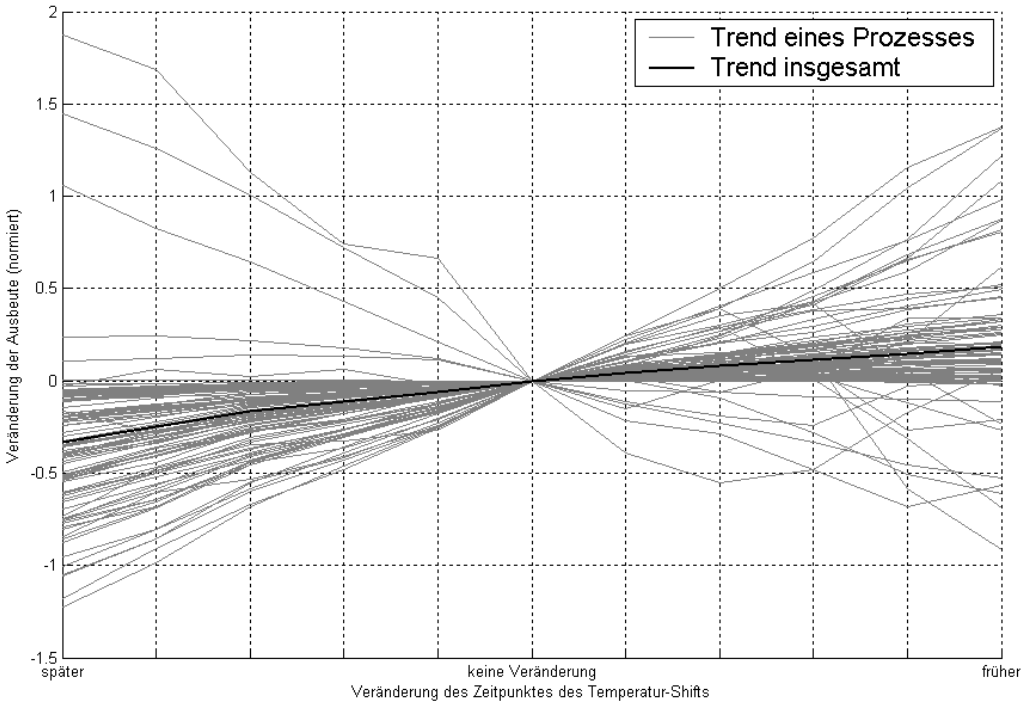

Abbildung 44. Von einem neuronalen Netz prognostizierte Veränderung der Ausbeute bei Veränderung des Zeitpunktes des Temperatur-Shifts (Modifikationsmethode 5). In grau sind die Prognosen einzelner Prozessdatensätze gezeigt, in schwarz der Gesamttrend.

In Abbildung 42 ist die vom neuronalen Netz prognostizierte Veränderung der Ausbeute bei Veränderung der Temperatur dargestellt. Die grauen Kurven basieren jeweils auf dem Datensatz eines Prozesses (insgesamt wurden Datensätze von 95 Prozessen verwendet). Man erkennt, dass bei der Mehrheit der Prozesse das neuronale Netz eine leichte Steigerung der Ausbeute[7] bei um 2,5 °C gesenkter Temperatur vorhersagt bzw. eine leichte Abnahme der Ausbeute bei um 2,5 °C erhöhter Temperatur. Bei einigen wenigen Prozessen ergeben sich starke Zuwächse bzw. Abnahmen der Ausbeute, bei einem einzigen Prozess zeigt sich ein gegenläufiger Trend.

Durch Mittelwertbildung wurde aus den grauen Kurven die schwarze Kurve erzeugt, die den Gesamttrend abbildet. Insgesamt ist durch ein Absenken der Temperatur also ein leichtes Ansteigen der Ausbeute zu erwarten. Bei der Untersuchung, wie sich eine Veränderung der Rührerdrehzahl auf die Ausbeute auswirkt (Modifikationsmethoden 6, 10 und 11), ergaben sich die größten Effekte bei der Ausbeute. Dies ist in Abbildung 43 am Beispiel der Ergebnisse, die aus einer treppenförmigen Veränderung der Rührgeschwindigkeit hervorgingen, dargestellt. Betrachtet man die Kurve für den Gesamttrend, so ergibt sich, dass bei Steigerung der Rührgeschwindigkeit um 10 rpm eine Steigerung der Ausbeute zu erwarten ist, die wesentlich stärker ist, als die Steigerung aus Abbildung 42, die beim Absenken der Temperatur zu erwarten ist. Bei einer Verringerung der Rührgeschwindigkeit wird ein sehr starkes Absinken der Ausbeute prognostiziert. Als Beispiel für eine Veränderung schließlich, die keine bzw. nur sehr geringen Einfluss auf die Ausbeute ausübt, ist in Abbildung 44 die zu erwartende Veränderung der Ausbeute bei Veränderung des Zeitpunktes des Temperatur-Shifts dargestellt. Die Veränderungen der Ausbeute sowohl bei früherem als auch bei späterem Temperatur-Shift sind vernachlässigbar gering.

In Tabelle 8 sind die Ergebnisse der Optimierungsrechnung für alle durchgerechneten Veränderungen an den Prozessvariablen zusammengefasst. Keine oder nur sehr geringe positive Auswirkungen auf die Ausbeute haben Veränderungen des Substrat-Feeds, des Temperatur-Shifts und des Drucks. Bei den Prozessgrößen Temperatur, Begasung und pH-Wert ergeben sich mittlere Auswirkungen auf die Ausbeute, während bei einer Steigerung der Rührgeschwindigkeit die größten Effekte auf die Ausbeute vorhergesagt werden.

[7] Aus Gründen des Datenschutzes können keine Absolutwerte für die Veränderungen der Ausbeute sondern nur normierte, dimensionslose Werte angegeben werden.

Tabelle 8. Zusammenfassung der Resultate der Optimierungsrechnung

Veränderung der Ausbeute	Modifikation			Veränderung der Ausbeute
+	früher	Start des Substrat-Feeds	später	-
-	klein	Höhe der Pulse des Substrat-Feeds	groß	0
++	**-2.5°C**	**Temperatur**	+2.5°C	--
+	klein	Temperatur-Shift (Größe)	groß	-
0	spät	Temperatur-Shift (Zeit)	früh	0
---	-10rpm	**Rührgeschwindigkeit**	**+10rpm**	+++
--	-0.1bar	Druck	+0.1bar	--
++	**schwächer**	**Begasung**	stärker	--
++	**-0.3**	**pH-Wert**	+0.3	--
---	-10rpm	Rührgeschwindigkeit (Pulsprofil)	+10rpm	++
---	-10rpm	**Rührgeschwindigkeit (Treppenprofil)**	**+10rpm**	+++
++	-0.3	pH-Wert (Treppenprofil)	+0.3	--

Bei der Interpretation und Anwendung dieser Ergebnisse spielt Expertenwissen eine sehr große Rolle. Nicht alle Prozessparameter, bei denen die Optimierungsrechnung Potenzial für eine Steigerung der Ausbeute nahe legt, lassen sich ohne weiteres direkt verändern. Oftmals ergeben sich aus einer Veränderung auch weitere Folgen für den Prozess und letztlich auch für die Ausbeute, die nicht im Prozessmodell, also dem neuronalen Netz berücksichtigt werden. Diese Faktoren sind aber bei der Bewertung der Optimierungsergebnisse entscheidend. So ist eine Veränderung der Temperatur, der Begasung und des pH-Wertes nicht ohne weiteres möglich. Da es sich bei dem gebildeten Produkt um ein Enzym handelt, kann z.B. die Veränderung des pH-Wertes oder der Temperatur zu Produktverlust durch Degenerierung oder Inaktivierung führen. Ein weiteres Kriterium ist die Datenbasis, die gerade für die Temperatur, den pH-Wert aber auch die Rührgeschwindigkeit zur Verfügung stand. Diese Werte werden über den Prozessverlauf durch einen Regler konstant gehalten und zeichnen sich innerhalb eines Prozesses nur durch kleine Schwankungen aus. Beim Vergleich mehrerer Prozesse untereinander gibt es zwischen diesen konstant gehaltenen Werten durchaus Unterschiede, wenn auch nur gering. Der Wertebereich bei diesen Prozessgrößen, der dem neuronalen Netz zum Lernen zur Verfügung stand, ist also stark begrenzt und daher muss die Extrapolation auf Temperaturen und Rührgeschwindigkeiten, die stark von dem ursprünglichen Wertebereich abweichen, vorsichtig betrachtet werden. Während sich die Veränderung der Temperatur aus den genannten Gründen als wenig sinnvolle

Maßnahme darstellt, ist eine Steigerung der Rührgeschwindigkeit, auch im Zusammenhang mit den Ergebnissen aus Kapitel 4.3, eine erfolgversprechende Strategie. In Kapitel 4.3 wurde der Einfluss der Viskosität auf den Prozessverlauf und die Ausbeute deutlich gemacht. Durch eine Erhöhung der Rührgeschwindigkeit und als Folge dessen verringerte Viskosität, wird auch vom Standpunkt der statistischen Datenanalyse aus betrachtet dieser Ansatzpunkt, die Erhöhung der Rührgeschwindigkeit, zur Ausbeutemaximierung nahegelegt. In der Praxis ist dieses Konzept jedoch nicht so einfach umzusetzen, da bereits mit maximal zulässiger Rührgeschwindigkeit für den vorhandenen Rührertyp gearbeitet wird. Erhöhter Leistungseintrag mit neuen Rührkonzepten muss neu bewertet werden, da ein verändertes Durchmischungsprofil sowie andere Schereigenschaften zu erwarten sind, die sich unterschiedlich zum existierenden System und den ausgewerteten Daten verhalten werden.

5 In-situ-Mikroskopie und digitale Bildverarbeitung

Einer der wichtigsten Parameter zur Überwachung und Steuerung von Kultivierungen bzw. Fermentationsprozessen ist die Zelldichte. Gängige Messmethoden zur Bestimmung der Zelldichte sind u. a. Trübungsmessung, Leitwertmessung, Messung der optischen Dichte oder Fluoreszenzmessung. Bei den genannten Verfahren ist eine regelmäßige Rekalibrierung erforderlich, da sie von weiteren Parametern abhängen. Störend auswirken kann sich z. B. ein Sensordrift im Kultivierungsverlauf oder Veränderungen in der physikalischen oder chemischen Umgebung. Üblicherweise werden die Ergebnisse dieser Messungen durch eine Offline-Methode, beispielsweise durch Zellzahlbestimmung in einer Zählkammer unter einem Mikroskop, verifiziert. Von Nachteil ist hier, dass die Probenahme und Auszählung zeitaufwändig und fehlerbehaftet ist und dass bei der Probenahme die Gefahr einer Kontamination des Kulturmediums durch kulturfremde Mikroorganismen besteht. Ein Konzept, das die genannten Probleme ausschließt, ist die Weiterentwicklung eines Lichtmikroskops zu einem In-situ-Mikroskop (*in situ* = am Ort, lokal), das das Messsignal, in diesem Fall Bilder aus dem Kulturmedium, direkt im Reaktor aufzeichnen kann. Ein solches In-situ-Mikroskop (ISM) ist am Institut für Technische Chemie der Universität Hannover entwickelt worden [FRERICHS], [BITTNER], [SUHR]. Im Rahmen dieser Doktorarbeit soll ein Softwarepaket entwickelt werden, dass die Steuerung des Mikroskops und die automatische Datenaufnahme und –auswertung erlaubt.

5.1 Das In-situ-Mikroskop

Das In-situ-Mikroskop ist ein Durchlicht-Hellfeld-Mikroskop mit endlich korrigierter Optik. Es besteht aus einem Mikroskopbody mit der Messzone und einem Lineartisch, an dem die Kamera und das Objektiv montiert sind. Das Mikroskop kann seitlich über einen 25 mm Port an einen Fermenter angeschlossen werden. Die Messzone, die sich durch Bewegung des äußeren Tubus öffnen und schließen lässt, ist permanent in das Kultivierungsmedium eingetaucht. Sie wird senkrecht zum Strahlengang durch zwei lichtdurchlässige Saphirglasscheiben (künstlicher Al_2O_3-Einkristall) begrenzt. Die

Beleuchtung der Messzone erfolgt durch eine LED unterhalb der Messzone. Die Bewegung des Tubus und somit auch Höhe und Volumen der Messzone werden durch einen mit Mikrometerschrauben verbundenen Schrittmotor gesteuert. Mit einem zweiten Schrittmotor lässt sich ein Objektiv-Tubus positionieren, mit dem fokussiert wird. In Abbildung 45 ist ein In-situ-Mikroskop der Baureihe III-T gezeigt. Eine detaillierte Abhandlung zu den verschiedenen Baureihen von In-situ-Mikroskopen sowie deren Eigenschaften ist in der Literatur zu finden [FRERICHS], [RUDOLPH].

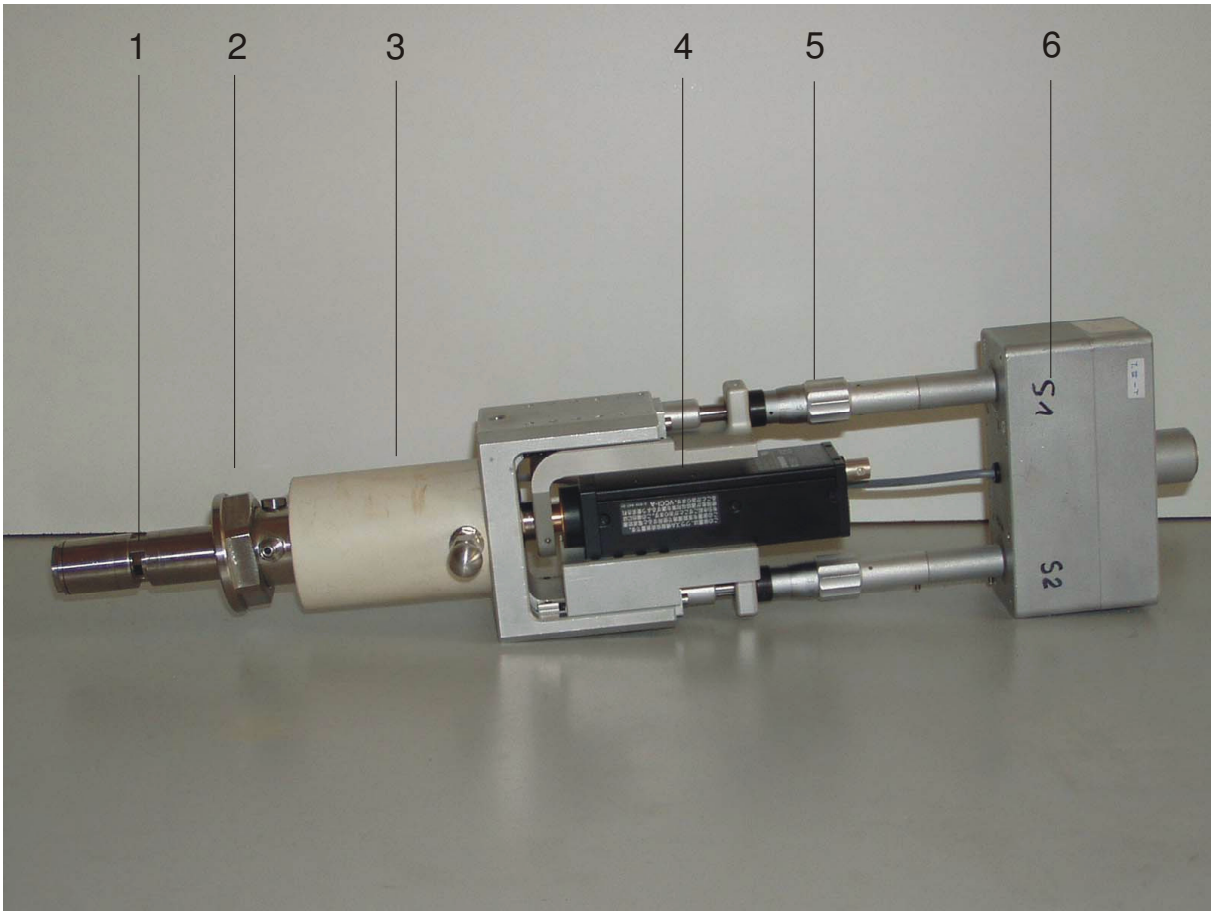

Abbildung 45. In-situ-Mikroskop Typ III-T. 1) Messzone (6,2 mm Höhe), 2) Überwurfmutter zur Befestigung am Reaktor, 3) Mikroskopbody, 4) CCD-Kamera (hier: Sony XCD X-700), 5) Mikrometerschrauben, 6) 2 Schrittmotoren zur Steuerung des inneren und äußeren Mikroskoptubus.

In Abbildung 46 ist schematisch ein an einen Reaktor angeschlossenes In-situ-Mikroskop mit einer Detailvergrößerung der Messzone dargestellt. Das Kulturmedium mit den Zellen passiert durch das Rühren die geöffnete Messzone. Mit der regelbaren LED werden die Objekte durchleuchtet und durch das Objektiv auf die CCD-Kamera

abgebildet. Die Bildinformationen der Kamera werden von der Software *In-situ-Control* empfangen und auf der Festplatte gespeichert. Das Programm soll neben dem Empfangen der Bilder zur Steuerung des Mikroskops dienen. Mit einem zweiten Programm, *In-situ Analyis*, werden die Bilder wahlweise online oder offline ausgewertet.

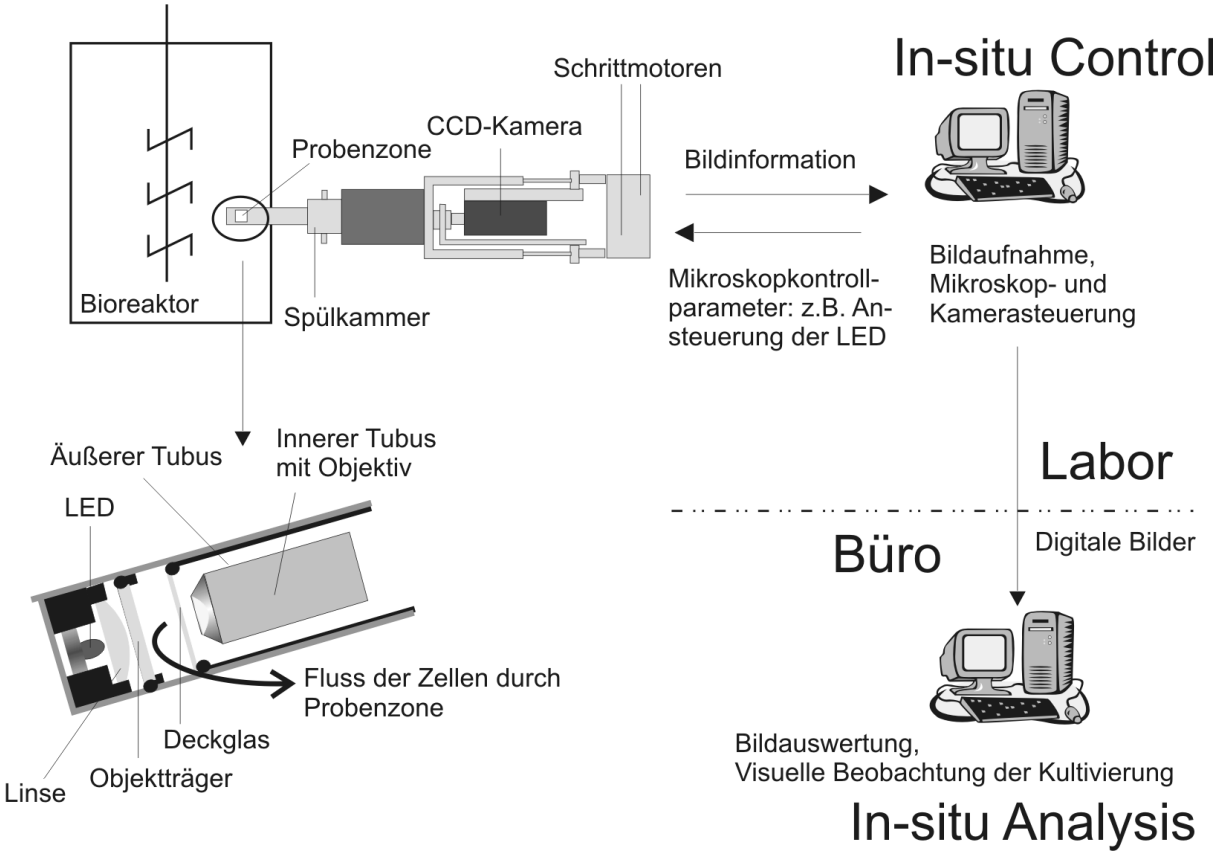

Abbildung 46. Darstellung des Prinzips der In-situ-Mikroskopie. Oben links ist ein an einen Reaktor angeschlossenes Mikroskop gezeigt, mit einer Detailskizze der Probenahmezone (darunter). Ansteuerung des Miskroskops und Aufzeichnung der Bilder geschieht mit der Software In-situ-Control im Labor. Die Auswertung der Bilder geschieht unabhängig davon mit der Software In-situ-Analysis im Büro. Die Auswertung kann online oder offline erfolgen.

5.2 In-situ-Control

In der Planungsphase wurden die gewünschten Eigenschaften des zu entwickelnden Programms in einem Pflichtenheft festgelegt. Die Software musste folgenden Anforderungen genügen:

- das Programm soll beim Starten nach einem angeschlossenen ISM-Controller[8] suchen und automatisch eine Verbindung aufbauen
- die Programmparameter sollen gespeichert und wieder geladen werden können
- die Messzone muss kalibriert werden können (d.h. der Nullpunkt der Messzonenhöhe muss eingestellt werden können)
- die Höhe der Messzone sowie die Objektivsteuerung soll durch grafische Elemente verstellbar sein (d.h. die Schrittmotoren des ISM müssen angesteuert werden können)
- die Intensität der LED im ISM soll einstellbar sein in Stufen von 0 (aus) bis 255 (volle Intensität)
- falls am ISM Abstandshalter (Spacer) zur Vermeidung der vollständigen Schließung der Messzone installiert sind, muss der Benutzer die Höhe der Abstandshalter dem Programm mitteilen können zur korrekten Berechnung des Messzonenvolumens
- die Kameraparameter (Videomodus z. B. 1024x768 oder 1280x1024 Pixel, Bildrate, Belichtungszeit und Kontrast) sollen verstellbar sein
- das von der Kamera empfangene Bild soll dargestellt werden (Live Video) und auch angehalten werden können (Freeze Funktion)
- das Bild soll verkleinert/vergrößert werden können (Zoomfunktion)
- der Bildausschnitt der zum Abspeichern relevant ist (ROI, Region of Interest) soll einstellbar sein
- die Parameter des Messzyklus (d. h. das automatische Abspeichern einer Sequenz von Bildern) müssen durch grafische Elemente einstellbar sein
- der Benutzer soll abgespeicherte (Einzel-)Bilder sowie ganze Messungen laden und komfortabel anzeigen können
- bei jedem Bild, das in einem Messzyklus abgespeichert wird, müssen die Parameter, die bei der Bildaufnahme relevant waren mit abgespeichert werden außerdem muss der Bildname eindeutig sein
- die Messzone soll während des Messzyklus automatisch auf- und zufahren können, um der Klumpenbildung in der Messzone entgegenzuwirken

[8] Zum Betreiben des In-situ Mikroskops notwendiges Zusatzgerät, an einen PC anschließbar über eine serielle Schnittstelle, Entwicklung erfolgte am Institut für Technische Chemie der Universität Hannover

Zur Erstellung der Software wurde die Programmiersprache Delphi 7.0 von Borland eingesetzt. Die Ansteuerung der Kamera erfolgt mit der Software ICImaging Control 1.41 von The Imaging Source Europe GmbH, die in Delphi als Active X Control eingebunden werden kann und somit die Verbindung mit einer über Firewire (IEEE 1394) angeschlossenen Digitalkamera von Delphi aus ermöglicht.

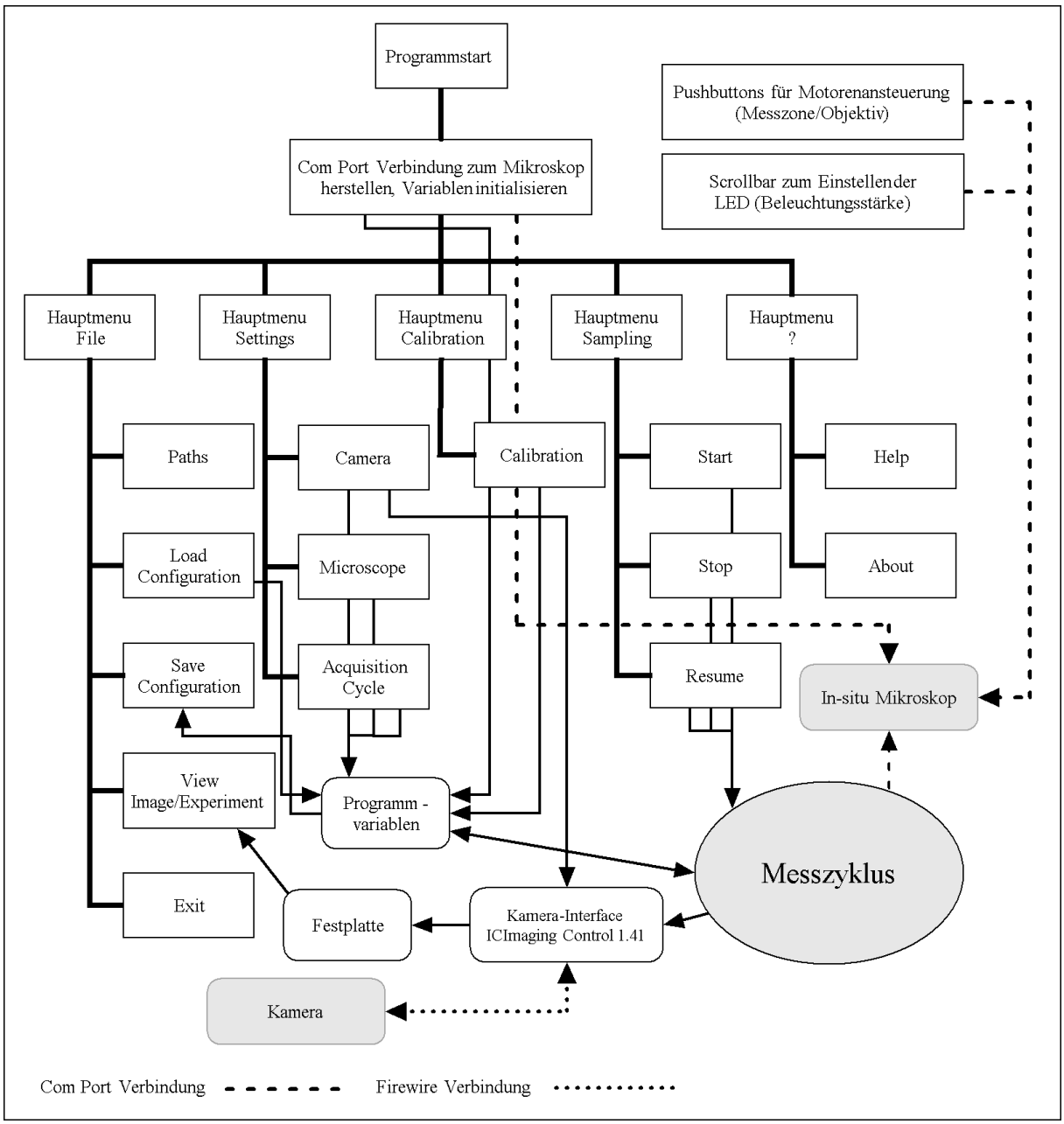

Abbildung 47. Struktur der Software In-situ-Control.

In Abbildung 47 ist die Struktur der Software In-situ-Control schematisch dargestellt. Nach dem Programmstart wird automatisch nach einem über eine serielle Schnittstelle

(Com Port) angeschlossenen Mikroskop bzw. dem ISM Controller (s. Fußnote 8) gesucht. Bei Erfolg wird die Controllernummer im Hauptfenster angezeigt (Abbildung 48 mitte rechts). Danach werden die Programmvariablen auf Standardwerte gesetzt.

Funktionen des Hauptmenus

In Abbildung 47 wird die Menustruktur von In-situ-Control wiedergegeben, sowie die Wirkung der mit den Menueinträgen ausgelösten Funktionen auf die Hardware und auf weitere Programmteile. Mit *Paths* können die Ausgabepfade für Bilddateien und Fehlerlogbücher gesetzt werden. Mit *Load Configuration* bzw. *Save Configuration* können die aktuellen Programmeinstellungen geladen bzw. abgespeichert werden.

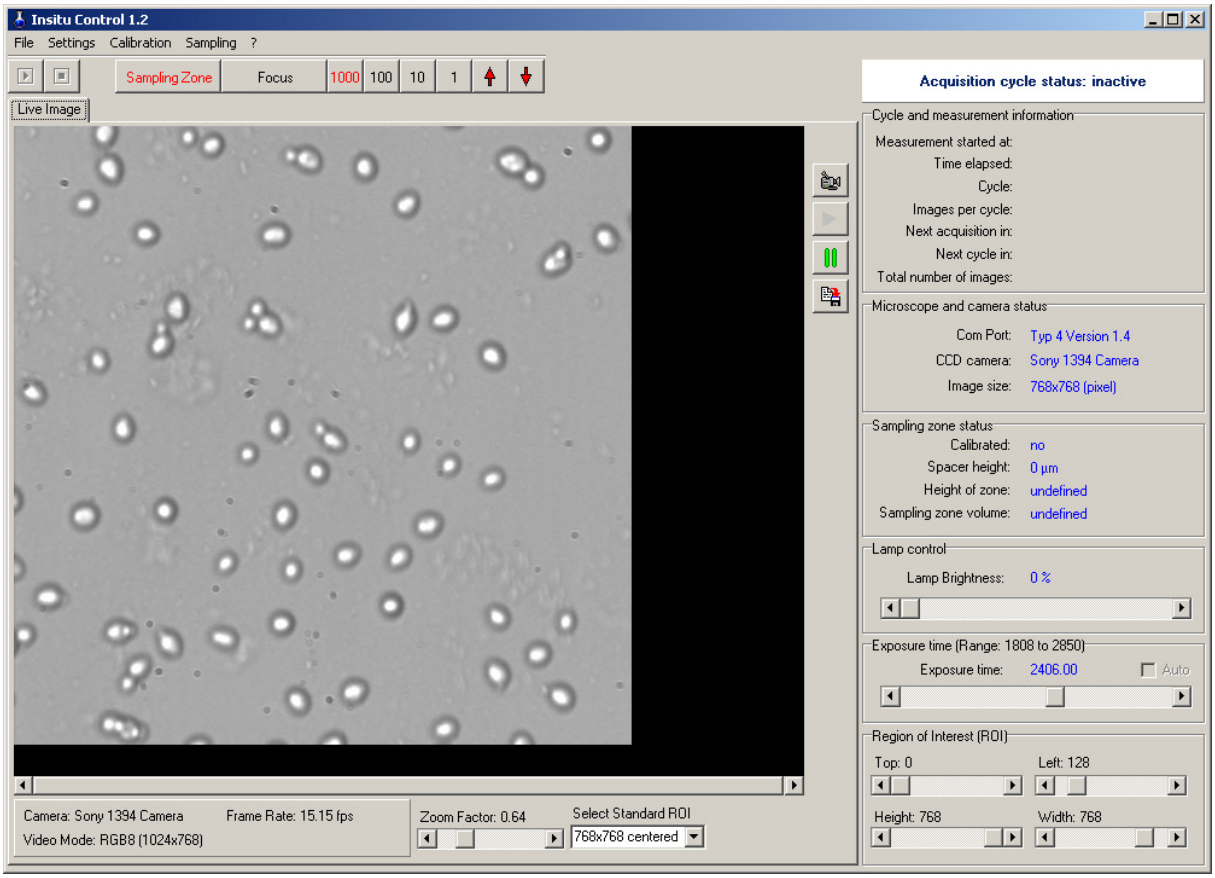

Abbildung 48. Screenshot des Hauptfensters von In-situ-Control.

Als Dateiformat wurde hierfür das gängige Ini-Datei-Format gewählt, das in Delphi durch die bereits implementierte Klasse TiniFile leicht zugänglich ist. Ein Vorteil dieses Formats ist, dass die Daten in Textform als Variablenbezeichner-Wert-Paare vorliegen

und somit der Inhalt der Datei leicht überprüfbar und vom Benutzer mit einem externen Texteditor ohne weiteres editierbar ist.

Mit *View Image* bzw. *View Experiment* können bereits abgespeicherte (Einzel-) Bilder sowie ganze Messreihen bequem visualisiert werden (View Experiment Funktion: siehe Abbildung 49). Durch Wahl des Menueintrags *Camera* wird ein Dialogfenster zur Herstellung der Verbindung mit einer Digitalkamera sowie zum Einstellen der Kameraparameter geöffnet. Mit *Microscope* werden Einstellungen bezüglich Spezifikation und Ausstattung des verwendeten In-situ-Mikroskops gemacht.

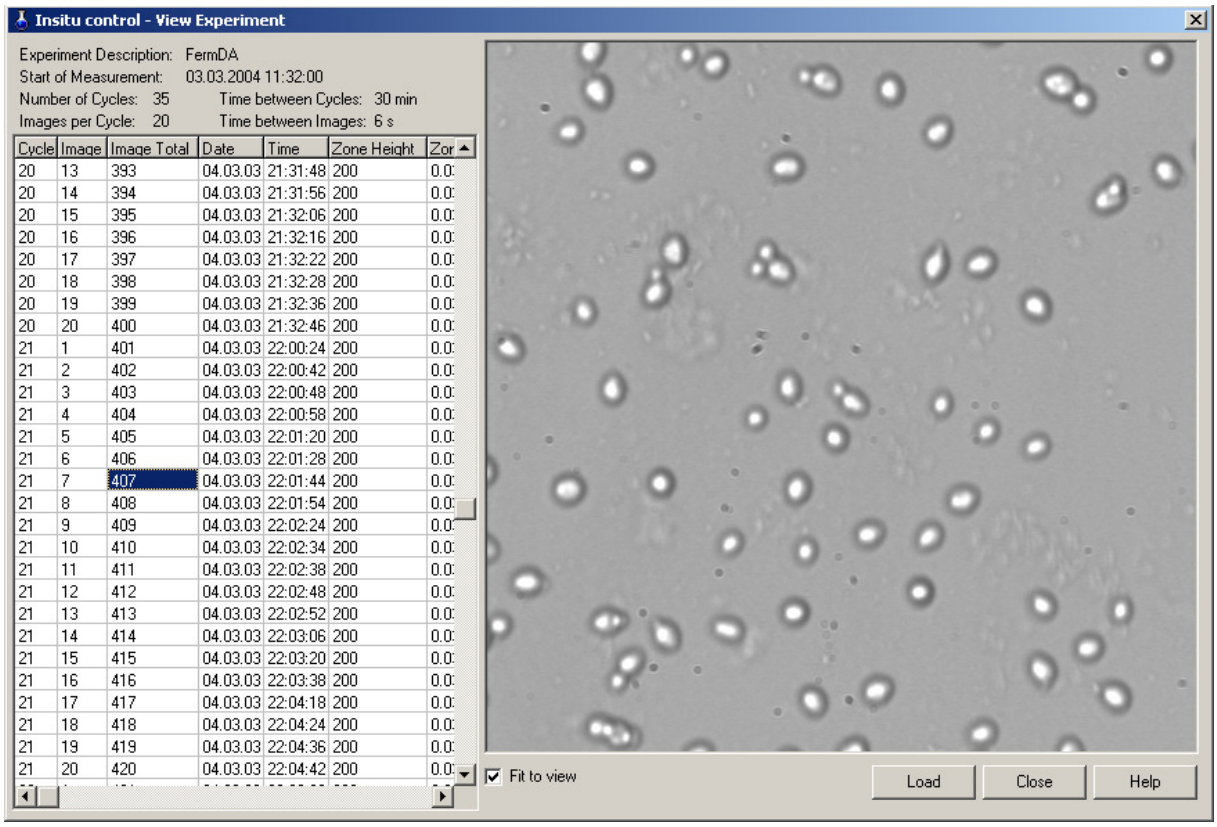

Abbildung 49. Screenshot der „View Experiment"-Funktion von In-situ-Control.

Dazu gehören unter Anderem die Höhe der Spacer, Spindelsteigung und Betriebsweise der Schrittmotoren mit denen die Mikroskoptuben bewegt werden und die Fläche eines Pixels bei dem aktuell gesetzten Videomodus der Kamera und dem aktuell verwendeten Objektiv (z. B. bei einer Auflösung von 1024x768 Pixeln und einem 4fach-Objektiv entspricht ein Pixel einer Fläche von 0,672 µm²). Diese Einstellung ist vor allem wichtig, damit das Programm das Volumen der Messzone berechnen kann. Über das Volumen der Messzone kann bei der späteren Auswertung der aufgezeichneten Bilder

der mit einem Bildverarbeitungsalgorithmus ermittelte Wert für die Zellzahl in eine Zelldichte umgerechnet werden.

Unter *Acquisition Cycle* können die Parameter des Messzyklus festgelegt werden: der Name des Experiments/der Messung, die Anzahl der Bilder pro Zyklus sowie das Zeitintervall zwischen den Bildern und die Anzahl der Messzyklen sowie das Zeitintervall zwischen den Zyklen. Das Aufnehmen der Bilder in Zyklen anstatt die Bilder kontinuierlich oder in einer anderen Art und Weise abzuspeichern hat folgenden Hintergrund: Um die Zellzahl bei einer Kultivierung beispielsweise einmal pro Stunde mittels der Bilder eines In-situ-Mikroskops zu erfassen, kann nicht nur ein Bild pro Stunde aufgezeichnet und ausgewertet werden, da dieses eine Bild nicht unbedingt die tatsächlich vorliegende Zellzahl im Reaktor wiederspiegelt sondern nur eine Momentaufnahme der Zellanzahl in der Messzone ist. Um in die Nähe der tatsächlichen Zellzahl zu kommen, ist das Aufnehmen einer großen Zahl von Bilden und anschließende Mittelwertbildung über die Auswertungsergebnisse vonnöten. Dies lässt sich durch entsprechende Einstellung der Messparameter erreichen, z. B. Zyklenintervall ist eine Stunde, mit 50 Bildern pro Zyklus, die im Abstand von 10 Sekunden aufgezeichnet werden. In diesem Beispiel wird also während des 3600 s andauernden Zyklus 500 s lang gemessen. Danach pausiert die Messung 3100 s bis der nächste Messzyklus beginnt.

Soll die ermittelte Zellzahl in eine Zelldichte umgerechnet werden können, muss das Volumen der Messzone bekannt sein. Dazu ist es nötig, die Messzone zu kalibrieren, was vor jeder Messung mit *Calibration* vorgenommen werden kann. Kalibrieren bedeutet hierbei, dass dem Programm bei geschlossener Messzone mitgeteilt wird, dass die Messzone jetzt geschlossen ist (d. h. den Programmvariablen wird zugewiesen: Messzonenhöhe = 0 µm). Diese Information ist wichtig, da vom Mikroskop keine Rückmeldung über die aktuelle Zonenhöhe verfügbar ist. Von diesem Startwert kann aber die aktuelle Zonenhöhe errechnet werden, da die Ansteuerungswerte des Schrittmotors, die vom Programm an den ISM Controller gesendet werden (Drehrichtung und Schrittanzahl), über die Spindelsteigung in eine Länge umgerechnet werden können. Über die Menüpunkte *Start, Stop* und *Resume* kann der Messzyklus gestartet, abgebrochen bzw. eine alte abgebrochene Messung wieder aufgenommen werden. Die Implementation des Messzyklus wird weiter unter beschrieben.

Funktionen des Hauptfensters

Die Erstellung des Hauptfensters der Anwendung sowie der bereits angesprochenen Dialog- und Eingabefenster wurde mit dem Delphi Form Designer durchgeführt. In Abbildung 48 ist ein Screenshot des Hauptfensters dargestellt. Den größten Teil der Fläche nimmt das Kameradisplay ein, in dem das aktuelle Bild der Kamera dargestellt ist. Zur Anzeige dient ein Objekt der Klasse TICImagingControl, die in Delphi durch Einbinden der oben genannten Software ICImaging Control 1.41 verfügbar wird. Das Objekt hat eine Doppelfunktion: Es ist sowohl das grafische Element, das die von der Kamera empfangenen Bilder anzeigt, als auch Schnittstelle zum Lesen und Setzen von Kameraparametern und zum Auslösen von Funktionen wie z. B. dem Abspeichern von Vollbildern, Filmen und dem Auslesen der Bilddaten Pixel für Pixel. Darunter befindet sich ein Schieberegler zur Einstellung der Vergrößerung (Zoom) und ein Auswahlkasten mit vordefinierten ROIs (Region of Interest, der Teilausschnitt des Bildes, der beim Abspeichern des Bildes berücksichtigt wird). Rechts davon kann mit vier Schiebereglern alternativ ein benutzerdefinierter Bereich angegeben werden. Über der ROI-Einstellung befindet sich je ein Schieberegler zur Einstellung der Belichtungszeit der Kamera sowie zur Wahl der Intensität der LED im Mikroskop. Über diesen befinden sich Statusanzeigen für die Messzone, den Mikroskop- und Kamerastatus und für den Messzyklus. Mit den vier Knöpfen im Kameradisplay kann das Dialogfenster zur Einstellung der Kamera aufgerufen werden, das Live Video gestartet bzw. eingefroren werden und es können Einzelbilder abgespeichert werden. Über dem Kameradisplay ist eine weitere Reihe von Knöpfen angeordnet. Mit den beiden linken Knöpfen kann, alternativ zur Benutzung der Menüpunkte *Start* und *Stop*, der Messzyklus gestartet bzw. gestoppt werden, mit den übrigen können die Schrittmotoren des Mikroskops gesteuert werden. Wahlweise wird der Schrittmotor, der den Objektiv- oder den Messzonentubus bewegt, angesprochen und eine Drehung um 1, 10, 100 oder 1000 Schritte rechts oder links herum ausgelöst. Ist die Messzone kalibriert wird bei einer Bewegung der Messzone aus der alten Höhe der Messzone zusammen mit der Anzahl der zurückgelegten Schritte die Höhe und das Volumen neu berechnet und in der Statusanzeige des Hauptfensters angezeigt.

Der Messzyklus

Der Ablauf des Messzyklus ist durch die im *Acquisition Cycle*-Dialog (siehe Funktionen des Hauptmenus) festgelegten Parameter definiert und stellt sich folgendermaßen dar:

- Initialisieren der Variablen, Vorbereiten der Kamera zur Aufnahme von Bildern
- Hauptschleife: Zyklen
    - Innere Schleife: Bilder
        - Eindeutigen Dateinamen generieren (mit Zeit- und Datumsstempel)
        - Bild unter diesem Namen abspeichern (Endung: *bmp*)
        - Logbuchdatei unten diesem Namen schreiben, die die aktuellen Programm-, Mikroskop- und Kameraparameter enthält (Endung: *log*)
        - Warten bis das eingestellte Zeitintervall für Bilder verstrichen ist
    - Innere Schleife Ende
    - Warten bis das eingestellte Zeitintervall für Zyklen verstrichen ist
- Hauptschleife Ende
- Messzyklus Ende

Durch das Programm werden während des Messzyklus folgende Dateien erzeugt:

- eine Listendatei, die die Namen sämtlicher zu dem jeweiligen Experiment gehörenden Bilddateien enthält (Endung: ise, in-situ experiment)
- die Bilddateien, die im Namen eine Datums- und Zeitsignatur haben sowie Zyklus- und Bildnummer und den Experimentnamen (Endung: bmp)
- pro Bilddatei eine Logbuchdatei mit den Parametern des Programms zum Zeitpunkt der Bildaufnahme (der Name ist mit dem der Bilddatei identisch, Endung: log)

In den ersten Programmversionen, die erstellt wurden (Versionsnummern 0.1 bis 0.9), wurde der Messzyklus als Unterprogramm der Anwendung selbst implementiert. Zusätzlich zur Verwaltung der grafischen Elemente des Hauptfensters, der Aufnahme und Auswertung von Benutzereingaben über Maus und Tastatur und der Ansteuerung von Mikroskop und Kamera, ist also der Haupt-Thread (in diesem Fall also die Anwendung selbst) auch für den Ablauf des Messzyklus zuständig. Bei Testdurchläufen

zeigte sich, dass dieser Umstand zu teilweise schwerwiegenden Problemen führen kann. Wird z. B. bei laufendem Messzyklus vom Benutzer die Titelzeile des Programms mit der Maus aufgenommen (das Fenster geht dadurch in den *Drag&Drop*-Modus) aber zunächst nicht wieder losgelassen, stoppen alle Prozesse der Anwendung, bis die Maus freigegeben wird. Dies hat zur Folge, dass unter Umständen etliche Zeitpunkte, an denen Bilder hätten abgespeichert werden müssen, übersprungen wurden. Weniger schwerwiegend, aber nicht benutzerfreundlich, ist der Effekt, dass das Programm zu den Zeitpunkten, an denen gerade ein Bild abgespeichert wird, auf Benutzereingaben nicht reagiert und der Mauszeiger sich nicht bewegen lässt. Weiterhin ist davon auszugehen, dass es durch Hintergrundanwendungen und Systemprozesse zu Ungenauigkeiten und Verschiebungen in der Zeitmessung und anderen unvorhersagbaren Auswirkungen auf den Ablauf des Messzyklus kommen kann. Damit der Messzyklus und im Besonderen die Zeitmessung der Intervalle zwischen den Bildern und Zyklen möglichst exakt und ungestört durch andere Anwendungen, Systemprozesse oder Benutzereingaben ablaufen kann, wurde in den folgenden Programmversionen (Versionsnummern 1.0 bis 1.2) zur Programmierung des Messzyklus die Klasse TThread genutzt. Durch diese Klasse wird das Starten neuer, von der Hauptanwendung unabhängiger, Threads ermöglicht. Die angesprochenen Probleme der anderen Programmstruktur können dann nicht mehr auftreten. Die Formulierung des Messzyklus mit Hilfe der Klasse TThread war allerdings mit erheblichem Mehraufwand verbunden, da alle Vorgänge, die eine Auswirkung auf das Hauptfenster haben mit diesem synchronisiert werden müssen. Dazu gehören z. B. alle Veränderungen an der grafischen Darstellung von Texten und Bildern im Hauptfenster, die vom Messzkylus-Thread aus vorgenommen werden.

Diese Programmversionen wurde nach ihrer Fertigstellung intensiv getestet. Es wurden alle grafischen Bedienelemente, Dialogfenster und sonstige Programmbestandteile auf ihre korrekte Funktion überprüft. Besonderes Augenmerk lag dabei auf dem korrekten Ablauf des Messzyklus. Es wurden einerseits Testdurchläufe „im Trockenen“, d. h. ohne, dass das Mikroskop an einem Fermenter angeschlossen war, da dies für das Programm keine Rolle spielt, durchgeführt, und andererseits Testdurchläufe während des realen Kultivierungsbetriebs. Erstere wurden, mit dem Rechner auf dem auch die Programmerstellung erfolgte, durchgeführt und letztere auf anderen Rechnern im Technikum des Instituts für Technische Chemie der Universität Hannover bzw. bei Bayer Berkeley in den USA. Bei diesen Tests zeigte sich, dass bei Messreihen von

kurzer bis mittlerer Dauer (weniger als 2 Tage Laufzeit, Bildanzahl kleiner als 10000) das Programm einwandfrei funktionierte. Bei langen Laufzeiten und großer Bildanzahl (länger als 6 Tage, mehr als 50000 Bilder) traten zwei Probleme auf. Zum Einen blieb das Bildintervall innerhalb der Messzyklen nicht konstant. So war bei einer der Testmessungen als Bildintervall eine Sekunde gewählt worden, die anfangs auch exakt eingehalten wurde, nach 50000 Bildern aber bei effektiv zwei Sekunden lag. Die Gesamtdauer der Messung dehnte sich dadurch von geplanten 24 Stunden auf über 72 Stunden aus. Das zweite Problem waren Programmabstürze bei nicht reproduzierbaren Laufzeiten bzw. Bildanzahlen, deren Auftrittswahrscheinlichkeit sich aber mit steigender Programmlaufzeit stark erhöhte. Beide Probleme konnten mit einer Akkumulation der Thread-Anzahl (Anzeige im Taskmanager des Betriebssystems) in Verbindung gebracht werden. Die Ursache dieses Anstiegs konnte trotz intensiver Durchsicht des Quellcodes sowie Recherche in mehreren Internetforen über Multithread-Programmierung in Delphi nicht zweifelsfrei geklärt bzw. behoben werden. Eine Vermutung ist, dass die Probleme dadurch entstehen, dass das Kamerainterface TICImagingControl Bestandteil der Hauptanwendung ist, aber vom Messzyklus-Thread auf dessen Funktionen zugegriffen werden muss. Es ist denkbar, dass das Kamerainterface dafür nicht geeignet ist. In der Programmversion 1.3 schließlich wurde der Messzyklus, basierend auf den Erfahrungen mit der zweiten Version, wieder als Unterprogramm der Hauptanwendung implementiert. Im Unterschied zur den Programmversionen 0.1 bis 0.9 wurde der Algorithmus so umstrukturiert, dass er nicht in mehreren Programmschleifen abgearbeitet wird, sondern als zeitgesteuerte Ereignisfunktion aufgerufen wird. Dazu wurde die Klasse TTimer verwendet. Mit dieser kann ein Unterprogramm, in diesem Fall das Messzyklus-Unterprogramm, in kontinuierlichen Zeitabständen, etwa alle 500 ms, immer wieder aufgerufen werden. In diesem Unterprogramm wird ausgehend von dem Startzeitpunkt der Messung bei jedem Aufruf durch Vergleich mit der aktuellen Zeit ermittelt, ob ein Bild abgespeichert werden muss, wenn ja zu welchem Messzyklus es gehört, und welche Bildnummer es hat (wichtig für die automatische Erstellung des Dateinamens). In seinen wesentlichen Eigenschaften ist dieser Timer-Ansatz mit dem Thread-Ansatz identisch. Unterschiede gibt es vor Allem bei kleinen Bildintervallen, bei der der Timer keine exakte Zeitmessung gewährleistet. Bei Zeitintervallen von ca. 500 ms aufwärts spielt das Problem keine Rolle, da die Ungenauigkeiten des Timers dann klein werden im

Vergleich zum eingestellten Intervall. Der wichtigste Unterschied zum Thread-Ansatz ist aber die verbesserte Langzeitstabilität des Programms. Bei Tests konnten Messungen mit einer Länge von 21 Tagen ohne Probleme durchgeführt werden.
Somit stehen zur Aufnahme von Bildern bzw. Messreihen von Bildern zwei Versionen der Software In-situ-Control zur Verfügung. Für kurze Messungen, bei denen Bilder mit Intervallen von wenigen Millisekunden aufgezeichnet werden müssen, kann In-situ-Control 1.2 herangezogen werden, da dieses Programm durch den Thread-Ansatz eine exaktere Zeitmessung gewährleistet. Für Langzeitexperimente kann die Version 1.3 genutzt werden.

5.3 In-situ-Analysis

Der Zweck der Software In-situ-Analysis liegt in der Auswertung von Bildern einer Zellkultur bzw. Fermentation, die mit einem In-situ-Mikroskop aufgenommen worden sind (aber auch Bildern, die aus anderen Quellen stammen können). Unter Auswertung ist zu verstehen, dass Informationen wie Zellzahl, Fläche bzw. Volumen der Zellen, Klassifizierung von Zellen usw. aus den Bildern extrahiert werden. Alle Funktionen müssen über eine grafische Benutzeroberfläche zugänglich sein. Des weiteren ist die Flexibilität der Software von entscheidender Bedeutung, sodass der Benutzer die Möglichkeit hat neue Auswertungsalgorithmen hinzuzufügen. Das bedeutet, dass die Algorithmen, die zum Lieferumfang von In-situ-Analysis gehören, nicht fest in die exe-Datei eingebunden sind, sondern erst zur Laufzeit des Programms aus einer DLL geladen werden. Die Funktionen, die In-situ-Analysis zur Anzeige und Nachbearbeitung der Rechenergebnisse anbietet, müssen so gestaltet sein, dass sie sich an den gerade aktiven Auswertungsalgorithmus anpassen. In der Planungsphase wurden somit folgende Eigenschaften für die Software festgelegt:

- Das Programm soll mit In-situ-Control zusammenarbeiten, d. h. eine laufende Messung in In-situ-Control soll mit In-situ-Analysis online ausgewertet werden können.
- Bereits abgeschlossene Experimente, die mit In-situ-Control erzeugt wurden, müssen auch ausgewertet werden können.

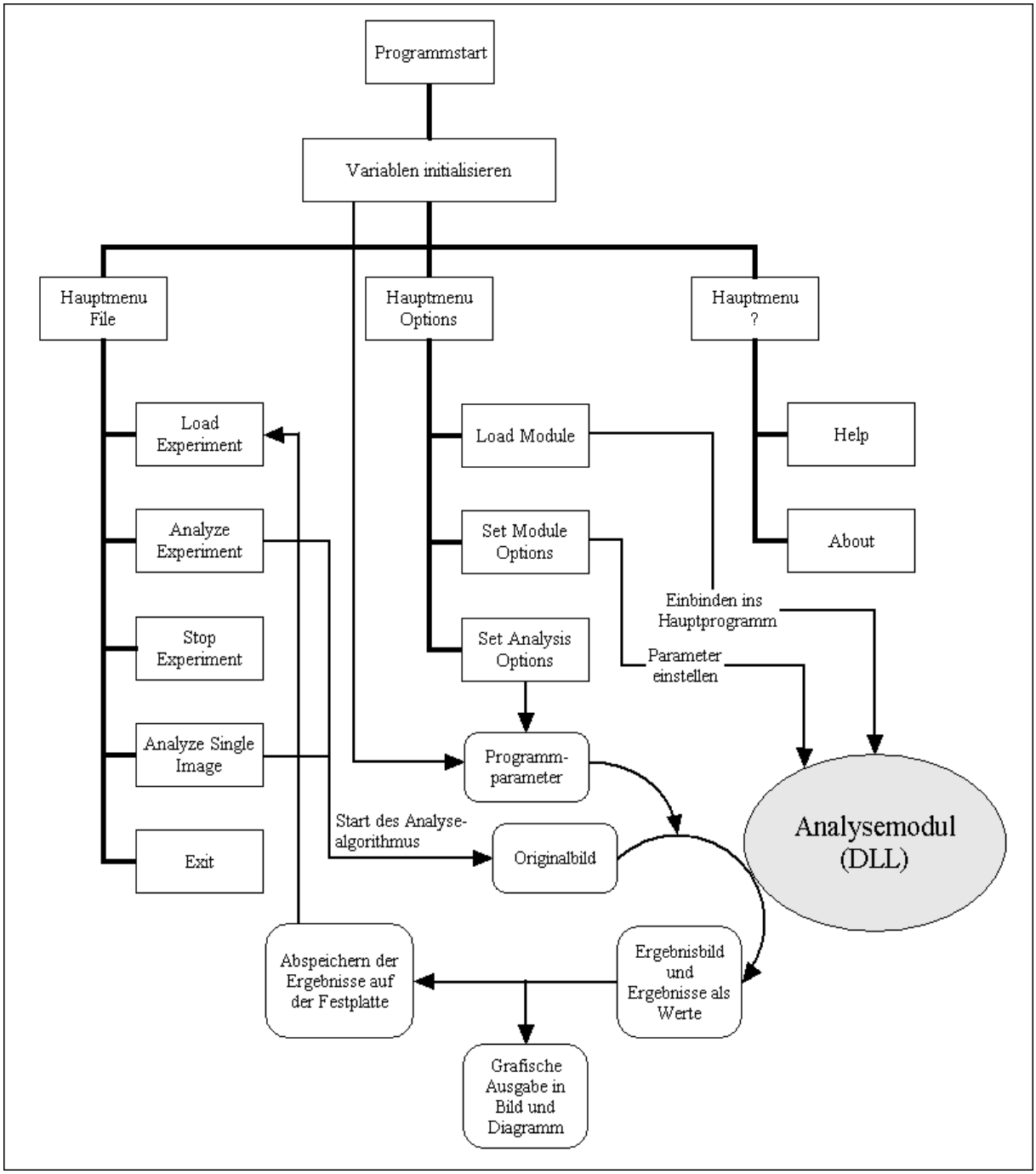

Abbildung 50. Struktur der Software In-situ-Analysis. Das Analysemodul ist vom Programm unabhängig und wird erst zur Laufzeit ins Programm eingebunden.

- Ergebnisse früherer Auswertungen müssen vom Programm wieder geladen und angezeigt werden können.
- Die Auswertung erfolgt nicht durch in das Programm eingebundene Auswertealgorithmen, sondern durch Algorithmen, die zur Laufzeit aus DLLs geladen werden.

- Da die Rechenergebnisse je nach geladener DLL unterschiedlich sind (in Anzahl und Qualität) muss die Visualisierung der Ergebnisse flexibel sein. Grundsätzlich soll es aber folgende Anzeigen geben: das Originalbild, das Resultatbild, eine Tabelle (Zellzahl pro Bild, Klassifizierung der gefundenen Objekte usw.) und Darstellung der Ergebnisse als Diagramm. Die so angezeigten Daten sollen exportiert werden können (als txt, bmp etc.)

Zur Erstellung der Software wurde die Programmiersprache Delphi 7.0 von Borland eingesetzt. In Abbildung 50 ist die Struktur der Software schematisch dargestellt.

Funktionen des Hauptmenus

Im Hauptmenu *File* können bereits ausgewertete Experimente geladen, Analysedurchläufe gestartet und abgebrochen werden und es können Einzelbilder ausgewertet werden. Eine Beschreibung des Analysevorgangs folgt weiter unten. Mit Ausnahme der Ladefunktion sind die genannten Funktionen erst dann verfügbar, wenn mit *Load Module* ein Auswertemodul in In-situ-Analysis eingebunden worden ist. Nach dem erfolgreichen Einbinden können mit *Set Module Options* die Parameter des Analysealgorithmus eingestellt werden. Die Programmparameter können durch *Set Analysis Options* eingestellt werden.

Aufbau eines Analysemoduls

Sinn des modularen Aufbaus von In-situ-Analysis ist es, dass das Hauptprogramm nicht mehr verändert werden muss, wenn neue Algorithmen zur Bildauswertung verfügbar werden bzw. wenn bestehende Algorithmen verändert werden sollen. Unerlässlich für den reibungslosen Betrieb in In-situ-Analysis ist aber, dass die Algorithmen in einer DLL mit einer bestimmten Strukturierung vorliegen. Zwischen Hauptprogramm und DLL muss ein Informationsfluss über die Anzahl und Art der vom Algorithmus generierten Ergebnisse stattfinden können, sodass diese vom Hauptprogramm in geeigneter Weise dargestellt und exportiert werden können. Die Art von Ergebnissen, die allgemein betrachtet aus einem Bild gewonnen werden können, sind: (i) Aussagen über das gesamte Bild (z. B. Anzahl von Zellen auf dem Bild), (ii) Aussagen über ein

Objekt auf dem Bild (z. B. Pixelfläche des Objekts) und (iii) Klassifizierung der gefundenen Objekte (z. B. Einzelzelle, Doppelzelle, Cluster, ...). Basierend auf dieser Überlegung muss jede DLL, die in In-situ-Analysis eingebunden werden soll, folgende Funktionen aufweisen (mit *genau* dem hier angegebenen Namen und *genau* den hier aufgeführten Parametern und Rückgabewerten):

- GetSummaryVarNum = function: integer;
- GetSummaryVarName = function(num: integer): PChar;
- GetSummaryVar = function(num: integer): extended;
- GetObjectVarNum = function: integer;
- GetObjectVarName = function(num: integer): PChar;
- GetObjectVar = function(num, objectnum: integer): extended;
- GetObjectClassNum = function: integer;
- GetObjectClassName = function(num: integer): PChar;
- GetObjectClass = function(objectnum: integer): integer;
- GetObjectClassSum = function(num: integer): integer;

Mit den „Num-Funktionen“ wird jeweils die Anzahl der vom Algorithmus generierten Ergebniswerte abgefragt, mit den „Name-Funktionen“ die Bezeichnung des betreffenden Wertes und mit den „Var-Funktionen“ kann der Wert selbst abgefragt werden (nach der durchgeführten Bildauswertung). Mit diesen Angaben ist es dem Hauptprogramm möglich, die im Analysemodul ermittelten Werte gemeinsam mit ihrer Bedeutung und einer für den Benutzer gut verständlichen Beschriftung zu speichern und in Form von Tabellen bzw. Diagrammen darzustellen.

Mit einem ähnlichen Konzept wurde das Lesen und Setzen der Algorithmusparameter realisiert. Wenn In-situ-Analysis ein neues Modul lädt, muss die Information ob, und wenn ja, wie viele Parameter das Modul hat, übermittelt werden. Dazu dient die Funktion GetParameterNum. Außerdem ist für das Setzen der Modulparameter vom Hauptprogramm aus (Hauptmenufunktion *Set Module Options*) entscheidend, welche Wertebereiche und Standardwerte die Parameter jeweils haben. Dies geschieht mit den Funktionen GetParameterMin, GetParameterMax und GetDefaultParameter. Um das *Set Module* Options-Dialogfenster des Hauptprogramms auch mit Beschriftungen und Erläuterungen zu den jeweiligen Parametern zu versehen, können mit den Funktionen GetParameterName bzw. GetParameterDescription entsprechende Textinformationen abgefragt werden. Die Parameterwerte selbst werden mit GetParameter bzw.

SetParameter gelesen bzw. gesetzt. Die Funktionen, die also im Zusammenhang mit den Parametern benötigt werden, sind:

- GetParameterNum = function: integer;
- GetParameterName = function(num: integer): PChar;
- GetParameter = function(num: integer): extended;
- SetParameter = function(num: integer; param: extended): integer;
- GetParameterMin = function(num: integer): extended;
- GetParameterMax = function(num: integer): extended;
- GetDefaultParameter = function(num: integer): extended;
- GetParameterDescription = function(num: integer): PChar;

Von den bisher genannten Funktionen sind einige optional und müssen nicht zwangsläufig in der DLL enthalten sein. So können z. B. alle Parameter-Funktionen weggelassen werden, wenn GetParameterNum Null zurückgibt, denn dann wird das Hauptprogramm keine dieser Funktionen jemals verwenden. Folgende Funktionen sind zum Betrieb aber zwingend erforderlich:

- GetDescription = function: PChar;
- Analyze = function(bmpfile: PChar): integer;
- SaveResultImage = function(filename: PChar): integer;

Die erste liefert einen Text, der eine Beschreibung der Anwendungsmöglichkeiten des Algorithmus beinhaltet. Dieser wird im „*Load Module*“-Dialogfenster angezeigt und soll den Anwender bei der Auswahl des für seine Daten passenden Algorithmus unterstützen. Der Funktion Analyze wird als Parameter der Dateiname eines Bildes übergeben, das ausgewertet werden soll. Diese Funktion der DLL ist somit der Auswertealgorithmus selbst, während alle bisher genannten Funktionen nur zur Vorbereitung und Einstellung dienen. Der Rückgabewert ist der Exitcode des Algorithmus: 0 bedeutet Erfolg, 1 bedeutet die Datei existiert nicht oder ist kein Bild im Bitmap-Format und 2 bedeutet, dass im Algorithmus selbst ein Fehler aufgetreten ist und das Bild nicht ausgewertet werden kann. Die letzte Funktion, SaveResultImage, dient zum Abspeichern eines Ergebnisbildes, das vom Algorithmus generiert wurde. In diesem können dann z. B. die gefundenen Objekte markiert dargestellt werden.

Wird vom Anwender das „*Load Module*“-Dialogfenster geöffnet, werden alle im Programmverzeichnis existierenden Dateien mit der Endung *dll* eingelesen und auf das Vorhandensein der oben genannten Funktionen untersucht. Sind alle nicht-optionalen

Funktionen vorhanden, ist eine DLL somit für In-situ-Analysis nutzbar. Nun werden mit GetDescription auch die Beschreibungen der Module geladen und im Dialogfenster gemeinsam mit den Namen der gefundenen Module als Liste angezeigt. Nun kann vom Anwender ein Modul ausgewählt und geladen werden. Daraufhin werden die Menupunkte im Hauptmenu, die sich auf das Starten der Auswertung beziehen, aktiviert.

Bildanalyse mit In-situ-Analysis

Nach der Auswahl von *Analyze Experiment* im Hauptmenu muss vom Anwender eine ise-Datei (siehe Kapitel 5.2) gewählt werden. In dieser Datei stehen die Namen aller zu dem Experiment gehörenden Bilddateien. In einer Programmschleife werden alle Bildnamen nacheinander an die Funktion Analyze des geladenen Auswertemoduls übergeben. Nach erfolgreicher Auswertung eines Bildes (Exitcode = 0) werden mit den o. g. Funktionen (GetObjectVar, etc.) die Ergebniswerte aus dem Speicher der DLL ins Hauptprogramm transferiert und in einer dynamischen Datenstruktur gespeichert. Die Ergebnisse werden im Hauptfenster in einer Tabelle und als Diagramm dargestellt. Die eingangs erwähnte Programmschleife wird mit einem Timer in bestimmten Zeitabständen immer wieder gestartet. Dies ist wichtig, wenn die Auswertung und Aufnahme des Experiments parallel verläuft. Der ise-Datei werden von In-situ-Control ständig neue Bildnamen hinzugefügt, die durch die Timer-Konstruktion von In-situ-Analysis in den Wartezeiten zwischen den Aufnahmezyklen sofort ausgewertet werden können.
Neben dem Übertragen der Auswertungsergebnisse in eine dynamische Datenstruktur werden die Resultate jedes einzelnen Bildes auch in einer Textdatei festgehalten. Die Textdatei erhält den Dateinamen des entsprechenden Bildes, aber mit der Endung txt. Diese Dateien sind auch später wichtig, wenn bereits ausgewertete Experimente zwecks komfortabler Anzeige der Ergebnisse wieder geladen werden sollen. Die Textdateien sind so gestaltet, dass sich aus ihnen die komplette dynamische Datenstruktur der Ergebnisse wieder generieren lässt.

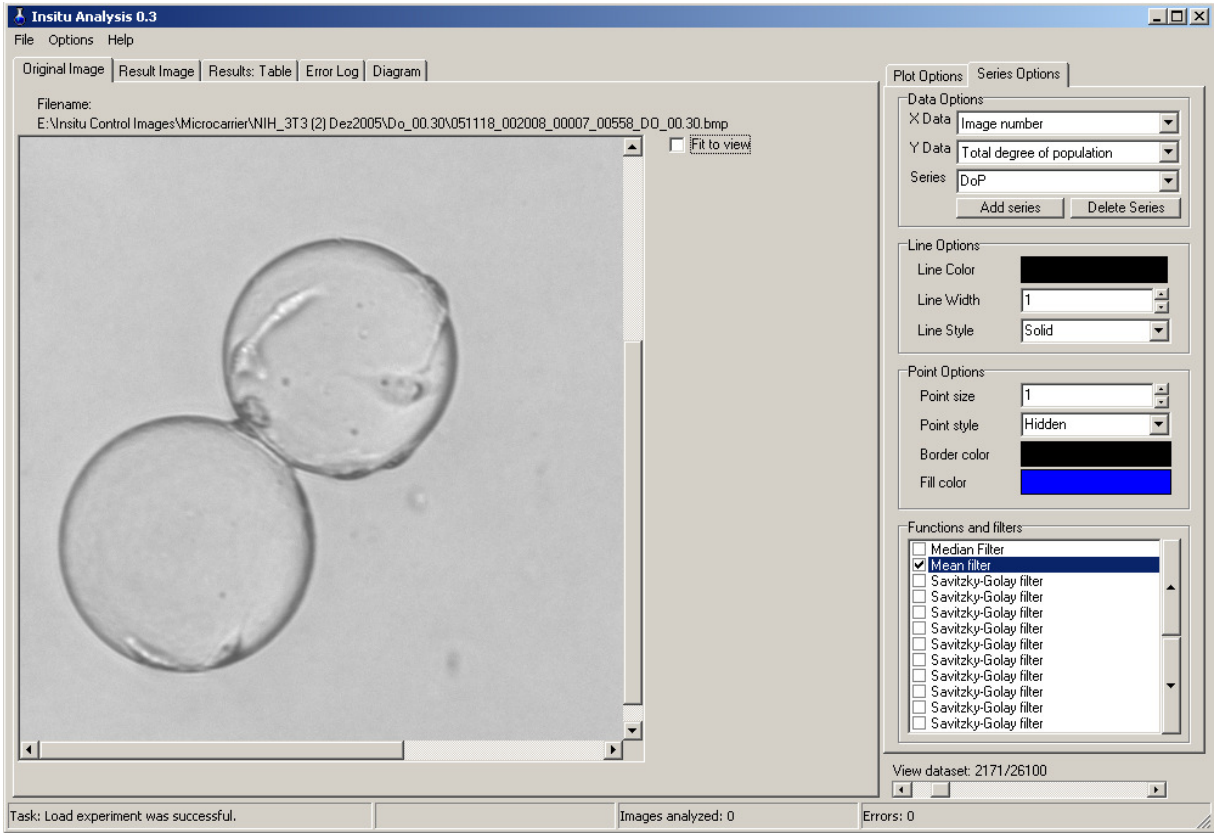

Abbildung 51. Screenshot von In-situ-Analysis. Im Anzeigefenster ist ein mit Tierzellen bewachsener Microcarrier zu sehen.

Funktionen des Hauptfensters

Den Großteil des Hauptfenster nimmt das Anzeigefenster für Grafiken und Tabellen ein, dessen Inhalt in fünf Klappkarten organisiert ist. Dargestellt werden können wahlweise das Originalbild (siehe Abbildung 51), das Ergebnisbild, die Analyseergebnisse des aktuell gewählten Bildes als Tabelle, das Logbuch der während der Analyse aufgetretenen Fehler und die Ergebnisse der Auswertung als Diagramm.

In der Statusleiste am unteren Bildrand wird die vom Programm aktuell durchgeführte Aufgabe und die Anzahl der ausgewerteten Bilder und der aufgetretenen Fehler angezeigt. Den rechten Teil des Hauptfensters bilden zwei Klappkarten mit Diagrammoptionen. Dies sind u. A. die Auswahl, welche Datenreihe gegen die Bildzahl oder die Kultivierungszeit aufgetragen werden soll, die Auswahl von Linienart, -farbe, -stärke und Markierungen sowie die Auswahl von Filtern. In der momentan vorliegenden Fassung sind drei Filter implementiert: Mittelwert-, Median- und Savitzky-Golay-Filter. Mit dem ersten Filter kann die schon in Abschnitt 5.2 angesprochene Mittelwertbildung

über alle Bilder eines Messzyklus vorgenommen werden. So kann z. B. bei vorliegenden Werten über die Zellzahl, von allen Bildern mit der gleichen Zyklusnummer jeweils der Mittelwert gebildet und ein Diagramm Zellzahl gegen Zyklusnummer bzw. Zeit erstellt werden. Alternativ zum Mittelwert kann auch der Median eingesetzt werden. Durch den dritten Filter, den Savitzky-Golay-Filter (Fensterbreite: 5 Werte), kann eine Glättung der Werte erreicht werden. Es ist in der Liste der Filter mehrfach anwählbar und kann somit auch mehrfach auf eine Datenreihe angewendet werden. Das fertige Diagramm kann schließlich als Bitmap-Grafik exportiert oder in der Zwischenablage abgelegt werden oder die Werte der angezeigten Datenreihen können als Textdatei abgespeichert werden.
Mit der entwickelten Software In-situ-Analysis können ISM-Experimente leicht eingelesen und mit entsprechenden Bildverarbeitungs-Modulen ausgewertet werden. Die Ergebnisse lassen sich in grafischer Form anzeigen oder zur weiteren Verarbeitung als Textdateien abspeichern. Durch die flexible Struktur ist das Nachrüsten neuer Analysemodule sehr leicht möglich. Somit kann das Programm bei einer großen Zahl verschiedenster Auswertungen eingesetzt werden.

5.4 Analysemodule

Im Folgenden werden zwei schon vorhandene sowie ein selbstentwickelter Analysealgorithmus und ihre Implementation in einem durch In-situ-Analysis verwendbaren Analysemodul beschrieben.

5.4.1 Double Yeast Cell Counter (DYCC)

Der DYCC-Algorithmus wurde am Institut für technische Chemie in Hannover entwickelt [KRABICHLER]. Der Algorithmus kann Bilder mit defokussiert aufgenommenen Zellen auswerten. Auf diese Weise aufgenommen, wirken die kugelförmigen Zellobjekte wie Linsen, bündeln das Durchlicht in der Zellmitte und erscheinen dadurch in der Mitte heller und am Rand dunkler als der Hintergrund. Der Algorithmus liefert über ein ausgewertetes Bild folgende Informationen: Anzahl der gefundenen Objekte, Fläche der Objekte (in Pixeln) und Anzahl der gefundenen Zellkerne (ermittelt durch die oben erwähnten hellen Bereiche in der Mitte der Zellen).

Außerdem können die Zellen in Einzel- und Doppelzellen sowie in Zellcluster klassifiziert werden. Der Algorithmus, ursprünglich in C++ geschrieben, wurde zwecks Optimierung und Transferierung in eine durch In-situ-Analysis nutzbare DLL-Form nach Delphi übersetzt. Dabei wurden gegenüber der ursprünglichen Version folgende Änderungen vorgenommen:

- In der C++-Version wird eine Klasse *Pixel* implementiert, die dazu dient Informationen über einen Pixel des auszuwertenden Bildes zu speichern. Dies ist z. B. die Information, zu welchem Objekt der Pixel gehört, ob er sich auf dem Rand, innerhalb oder zwischen zwei Objekten befindet usw. Diese Informationen werden im Laufe der Auswertung über get- und set-Methoden der Klasse *Pixel* gespeichert bzw. ausgelesen. Um jeden Pixel des Bildes zu erfassen, wird somit ein „array of *Pixel*„ benötigt. In der Delphi-Version des Algorithmus wurde der Ansatz mit der Klasse *Pixel* nicht übernommen, sondern es wurden mehrere Arrays verwendet, die die zu den Bildpunkten gehörenden Informationen direkt speichern.
- Der Algorithmus wurde um die Berechnung der durchschnittlichen Zellgröße sowie der Zellkonzentration (in Anzahl pro Milliliter, berechnet mit Hilfe des Volumens der Messzone des In-situ-Mikroskops) erweitert.

Der Algorithmus wurde zusammen mit den entsprechenden Schnittstellenfunktionen zum Anbinden an In-situ-Analysis (siehe Abschnitt 5.3) in eine DLL kompiliert. Die korrekte Umsetzung des Algorithmus wurde anhand einer Hefezellenkultivierung validiert. Die Bilddaten wurden sowohl mit dem ursprünglichen Algorithmus als auch mit dem neuen DYCC-Modul in In-situ-Analysis ausgewertet. Die Analyseergebnisse waren identisch. Einzig die Bearbeitungszeit war stark unterschiedlich: Der ursprüngliche Algorithmus benötigt 45 Sekunden für 100 Bilder im Format 512 mal 512 Pixel, während der nach Delphi übersetzte Algorithmus 11 Sekunden für dieselben 100 Bilder benötigt (gemessen auf einem Rechner mit einer Intel Pentium 4 CPU mit 3,2 GHz, 1 MB DDR-SDRAM und Windows XP mit Service Pack 2). Die erhöhte Bearbeitungsgeschwindigkeit konnte, zumindest zum größten Teil, auf den oben angesprochenen Unterschied bei der Speicherung der Pixelinformationen zurückgeführt werden. Die Speicherung der Werte mit Objekten läuft über set- und get-Methoden, was einen Funktionsaufruf dieser Methoden *und* das Ablegen des Wertes nötig macht, während im anderen Fall der Wert einfach im Speicher abgelegt wird. Die Zeitverluste

durch die zusätzlichen Funktionsaufrufe sind gering. Da aber der Schreib-/Lesevorgang millionenmal im Algorithmus ausgeführt wird, summieren sich die Verluste bei der alten Version stark auf. In einem Testprogramm, welches Werte sowohl durch die set-Methode eines Objekts speichert als auch direkt in einem Array, konnte gemessen werden, dass ersteres ca. die dreifache Zeit in Anspruch nimmt.

5.4.2 Auswertungsalgorithmus für BHK-Zellen

Für einen schon in Form einer DLL vorliegenden Auswertungsalgorithmus [MARTINEZ], der dazu dient, Bilder mit BHK-Zellen auszuwerten, wurde in Delphi eine Interface-DLL erstellt. In dieser sind die in Abschnitt 5.3 angesprochenen Funktionen implementiert, die zum Aufrufen eines Algorithmus aus In-situ-Analysis benötigt werden. Die Funktion *Analyze* der Interface-DLL dient dann lediglich dazu, den Funktionsaufruf, der bei der Auswertung eines Bildes ausgelöst wird, an die eigentliche DLL weiterzuleiten.

5.4.3 Bestimmung des Bewuchsgrades von mit Zellen bewachsener Microcarrier

Da das Anwendungsspektrum des In-situ-Mikroskops durch die Beobachtung von Suspensionszellen nicht ausgeschöpft ist, wurden Untersuchungen zur Beobachtung von adhärenten Zellen unternommen. Hierbei wurde als Ansatzpunkt die Kultivierung von Tierzellen auf Trägern, sogenannten Microcarriern, gewählt.
Am Institut für Technische Chemie in Hannover wurde eine Kultivierung von NIH-3T3 Zellen (Mausmuskel, Fibroblasten) in einem 2 L Fermenter durchgeführt. Die Konzentration von Cytodex 1 (Amersham, Schweden) Microcarriern in der Kulturbrühe betrug 2 g/L. Weitere experimentelle Bedingungen sind in der Literatur beschrieben [RUDOLPH 2]. Während des Experiments sind mit einem In-situ-Mikroskop 26100 Bilder in 29 Messzyklen mit 40 Minuten Zeitintervall und je 900 Bildern aufgezeichnet worden. Zur Auswertung dieser Daten war es notwendig einen Algorithmus zu entwickeln, der es erlaubt die Microcarrier auf den Bildern zu erkennen und den Bewuchsgrad der Microcarrier zu berechnen. Die Entwicklung des Algorithmus sowie

dessen Einbindung in ein Analysemodul wurde mit der Programmiersprache Delphi 7.0 von Borland durchgeführt.

Erkennung der Microcarrier mit Methoden der digitalen Bildverarbeitung

Der erste Schritt für die Erkennung der Microcarrier ist ihre Separation vom Hintergrund des Bildes. Dabei kann die Tatsache ausgenutzt werden, dass unabhängig davon ob sie nicht, schwach oder stark bewachsen sind, zwischen Hintergrund und Microcarrier immer ein deutlich erkennbarer Grauwertunterschied auftritt. Zum Auffinden von Grauwertunterschieden werden in der digitalen Bildverarbeitung lokale Operatoren wie z. B. das Roberts-Cross oder der Sobel-Operator (s. Abschnitt 2.6.1) eingesetzt. In einer Voruntersuchung wurden beide Operatoren auf ihre Tauglichkeit zur Erkennung der Kanten der Microcarrier überprüft. Dazu wurden aus dem vorliegenden Datenmaterial 35 Bilder ausgewählt. In dem mit Delphi erstellten Testprogramm wurden die Bilder eingelesen und in eine Grauwertmatrix g umgewandelt. Mit Formel {36} wurde die Matrix G errechnet, die die Beträge der Grauwertgradienten enthält. Die Randbereiche von G (ein Pixel Breite, nach allen Seiten) konnten nicht berechnet werden, da dort der Konvolutionskern über den Rand des Originalbildes hinausgeht. Die entsprechenden Werte in G wurden auf 0 gesetzt. Wird G als Graustufenbild dargestellt, erscheinen Kanten des Originalbildes, also z. B. die Begrenzungslinien der Microcarrier, weiß und der Hintergrund sowie ein Großteil der Innenbereiche der Microcarrier schwarz. In Abbildung 52 ist dies beispielhaft für ein Bild mit zwei Microcarriern gezeigt.

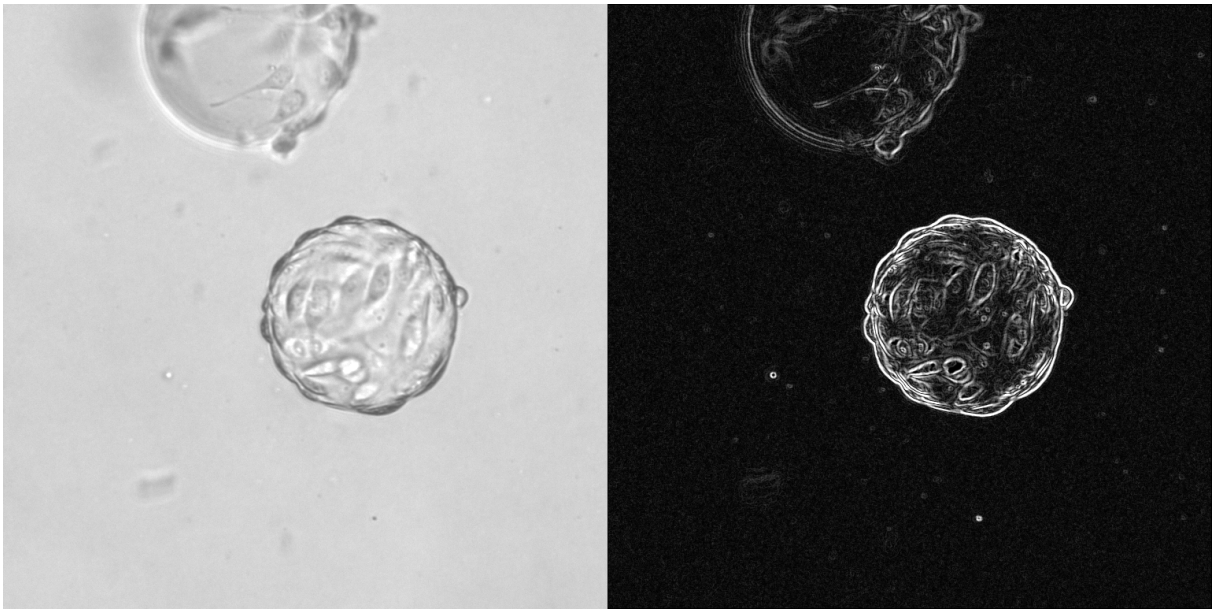

Abbildung 52. Links: ISM-Originalbild mit zwei Microcarriern; rechts: Ausgabebild nach Anwendung des Sobel-Operators.

Auf analoge Weise wurde die Kantendetektion auch mit einem Roberts-Cross-Operator durchgeführt. In den Untersuchungen zeigte sich aber, dass bei leicht unscharf aufgenommenen Bildern die Kanten der Microcarrier mit dieser Methode nicht sicher erkannt werden konnten. Mit dem Sobel-Operator gelingt dies weitaus besser, wie am Beispiel des etwas außerhalb der Fokusebene der Kamera liegenden Carriers am oberen Bildrand in Abbildung 52 zu sehen ist. Anschließend an die Berechnung von G wurden sehr kleine Werte in G durch ein einfaches Thresholding-Verfahren ausgeblendet und auf Null gesetzt. Im späteren Verlauf der Untersuchung erwies sich letztlich ein Thresholding-Wert von 40 als optimal. Dadurch wird das Hintergrundrauschen der Grauwerte im Originalbild herausgefiltert. Auch störende Objekte auf den Bildern (Schmutz auf dem Objektiv oder den Saphirscheiben des ISM, kleine Partikel im Medium), die für die weitere Auswertung keine Rolle spielen, werden dadurch ausgeblendet.

Der nächste Schritt bestand darin, die Information über die Lage von Kanten im Originalbild, die in G gespeichert ist, in eine Ortsinformation umzuwandeln. Diese Information beinhaltet, an welchen xy-Koordinaten des Bildes sich Microcarrier befinden. Dazu wurde die in Abschnitt 2.6.2 beschriebene Methode der Kantenverfolgung eingesetzt. In dem dafür entwickelten Algorithmus wird ausgehend von der linken, oberen Ecke das Bild zeilenweise von links nach rechts durchlaufen und

nach Stellen gesucht, an denen auf einen Wert von Null in G ein von Null verschiedener Wert folgt. Dann wird eine geschlossene Linie um das Objekt gebildet. Pro Objekt wird von dem Algorithmus eine eindeutige Objektnummer vergeben und der Bereich innerhalb der geschlossen Linie mit dieser Nummer markiert. So markierte Bereiche werden bei der weiteren Behandlung des Bildes übersprungen und somit nicht doppelt ausgewertet. Außerdem kann dadurch jeder markierte Bildpunkt genau einem Objekt zugeordnet und die Fläche eines Objekts über die Anzahl der mit der entsprechenden Objektnummer markierten Pixel berechnet werden. Die Fläche wird als einfaches Unterscheidungskriterium verwendet, ob es sich bei einem gefundenen Objekt um einen Microcarrier handelt. Im Rahmen der Voruntersuchungen stellte sich heraus, dass ein Grenzwert von 40000 Pixel die besten Ergebnisse liefert. Alle Objekte mit einer Fläche von weniger als 40000 Pixel sind keine Microcarrier oder nur Microcarrier-Bruchstücke und werden vom Algorithmus bei der weiteren Auswertung ignoriert.

Weiterverarbeitung der als Microcarrier identifizierten Bildausschnitte

Mit dem beschriebenen Verfahren können diejenigen Grauwerte, die zu einem Microcarrier gehören, aus der Gesamtheit der Grauwerte eines Bildes extrahiert werden. Aus diesen Grauwerten musste nun mit einer geeigneten Methode der Bewuchsgrad des Microcarriers errechnet werden. Dabei wurde folgende Annahme gemacht: Die mit obigem Verfahren ausgeschnittenen Teilbilder bilden den Microcarrier naturgemäß nur zweidimensional ab. Auswertbar ist somit lediglich eine Aufsicht auf den Carrier. Zellen, die seitlich angewachsen sind oder verdeckte Zellen auf der Rückseite können bei der Auswertung nicht berücksichtigt werden. Homogenen Bewuchs annehmend muss also der Bewuchs, der sich aus der Auswertung der Aufsicht ergibt, als Maß für den Bewuchs des ganzen Microcarriers verwendet werden.
Ein Ansatz zur Weiterverarbeitung der Teilbilder baut darauf auf, die Methode, mit der die Carrier vom Hintergrund getrennt wurden, auf das Innere der Objekte nochmals anzuwenden. Dadurch können die Kanten der auf dem Microcarrier festgewachsenen Zellen gefunden werden. Nach Anwendung der Objekterkennung durch Kantenverfolgung kann die Fläche der Zellen berechnet werden und über Vergleich mit der Gesamtfläche des Microcarriers ist der Bewuchsgrad berechenbar. In der Praxis stellte sich bereits nach einigen Voruntersuchungen heraus, dass dieser Ansatz große

Probleme mit sich bringt: Je nach Bewuchsgrad liegen die Zellen sehr dicht zusammen, haben stark unterschiedliche Formen und haben je nach Schattenwurf und Beleuchtung unterschiedliche Grauwerte. Dies erschwert die Objekterkennung beträchtlich und erfordert es, den oben beschriebenen Algorithmus um die Behandlung vieler Spezialfälle zu erweitern. Neben der stark angestiegenen Komplexität des Algorithmus und dem hohen Rechenaufwand pro Bild konnten mit dieser Methodik keine überzeugenden Resultate erhalten werden. Daher wurde dieser Ansatz verworfen und einfacher auszuwertende Eigenschaften der Bildausschnitte von Microcarriern untersucht.

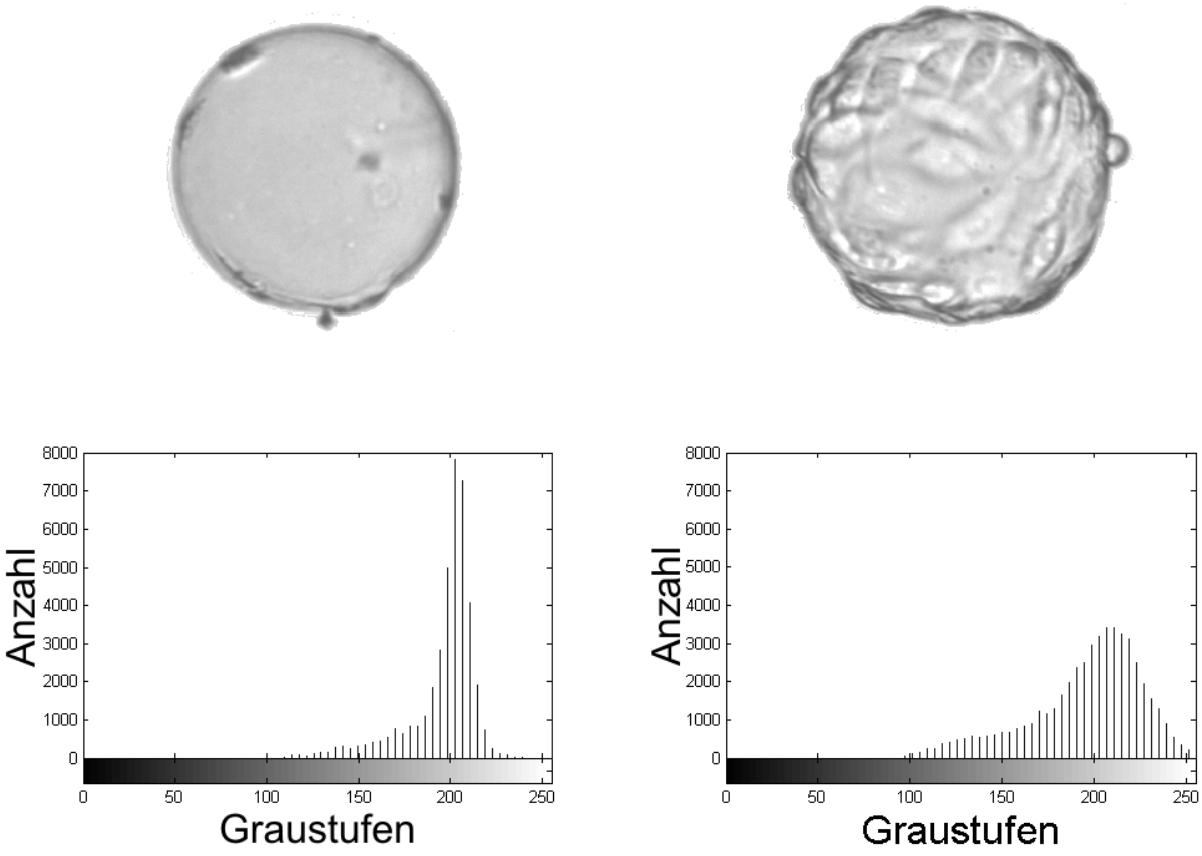

Abbildung 53. Dargestellt sind ein unbewachsener (links) und ein nahezu vollständig bewachsener (rechts) Microcarrier. Darunter sind die dazu gehörenden Histogramme der Grauwerte gezeigt (Anzahl Bildpunkte pro Graustufenwert). Die Unterschiede in den Histogrammen zwischen dem bewachsenen und dem unbewachsenen Microcarrier sind deutlich zu erkennen.

Dabei zeigte sich, dass die Grauwert-Histogramme der Bildausschnitte ein vielversprechender Ansatzpunkt zur Berechnung des Bewuchsgrades sind. Bei unbewachsenen oder wenig bewachsenen Microcarriers ergeben sich schmale

Verteilungen der Grauwerte mit Maxima bei ungefähr 200. Je stärker ein Carrier bewachsen ist, desto breiter und flacher wird die Verteilung. Das Maximum verschiebt sich zunehmend zu höheren Grauwerten. Dies ist exemplarisch für einen sehr schwach bewachsenen und einen vollständig bewachsenen Microcarrier in Abbildung 53 dargestellt.
Das Histogramm wurde zunächst normiert indem jeder Wert durch die Gesamtzahl der Pixel geteilt wurde (die Summe über alle Werte des Histogramms ergibt danach 1). Nun wurden fünf zur Berechnung des Bewuchsgrades relevante Werte daraus errechnet: Der Wert des Maximums sowie der Grauwert an dem dieser auftritt, der Wert der Graustufe 255 (weiß) und der Mittelwert und die Varianz der Verteilung. Diese fünf Werte wurden als Eingangsgrößen für ein neuronales Netz verwendet, das zuvor mit einem entsprechenden Trainingsdatensatz auf die Berechnung des Bewuchsgrades trainiert wurde.

Training und Anwendung neuronaler Netze zur Berechnung des Bewuchsgrades

Zum Training eines neuronalen Netzes ist ein Trainingsdatensatz notwendig. Aus den vorliegenden Daten (s. o.) wurden 64 Bilder ausgewählt und nach dem oben beschriebenen Verfahren ausgewertet. Zusätzlich zu den so erhaltenen 64x5 Werten, die als Eingangsgrößen für das neuronale Netz dienten, war jeweils die Bestimmung der Zielgröße, des Bewuchsgrades, vonnöten. Die 64 Zielwerte wurden durch visuelle Begutachtung der Bilder festgelegt. Darüber hinaus wurde auf die gleiche Art und Weise ein Validationsdatensatz mit 32 Bildern erstellt.
Um ein neuronales Netz in der Programmiersprache Delphi trainieren und verwenden zu können, mussten die entsprechenden Daten- und Programmstrukturen selbst erstellt werden. Es wurde die Funktionsweise einfacher, dreischichtiger Feed-Forward-Netze mit einer sigmoiden Transferfunktion (Gleichung {52}) in der versteckten Schicht und einer linearen Transferfunktion (Identität) in der Ausgabeschicht implementiert. Die Berechnung der Netzausgabe y aus einem Eingabevektor p erfolgt nach Gleichung {53} , die Berechnung des Fehlers während des Trainingsvorgangs nach Gleichung {27}. Der zum Training verwendete Datensatz wurde zuvor normiert, sodass jede enthaltene Variable, die fünf oben genannten Werte aus dem Histogramm sowie der Zielwert (der Bewuchsgrad), einen Wertebereich von -1 bis +1 hatte. Das Training des Netzes

erfolgte durch einen genetischen Algorithmus in Kombination mit einem Simplex-Verfahren. Der in Delphi erstellte genetische Algorithmus ist identisch mit der GAS-Toolbox für Matlab der Universität Sheffield (The Genetic Algorithm Toolbox, Department of Automatic Control and Systems Engineering, University of Sheffield, www.shef.ac.uk/acse/research/ecrg/gat.html) bei realer Kodierung der Individuen. Die Implementation des Simplex-Verfahrens entspricht dem in Abschnitt 2.5.1 dargelegten Ablauf für den n-dimensionalen Fall.

$$f^1(x) = \frac{2}{1+e^{-2x}} - 1 \qquad \{52\}$$

$$y = f^2\left(LW^{2,1} f^1\left(IW^{1,1} p + b^1\right) + b^2\right) \qquad \{53\}$$

$$RMSEV = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(y_i - t_i\right)^2} \qquad \{54\}$$

Die Parameterwerte des neuronalen Netzes, die Werte aus den Wichtungsmatrizen und Biasvektoren, bilden ein Individuum des genetischen Algorithmus. Bei der Initialisierung des genetischen Algorithmus wurden 500 Individuen mit Zufallswerten im Bereich von –100 bis +100 erzeugt. Dieser Wertebereich kann auch später, z. B. in den Crossover-, Mutations- und Rekombinationsschritten, nicht unter- bzw. überschritten werden. Nach je 25 durchlaufenen Generationen wurde das beste Individuum aus dem Chromosomensatz entnommen, mit dem Simplex-Verfahren optimiert und danach in den Chromosomensatz zurückgeschrieben.

Der genetische Algorithmus wurde abgebrochen, sobald der Fehler des Validationsdatensatzes anzusteigen begann. Es wurden mehrere Netze mit unterschiedlichen Neuronenanzahlen in der versteckten Schicht trainiert. Das optimale Resultat wurde beim Training eines Netzes mit 2 Neuronen in der versteckten Schicht über 400 Generationen erhalten.

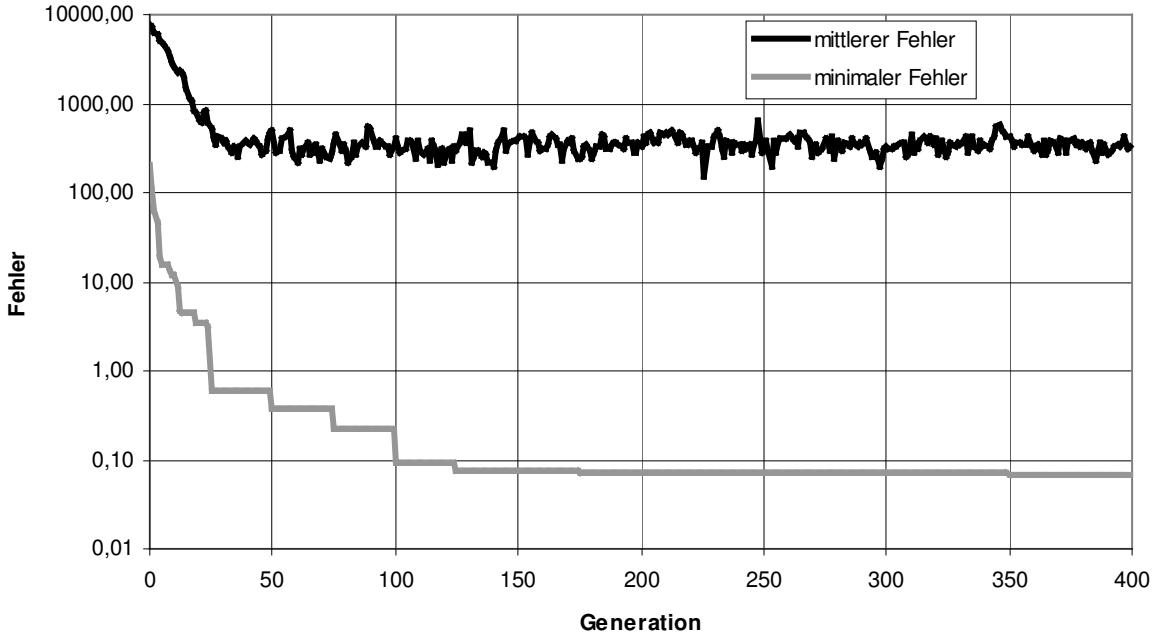

Abbildung 54. Verlauf des mittleren Fehlers aller 500 Individuen (schwarz) sowie des Fehlers des besten Individuums (grau) während 400 Generationen. Zur Berechnung der Fehler wurde Gleichung {27} verwendet. Wird der Fehler des besten Individuums nach 400 Generationen als RMSEP ausgedrückt, der sich analog zum RMSEV (Gleichung {54}) berechnet, ergibt sich ein Wert von 12,8.

In Abbildung 54 ist der mittlere Fehler aller 500 Individuen und der Fehler des besten Individuums im Verlauf der 400 Generationen dargestellt. Erwartungsgemäß nahmen beide Fehler anfangs stark ab und erreichten nach ca. 30 bzw. 125 Generation annähernd konstante Werte. Während des Ablaufs des genetischen Algorithmus wurde in jeder Generation der momentan beste Parametersatz in das neuronale Netz übertragen und mit diesem die Bilder des Validationsdatensatzes ausgewertet. Dies diente zur Überprüfung, ob es während des Trainingsvorgangs zu Overfitting kommt (siehe auch Abschnitt 2.4.4). Zur Bestimmung der Vorhersagegüte wurde nach Gleichung {54} der RMSEV (*root mean square error of validation*) berechnet. Sein Verlauf über die 400 Generationen ist in Abbildung 55 dargestellt. In Generation 375 ist ein sehr geringer Anstieg erkennbar, daher wurde der Algorithmus nach dem nächsten Rechenzyklus, also bei Generation 400, gestoppt.

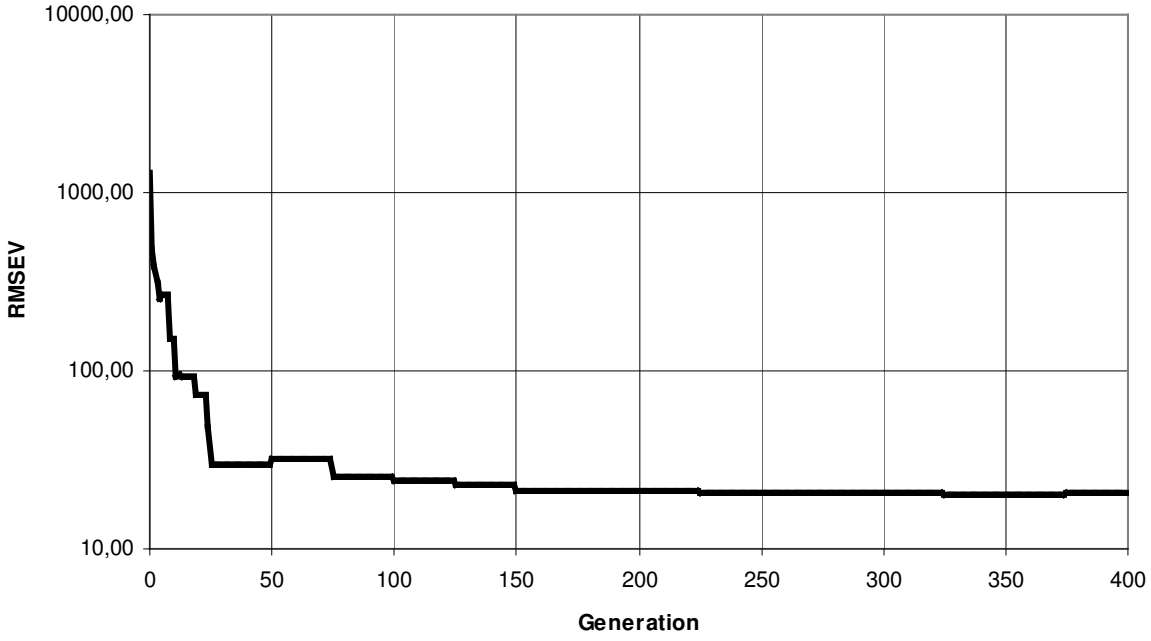

Abbildung 55. Fehler des Validationsdatensatzes ausgedrückt als RMSEV (Gleichung {54}). In Generation 375 erfolgte ein leichter Anstieg, daher wurde der genetische Algorithmus danach abgebrochen.

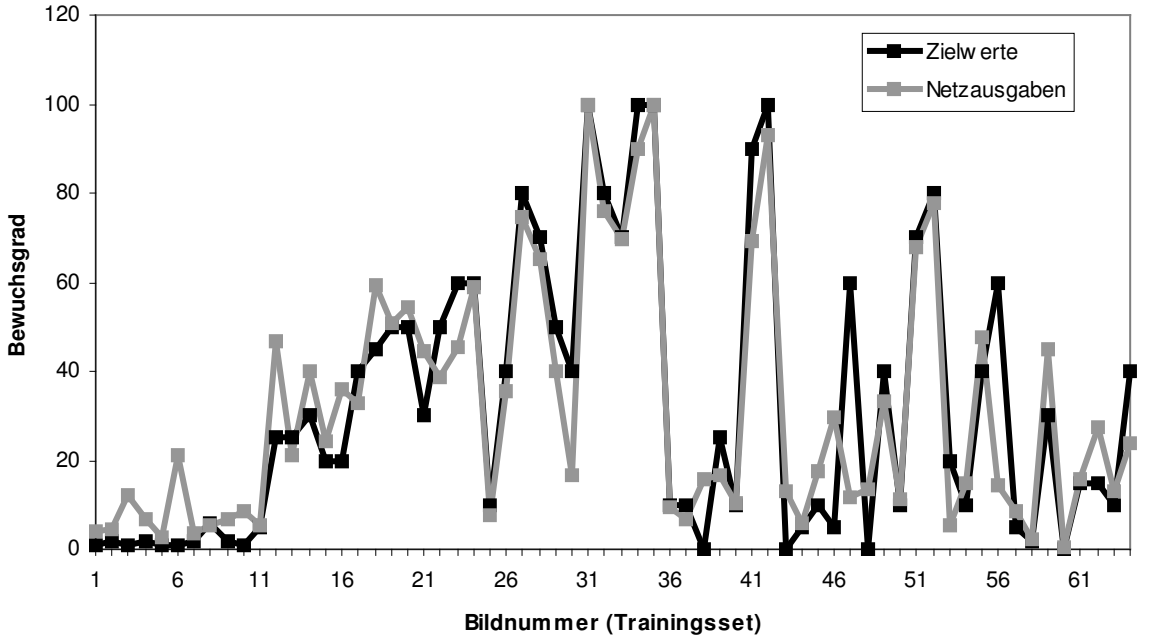

Abbildung 56. Gegenüberstellung der Zielwerte und der Netzausgaben bei den 64 Bildern des Trainingsdatensatzes.

Die Auswertung des Trainingsdatensatzes mit dem optimierten Netz ergab einen RMSEP (*root mean square error of prediction*) von 12,8 und die Auswertung des Validationsdatensatzes einen RMSEV von 20,5. In Abbildung 56 sind die Zielwerte des Trainingsdatensatzes den berechneten Bewuchsgraden des neuronalen Netzes gegenübergestellt. Bei sechs der 64 Bilder gab es zum Teil deutliche Abweichungen, sowohl nach oben als auch nach unten. Eine Diskussion möglicher Fehlerquellen der Auswertung folgt unten. Bei den restlichen Bildern zeigte sich eine gute bis sehr gute Übereinstimmung von Ziel- und Vorhersagewert.
Zum Training eines neuronalen Netzes, basierend auf einem aus experimentellen Daten erstellten Trainings- und Validationsdatensatz, wurde ein einfach zu bedienendes Hilfsprogramm erstellt. Die Datensätze werden eingelesen, die optimalen Netzparameter mit dem genetischen Algorithmus und dem Simplex-Verfahren bestimmt und in einer Datei abgespeichert. Somit lässt sich für Experimente, bei denen die Bilder der Microcarrier deutlich anders aussehen und somit auch der Zusammenhang zwischen Histogramm und Bewuchsgrad ein anderer ist als bei dem hier ausgewerteten Experiment, jederzeit ein angepasstes neuronales Netz erzeugen.

Zusammenfassung: Ablauf des Algorithmus

Zusammengefasst läuft der Algorithmus folgendermaßen ab:

- Einlesen der Parameter des neuronalen Netzes aus einer Datei
- Einlesen des Bildes aus einer Bitmapdatei
- Anwenden des Sobel-Operators zur Detektion von im Bild vorhandener Kanten
- Ausblenden kleiner Grauwertgradienten durch Thresholding (1. Parameter des Algorithmus: *Sobel Threshold*)
- Objekterkennung durch Kantenverfolgung
    - Geschlossene Linien um die Objekte bilden und die Bildpunkte in Inneren der Linien mit einer fortlaufenden Objektnummer markieren
    - Fläche des Objekts berechnen (Anzahl Pixel), bei Objekten deren Fläche unterhalb eines Grenzwertes liegt wird das Objekt bei der Auswertung ignoriert (2. Parameter des Algorithmus: *Pixel Amount Threshold*)
- Grauwerte-Histogramme der gefundenen Objekte errechnen
- Eingabewerte des neuronalen Netzes aus den Histogrammen berechnen

- Anwendung des neuronalen Netzes zur Bestimmung des Bewuchsgrades

Sind auf einem Bild mehrere Microcarrier, wird jeder getrennt ausgewertet und jeweils die Fläche und der Bewuchsgrad errechnet. Neben diesen objektspezifischen Rückgabewerten hat der Algorithmus noch folgende Rückgabewerte: Gesamtfläche (Anzahl der gesamten durch Carrier eingenommenen Pixel), mittlerer Bewuchsgrad und Anzahl der gefundenen Objekte. Zum Export dieser Rückgabewerte nach In-situ-Analysis wurde der Algorithmus um Funktionen, die den in Abschnitt 5.3 aufgeführten Anforderungen genügen, erweitert und als DLL kompiliert.

Einsatz des Analysemoduls zur Auswertung experimenteller Daten

Zur Überprüfung der Leistungsfähigkeit des Algorithmus wurde das eingangs erwähnte Kultivierungsexperiment (NIH-3T3 Zellen mit Cytodex 1 Microcarriern, Aufnahme von 29100 Bildern in 29 Zyklen à 900 Bildern) mit dem erstellten Microcarrier-Analysemodul in In-situ-Analysis ausgewertet. Bei der Analyse zeigte sich, dass von den 29100 vorhandenen Bildern 23315 Bilder leer waren (keine Microcarrier und sonstige Objekte vorhanden) und vom Algorithmus somit bei der Auswertung übersprungen wurden. Bei der Auswertung der 5785 restlichen Bilder wurde jeweils der Bewuchsgrad errechnet und in In-situ-Analysis gegen die Kultivierungszeit aufgetragen. Diese Form der Auswertung erlaubt wenig Rückschlüsse auf die Vorgänge während der Kultivierung, da ein Einzelbild und der daraus ermittelte Wert für den Bewuchsgrad nicht den wahren Wert des Bewuchsgrades zur gegebenen Zeit wiedergibt, sondern nur *einen* um den gesuchten Wert gestreuten Messwert. Aus diesem Grund wurden pro Messzyklus 900 Bilder in schneller Folge hintereinander aufgenommen. Aus den ermittelten Werten für den Bewuchsgrad eines Messzyklus wurde der Mittelwert berechnet. In Abbildung 57 sind die so erhaltenen 29 Werte für den Bewuchsgrad gegen die Kultivierungszeit aufgetragen.

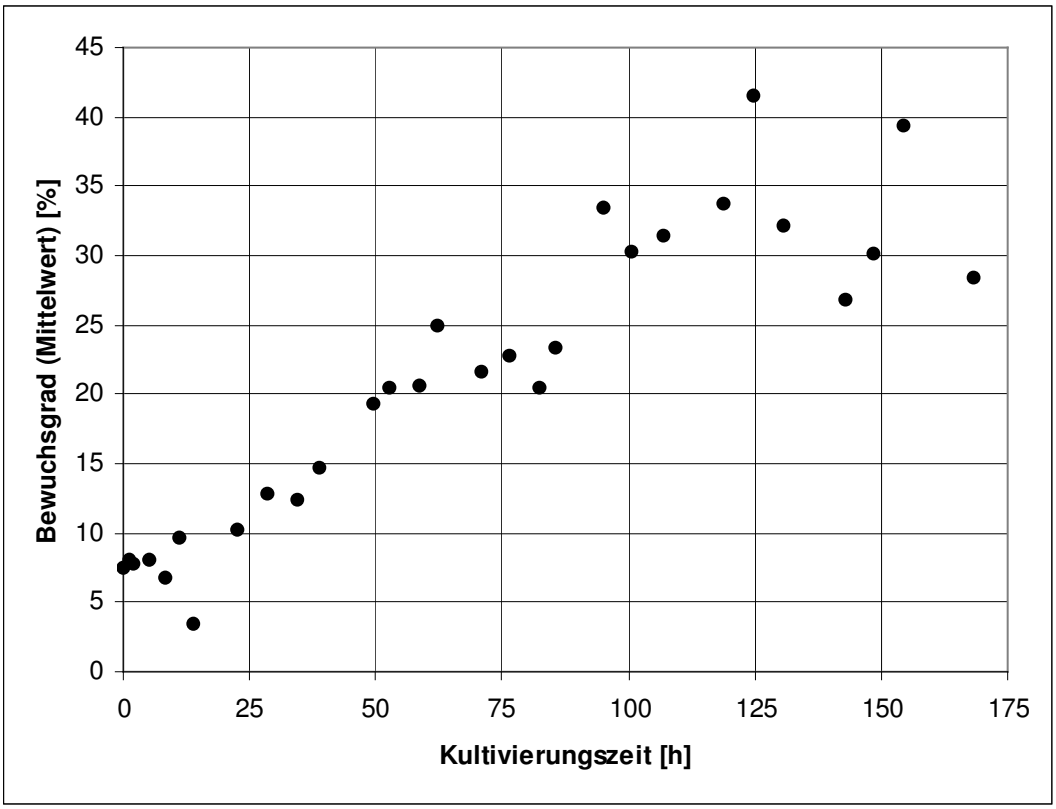

Abbildung 57. Auswertungsergebnis des Microcarrier-Experiments. Dargestellt ist der Bewuchsgrad (Mittelwert der Werte eines Messzyklus) gegen die Kultivierungszeit.

Der Bewuchsgrad war erwartungsgemäß anfangs niedrig und lag bei etwa 5 %. In den ersten 100 Stunden der Kultivierung stieg er bis auf ca. 25 % an. Im späten Kultivierungsverlauf (Stunde 100-175) blieb der Bewuchsgrad zunächst konstant und sank leicht auf einen Wert von 20 % ab. Bei zwei Messzyklen ergaben sich deutlich höhere Werte von 40 % bzw. 38 %. Bei Betrachtung der entsprechenden Bilddaten wurde festgestellt, dass bei diesen beiden Messzyklen sehr häufig voll bewachsene Microcarrier im Bild auftraten und sich somit ein höherer Durchschnittswert für diese beiden Zyklen ergab.

Eine vollständige Überprüfung der Auswertungsergebnisse wurde nicht durchgeführt, da das „per Hand-Auswerten“ aller 26100 Bilder extrem zeitaufwändig gewesen wäre. Es wurden die Bilder von drei Messzyklen, jeweils einer von Beginn, Mitte und Ende der Kultivierung, begutachtet und die Werte für den Bewuchsgrad ermittelt. Durch gerade diese Auswahl der Zyklen sollte erreicht werden, dass sowohl Bilder mit tendenziell eher niedrigem, mittlerem und hohem Bewuchsgrad bei der Überprüfung berücksichtigt werden. Nun wurde pro Zyklus die mittlere absolute Abweichung der

Werte berechnet (Gleichung {55}). Diese, zusammen mit der jeweils kleinsten bzw. größten absoluten Abweichung, sind in Tabelle 9 für die drei Zyklen aufgeführt. Die Abweichungen steigen von 5 % bei Zyklus 2 (Kultivierungsbeginn) auf 16 % beim letzten Zyklus. Bei jedem Zyklus gibt es mindestens ein Bild, dass bei beiden Auswertungen identische Werte für den Bewuchsgrad ergab (siehe kleinste Abweichungen). Die maximale Abweichung steigt, ebenso wie die mittlere Abweichung, bei Zyklen aus dem späteren Kultivierungsverlauf an.

$$m = \frac{1}{N} \sum_{i}^{N} \left| y_i - t_i \right| \qquad \{55\}$$

m: mittlere Abweichung; y: Bewuchsgrad (Algorithmus)
t: Bewuchsgrad (per Hand); N: Bildanzahl im Messzyklus

Tabelle 9. Vergleich der Auswertungsergebnisse.

Zyklus	2	14	29
Bewuchsgrad (Algorithmus)	8,1	20,6	27,8
Bewuchsgrad (per Hand)	6,7	26,4	34,9
mittlere Abweichung	5,4	11,2	16,1
kleinste Abweichung	0,0	0,1	0,0
größte Abweichung	49,7	71,2	90,3

Fehlerdiskussion

Das Ansteigen der mittleren und der maximalen Abweichung ist darauf zurückzuführen, dass Fehler bei der Erkennung der Microcarrier sich bei Zyklen aus dem späteren Verlauf der Kultivierung viel stärker auswirken. Ist z. B. auf einem Bild ein unscharf aufgenommener Microcarrier mit einem Bewuchsgrad von 10 % abgebildet, der vom Algorithmus nicht erkannt wird (Rückgabe des Algorithmus folglich Bewuchsgrad = 0 %), ergibt dies einen Fehler von 10 %. Bei einem unscharf aufgenommenen Microcarrier, der nahezu vollständig bewachsen ist (z. B. zu 90 %), ergibt sich so ein Fehler von 90 %. Ist ein Carrier, trotz Unschärfe, korrekt identifiziert worden, ergibt sich ein weiteres Problem: Bei unscharf aufgenommenen Bildern ist der Zusammenhang zwischen Histogramm und Bewuchsgrad anders als bei scharf aufgenommenen Bildern. Die Abweichung ist somit auf nicht richtige Verarbeitung des Histogramms durch das neuronale Netz zurückzuführen. Zur Verminderung dieses

Problems wurde dieser Tatsache bei der Erstellung des Trainingsdatensatzes durchaus Rechnung getragen, indem unscharfe Bildern und auch auf andere Weise von der Norm abweichende Bilder berücksichtigt wurden. Dennoch ist es fraglich, ob durch das Hinzunehmen von mehr und mehr Sonderfällen in den Trainingsdatensatz, die generelle Vorhersagequalität des Algorithmus zunimmt. Das Problem ist also die möglichst repräsentative Auswahl einiger Bilder aus dem vorhandenen Datenmaterial zur Erstellung eines Trainingsdatensatz vorzunehmen, ohne dabei die Spezialfälle zu wenig oder zu viel Einfluss nehmen zu lassen. Neben unscharfen Aufnahmen sind, besonders im späteren Kultivierungsverlauf, Bilder mit zerstörten (geplatzten, zerrissenen etc.) Microcarriern und Bilder von Carrier-Cluster problematisch. Bei letzteren kommt es zwischen den Microcarriern zu einem Schattenwurf, der zu einem stark veränderten Histogramm führt. Niedrige Grauwerte sind dort sehr stark vertreten. Die Auswertung solcher Histogramme durch das neuronale Netz führt häufig zu ungenauen Vorhersagewerten für den Bewuchsgrad.

Die größte Fehlerquelle ist aber, das Begutachten der Microcarrierbilder zwecks Ermittlung des „wahren“ Bewuchsgrades. Durch Begutachten der Bilder kann der Bewuchsgrad, vom Fall vollständig unbewachsener oder bewachsener Carrier abgesehen, nur geschätzt werden. Dieser Fehler wirkt sich gleich zweimal im Verlauf der dargelegten Auswertung aus: Zum ersten Mal beim Erstellen des Trainingsdatensatzes für den Bewuchsgradwerte benötigt werden und zum zweiten Mal bei der Überprüfung der Berechnungen des Algorithmus.

6 Zusammenfassung

Die Themenschwerpunkte dieser Arbeit waren Experimental Design, Prozessanalyse und Prozessoptimierung sowie Messtechnik und Auswertung von Messdaten. Mit den drei in dieser Arbeit behandelten Themen wurde ein breites Spektrum der für das Verständnis und die optimale Führung von Bioprozessen notwendigen Verfahren und Methoden abgedeckt.

Um das Verstehen komplexer biologischer Prozesse zu erleichtern, ist es hilfreich die Kinetik der ablaufenden enzymatischen Reaktionen zu kennen. Die Parameter der Enzymkinetik müssen durch Messungen bestimmt werden. Die Genauigkeit, mit der dies gelingt, hängt dabei von der Durchführung der Experiments und der Messungen ab. Für eine enzymatische Reaktion, die durch eine Michaelis-Menten-Kinetik beschrieben werden kann, wurden mit einem Experimental-Design-Ansatz optimale Prozess- und Messbedingungen errechnet, die es gestatten die Parameter der Kinetik mit hoher Genauigkeit zu messen. Dazu wurde ein dynamisches Modell der Reaktion aufgestellt und die Fisher-Informations-Matrix (FIM) berechnet. Als Gütekriterium der Parameterbestimmung wurde der kleinste Eigenwert der FIM betrachtet (E-Kriterium). Mit einem Optimierungsalgorithmus wurden diejenigen Bedingungen (Messzeitpunkte, Startkonzentration des Substrates, Reaktionsvolumen etc.) gesucht, bei denen die Parameterbestimmung optimal erfolgen kann. Wird das Messexperiment als Batch-Prozess durchgeführt, zeigte sich, dass im Vergleich zu äquidistanten Messungen eine Optimierung der Messzeitpunkte die Genauigkeit der Parameterbestimmung wesentlich verbessert. Die Cramer-Rao-Lower-Bound (CRLB) sank dadurch im Mittel um 26 %. Eine noch größere Verbesserung ist aber dadurch zu erreichen, dass man den Prozess als Fed-Batch-Prozess durchführt. Durch Optimierung der Zeitpunkte für die Substratzugabe und die Messungen kann die Genauigkeit der Parameterbestimmung um 40 % für K_m und um 18 % für v_{max} gesteigert werden. Für zukünftige Studien kann der hier vorgestellte Berechnungsansatz leicht auf andere enzymatische Reaktionen übertragen werden, indem in das Prozessmodell eine andere Kinetik eingesetzt wird.

Im zweiten Teil dieser Arbeit wurde ein industrieller Fermentationsprozess untersucht. Die Zielsetzung bestand darin die für die Produktausbeute wichtigsten Einflussgrößen zu ermitteln und ein Prozessmodell zu erstellen, mit dem eine Simulation und eine

Optimierung des Prozesses durchgeführt werden kann. Der verwendete Datensatz bestand aus Online- und Offline-Messungen von etwa 100 Prozessdurchläufen. Die Anwendung statistischer Verfahren (Korrelations- und Hauptkomponentenanalyse) ergab, dass die Produktausbeute wesentlich von der Viskosität, der Zitronensäurekonzentration, der Menge des zugegebenen Substrats sowie von der Anzahl und den Zeitpunkten der Zugaben abhängt. Zur Erstellung eines Prozessmodells wurde ein neuronales Netz verwendet. Aus einem Eingabevektor mit Prozessdaten des Zeitpunktes t sollte durch das Netz die zu erwartende Produktkonzentration bei t+12 h errechnet und eine Prognose der Endausbeute erstellt werden. Mit dem vorhandenen Datensatz wurde ein dreischichtiges Feed-Forward-Netz trainiert. Dieses wurde erfolgreich bei der Echtzeit-Simulation des Prozesses eingesetzt. Im Mittel lag der Vorhersagefehler bei 6 %. Neben der Anwendung zur Prozesssimulation wurde das Netz auch verwendet um abzuschätzen, wie sich Änderungen der Prozessführung auf die Ausbeute auswirken. Dazu wurden modifizierte Datensätze erzeugt, bei denen z. B. der Temperaturverlauf oder die Rührerdrehzahl verändert worden sind. Insgesamt wurden zwölf verschiedene Änderungen der Prozessführung betrachtet. Die Resultate zeigten, dass die größten Zugewinne an Produkt bei einer Erhöhung der Rührerdrehzahl zu erwarten sind. Eine geringere Steigerung der Ausbeute ist bei einer Absenkung der Temperatur, der Begasungsrate und des pH-Wertes zu erwarten. Diese drei Ergebnisse bieten wenig Spielraum für eine Optimierung. Sie stimmen nicht mit den Ergebnissen der statistischen Auswertung und mit vorhandenem Expertenwissen über den Prozess überein. Außerdem ist eine Extrapolation der zu erwartenden Ausbeute bei diesen drei Prozessgrößen mit einer relativ hohen Ungenauigkeit behaftet, da diese in nahezu allen vorhandenen Datensätzen identische Verläufe haben. Das neuronale Netz kann mangels geeigneter Lernbeispiele daher keine präzisen Vorhersagen liefern. Demgegenüber ist die zu erwartende Ausbeutesteigerung bei Veränderung der Substratzufütterung ein plausibles Ergebnis. So wird z. B. durch eine frühere Substratzufütterung als Nebeneffekt die Fermentationsbrühe verdünnt, was ein Absinken der Viskosität und eine Erhöhung der Ausbeute zur Folge hat. Am Prozess zeigte sich, dass der gleiche positive Effekt auf die Ausbeute sich auch durch Zugabe von Wasser bei Prozessbeginn erreichen lässt.

Der dritte Schwerpunkt dieser Arbeit war die Erstellung einer Steuerungs- und Auswertesoftware für ein In-situ-Mikroskop. Das Programm *In-situ-Control* bietet über

eine grafische Benutzeroberfläche die zum Betrieb des Mikroskops und der verwendeten Digitalkamera notwendigen Steuer- und Kontrollfunktionen. Außerdem kann die Software zur automatisierten Aufnahme von Messreihen verwendet werden. Dabei wird das von der Kamera aufgezeichnete Bild als Bitmapdatei zusammen mit den aktuellen Messbedingungen und –parametern abgespeichert. Die so erstellten Messreihen lassen sich mit dem Programm *In-situ-Analysis* auswerten. Dieses ist modular aufgebaut: Nur die grundlegenden Abläufe beim Auswertungsvorgang und die grafische Benutzeroberfläche sind in der Hauptanwendung festgelegt und die Vorschrift zur Auswertung der Bilder (der Analysealgorithmus) wird aus einer DLL geladen. So kann der Funktionsumfang der Software leicht erweitert werden, wenn dies Veränderungen in der Aufnahmetechnik oder Bilder neuer Zelllinien erforderlich machen. Ein vorhandener Algorithmus zur Auszählung und Klassifizierung von Hefezellen wurde optimiert und für die Verwendung als DLL angepasst. Für eine vorhandene DLL mit einem Algorithmus zur Auswertung von Tierzellen wurde eine Interface-DLL erstellt, sodass eine Anbindung an die Software ermöglicht wurde. Für die Auswertung des Bewuchsgrades von mit Tierzellen bewachsener Microcarrier wurde ein Analysealgorithmus erstellt. Mit Hilfe von Methoden der digitalen Bildverarbeitung konnte die Microcarrier vom Hintergrund separiert werden. Durch Auswertung der Grauwert-Histogramme der Bildausschnitte mit einem neuronalen Netz ließ sich der Bewuchsgrad der Microcarrier berechnen. Die Fehler lagen dabei zwischen 5-15 %, in Abhängigkeit der Beschaffenheit des Originalbildes. Als problematisch erwiesen sich vor allem unscharfe Bilder und Bilder mit Clustern aus Microcarriern, Carrier-Fragmenten und durch andere Objekte überlagerte Microcarrier.

Die in dieser Arbeit verwendeten Methoden konnten zur Planung, Analyse und Optimierung von Bioprozessen sehr erfolgreich eingesetzt werden.

7 Literaturverzeichnis

[ALEKSANDER] Aleksander, I., Morton, H. (1990): *An Introduction to neural computing*. First edition. London: Capman and Hall

[ANDERSON] Anderson, T.W. (1984): *An introduction to multivariate statistical analysis.* Second Edition. New York: John Wiley & Sons

[ATKINSON] Atkinson, A.C. (1996): „The usefulness of optimum experimental designs." *J. R. Stat. Soc.* B 58 (1), 59-76

[BECKER] Becker, T., Enders, T., Delgado, A. (2002): „Dynamic neural networks as a tool for the online optimization of industrial fermentation." *Bioprocess Biosyst Eng* 24, 347-354

[BITTNER] Bittner, C., Wehnert, G., Scheper, T. (1998): "In-situ microscopy for online determination of biomass." *Biotechnology and Bioengineering*, Vol. 60, Nr. 1

[CICHOCKI] Cichocki, A., Unbehauen, R. (1993): *Neural Networks for optimization and Signal Processing.* Wiley

[DANZER] Danzer, K. et al. (2001): *Chemometrik. Grundlagen und Anwendungen.* Berlin : Springer

[DUGGLEBY] Duggleby, R.G., Clarke, R.B. (1991): „Experimental design for estimating the parameters of the Michaelis-Menten equation from progress curves of enzyme-catalyzed reactions." *Biochim. Biophys. Acta* 1080, 231-236

[DOKUNNT] Dokumentation zur Neural Net Toolbox Version 4.0.2 in Matlab (Version 6.5)

[EHRENBERG] Ehrenberg, A.S.C. (1986): *Statistik oder der Umgang mit Daten.* 1. Auflage. Weinheim: VCH

[ENDRENYI] Endrenyi, L, Chan, F.-Y. (1981): „Optimal design of experiments for the estimation of precise hyperbolic kinetic and binding parameters." *J. Theor. Biol.* 90, 241-263

[ESBENSEN] Esbensen, K.H. (2000): *Multivariate Data Analysis -in practice.* 4th edition. CAMO ASA, Norwegen

[FRERICHS] Frerichs, J.G. (2000): Entwicklung eines In-situ-Mikroskops zur bildgestützten Online-Überwachung von Bioprozessen, Dissertation am Fachbereich Chemie der Universität Hannover

[GATUTORIAL] Tutorial: An Overview of Genetic Algorithms, Genetic Algorithms User´s Guide (Matlab), University of Sheffield

[GESTEWITZ] Gestewitz, M, Heußner, T. (2005): *Skript zur Vorlesung Kognitive Robotik*. Institut für Informatik, Humboldt-Universität zu Berlin

[GOODWIN] Goodwin, G.C. (1987): Identification: experiment design. In: Singh, G. (Ed.), *System and Control Encyclopedia*, vol. 4. Oxford, Pergamon, S. 2257-2267

[GOLDBERG] Goldberg, D.E. (1989): *Genetic Algorithms in Search, Optimization, and Machine Learning*. Reading: Addison-Wesley

[HAYKIN] Haykin, S. (1994) : *Neural Networks. A Comprehensive Foundation.* New York: Macmillan

[HEBB] Hebb, D.O. (1949): *The Organization of Behavior*. New York: Wiley

[HECHT-NIELSEN] Hecht-Nielsen, R. (1990): *Neurocomputing*. Addison-Wesley Publishing Company

[HELLMUTH] Hellmuth, K. (2006): mündliche Mitteilung.

[KAY] Kay, S.M. (1993): *Fundamentals of statistical signal processing: estimation theory*. Prentice-Hall, London

[KRABICHLER] Krabichler, C. (2005): Erkennen von Hefezellen in der In-situ-Mikroskopie mit Methoden der Bildverarbeitung, Diplomarbeit

[MALMGREN] Malmgren, B., Nordlund, U. (1997): „Application of artifical neural networks to chemostratigraphy“. *Paleoceanography*, 11:505-512.

[MARTINEZ] Martinez, G., Frerichs, J.G., Joeris, K., Konstantinov, K., Scheper, T. (2005): „Cell density estimation from a still image for in-situ microscopy“, IEEE ICASSP, Philadelphia

[MASSART] Massart, D.L., Vandeginste, B.G.M., Deming, S.N., Michotte, Y., Kaufman, L. (1988): *Data Handling in Science and Technology. Volume 2. Chemometrics a textbook.* Amsterdam: Elsevier

[MURPHY] Murphy, E.F., Gilmour, S.G., Crabbe, M.J.C. (2002): „Effective experimental design: enzyme kinetics in the bioinformatics era.” *Drug Discovery Today* 7 (Suppl.), 187-191

[NELDER] Nelder, J., Mead, R. (1965): “A Simplex-Method for function minimization”. *Comput. J.* 7(4), 308-313

[OTTO] Otto, M. (1997): *Chemometrie. Statistik und Computereinsatz in der Analytik.* Weinheim : VCH

[PUKELSHEIM] Pukelsheim, F. (1993): *Optimal Design of Experiments*, Wiley, New York

[RITTER] Ritter, H., Martinetz, T., Schulten, K. (1990): *Neuronale Netze. Eine Einführung in die Neuroinformatik selbstorganisierender Netzwerke.* Bonn: Addison-Wesley

[RUDOLPH] Rudolph, G. (2004): Untersuchungen für den Einsatz bildgebender Verfahren zum Inline-Monitoring biotechnologischer Prozesse. Diplomarbeit, Universität Hannover

[RUDOLPH 2] Rudolph, G., Gierse, A., Lindner, P., Kasper, C., Hitzmann, B., Scheper, T. (2005): Observation and analysis of lab scaled microcarrier cultivation by in-situ microscopy with image processing tools, ESACT Meeting

[SITTER] home.arcor.de/ralf.sitter/kyb/neuro/neuron.jpg, Abrufdatum: 22.03.2006

[SUHR] Suhr, H., Wehnert, G., Schneider, K., Bittner, C., Scholz, T., Geißler, P., Jähne, B., Scheper, T. (1995): "In-situ microscopy for online characterisation of cell-populations in bioreactors, including cell concentration measurements by depth from focus." *Biotechnology and Bioengineering*, Vol. 47, 106-116

[STRYER] Stryer, L. (1996): *Biochemie*. 4. Auflage. Spektrum Akad. Verlag.

[UNSCRAMBLER] Online Hilfe in der Software The Unscrambler, Version 7.6, CAMO ASA, Norwegen

[VALDEZ-CASTRO] Valdez-Castro, L., Baruch, I., Barrera-Cortés, J. (2003): "Neural networks applied to the prediction of fed-batch fermentation kinetics of *Bacillus thuringiensis*" *Bioprocess Biosyst Eng* 25, 229-233

[VANDEGINSTE] Vandeginste, B.M.G., Massart, D. L., Buydens, L. M. C., De Jong, S., Lewi, P. J., Smeyers-Verbeke, J. (1998): *Handbook of Chemometrics and Qualimetrics – Part B*, Elsevier Science, Amsterdam

[VELOSO] Veloso, A.C.A., Rocha, I., Ferriera, E.C. (2004): "Identification of yield coefficients in an *E. coli* model – an optimal experimental design using genetic algorithm." *Preprints of the Ninth International Symposium on Computer Application in Biotechnology*, Nancy, Frankreich 28-31 Mai 2004

[WEB1] http://de.wikipedia.org/wiki/Sobel-Operator, Abrufdatum 30.03.2006

[WEB2] http://homepages.inf.ed.ac.uk/rbf/HIPR2/hipr_top.htm, Abrufdatum 30.03.2006

[ZELL] Zell, A. (1997): *Simulation neuronaler Netze*. 2. Nachdruck. München: R. Oldenbourg Verlag

[ZUPAN] Zupan, J., Gasteiger, J (1993): *Neural Network for Chemists.* VCH

8 Anhang

8.1 Abkürzungsverzeichnis

a^n	Ausgabe der n-ten Schicht
ANN	artificial neural networks
BHK	Baby Hamster Kidney
cov	Kovarianz
DDE	Dynamic Data Exchange
DLL	Dynamic Link Library
CCD	Charge Coupled Device (Ladungsgekoppeltes, analoges Bauelement, das aus einer Reihe von Speicherelementen besteht; CCD-Chip: Bildaufnahmeteil elektronischer CCD-Kameras)
$E\langle x\rangle$	Erwartungswert von x
EoF	End of Fermentation (Produktausbeute am Ende der Fermentation)
FIM	Fisher-Informations-Matrix
f^n	Transferfunktion der n-ten Schicht
IMCU	International Milk Clotting Units
ISM	In-situ-Mikroskop
γ	performance ratio
g(x,y)	Grauwert eines Bildes an der Stelle x,y
$G_x(x,y)$, $G_y(x,y)$	Sobelgradient in x- bzw. y-Richtung (ohne Index: Gesamtgradient)
GA	Genetischer Algorithmus
GUI	Graphical User Interface
$IW^{n,m}$	Input Weight (Wichtungsmatrix) der Verbindung der n-ten Schicht mit dem m-ten Eingabevektor
KNN	künstliche neuronale Netze
LED	Light emitting diode

$LW^{n,m}$	Layer Weight (Wichtungsmatrix) der Verbindung der n-ten Schicht mit der m-ten Schicht
n^j	net input der j-ten Schicht
OUR	Oxygen Uptake Rate (Sauerstoffaufnahmerate)
p	Eingabevektor bei Neuronalen Netzen
pO_2	Gelöstsauerstoffkonzentration
Q	Messfehlerkovarianz
MoF	Moment of Force (Leistungseintrag)
NaN	Not a number, Leerelement bzw. fehlendes Element in Vektoren oder Matrizen
NIPALS	Non-linear Iterative Projection by Alternating Least-Squares
PChar	Datentyp in Delphi, Zeiger auf einen 0-terminierten string
R	Anzahl der Elemente in Eingabevektoren bei neuronalen Netzen
RMSEP, RMSEV	Root mean square error of prediction/validation
ROI	Region of Interest (Bereich eines Bildes der zum Auswertung besonders relevant ist)
rpm	Rotations per minute
S^n	Anzahl der Neuronen in der n-ten Schicht
std, STD	Standardabweichung, Standard deviation
t	Vektor mit Zielwerten eines neuronalen Netzes
$\theta(x, y)$	Richtung des (Sobel-) Gradienten an der Stelle x,y

8.2 Lebenslauf

Zur Person

geboren am 11.Juli 1977
in Altdorf bei Nürnberg
Geburtsname: Müller

Nelkenstrasse 18
30167 Hannover
Telefon: 0511/4739881
Email: lindner@iftc.uni-hannover.de

Familienstand:
ledig, keine Kinder

Staatsangehörigkeit:
deutsch

Berufserfahrung

seit 08/2003	Wissenschaftlicher Mitarbeiter am Institut für Technische Chemie, Universität Hannover Tätigkeiten: Experimental Design, statistische Datenauswertung (Korrelations- und Hauptkomponentenanalyse), Prozessoptimierung, Einsatz von Optimierungsmethoden (GA etc.), Modellierung mit neuronalen Netzen, Planung und Entwicklung von Software-Systemen Anfertigung der Doktorarbeit, Titel: „Entwicklung von Software-System zur Planung, Datenaufnahme und –auswertung bei Bioprozessen“ Abschluss der Promotion: voraussichtlich Ende 2006

Studium

10/1997 – 05/2003	Chemiestudium an der Universität Hannover Diplomarbeit: „Mathematische Modellierung von methylglyoxalinduzierten Oszillationen bei einer kontinuierlichen Kultivierung von *E. coli*“ Note: Sehr gut

Praktika und Zusatzqualifikationen

09/2003	Marketing Seminar
09/2003 – 05/2004	Seminar Spezielles Recht für Chemiker
08/2000 – 05/2001	Praktikum an der Stanford University, Stanford CA, USA Forschungstätigkeit am Institut für Organische Chemie
09/1999	Seminar Betrieblicher Umweltschutz
09/1998 – 05/1999	Fremdsprachenkurs Englisch für Chemiker am Fachsprachenzentrum der Universität Hannover

Zivildienst

06/1996 – 09/1997	Zivildienst am Eilenriedestift in Hannover Tätigkeit: Pflege und Betreuung

Schule

05/1996	Abitur am Kurt-Schwitters-Gymnasium Hannover

Sprachkenntnisse

Englisch: Verhandlungssicher
Französisch: Grundkenntnisse

EDV – Kenntnisse

MS Office, Matlab, Maple, The Unscrambler
Delphi, Corel Draw

8.3 Veröffentlichungen

Lindner, P., Hitzmann, B. (2006): "Experimental design for optimal parameter estimation of an enzyme kinetic process based on the analysis of the Fisher information matrix." *Journal of theoretical Biology* 238, 111-123

Lindner, P., Hellmuth, K., Hitzmann, B. (2005): "Correlation analysis of fermentation data." Bioperspectives 2005, Wiesbaden (Poster)

Rudolph, G., Gierse, A., Lindner, P., Hitzmann, B., Scheper, T. (2005): „Observation and Analysis of Lab Scale Microcarrier Cultivation by In-situ Microscopy with Image Processing Tools" Bioperspectives 2005, Wiesbaden (Poster)