Benjamin Hartmann

Lokale Modellnetze zur Identifikation und Versuchsplanung nichtlinearer Systeme

Dissertation

Schriftenreihe der Arbeitsgruppe
Mess- und Regelungstechnik – Mechatronik
Department Maschinenbau

Herausgeber: Oliver Nelles

UNIVERSITÄT SIEGEN

Band 2

Impressum
Prof. Dr.-Ing. Oliver Nelles
Arbeitsgruppe Mess- und Regelungstechnik
Department Maschinenbau
Universität Siegen
57068 Siegen
ISSN 2193-0538
URN urn:nbn:de:hbz:467-7865
Zugl.: Siegen, Univ., Diss., 2014

Lokale Modellnetze zur Identifikation und Versuchsplanung nichtlinearer Systeme

Dissertation
zur Erlangung des akademischen Grades
DOKTOR-INGENIEUR

vorgelegt von
Dipl.-Ing. Benjamin Hartmann
aus Siegen

eingereicht beim
Department Maschinenbau
der Universität Siegen

Referent: Prof. Dr.-Ing. Oliver Nelles
Korreferent: Prof. Dr.-Ing. Claus-Peter Fritzen

Tag der mündlichen Prüfung
24. Januar 2014

Vorwort

Diese Arbeit entstand während meiner Tätigkeit als wissenschaftlicher Mitarbeiter am Institut für Mechanik und Regelungstechnik – Mechatronik der Universität Siegen. Für die finanzielle Unterstützung des Forschungsprojekts danke ich der Deutschen Forschungsgemeinschaft (DFG).

Meinem Doktorvater, Herrn Prof. Dr.-Ing. Oliver Nelles, danke ich für das entgegengebrachte Vertrauen in allen Tätigkeiten, welches die eigenverantwortliche und freie Forschung am Institut ermöglichte. Außerdem bedanke ich mich für die äußerst wertvolle und großzügige Förderung bei der Durchführung dieser Arbeit, die Diskussionsbereitschaft und die vielen motivierenden Anregungen.

Herrn Prof. Dr.-Ing. Claus-Peter Fritzen möchte ich für das Interesse an meiner Arbeit und die freundliche Übernahme des Korreferats danken.

Ich bedanke mich ganz herzlich bei meinen Kollegen Dr. Geritt Kampmann, Oliver Bänfer, Tobias Ebert, Torsten Fischer und Julian Belz für die Unterstützung und die vorbildliche Zusammenarbeit. Die immer freundliche, offene und motivierende Atmosphäre habe ich all die Jahre sehr genossen. Dr. Jochen Moll danke ich für die konstruktive Zusammenarbeit, durch die ich wertvolle Einblicke in sein Forschungsfeld gewinnen durfte. Außerdem gilt mein Dank allen nicht namentlich genannten Kollegen, sowie allen Studierenden, die meine Tätigkeit im Rahmen von Studien- und Diplomarbeiten oder als studentische Hilfskräfte unterstützt haben.

Für die Förderung und Durchführung der Versuche am Motorenprüfstand der Daimler AG danke ich insbesondere Dr. Frank Kirschbaum, Dr. Philipp Klein, Nataša Kieft und Dr. René Linssen. Die vielen Diskussionen haben mir immer wieder neue Anregungen gegeben, um die Algorithmen gezielt für die praktischen Anforderungen weiterzuentwickeln. Bei Dr. Wolf Baumann und Dr. Karsten Röpke bedanke ich mich ebenfalls für die wertvolle Zusammenarbeit und die Möglichkeit, einen Beitrag zur Entwicklung zukünftiger Softwarelösungen bei der Firma IAV GmbH leisten zu dürfen.

Für die große Gastfreundschaft und die freundschaftliche Zusammenarbeit während meiner Aufenthalte an der University of Ljubljana in den Jahren 2008, 2011 und 2012 danke ich in besonderem Maße Prof. Dr. Igor Škrjanc und allen Mitarbeitern seiner Arbeitsgruppe.

Nicht zuletzt danke ich meinen Eltern, meiner Familie und meiner Freundin Franziska, die mir stets Mut zugesprochen und mich in meiner Arbeit bestärkt haben.

Siegen, Juni 2013 Benjamin Hartmann

Inhaltsverzeichnis

Symbole und Abkürzungen

Allgemeine Symbole

α	Regularisierungsparameter
$\underline{c_i}$	Zentrumskoordinaten eines Teilmodells
C	Metaparameter zur Strukturabwägung
$\underline{D}$, d_{ij}	Abstandsmatrix, Abstandswert
e	Modellfehler $y - \hat{y}$
J	Verlustfunktion
k	Allgemeiner Index, Zeitindex bei zeitdiskreten Größen
k_σ	Proportionalitätsfaktor zur Einstellung der Interpolationsglattheit
l	Polynomgrad
$\underline{K}$	Regularisierungsmatrix
M	Anzahl lokaler Modelle
$\mu(\cdot)$	Zugehörigkeitsfunktion
n	Anzahl der Modellparameter
n_{eff}	Anzahl der effektiven Modellparameter
nx/nz	Anzahl der x-/z-Regressoren
N	Anzahl Messpunkte
N_c	Anzahl Kandidatenpunkte
p	Eingangsraumdimension, Wahrscheinlichkeit
$\Psi(\cdot)$	Teilungsfunktion
$\Phi(\cdot)$	Gültigkeitsfunktion
$\underline{Q}$	Gewichtungsmatrix
$\mathbb{R}$	Menge der reellen Zahlen
R^2	Bestimmtheitsmaß
$\underline{S}$	Glättungsmatrix
σ	Standardabweichung
$\underline{\sigma_n}$	Standardabweichung des Rauschens
$\underline{\Sigma}$	Kovarianzmatrix
t	t-Wert (Hypothesentest)
$\underline{\theta}$	Allgemeiner Modellparametervektor
u	Prozesseingangsgröße
$\underline{v}$	Parametervektor der Teilungsfunktionen
$\underline{w}$	Parametervektor der Teilmodelle
$\underline{x}$	Eingangsvektor der Teilmodelle (Regelkonklusionen)
$\underline{X}$	Regressionsmatrix
$\underline{\xi}$	Vektor eines Kandidatenpunktes

y Prozessausgangsgröße
$\hat{y}$ Modellausgangsgröße
$\underline{z}$ Eingangsvektor der Gültigkeitsfunktionen (Regelprämissen)

Abkürzungen

AIC	Akaike Informationskriterium (*Akaike Information Criterion*)
AIC_c	Korrigiertes Akaike Informationskriterium (*Corrected AIC*)
CV	Kreuzvalidierung (*Cross Validation*)
GPM	Gaußprozessmodell (*Gaussian Process Model*)
HILOMOT	*Hierarchical Local Model Tree* (Trainingsalgorithmus)
LLM	Lokal Lineares Modell
LM	Lokales Modell
LOLIMOT	*Local Linear Model Tree* (Trainingsalgorithmus)
LOOCV	LOO-Kreuzvalidierung (*Leave One Out Cross Validation*)
LQM	Lokal Quadratisches Modell
LS	Least Squares-Verfahren (Fehlerquadratmethode)
MAP	*Maximum a posteriori*-Methode
ML	*Maximum Likelihood*-Methode
MLP	*Multi Layer Perceptron* (Neuronaler Netztyp)
NF	*Neuro Fuzzy*
NN	Nächster Nachbar (*Nearest Neighbor*)
NMSE	Normierter mittlerer quadratischer Fehler (*Normalized Mean Squared Error*)
NOE	Nichtlinearer Ausgangsfehler (*Nonlinear Output Error*)
NRMSE	Normierte Wurzel des mittleren quadratischen Fehlers (*Normalized Root Mean Squared Error*)
PRESS	*Prediction Sum of Squares* (Kreuzvalidierung)
RBF	Radialbasisfunktion
RMSE	Wurzel des mittleren quadratischen Fehlers (*Root Mean Squared Error*)
SUHICLUST	*Supervised Hierarchical Clustering* (Trainingsalgorithmus)
SVD	Singulärwertzerlegung (*Singular Value Decomposition*)
SVM	*Support Vector Machine*
VP	Versuchsplanung
WLS	Gewichtetes (*weighted*) Least Squares-Verfahren

Mathematische Symbole

$\hat{\cdot}$	Geschätzte Größe
$\tilde{\cdot}$	Transformierte Größe
$\lvert \cdot \rvert$	Betrag einer Zahl
$\lVert \cdot \rVert_2$	2-Norm
$\lVert \cdot \rVert_\mathrm{F}$	Frobenius-Norm
$\lVert \cdot \rVert_{\underline{\Sigma}}$	Mahalonobis-Norm
$(\cdot)^{-1}$	Inverse einer Matrix
$\mathrm{diag}(\cdot)$	Diagonalelemente einer Matrix
$\mathrm{spur}(\cdot)$	Spur einer Matrix
$(\cdot)^T$	Transponierte einer Matrix
$\circ$	Hadamard-Produkt
$\mathrm{E}\{\cdot\}$	Erwartungswert
$\mathrm{cov}\{\cdot\}$	Kovarianzmatrix
$\mathcal{L}(\cdot)$	Likelihood-Funktion
$p(\cdot)$	Wahrscheinlichkeitsdichtefunktion

Kurzfassung

In der vorliegenden Arbeit werden neue Verfahren zur experimentellen, datenbasierten Modellbildung (Identifikation) und zur Versuchsplanung nichtlinearer Systeme mit Hilfe von lokalen Modellnetzen vorgestellt.

Die aus dieser Arbeit hervorgehenden Algorithmen basieren einerseits auf einer detaillierten Analyse und Weiterentwicklung der linearen Optimierung lokal gewichteter, polynomialer Modelle hinsichtlich Parameterschätzung, Regularisierung, Validierung und Strukturselektion. Andererseits bedarf es nichtlinearer Optimierungsverfahren, um die Gültigkeitsgebiete der lokalen Modelle an den nichtlinearen Prozess anzupassen. Dazu werden geeignete, heuristische Partitionierungsstrategien entwickelt, die durch sukzessive Unterteilung des Eingangsraums in der Lage sind, lokale Modellnetze effizient an die zugrunde liegenden Prozessdaten anzupassen. Dazu bedienen sich die untersuchten Verfahren achsenschräger Teilungsstrategien, welche sich durch eine flexible Partitionierung insbesondere für höherdimensionale Probleme eignen. Zudem wird eine Strategie vorgestellt, bei der automatisch eine Abwägung stattfindet, ob das Modell entweder mit einer achsenschrägen Teilung flexibilisiert oder ob die Anzahl der zu selektierenden Regressoren der Teilmodelle erhöht werden soll. Die Partitionierung und die lokale Selektion der signifikanten Polynomterme erfolgen hierbei simultan.

Darüber hinaus bietet die Umsetzung des neu entwickelten „Hierarchical Local Model Tree"-Algorithmus' (HILOMOT) als statistisches Versuchsplanungsverfahren neue Möglichkeiten für praktische Anwendungen. Neben einer Offline-Versuchsplanung können nichtlineare Prozesse, für die kein a priori-Wissen zur Verfügung steht, mit dem Algorithmus aktiv gelernt werden. Das aktive Lernverfahren ist in der Lage, online mit dem Prozess zu interagieren und den Anforderungen entsprechend die Messpunkte effizient zu platzieren.

Die vorgestellten Algorithmen werden durch die Anwendung an realen Systemen verifiziert und deren Leistungsfähigkeit unter Beweis gestellt. Dazu zählt die Modellierung des Verbrauchs und der Emissionskenngrößen eines modernen Dieselmotors, die aktive Versuchsplanung im Rahmen eines Structural Health Monitoring-Systems und die adaptive Online-Versuchsplanung am Motorenprüfstand zur Kalibrierung eines Dieselmotors.

Stichwörter: Experimentelle Modellbildung, Nichtlineare Systemidentifikation, Takagi-Sugeno Neuro-Fuzzy Systeme, Lokale Modellnetze, Inkrementelle Baumkonstruktion, Achsenschräge Partitionierung, Strukturselektion, Statistische Versuchsplanung, Aktives Lernen.

Abstract

This thesis proposes new approaches for experimental, data-based modeling (identification) and for experimental design of nonlinear systems based on local model networks.

On the one hand, the proposed algorithms are based on a detailed analysis and further development of the linear optimization for locally weighted, polynomial models. This concerns parameter estimation, regularization, validation and subset selection. On the other hand, nonlinear optimization methods are required in order to adapt the validity regions of the local models to the underlying nonlinear process. This is achieved by the development of heuristic partitioning strategies that incrementally subdivide the input space. These methods enable an efficient adaptation of local model networks to the given data. The basic idea of the investigated approaches is the application of flexible axes-oblique partitioning strategies that are well-suited for higher-dimensional problems. In addition, a strategy is proposed that applies a structure trade-off where the model is improved either by performing an axes-oblique split or by increasing the number of regressors to be selected for the local sub-models. The partitioning as well as the local selection of the significant polynomial terms are carried out simultaneously.

Furthermore, the newly developed „hierarchical local model tree“ (HILOMOT) algorithm is extended for the application as an experimental design approach. Next to an offline strategy for design of experiments (DoE) the algorithm is enhanced with active learning strategies in order to perform online measurements of processes where no prior knowledge is available. The active learning algorithm is able to interact with the process and can meet the requirements to efficiently place the measurement points.

The proposed algorithms are verified on real system applications where their performance has been demonstrated. That incorporates the modeling of the fuel consumption and emissions of a modern diesel engine as well as the active learning of damage locations in the framework of a structural health monitoring system. Moreover, an adaptive online experimental design is implemented on an engine test bed for the calibration of a diesel engine.

Keywords: experimental modeling, nonlinear system identification, Takagi-Sugeno neuro-fuzzy systems, local model networks, incremental tree-construction, axes-oblique partitioning, subset selection, design of experiments (DoE), active learning.

1 Einführung

1.1 Motivation

Modelle sind die Grundlage nahezu aller leistungsfähigen Verfahren in der Automatisierungstechnik und in vielen anderen Bereichen. Modellbasierte Verfahren werden u.a. zur Prädiktion, Simulation, Optimierung, Data Mining, Regelung, Fehlerdiagnose benötigt, siehe Bild 1.1. Oft können *theoretische* Modelle aufgestellt werden, die z.B. auf physikalischen oder chemischen Gleichungen basieren. Allerdings gibt es viele technische Vorgänge, die man unter Umständen *nicht* mehr theoretisch beschreiben kann. In diesem Fall bleibt keine andere Wahl, als das Prozessverhalten *experimentell* zu ermitteln. Die aus Experimenten gewonnenen Daten können dann mit einer *experimentellen Modellbildung (Identifikation)* beschrieben werden.

In technischen Disziplinen herrscht ein permanenter Trend zur Modellierung immer höherdimensionaler Zusammenhänge. Im Wesentlichen wird diese Entwicklung durch drei Merkmale weiter verschärft:

1. *Höhere Komplexität*: Technische Innovationen werden immer komplexer, um am Markt zu bestehen. Beispielsweise kamen im Lauf der Jahre zur Verbesserung von Verbrennungsmotoren viele Stellgrößen hinzu: Abgasrückführung, Mehrfacheinspritzung, Variable Turbinengeometrie (VTG-Lader), variables Saugrohr usw.
2. *Höhere Genauigkeit*: Strengere Gesetzgebung und Marktanforderungen zwingen die Hersteller, eine immer größere Präzision ihrer Produkte zu gewährleisten. Dies geht auch einher mit der Forderung hoher Modellgüten. Zusammenhänge, die früher vielleicht vernachlässigt wurden, müssen nun mit dem Modell berücksichtigt werden.
3. *Dynamische Zusammenhänge*: Höhere Genauigkeit kann auch bedeuten, dynamische statt statische Modelle zu verwenden. Die dann benötigten Verzögerungen der Eingangs- und Ausgangsgrößen führen u.U. zu einer drastischen Vergrößerung der Dimensionalität.

Wie gravierend sich der Einfluss der Dimensionalität auf die Modellbildung auswirken kann, zeigt folgendes Beispiel: Betrachtet man die Anzahl der Stützstellen eines Rasterkennfelds und legt pro Eingangsgröße fünf Stützstellen fest, so ergibt sich bei einer einzigen Eingangsgröße die Stützstellenanzahl zu $N = 5^1$, bei zwei Größen $N = 5^2 = 25$ usw., siehe Bild 1.2. Möchte man schließlich ein Kennfeld bezüglich zehn Eingangsgrößen realisieren, hat das Kennfeld $5^{10} \approx 10$ Millionen Stützstellen. Dieses exponentielle Wachstum des Aufwands hat Richard E. Bellman in den 60er Jahren im Zusammenhang mit der Dynamischen Programmierung als *Curse of Dimensionality* bezeichnet. Es bezieht sich in den meisten Fällen allerdings lediglich auf den *Lösungsansatz*, nicht aber auf das wirkliche Problem. Im Umkehrschluss bedeutet dies: Wenn tatsächlich die Komplexität des Problems exponentiell ansteigt, muss man dies wohl oder übel so akzeptieren. Wenn jedoch nur der Aufwand des Lösungsansatzes exponentiell ansteigt, sollte nach einem effizienteren Verfahren gesucht werden.

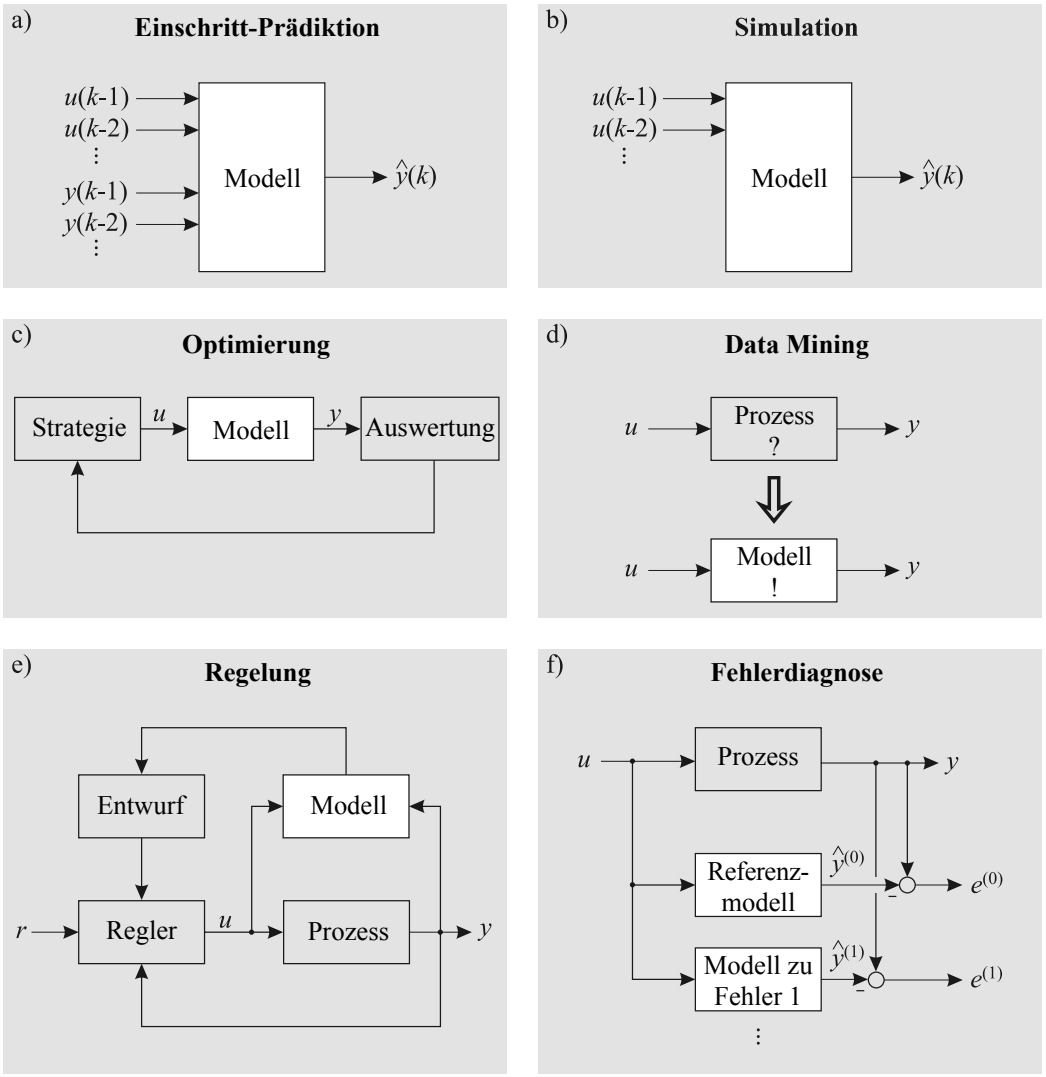

Bild 1.1: Modelle sind die Basis vieler technischer Verfahren. Dazu zählen beispielsweise: a) Einschritt-Prädiktion, b) Simulation, c) Optimierung, d) Data Mining, e) Regelung oder f) Fehlerdiagnose [116].

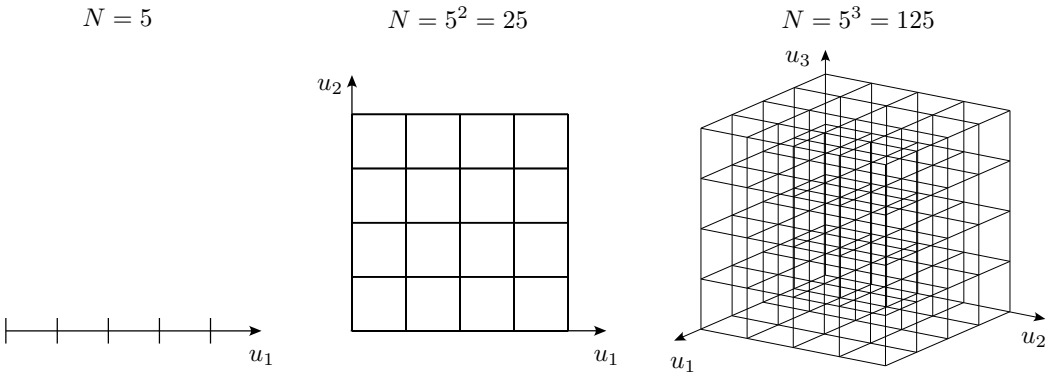

Bild 1.2: Beispiel eines Rasterkennfelds: Mit zunehmender Anzahl an Eingangsgrößen wächst die Anzahl der Stützstellen exponentiell an.

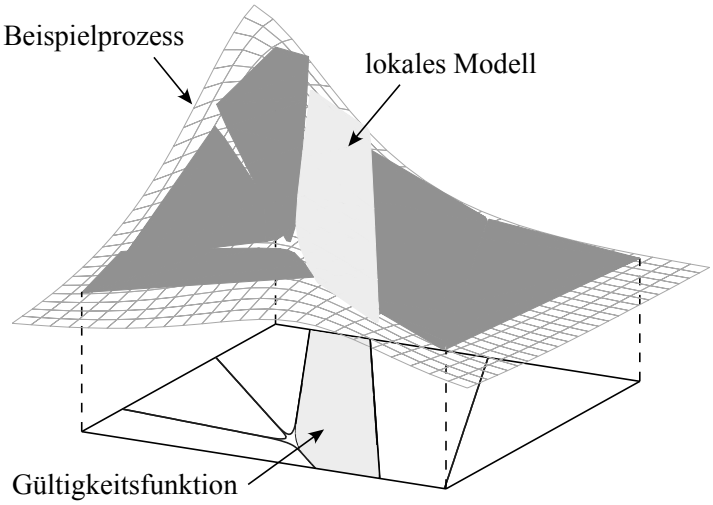

Bild 1.3: Lokales Modellnetz: Superposition lokaler Teilmodelle, die mit den assoziierten Gültigkeitsfunktionen interpoliert werden.

Grundlegende Fragen bei jeder Art von experimenteller Modellbildung sind:

- Welcher Modellansatz ist der geeignete: z.B. Polynom, RBF, NRBF, Gaußprozessmodell, Support Vector Machine, lokales Modellnetz?
- Welche Einstellungsmöglichkeiten gibt es und wie sind diese vorzunehmen: z.B. Kernelfunktion, Verlustfunktion, Validierungsverfahren, Glattheit, Trainingsoptionen?
- Wie flexibel muss das Modell sein?
- Nach welchem Kriterium wird die Modellqualität beurteilt?

In dieser Arbeit wird gezeigt, dass lokale Modellnetze ein geeignetes Werkzeug darstellen, um diesen Fragestellungen sinnvoll zu begegnen und im Zuge dessen einer vollständig automatisierten Modellbildung möglichst nahe zu kommen. Ein nichtlinearer Prozess wird dabei durch stückweise linear parametrierte Modelle lokal beschrieben, siehe Bild 1.3. In welchen Bereichen des Eingangsraums diese Teilmodelle den Prozess regional beschreiben, wird durch die Gültigkeitsfunktionen festgelegt.

Die Entwicklung effizienter Verfahren zur Erstellung lokaler Modellnetze, die auch mit immer mehr Eingangsgrößen umgehen können, ist Gegenstand dieser Arbeit. Die Effizienzsteigerung besteht darin, existierende Verfahren für die Modellbildung mit lokalen Modellnetzen durch die Ausnutzung heutiger Rechnerkapazitäten immer flexibler und damit geeigneter für komplexere Problemstellungen zu gestalten.

Die experimentelle Modellbildung hängt zudem in starkem Maße davon ab, welchen Umfang und welche Qualität die *Messdaten* zum Trainieren des Modells haben. Die Modellbildung ist, wie in Bild 1.4 gezeigt, schließlich nur Teil einer ganzen Prozesskette. Die Qualität der gesamten Prozesskette steht und fällt mit einer gut oder schlecht durchgeführten *Versuchsplanung*, da alle nachfolgenden Schritte auf ihr basieren. Eine über die Modellbildung hinaus gehende Motivation liegt daher in der effizienten Platzierung von Messungen. Die klassische Aufgabe hierbei lautet, mit so wenigen Messungen wie möglich ein Modell zu erhalten, welches den zugrunde liegenden Prozess gut repräsentiert.

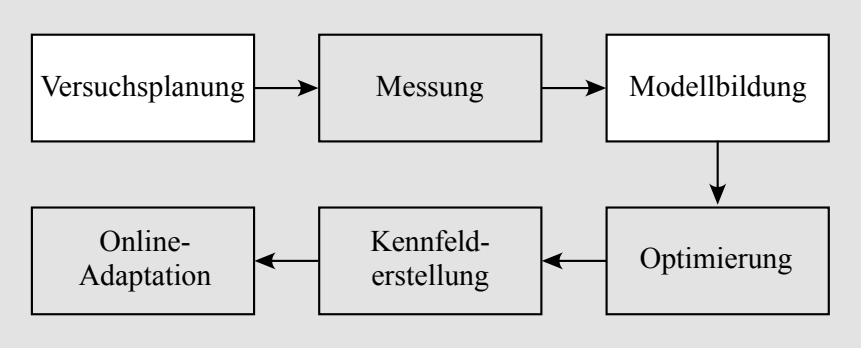

Bild 1.4: Typische Prozesskette einer praktischen Anwendung.

1.2 Zielsetzung und Aufbau der Arbeit

Zielsetzung dieser Arbeit ist die Entwicklung eines Verfahrens zur Modellbildung und Versuchsplanung, welches auf lokalen Modellnetzen basiert und sich durch einfache Bedienung und große Robustheit auszeichnet. Trotzdem soll es Modellgüten ermöglichen, die mit State of the art-Methoden konkurrieren können und heutige Rechnerkapazitäten sinnvoll nutzen. Hierbei liegt der Fokus auf *parametrischen* Methoden zur Modellbildung *stationärer* Zusammenhänge. Dabei liegt die besondere Herausforderung darin, die in dieser Arbeit entwickelten Methoden bewusst darauf auszurichten, Anwendern in der Praxis ein leistungsfähiges Werkzeug bereitzustellen. Spezialwissen auf dem Gebiet der Systemidentifikation und Modellbildung soll zur Benutzung der Verfahren so wenig wie möglich erforderlich sein.

Im Speziellen soll ein inkrementelles Baumkonstruktionsverfahren zur Identifikation von Netzen mit lokal polynomialen Modellen nach dem Vorbild eines bereits existierenden Algorithmus entwickelt werden, welches dessen Hauptschwäche vermeidet, nämlich die Beschränkung auf achsenorthogonale Unterteilungen. Die Flexibilität des Modells kann durch eine achsenschräge Partitionierung (Bild 1.5) deutlich verbessert werden.

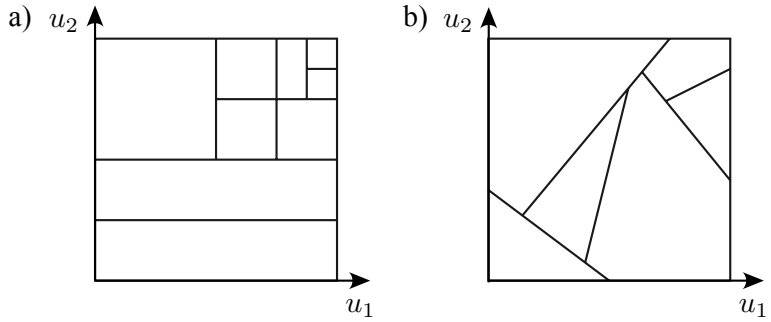

Bild 1.5: Partitionierung a) achsenorthogonal und b) achsenschräg.

Außerdem kann durch die geeignete Wahl der Struktur der lokalen, polynomialen Modelle die Modellgüte signifikant verbessert werden. Die Einbringung des Vorwissens in Form der Modellordnung gestaltet sich jedoch praktisch sehr schwierig, insbesondere wenn der Prozess von vielen Eingangsgrößen abhängt. Aus diesem Grund wird mit dieser Arbeit zudem ein neues, achsenschräges Partitionierungsverfahren entwickelt, welches einen automatischen Kompromiss zweier Alternativen zur Flexibilisierung des Modells ermöglicht. Es

wird zwischen einer Verfeinerung der Partitionierung und einer Erhöhung des Polynomgrades vergleichend abgewägt. Dabei kommt ein leistungsfähiges Variablenselektionsverfahren zum Einsatz.

Neben der experimentellen Modellbildung ist das Ziel, die positiven Eigenschaften der untersuchten Verfahren für eine Versuchsplanung auszunutzen. Zunächst wird untersucht, inwieweit eine Offline-Versuchsplanung mit dem entwickelten achsenschrägen Partitionierungsalgorithmus möglich ist. Dann wird durch die Entwicklung und den Einsatz aktiver Lernstrategien der Algorithmus derart weiterentwickelt, dass er sich auch für eine Online-Versuchsplanung eignet. Das nichtlinear parametrierte Lernverfahren ist in der Lage, mit dem Prozess zu interagieren und den Anforderungen entsprechend die Messpunkte effizient zu platzieren.

Um die genannten Ziele zu erreichen, wird die Arbeit in folgende Abschnitte gegliedert:

Kapitel 2 stellt zunächst die wichtigsten theoretischen Grundlagen zu Polynommodellen, neuronalen Netzen und lokalen Modellnetzen dar.

Kapitel 3 erläutert kurz die Bayes'sche Methode, das Maximum-Likelihood-Verfahren und welche Annahmen bei der Verwendung einer Least Squares-Schätzung zu treffen sind. Danach stellt das Kapitel sowohl die globale als auch lokale Schätzung vor und geht auf den impliziten Regularisierungseffekt der lokalen Schätzung ein. Zur Verwendung expliziter Regularisierungsverfahren wird die Methode der Ridge Regression, der approximativen Tikhonov-Regularisierung und der global-lokalen Regularisierung vorgestellt. Darüber hinaus befasst sich das Kapitel mit der Optimierung der Modellkomplexität linear geschätzter Modelle und leitet lineare Strukturselektionsverfahren her, die speziell bei einer gewichteten Least Squares-Schätzung Einsatz finden.

Kapitel 4 befasst sich mit der nichtlinearen Optimierung der Struktur lokaler Modellnetze. Nachdem die in der Literatur gängigen Partitionierungsverfahren vorgestellt wurden, zeigt das Kapitel Strategien sowohl zur Optimierung paralleler als auch hierarchischer Modellstrukturen auf. Hierzu werden die zwei Partitionierungsverfahren SUHICLUST („*Supervised Hierarchical Clustering*“) und HILOMOT („*Hierarchical Local Model Tree*“) vorgestellt. Nachdem auf die Komplexitätsoptimierung lokaler Modellnetze eingegangen wurde, befasst sich das Kapitel zudem mit einer Strategie, die eine simultane Anpassung von lokaler Modellstruktur und achsenschräger Partitionierung ermöglicht. Die Regressoren der lokalen Modelle werden dazu so selektiert, dass sie als lokale Taylorreihenentwicklungen des Prozesses interpretiert werden können.

Kapitel 5 beschreibt die passive und aktive Messdatengewinnung mit dem zuvor hergeleiteten HILOMOT-Algorithmus. Zunächst stellt das Kapitel die wichtigsten Grundlagen zu statistischer Versuchsplanung und zu aktivem Lernen dar. Danach wird aufgezeigt, wie man eine hierarchische, achsenschräge Partitionierung für eine passive Versuchsplanung nutzen kann. Daraufhin stellt das Kapitel auf HILOMOT basierende, neue Methoden zur aktiven Versuchsplanung für nichtlineare Prozesse vor, die in dieser Arbeit als HILOMOTDOE („*Hierarchical Local Model Tree for Design of Experiments*“)

bezeichnet werden. Abschließend wird ein Ausblick auf die Verwendung von Modellkomitees gegeben. Die Methode des Bootstrappings und die Verwendung heterogener HILOMOT-Komitees bieten viel versprechende Eigenschaften für zukünftige Lösungen.

Kapitel 6 beschreibt die Anwendung von HILOMOT und HILOMOTDOE für drei verschiedene reale Prozesse. Hierbei handelt es sich zunächst um ein Structural Health Monitoring System, bei dem durch aktives Lernen eine Schadenslokalisation erfolgt. Daraufhin werden für die Messdaten eines elfdimensionalen Datensatzes zu einem Dieselmotor Verbrauchs- und Emissionsmodelle erstellt, die z.B. zur Stellgrößenoptimierung genutzt werden können. Der HILOMOTDOE-Algorithmus wurde zudem an einem realen Motorenprüfstand eingesetzt und für die Interaktion mit der Prüfstandssoftware implementiert. Das Kapitel zeigt die Ergebnisse, die mit der automatisierten Online-Vermessung eines Dieselmotors erzielt wurden.

Kapitel 7 enthält eine abschließende Zusammenfassung und einen Ausblick auf zukünftige Forschungsthemen.

Die wesentlichen Beiträge und Neuerungen dieser Arbeit sind:

- Detaillierte Analyse und Weiterentwicklung der linearen Optimierung lokal gewichteter Modelle.
- Entwicklung der neuen Partitionierungsverfahren SUHICLUST und HILOMOT zur nichtlinearen Optimierung der Struktur von lokalen Modellnetzen.
- Entwicklung eines neuen Trainingsverfahrens, das die individuelle Selektion der Regressoren lokal polynomialer Modelle im Rahmen eines achsenschrägen Partitionierungsverfahrens ermöglicht.
- Entwicklung des neuen Versuchsplanungsverfahrens HILOMOTDOE, welches vollständig automatisiert abläuft und zum aktiven Lernen eines unbekannten, nichtlinearen Prozessverhaltens eingesetzt werden kann.
- Implementierung und Anwendung des HILOMOT-Algorithmus' zur Verbrauchs- und Emissionsmodellierung von Verbrennungsmotoren.
- Implementierung und Anwendung des HILOMOTDOE-Algorithmus' zur Schadenslokalisation im Rahmen eines Structural Health Monitoring Systems und zur automatisierten Online-Vermessung eines Verbrennungsmotors am Motorenprüfstand.

2 Nichtlineare Systemidentifikation statischer Prozesse

Im Fachgebiet der deskriptiven Statistik existiert ein großer Katalog an Methoden für die datengetriebene Identifikation nichtlinearer, statischer Prozesse. Die auszuwählende Modellarchitektur hängt im Wesentlichen vom zugrunde liegenden Anwendungsfall und damit von der Art der Nichtlinearität des Prozesses ab. Klassische Polynomansätze, die optional durch statistische Strukturselektionsverfahren ergänzt werden, zählen neben Kennfeldmethoden zu den gängigsten parametrischen Regressionsverfahren. Weitere sehr populäre Ansätze sind Methoden, die auf lokalen Basisfunktionen beruhen und die sich daraus ergebende Erweiterung auf lokale Modellnetze. Dieses Kapitel widmet sich daher der genaueren Beschreibung dieser Verfahren.

2.1 Modelle basierend auf einer Polynomstruktur

Polynome sind in der Praxis neben Rasterkennfeldern vermutlich die verbreitetste Modellbildungsmethode. Gründe dafür liegen einerseits in der guten Interpretierbarkeit, andererseits in der einfachen Implementierung und dem geringen Rechenzeitbedarf dieser Methodenklasse. Daher geht dieser Abschnitt zunächst kurz auf die wichtigsten Eigenschaften klassischer Polynomansätze ein, reflektiert deren Vor- und Nachteile und gibt anschließend einen Überblick zur Erweiterung des klassischen Polynomansatzes mit Strukturselektionsverfahren. Dadurch wird das Optimierungsproblem zwar deutlich komplizierter, allerdings können durch diese Verfahren Polynome auch zugänglich für höherdimensionale Problemstellungen gemacht werden.

Als Einleitung zur Thematik zeigt Bild 2.1 exemplarisch ein Polynommodell des Grades m mit einer einzigen Eingangsgröße u, welches als einfaches Beispiel aus [116] entnommen ist. Die Problemstellung dabei ist, gemessene Daten $\{\underline{u}(i), y(i)\}$ mit $i = 1, \ldots, N$, die mit dem Prozessrauschen n überlagert sind, durch die vorab angenommene Polynomstruktur mit den Parametern $c_0,\ c_1, \ldots, c_m$ möglichst repräsentativ abzubilden. Der Modellausgang kann in diesem Beispiel durch

$$\hat{y} = c_0 + c_1\, u + c_2\, u^2 + \ldots + c_m\, u^m = \sum_{i=0}^{m} c_i\, u^i\,. \tag{2.1}$$

formuliert werden und der Modellfehler $e(k) = y(k) - \hat{y}(k)$ ergibt sich zu

$$e(k) = y(k) - c_0 - c_1\, u(k) - c_2\, u^2(k) - \ldots - c_m\, u^m(k)\,. \tag{2.2}$$

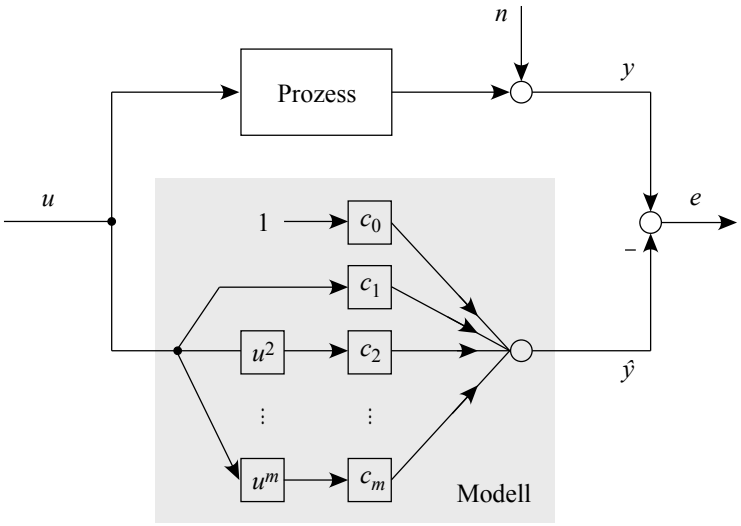

Bild 2.1: Polynommodell m-ten Grades mit $\hat{y} = c_0 + c_1 u + c_2 u^2 + \ldots + c_m u^m$. Zur besseren Übersicht besitzt dieses Beispiel nur eine einzige Eingangsgröße u [116].

Dann folgt daraus das lineare Gleichungssystem für N Messungen mit der Regressionsmatrix $\underline{X}$ und dem Parametervektor $\underline{w}$:

$$\underline{X} = \begin{bmatrix} 1 & u(1) & u^2(1) & \cdots & u^m(1) \\ 1 & u(2) & u^2(2) & \cdots & u^m(2) \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & u(N) & u^2(N) & \cdots & u^m(N) \end{bmatrix}, \qquad \underline{w} = \begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ \vdots \\ c_m \end{bmatrix}. \tag{2.3}$$

Unter der Annahme[1], dass es sich um weißes Prozessrauschen handelt, kann das lineare Optimierungsproblem nach der Methode der kleinsten Fehlerquadrate (engl.: *least squares*, kurz: *LS*) mit folgender Gleichung in einem Schritt gelöst werden

$$\underline{\hat{w}} = (\underline{X}^T \underline{X})^{-1} \underline{X}^T \underline{y}\,, \tag{2.4}$$

wobei dadurch die Summe der Fehlerquadrate

$$J = \sum_{i=0}^{N} e^2(i) = \sum_{i=0}^{N} \left[y(i) - \hat{y}(u(i))\right]^2 \to \min_{\underline{w}} \tag{2.5}$$

minimiert wird.

2.1.1 Klassische Polynommodelle

Ein Spezialfall aus der Funktionsklasse der Polynome sind lineare[2] Modelle, d.h. Polynommodelle, die ausschließlich die Steigungsparameter in jede Eingangsrichtung und optional

[1] Eine vollständige Ausführung zu den getroffenen mathematischen Annahmen bei der Least Squares-Schätzung kann beispielsweise aus [116] entnommen werden.

[2] Wenn außer den Steigungsparametern eines linearen Modells ein zusätzlicher Offset-Parameter hinzukommt, spricht man streng genommen von *affinen* Modellen. Der Einfachheit halber werden die affinen Modelle in dieser Arbeit ebenfalls als *linear* bezeichnet.

einen Gleichwert (engl.: *offset*) aufweisen. Oft ist ein lineares Modell eine gute, erste Annäherung des nichtlinearen Prozessverhaltens, sofern die dem Prozess zugrunde liegende Nichtlinearität eher schwach ausgeprägt ist. Zudem ist diese Modellklasse oft die einzig sinnvolle Alternative, um mit spärlichen, stark verrauschten oder schlecht verteilten Daten eine sinnvolle Regression durchzuführen. Nach [116] ist ein lineares Modell beschrieben durch

$$\hat{y} = w_0 + w_1 u_1 + w_2 u_2 + \ldots + w_p u_p \tag{2.6}$$

oder auch

$$\hat{y} = \sum_{i=0}^{p} w_i u_i \qquad \text{mit} \quad u_0 = 1\,. \tag{2.7}$$

Baut man das lineare Modell zu einem allgemeinen Polynom aus, erhöht sich die Flexibilität des resultierenden Modells mit dem Grad des Polynoms. Ein p-dimensionales Polynom des Grades l ist beschrieben durch:

$$\hat{y} = w_0 + \sum_{i=1}^{p} w_i u_i + \sum_{i_1=1}^{p} \sum_{i_2=i_1}^{p} w_{i_1 i_2} u_{i_1} u_{i_2} + \ldots + \sum_{i_1=1}^{p} \cdots \sum_{i_l=i_{l-1}}^{p} w_{i_1 \cdots i_l} u_{i_1} \cdots u_{i_l}\,. \tag{2.8}$$

Der Gleichanteil w_0 und die erste Summe beschreiben ein lineares Modell, wohingegen alle weiteren Terme die Polynomanteile höheren Grades darstellen. Die resultierende Anzahl der Regressoren beziehungsweise Parameter berechnet sich nach [116, 97] wie folgt:

$$M = \frac{(l+p)!}{l!\,p!} - 1\,, \tag{2.9}$$

wobei M die Anzahl der Basisfunktionen und $M+1$ die Anzahl der Parameter inklusive des Gleichwerts w_0 sind. In diesem Zusammenhang lässt sich das Polynom auch als

$$\hat{y} = \sum_{i=0}^{M} \tilde{w}_i x_i \qquad \text{mit} \quad x_0 = 1\,, \tag{2.10}$$

formulieren, wobei die Terme $\tilde{w}_i x_i$, $i = 0, \ldots, M$, den i-ten Term in (2.8) repräsentieren.

Wie bereits in der Einleitung dieses Abschnitts beschrieben, kann für ein klassisches Polynommodell ein Gleichungssystem aufgestellt werden, welches linear in den Parametern ist und daher mit der Methode der kleinsten Quadrate schätzbar ist. Die Regressionsmatrix $\underline{X}$ und der Parametervektor $\underline{w}$ eines Polynoms l-ten Grades mit p Eingangsgrößen lautet

$$\underline{X} = \begin{bmatrix} 1 & u_1(1) & \cdots & u_p(1) & u_1^2(1) & \cdots & u_p^l(1) \\ 1 & u_1(2) & \cdots & u_p(2) & u_1^2(2) & \cdots & u_p^l(2) \\ \vdots & \vdots & & \vdots & \vdots & & \vdots \\ 1 & u_1(N) & \cdots & u_p(N) & u_1^2(N) & \cdots & u_p^l(N) \end{bmatrix} \qquad \underline{w} = \begin{bmatrix} \tilde{w}_0 \\ \tilde{w}_1 \\ \tilde{w}_2 \\ \vdots \\ \tilde{w}_M \end{bmatrix} \tag{2.11}$$

mit M als Anzahl der Basisfunktionen nach (2.9).

Neben den eingangs genannten Vorteilen muss dem Anwender allerdings bewusst sein, dass eine Polynomstruktur je nach Problemstellung gravierende Nachteile mit sich bringen kann:

1. Die Anzahl der Regressoren wird aufgrund der Kreuzterme mit steigender Dimensionalität äußerst groß (*Curse of Dimensionality* [15]).

2. Bei Überanpassung des Modells neigen Polynome zu unerwünschtem Schwingen im Interpolationsbereich.

3. Die Extrapolation strebt sehr schnell gegen $\pm\infty$, schneller als lineare Modelle.

Eine detaillierte Übersicht zu den Vor- und Nachteilen von linearen Modellen und Polynomen ist in [116] zu finden. Abhilfe bei höherdimensionalen Problemen können Strukturselektionsverfahren bieten. Allerdings steigt dadurch der Rechenaufwand u.U. beträchtlich.

2.1.2 Polynommodelle unter Verwendung von Strukturselektionsverfahren

Die Problematik der Struktur- oder auch Variablenselektion wurde vielfach in der Statistik-Literatur behandelt (siehe beispielsweise [31, 160, 111]). Warum eine Variable *wichtig* ist oder nicht, ist noch nicht vollständig verstanden. Man kann beispielsweise annehmen, dass eine Variable dann wichtig ist, wenn ihr Weglassen einen erheblichen Einfluss auf die *Vorhersagegenauigkeit* hat, siehe [76]. Die Beurteilung der Modellgüte erfolgt meist über statistische Informationskriterien, welche ohne zeitintensive Berechnungen auskommen. Alleine auf der Grundlage statistischer Überlegungen und der Trainingsdaten wird hierbei beurteilt, ob das Modell eine gute Vorhersagegenauigkeit hat. Häufig verwendet man beispielsweise Kriterien, die auf Mallow's C_p-Statistik [104] basieren. Äquivalente Auswahlkriterien sind *Akaike's Information Criterion* (AIC) [3] oder das *Bayesian Information Criterion* (BIC) [135].

Die treibende Kraft hinter der Strukturselektion (engl.: *Subset Selection*), d.h. die Auswahl eines Untermodells mit weniger Variablen, ist der Wunsch nach einem reduzierten Regressionsmodell, kombiniert mit der benötigten hohen Vorhersagegenauigkeit. Das Modell ist dann einfacher aufgebaut und leichter zu interpretieren als das Modell mit vollständigem Variablensatz. Wenig einflussreiche, redundante Details fallen durch die Selektion weg, um die starken Effekte besser erkennen und interpretieren zu können. Die Koeffizienten der Eingänge, die in dem Subset-Modell übrig geblieben sind, werden mit Least Squares geschätzt.

Eine Auswahl der wichtigsten Variablen eines Regressionsmodells präsentiert sich generell als kompliziertes Problem. Gleichwohl ist die Frage, welcher Selektionsalgorithmus sich am besten eignet, noch nicht abschließend geklärt.

Die gängigsten Verfahren zur Strukturselektion sind:

- *Best-Subset Selection*: Dieser Algorithmus testet alle Anzahlen und alle Kombinationen an Variablen, um das beste „Subset“ zu finden. Aus diesem Grund ist die Methode nur dann brauchbar, wenn die Anzahl der Variablen $30 - 40$ nicht übersteigt. Das beste Untermodell mit zwei Variablen muss z.B. nicht zwangsläufig dieselben Variablen enthalten, die bei dem besten Untermodell mit nur einer Variablen ausgewählt wird. In [154] wird z.B. vorgeschlagen, die Lösung dieses kombinatorischen Optimierungsproblems durch genetische Algorithmen anzunähern.

- *Stepwise Regression*: Die Stepwise Regression-Algorithmen lassen sich in *Backward Elimination*, *Forward Selection* und Mischformen, welche die Hauptideen der beiden Typen beinhalten, unterteilen [61, 76]. Die Backward Elimination startet mit allen Variablen und streicht sukzessive die für das Modell unwichtigsten Variablen. Umgekehrt startet die Forward Selection mit der wichtigsten Variable und nimmt Schritt für Schritt eine weitere Variable hinzu. Bei den Mischformen können Variablen, die zuvor eliminiert wurden, später wieder hinzu genommen werden und umgekehrt.
- *Strukturselektion durch Regularisierung*: Mit Hilfe eines zusätzlichen Regularisierungsmechanismus' ist es möglich, im Rahmen der Least Squares-Schätzung eine Strukturselektion durchzuführen. Ein bekanntes Verfahren, das auf dem Prinzip der Ridge Regression[3] beruht, ist der *LASSO*-Algorithmus[4] [148, 8].
- *Least Angle Regression*: Der Least Angle Regression-Algorithmus (LAR) ist eine automatische Strukturselektionsmethode, welche die Vorwärts-Selektion verbessert. Er kann auch für Fälle verwendet werden, in denen die Anzahl der Eingangsgrößen größer als die Anzahl der Datenpunkte ist. Einfache Änderungen am LAR ermöglichen es, den Forward-Stagewise-Algorithmus und den Lasso-Algorithmus effizienter zu berechnen. Die Algorithmen werden zusammen LARS genannt [32, 61].

Die genannten Verfahren bieten eine hervorragende Möglichkeit, das Modell im Hinblick auf Overfitting robuster zu gestalten, da redundante Parameter des Modells eliminiert werden. In der Praxis liegt jedoch oft ein Prozessverhalten vor, bei dem die Flexibilität eines Polynoms nicht ausreichend ist, um ausgeprägte Nichtlinearitäten abzubilden (Biasfehler). In diesem Fall ist der Einsatz von globalen Polynommodellen nicht mehr sinnvoll. Aus diesem Grund beschäftigt sich der folgende Abschnitt mit der Modellklasse der neuronalen Netze.

2.2 Neuronale Netze

Gerade im Hinblick auf die Modellierung höherdimensionaler Zusammenhänge bietet sich die Methodenklasse der neuronalen Netze an. Die Beschaffenheit eines neuronalen Netzes hängt vom verwendeten Netztyp ab. Aus Gründen der Übersichtlichkeit wendet sich dieser Abschnitt ausschließlich den verbreitetsten Klassen neuronaler Netze zu:

- Netze mit radialen Basisfunktionen (RBF-Netze) und
- Multilayer Perceptron Netze (MLP-Netze).

Neuronale Netze bestehen im Allgemeinen aus einer Eingangsschicht, einer oder mehreren verdeckten Schichten und der Ausgangsschicht (siehe Bild 2.2). Die Besonderheit neuronaler Netze ist, dass im Gegensatz zu Polynomen alle Basisfunktionen vom gleichen Typus stammen. Die verschiedenen Ausprägungen neuronaler Netze unterschieden sich im Wesentlichen in der Wahl des Typs der Basisfunktionen, welche auch als Neuronen bezeichnet werden.

[3] Bei Ridge Regression wird zur Summe der Fehlerquadrate zusätzlich die Summe der Parameter-Beträge addiert, gewichtet mit einem Regularisierungsparameter α.

[4] LASSO = Least Absolute Shrinkage and Selection Operator.

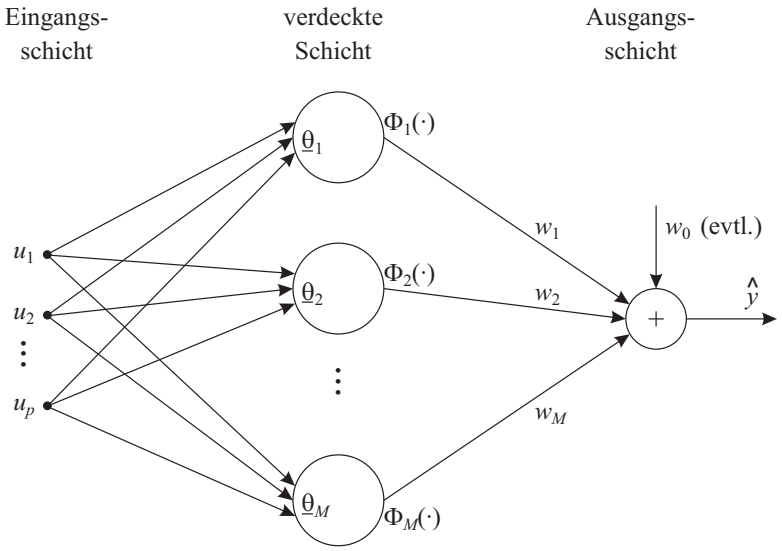

Bild 2.2: Ein Netz besteht aus überlagerten Basisfunktionen [116].

Wie bei den linearen und polynomialen Modellen angedeutet, lassen sich neuronale Netze ebenfalls als eine Summe von M gewichteten Basisfunktionen Φ_i darstellen, wobei der Gleichwert w_0 optional hinzuaddiert wird:

$$\hat{y}(\underline{u}) \;=\; w_0 + \sum_{i=1}^{M} w_i \Phi_i\left(\underline{u}, \underline{\theta}_i\right) . \tag{2.12}$$

Die Gewichte w_i, $i = 0, 1, \ldots, M$, entscheiden darüber, wie stark ein jeweiliges Neuron zum Gesamtmodellausgang beiträgt. Die Struktur von Gl. (2.12) lässt erkennen, dass es sich bei der Bestimmung der Gewichte w_i um ein lineares Optimierungsproblem handelt, welches mit der Methode der kleinsten Fehlerquadrate effizient lösbar ist. Im Unterschied zum Polynommodell ergibt sich hier die Regressionsmatrix mit N Messdaten $\{\underline{u}(1), y(1)\}, \{\underline{u}(2), y(2)\}, \ldots, \{\underline{u}(N), y(N)\}$ zu:

$$\underline{\Phi} = \begin{pmatrix} 1 & \Phi_1(\underline{u}(1)) & \Phi_2(\underline{u}(1)) & \cdots & \Phi_M(\underline{u}(1)) \\ 1 & \Phi_1(\underline{u}(2)) & \Phi_2(\underline{u}(2)) & \cdots & \Phi_M(\underline{u}(2)) \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & \Phi_1(\underline{u}(N)) & \Phi_2(\underline{u}(N)) & \cdots & \Phi_M(\underline{u}(N)) \end{pmatrix} . \tag{2.13}$$

Mit dem Ausgangsdatenvektor $\underline{y} = [y(1)\ y(2)\ \cdots\ y(N)]^T$ berechnet sich der optimale Gewichtsvektor $\underline{\hat{w}} = [\hat{w}_0\ \hat{w}_1\ \cdots\ \hat{w}_M]^T$ aus

$$\underline{\hat{w}} = \left(\underline{\Phi}^T \underline{\Phi}\right)^{-1} \underline{\Phi}^T \underline{y} . \tag{2.14}$$

Ein Neuron i der verdeckten Schicht (siehe Bild 2.3) bildet einen p-dimensionalen Eingangsvektor $\underline{u} = [u_1\ u_2\ \cdots\ u_p]^T$ mit Hilfe der Parameter, die im Vektor $\underline{\theta}_i$ zusammengefasst sind, auf eine skalare Größe ab, die sog. Aktivierung x_i, welche dann mittels der sog. Aktivierungsfunktion auf den Neuronenausgangswert nichtlinear transformiert wird [116]. Diese Abbildung kann nach verschiedenen Konstruktionsprinzipien erfolgen. Die wichtigsten Vertreter sind nach [116]:

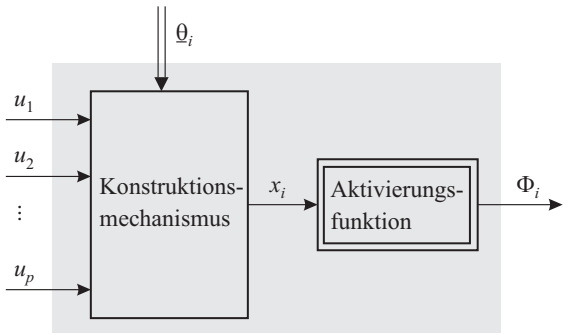

Bild 2.3: Der Konstruktionsmechanismus bildet den hochdimensionalen Eingangsvektor auf eine skalare Größe ab [116].

- die Skalarprodukt-Konstruktion,
- die radiale Konstruktion und
- die Tensorprodukt-Konstruktion.

An dieser Stelle werden allerdings nur die ersten beiden Konstruktionsmechanismen näher behandelt, da die Tensorprodukt-Konstruktion insbesondere bei Rasterkennfeldern und Splines benutzt werden, die i.A. nur für zwei bis drei Eingangsgrößen sinnvoll anwendbar sind [116]. Die radiale Konstruktion findet Anwendung bei sogenannten RBF-Netzen, die Skalarprodukt-Konstruktion bei Multilayer-Perceptron-Netzen (MLP-Netzen). Beide Netz-Typen werden in den folgenden Abschnitten beschrieben. Für einen detaillierten Einblick in die Theorie neuronaler Netze bieten sich insbesondere die Arbeiten von [17, 63, 116, 89] an.

2.2.1 Multilayer Perceptron Netze

Für ein Multilayer-Perceptron-Netz (MLP-Netz) ist die Verwendung einer Skalarprodukt-Konstruktion in Verbindung mit globalen Aktivierungsfunktionen, z.B. Sigmoiden, typisch [116]. Die Skalarprodukt-Konstruktion (engl. *ridge construction*) bildet Nichtlinearitäten nur entlang einer Richtung im Eingangsraum ab, nämlich in Richtung des Vektors $\underline{\theta}_i = [w_{i0}\ w_{i1}\ w_{i2}\ \cdots\ w_{ip}]^T$. Entlang aller anderer Richtungen ergibt sich ein konstantes Verhalten. Die Aktivität x_i berechnet sich mit $\underline{\tilde{u}} = [1\ u_1\ u_2\ \cdots\ u_p]$ nach:

$$x_i = \underline{\tilde{u}} \cdot \underline{\theta}_i = w_{i0} + w_{i1}u_1 + w_{i2}u_2 + \ldots w_{ip}u_p\,. \tag{2.15}$$

Mit Sigmoiden als Aktivierungsfunktionen ergibt sich für den Ausgang des sogenannten Perceptrons:

$$\Phi_i(\underline{u}, \underline{\theta_i}) = \frac{1}{1 + \exp\left(-x_i\right)}\,. \tag{2.16}$$

Beim MLP-Netz werden standardmäßig *alle* Parameter *nichtlinear* optimiert, d.h. sowohl die Parameter der verdeckten Schicht als auch die Ausgangsgewichte [116]. Diese Vorgehensweise hat allerdings zur Folge, dass das Training eher langsam abläuft. Außerdem besteht eine hohe Gefahr für lokale Optima, zumal die Gewichte des MLP-Netzes oft nur zufällig initialisiert werden. Einerseits macht die beliebige Ausrichtung und Anordnung

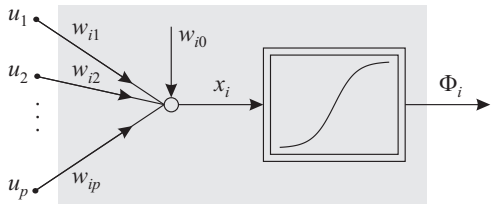

Bild 2.4: Perceptron: Ein Neuron der verdeckten Schicht eines MLPs [116].

der Sigmoiden im Eingangsraum eine Interpretation des resultierenden Modells äußerst schwierig. Andererseits eignet sich die hohe Flexibilität dieses Ansatzes insbesondere für hochdimensionale Abbildungen.

Die praktische Anwendung von MLP-Netzen kann sich oft als problematisch darstellen, sofern kein robuster Ansatz zur Initialisierung der Parameter der verdeckten Schicht vorliegt. Bei zufälliger Initialisierung ergeben sich nämlich keine reproduzierbaren Ergebnisse und der Anwender ist gezwungen, eine größere Lösungs*menge* zu erzeugen, um daraus letztendlich das „beste“ Modell auszuwählen.

In der Literatur sind viele verschiedene Ausprägungen von MLP-Netzen zu finden. Als Varianten des vorgestellten Ansatzes kann man z.B. mehr als nur eine verdeckte Schicht verwenden oder auch in der Ausgangsschicht sigmoide Aktivierungsfunktionen einsetzen. Darauf wird aber an dieser Stelle nicht näher eingegangen und der Vollständigkeit halber auf [62] verwiesen.

2.2.2 Netze mit radialen Basisfunktionen

Netze mit radialen Basisfunktionen oder auch RBF-Netze verwenden, wie der Name dieses Netztypus erkennen lässt, zur Berechnung des Neuronenausgangs eine *radiale* Konstruktion. Die Aktivität x_i des Neurons hängt vom radialen Abstand zum Zentrum $\underline{c}_i = [c_{i1}\ c_{i2}\ \cdots\ c_{ip}]^T$ des jeweiligen Neurons folgendermaßen ab:

$$x_i = \|\underline{u} - \underline{c}_i\|_{\underline{\Sigma}_i} = \sqrt{(\underline{u} - \underline{c}_i)^T \underline{\Sigma}_i^{-1} (\underline{u} - \underline{c}_i)} \tag{2.17}$$

Im allgemeinsten Fall ist das Abstandsmaß durch die Mahalonobisnorm definiert, welche den Einsatz symmetrischer Kovarianzmatrizen $\underline{\Sigma}$ erlaubt. In der Praxis wird das Abstandsmaß jedoch oft auf die Diagonalform von $\underline{\Sigma}$ eingeschränkt, weil dies die Anzahl der zu definierenden Parameter erheblich reduziert. Auf die Eigenschaften der Kovarianzmatrix wird in Kapitel 4.2 detaillierter eingegangen.

Üblicherweise kommen bei RBF-Netzen lokale oder streng lokale Aktivierungsfunktionen zum Einsatz [116]. Dadurch kann jedem Neuron ein bestimmter Bereich im Eingangsraum zugeordnet werden. Klassischer Vertreter dieser Gattung von Aktivierungsfunktionen ist die Gauß-Funktion, sodass sich der Neuronenausgang nach

$$\underline{\Phi}_i(\underline{u}, \underline{\theta_i}) = \exp\left(-\frac{1}{2}\|\underline{u} - \underline{c}_i\|^2_{\underline{\Sigma}_i}\right) \tag{2.18}$$

berechnet. Die Ausgangsgewichte w_i gehen *linear* in die Berechnung des Ausgangs $\hat{y}$ ein und

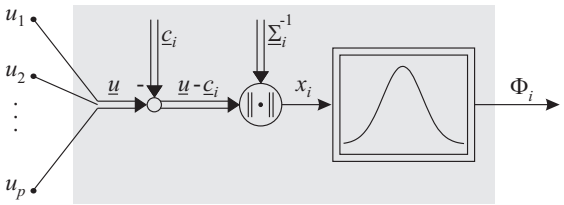

Bild 2.5: RBF-Neuron der verdeckten Schicht [116].

können daher mit der Least Squares-Methode geschätzt werden. Im Gegensatz zu MLP-Netzen wird diese Methode gewöhnlich der nichtlinearen Optimierung aller Netz-Parameter vorgezogen [118]. Zur Bestimmung der Gewichte der verdeckten Schicht lässt sich die gute Interpretierbarkeit von RBF-Neuronen ausnutzen. Die Zentren und Standardabweichungen der Gauß-Funktionen sind mittels verschiedener *Heuristiken* gut festlegbar. Dabei nimmt man bewusst in Kauf, das globale Optimum unter Umständen nicht zu treffen, um die Vorteile der Methode voll nutzen zu können: Das Netz lässt sich wesentlich schneller trainieren und gewährleistet eine gute Interpretierbarkeit. Auf diese Weise trainierte RBF-Netze eignen sich für mitteldimensionale Probleme [118]. Einen guten Überblick zu heuristischen Verfahren für RBF-Netze bieten z.B. [116, 118].

Eine wichtige Variante stellen *normierte* RBF-Netze (NRBF-Netze) dar, da sie ein wesentlich verbessertes Inter- und Extrapolationsverhalten gegenüber dem klassischen RBF-Netz ermöglichen. Die Ausgänge aller RBF-Neuronen werden in diesem Fall mit der Summe aller Neuronen-Ausgangswerte normiert:

$$\hat{y}(\underline{u}) = \sum_{i=1}^{M} w_i \widetilde{\Phi}_i\left(\underline{u}, \underline{\theta}_i\right) \quad \text{mit} \quad \widetilde{\Phi}_i\left(\underline{u}, \underline{\theta}_i\right) = \frac{\Phi_i\left(\underline{u}, \underline{\theta}_i\right)}{\sum_{j=1}^{M} \Phi_j\left(\underline{u}, \underline{\theta}_j\right)} . \tag{2.19}$$

Zum einen ist das Netz unempfindlicher gegenüber der Wahl der Standardabweichungen, da sich durch die Normierung glattere Interpolationsverläufe ergeben. Zum anderen ist es oft von Vorteil, dass am Rand liegende Messwerte konstant extrapoliert werden und nicht wie beim RBF-Netz die Extrapolation den Modellausgang $\hat{y}$ auf den Wert 0 zwingt.

Diese Vorteile sind jedoch mit der Einschränkung verbunden, dass durch den gemeinsamen Normierungsnenner alle Neuronen aneinander gekoppelt sind und dadurch z.B. eine unabhängige Strukturselektion der RBF-Neuronen nicht mehr möglich ist [118]. Für hochdimensionale Prozesse schlägt dieser Nachteil ganz besonders zu Buche, weil keine Möglichkeit besteht, den „Fluch der Dimensionalität“ durch Auswahl der wichtigsten Neuronen abzuschwächen. Darüber hinaus können sich durch die Normierung unerwünschte Effekte einstellen, die in Kapitel 4.2.2 näher behandelt werden.

Generell kann sowohl bei RBF- als auch bei NRBF-Netzen die resultierende Modellgüte aufgrund der verwendeten lokal *konstanten* Modelle starken Schwankungen unterliegen. Eine wichtige Erweiterung der bisher vorgestellten Basisfunktionennetze sind daher lokale Modellnetze, bei denen diese Einschränkung der lokalen Modellklasse aufgehoben ist.

2.3 Lokale Modellnetze

Lokale Modellnetze (kurz: LMN) können nach [116] als Erweiterung von NRBF-Netzen beziehungsweise als Erweiterung von Singleton-Neuro-Fuzzy-Netzen interpretiert werden. Dank ihrer breiten Anwendbarkeit (siehe z.B. [116, 139]) sind sie besonders gut zur Identifikation nichtlinearer statischer und dynamischer Prozesse geeignet.

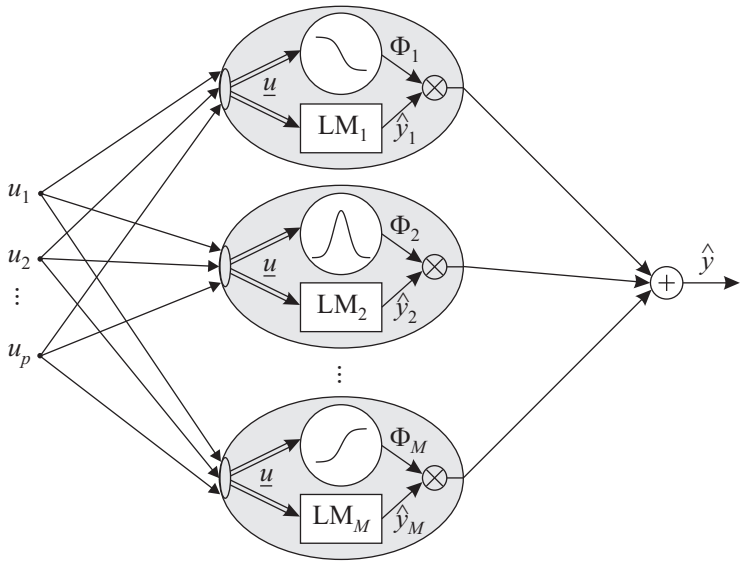

Bild 2.6: Lokales Modellnetz nach [116]. Der Modellausgang $\hat{y}$ ergibt sich durch Aufsummieren der gewichteten lokalen Modelle. Die Gewichtung der lokalen Modelle LM_i erfolgt durch die Gültigkeitsfunktionen $\Phi_i(\cdot)$.

Der Ausgang $\hat{y}$ eines lokalen Modellnetzes mit p Eingängen $\underline{u} = [u_1\ u_2\ \cdots\ u_p]^T$ ergibt sich aus der Summe von M lokalen Modellen $\hat{y}_i$, $i = 1, \ldots, M$, die mit ihren Gültigkeitsfunktionen $\Phi_i(\cdot)$ gewichtet werden [116], siehe Bild 2.6:

$$\hat{y} = \sum_{i=1}^{M} \hat{y}_i(\underline{u})\Phi_i(\underline{u})\,. \tag{2.20}$$

Eine wichtige Eigenschaft der Gültigkeitsfunktionen besteht darin, dass sie im Intervall $(0, 1] \in \mathbb{R}$ liegen und sich für jeden Eingangsvektor $\underline{u}$ die Summe aller $\Phi_i(\cdot)$ zu Eins ergibt:

$$\sum_{i=1}^{M} \Phi_i(\underline{u}) = 1\,. \tag{2.21}$$

Letzteres Merkmal stellt die Interpretierbarkeit des Netzes sicher, da man für jeden Wert von $\underline{u}$ eine prozentuale Aussage über die Gültigkeit aller lokalen Modelle treffen kann. Deswegen spricht man auch von der sog. *Einheitspartitionierung* (engl.: *partition of unity*).

Zur datenbasierten Modellbildung ist es generell wünschenswert, die Interpolation der lokalen Modelle möglichst glatt zu gestalten und deshalb auch glatte Funktionen wie z.B.

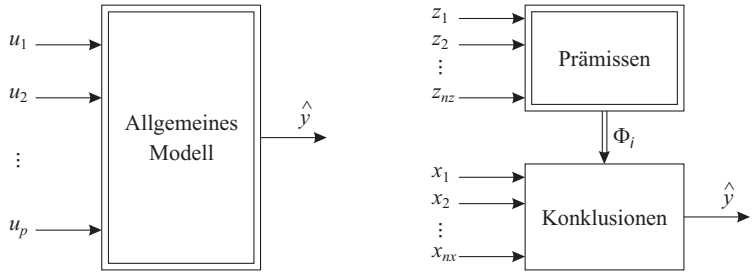

Bild 2.7: Die Eingangsgrößen können je nach ihrem linearen bzw. nichtlinearen Einfluss nur den Regelprämissen, nur den Regelkonklusionen oder beiden zugeführt werden [116].

Sigmoide oder normierte Gaußfunktionen als Gültigkeitsfunktionen zu wählen. In der Anwendung stellt der Kompromiss zwischen Glattheit und lokaler Interpretierbarkeit aber oft eine Herausforderung dar.

Oft ist es sinnvoll, den Eingangsraum der Gültigkeitsbereiche vom Eingangsraum der lokalen Modelle zu unterscheiden. In diesem Fall kann ein lokales Modellnetz verallgemeinert als Takagi-Sugeno-Fuzzy-System [146] interpretiert werden und die Erweiterung von (2.20) ergibt sich zu [116]:

$$\hat{y} = \sum_{i=1}^{M} \hat{y}_i(\underline{x})\Phi_i(\underline{z})\,, \tag{2.22}$$

wobei $\underline{z} = [z_1 \; z_2 \; \cdots \; z_{nz}]^T$ den Eingangsraum für die Regelprämissen (Wenn...) und $\underline{x} = [x_1 \; x_2 \; \cdots \; x_{nx}]^T$ den Eingangsraum für die Regelkonklusionen (Dann...) aufspannen, siehe Bild 2.7. Durch diese Vorgehensweise lässt sich die Komplexität der Modellierungsaufgabe oft drastisch reduzieren. Dies gilt insbesondere für dynamische Modelle, deren Eingangsräume wegen der nötigen Zeitverschiebungen $(u(k-1), u(k-2), \ldots, y(k-1), y(k-2), \ldots)$ meist sehr hochdimensional werden. Als Eingänge für die Regelprämissen müssen nur die Größen verwendet werden, deren vermuteter Einfluss *nichtlinearer* Natur ist, während die Regelkonklusionen die *linear* beeinflussenden Größen verarbeiten. Prinzipiell können die lokalen Modelle $\hat{y}_i(\cdot)$ als beliebige lineare oder nichtlineare Funktionen gewählt werden. Es ist allerdings extrem vorteilhaft, die lokalen Modelle *linear* zu parametrieren, damit sie sich mit Least Squares linear optimieren lassen. Üblicherweise werden deshalb Polynome als lokale Modelle verwendet [116].

Zur besseren Übersicht ist es hilfreich, einige Spezialfälle zu betrachten: Falls die $\hat{y}_i(\cdot)$ konstant gewählt werden, erhält man ein NRBF-Netz. Für lineare bzw. affine $\hat{y}_i(\cdot)$ erhält man ein lokal lineares Neuro-Fuzzy-Modell. Die Funktionen $\hat{y}_i(\cdot)$ können als *lokale Approximatoren* der unbekannten Funktion angesehen werden, für welche das Modell erstellt wird. In diesem Zusammenhang lassen sich konstante und lineare lokale Modelle (LLM) als *Taylorreihenentwicklungen* nullter und erster Ordnung der unbekannten Funktion ansehen. Je höher man die Ordnung der Taylorreihenentwicklung wählt, desto flexibler gestalten sich die lokalen Modelle und resultieren damit zu Polynomen immer höheren Grades. Bild 2.8 illustriert die Approximation einer nichtlinearen Funktion mit einem lokalen Neuro-Fuzzy-Modell. Dabei kommen a) zwölf Polynome nullten Grades, b) fünf Polynome ersten Grades, c) zwei Polynome zweiten Grades (bei diesem Beispiel die beste Alternative) und schließlich d) ein Polynom dritten Grades zum Einsatz.

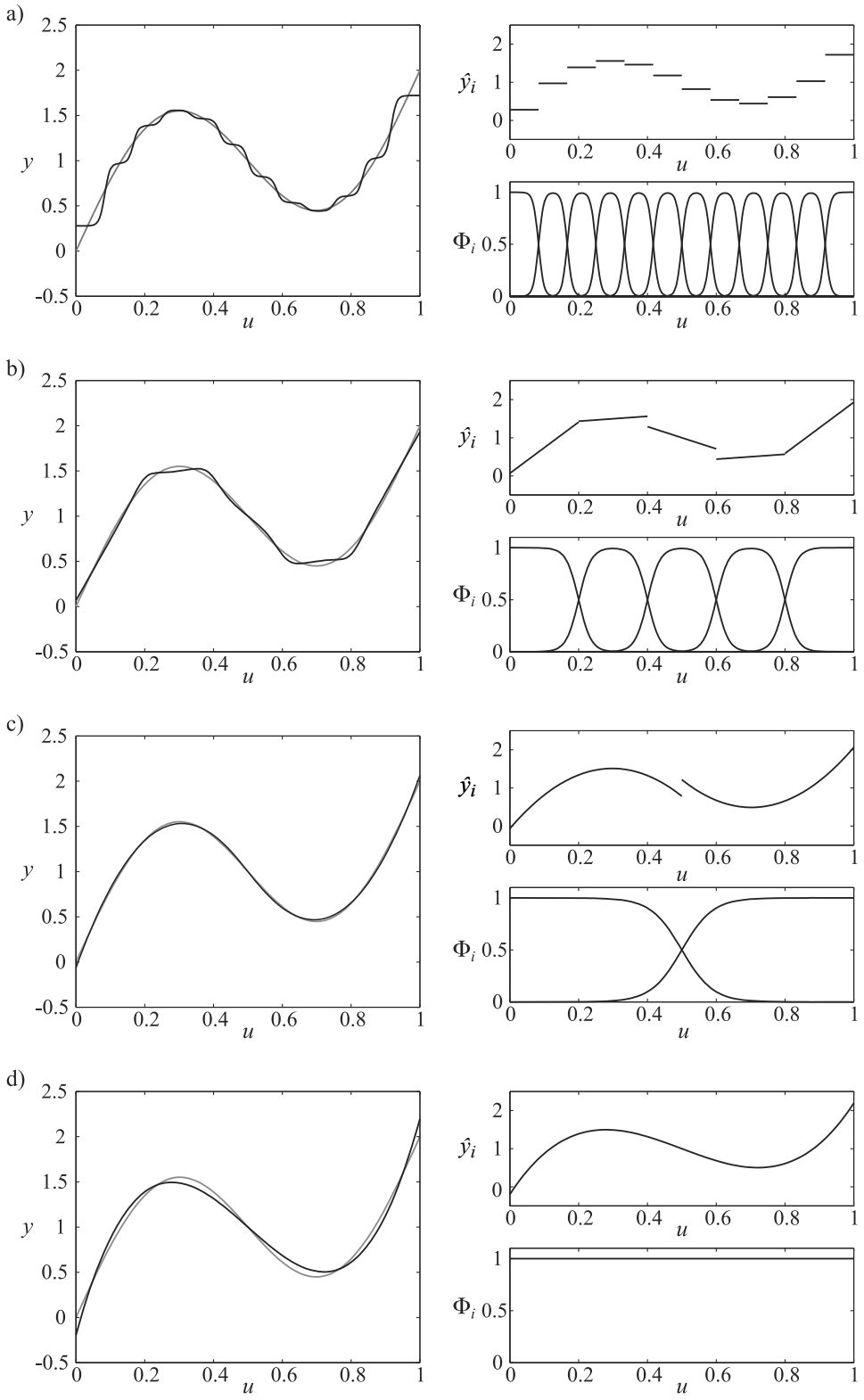

Bild 2.8: Approximation mit einem lokal polynomialen Modellnetz mit a) konstanten, b) linearen, c) quadratischen und d) kubischen lokalen Modellen. Links ist der Prozess im Vergleich zum Modell und rechts die lokalen Modelle $\hat{y}_i$ mit den korrespondierenden Gültigkeitsfunktionen Φ_i dargestellt [116].

Demnach stellt die Modellbildung mit lokalen polynomialen Modellen einen Kompromiss zwischen zwei Extremfällen dar: Einerseits ein NRBF-Netz (viele lokal konstante Modelle) und andererseits ein reines Polynommodell. Beide Ansätze können je nach Anwendungsgebiet gravierende Nachteile mit sich bringen. Die Komplexität des Gesamtmodells hängt sowohl von der Anzahl der benötigten lokalen Modelle M (Neuronen) als auch von der Anzahl der Parameter pro lokalem Modell ab. Wie Tabelle 2.1 zeigt, besitzt ein NRBF-Netz nur einen einzigen (linearen) Parameter pro lokalem Modell. Allerdings wird eine sehr große Zahl an Neuronen notwendig. Im Gegensatz dazu benötigt ein Polynom hohen Grades eine große Anzahl an Parametern, jedoch reicht lediglich ein einzelnes Neuron aus.

Tabelle 2.1: Vom NRBF-Netz zum Polynom.

Ordnung	Parameter pro lokalem Modell	M	Name
konstant	1	sehr groß	NRBF-Netz
linear	$p+1$	groß	lokal lineares NF-Modell
quadratisch	$(p+2)(p+1)/2$	mittel	lokal quadrat. NF-Modell
$\vdots$	$\vdots$	$\vdots$	$\vdots$
l-te Ordnung	$(l+p)!/(l!\,p!)$	1	Polynom

p = Anzahl der Eingangsgrößen, M = Anzahl der lokalen Modelle, NF = Neuro-Fuzzy.

In der Vergangenheit haben sich lokal *lineare* Ansätze etabliert, da diese in vielen Fällen einen sehr guten Kompromiss bei der Modellbildung darstellen. Bei einigen Anwendungen, vor allem im Rahmen einer modellbasierten Optimierung, lohnt sich zudem der Schritt zur nächst höheren Komplexitätsstufe, siehe z.B. [130, 116, 12, 11]. In [116] wird gezeigt, dass lokal quadratische Modelle ein Prozessoptimum vergleichsweise besser approximieren können als lokal lineare Modelle. Neuere Ansätze schlagen außerdem vor, die Struktur der lokalen Modelle automatisch mittels Strukturselektionsverfahren individuell aussuchen zu lassen [136, 11]. Dies bietet die Möglichkeit, lokale Modelle unterschiedlicher Struktur gleichzeitig im Gesamtmodell unterzubringen und somit ein wesentlich flexibleres Modell zu erzeugen.

Für alle Varianten eines lokalen Modellnetzes ist jedoch eine gute Partitionierung, d.h die richtige Wahl der Gültigkeitsfunktionen $\Phi_i(\cdot)$, entscheidend für den Erfolg. Allerdings stellt die Optimierung der Partitionierungsparameter eine große Herausforderung dar, weil dies ein komplexes nichtlineares Optimierungsproblem ist. Auf diese Thematik wird daher in Kapitel 4 besonders detailliert eingegangen.

2.4 Zusammenfassung

In diesem Kapitel wurden gängige Verfahren zur nichtlinearen Regression vorgestellt. Wichtigste Methode im praktischen Umfeld ist wohl die klassische Polynomregression, die üblicherweise durch Strukturselektionsverfahren ergänzt wird. Eine weitere Klasse nichtlinearer Regressionsverfahren stellen neuronale Netze dar. Daher wurden außerdem das Multilayer-Perceptron und radiale Basisfunktionennetze näher erklärt. Die daraufhin vorgestellte Methode der lokalen Modellnetze bildet die Basis dieser Arbeit und wurde daher im Detail beschrieben.

3 Lineare Optimierung lokaler Modellparameter

Beim Training lokaler Modellnetze werden sowohl die Modellstruktur, das heißt die Lage und Ausrichtung der Gültigkeitsfunktionen $\Phi_i(\cdot)$, als auch die Parameter der lokalen Modelle bestimmt. Prinzipiell ist es möglich, die lokalen Modelle beliebig zu wählen. In der Praxis werden jedoch fast ausschließlich *linear* parametrierte lokale Modelle, z.B. Polynome, bevorzugt. Dies ist äußerst vorteilhaft, da so die Parameter mit der großen Palette linearer Schätzverfahren optimiert werden können. Mit genau dieser Problemklasse setzt sich dieses Kapitel auseinander.

Das wichtigste Verfahren ist die Methode der kleinsten Fehlerquadrate bzw. das Least Squares-Verfahren. Um zu verstehen, welche Annahmen für diese Methode zu treffen sind, gehen die Kapitel 3.1 und 3.2 zunächst auf die Bayes'sche Methode und die Maximum Likelihood-Schätzung ein. Dies bildet zudem die Basis zur Herleitung des AIC-Kriteriums in Kapitel 3.5.3. Die Grundlagen zur gewichteten Least Squares-Schätzung werden in Kapitel 3.3 behandelt. Explizite Regularisierungsverfahren können durch gezielte Reduktion der Modellflexibilität die Vorhersagegenauigkeit des Modells verbessern. Diese Thematik greift Kapitel 3.4 auf. Anschließend werden in Kapitel 3.5 statistische Methoden zur Abschätzung der optimalen Modellkomplexität vorgestellt. Zum Schluss widmet sich Kapitel 3.6 der Strukturselektion, speziell für die lokal gewichtete Least Squares-Schätzung, gefolgt von einer Zusammenfassung des Kapitels.

3.1 Bayes'sche Methode

Die Bayes'sche Methode nimmt die Annahme zum Ausgangspunkt, dass die Messungen $\underline{y} = [y(1)\ y(2)\ \ldots\ y(N)]^T$ und die Parameter $\underline{\theta} = [\theta_1\ \theta_2\ \ldots\ \theta_n]^T$ durch eine *gemeinsame* Wahrscheinlichkeitsdichte $p(\underline{y}, \underline{\theta})$ beschrieben werden können [116, 17]. Diese Art der Parameterschätzung ist sehr allgemein gehalten und bedarf eines großen Vorwissens über den zu modellierenden Prozess, welches in dieser Form in den wenigsten Fällen vorhanden sein dürfte. Allerdings bildet sie die theoretische Grundlage aller Parameterschätzverfahren in diesem Kapitel und findet daher an dieser Stelle Beachtung.

Nach dem Satz von Bayes lässt sich die gemeinsame Dichtefunktion $p(\underline{y}, \underline{\theta})$ mit Hilfe der *bedingten* Wahrscheinlichkeitsdichten $p(\underline{\theta}|\underline{y})$ und $p(\underline{y}|\underline{\theta})$ folgendermaßen berechnen:

$$p(\underline{y}, \underline{\theta}) = \underbrace{p(\underline{\theta}|\underline{y})}_{a\ posteriori} \cdot p(\underline{y}) = p(\underline{y}|\underline{\theta}) \cdot \underbrace{p(\underline{\theta})}_{a\ priori} . \tag{3.1}$$

Dabei entspricht $p(\underline{\theta}|\underline{y})$ der *a posteriori*- und $p(\underline{\theta})$ der *a priori*-Wahrscheinlichkeitsdichte. Um das wahrscheinlichste Modell zu finden, muss die Dichte $p(\underline{y}, \underline{\theta})$ maximiert werden.

Im Fall der Modellbildung geht man von vorab gegebenen Messdaten $\underline{y}$ bzw. einer gegebenen Verteilungsdichte $p(\underline{y})$ aus. Daraus folgt, dass die Modellschätzung von der Verteilungsdichte $p(\underline{y})$ unberührt bleibt. Die Information der Messdaten lässt sich in das Optimierungsproblem einbeziehen und die Formulierung ergibt sich zu

$$p(\underline{\theta}|\underline{y}) = \frac{p(\underline{y}|\underline{\theta})p(\underline{\theta})}{p(\underline{y})} \rightarrow \max_{\underline{\theta}}, \tag{3.2}$$

welche als *Maximum a posteriori*-Methode (Abk.: MAP) bezeichnet wird.

3.2 Maximum Likelihood-Verfahren

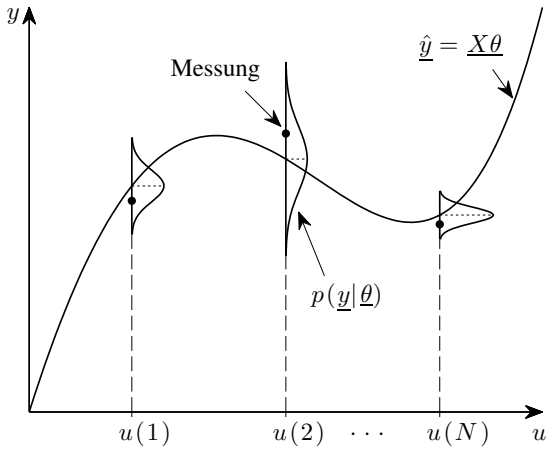

Bild 3.1: Bei der Modellbildung mit der ML-Methode möchte man die Likelihoodfunktion maximieren. In dieser Skizze wird beispielhaft gaußverteiltes Messrauschen dargestellt.

Durch das Einbinden weiterer Annahmen lässt sich die *Maximum a posteriori*-Methode zum *Maximum Likelihood*-Verfahren (Abk.: ML) vereinfachen. Ziel bei der ML-Methode ist die Maximierung der Likelihoodfunktion $\mathcal{L}(\underline{\theta}|\underline{y})$, die sich unter folgenden Voraussetzungen aus der MAP-Dichte $p(\underline{\theta}|\underline{y})$ in Gl. (3.2) ergibt:

- Die Verteilungsdichte der Messdaten $p(\underline{y})$ ist nicht beeinflussbar durch die Parameter $\underline{\theta}$. Das Äquivalent zum Optimierungsproblem in Gl. (3.2) ist demnach:

$$p(\underline{y}|\underline{\theta})p(\underline{\theta}) \rightarrow \max_{\underline{\theta}} . \tag{3.3}$$

- Darüber hinaus lässt sich in den meisten Fällen keine *a priori*-Aussage über die Verteilungsdichte der Parameter $p(\underline{\theta})$ treffen. Dann bleibt nur die Möglichkeit, alle Parameter als gleichwahrscheinlich anzunehmen und damit $p(\underline{\theta}) = \text{konst.}$ zu setzen. Dadurch wird der Einfluss der Parameterverteilung bei der Optimierung eliminiert. Diese Vereinfachung führt dann auf:

$$p(\underline{y}|\underline{\theta}) \rightarrow \max_{\underline{\theta}} . \tag{3.4}$$

- Das Modell $\underline{\hat{y}}$ ist gleichbedeutend mit den Mittelwerten der Dichten $p(\underline{y}|\underline{\theta})$ der Messungen, weil dort die Wahrscheinlichkeit am größten ist. Eine Veranschaulichung dazu zeigt Bild 3.1.
- Für den Fehler des i-ten Messwerts $e(i) = y(i) - \hat{y}(i)$ wird gaußverteiltes, mittelwertfreies Rauschen angenommen. Letztere Annahme deckt sich mit der vorangegangenen Forderung, dass das Modell durch die Mittelwerte der Dichten $p(y(i)|\underline{\theta})$ beschrieben ist. Die Gaußverteilung ist in der Praxis oft eine gute Wahl, da sich nach dem zentralen Grenzwertsatz der Statistik die Superposition der Verteilungen vieler statistisch unabhängiger Zufallsvariablen einer Gaußverteilung annähert [46]. Je mehr Effekte sich überlagern, desto mehr ähneln die Verteilungen einer Gaußverteilung.
- Zur Formulierung der Likelihoodfunktion wird vorausgesetzt, dass alle Messungen $y(i)$ statistisch unabhängig und die Fehler $e(i)$ unkorreliert sind. Der Fehler findet daher folgende Beschreibung [116]:

$$p\left(y(i)|\underline{\theta}\right) := p\left(e(i)\right) = \frac{1}{\sqrt{2\pi}\sigma(i)} \exp\left(-\frac{1}{2}\frac{e^2(i)}{\sigma^2(i)}\right) . \tag{3.5}$$

  In diesem Fall dürfen die Wahrscheinlichkeitsdichten der einzelnen Messpunkte miteinander multipliziert werden, um auf die Likelihoodfunktion $\mathcal{L}$ zu kommen:

$$\mathcal{L}(\underline{\theta}|\underline{y}) = p\left(e(1)\right) \cdot p(e(2)) \cdot \ldots \cdot p(e(N)) \rightarrow \max_{\underline{\theta}} . \tag{3.6}$$

  Durch Maximierung der Likelihoodfunktion lässt sich der optimale Parametervektor $\underline{\theta}$ bestimmen.

Insbesondere bei der Annahme unkorrelierten Rauschens ist es mathematisch geschickter, nicht direkt die Likelihoodfunktion zu maximieren, sondern zunächst den negativen Logarithmus anzuwenden, um auf die sogenannte Log-Likelihoodfunktion zu kommen. Aus den Produkten wird dadurch eine Summe und das äquivalente Optimierungsproblem lautet:

$$J(\underline{\theta}) = -\ln\left(p(e(1))\right) - \ln\left(p(e(2))\right) - \ldots - \ln\left(p(e(N))\right) \rightarrow \min_{\underline{\theta}} . \tag{3.7}$$

Wendet man darauf Gl. (3.5) an, resultiert daraus die Verlustfunktion:

$$J(\underline{\theta}) = \frac{1}{\sigma^2(1)} e^2(1) + \frac{1}{\sigma^2(2)} e^2(2) + \ldots + \frac{1}{\sigma^2(N)} e^2(N) \rightarrow \min_{\underline{\theta}} . \tag{3.8}$$

Zuletzt geht die lineare Least Squares-Methode (LS) davon aus, dass die Varianz des Rauschens über alle Messpunkte identisch und konstant ist. In der Folge ist das Optimierungsproblem unabhängig von den Varianzen σ^2:

$$J(\underline{\theta}) = e^2(1) + e^2(2) + \ldots + e^2(N) \rightarrow \min_{\underline{\theta}} . \tag{3.9}$$

Diese Verlustfunktion entspricht der klassischen Least Squares-Formulierung und ist mit den getroffenen Annahmen gleichbedeutend mit der Maximum Likelihood-Schätzung. Allerdings bietet die ML-Methode darüber hinaus die Möglichkeit, auch andere Verteilungen wie z.B. eine Exponentialverteilung des Rauschens für die Schätzung anzunehmen:

$$p\left(e(i)\right) = \frac{1}{2\sigma} \exp\left(-\frac{|e(i)|}{\sigma}\right) . \tag{3.10}$$

Die Parameterschätzung mit Exponentialverteilung ist dann weniger anfällig bezüglich Ausreißern in den Messdaten als die LS-Schätzung, weil dabei die Summe der Fehlerbeträge minimiert wird:

$$J(\underline{\theta}) = |e(1)| + |e(2)| + \ldots + |e(N)| \rightarrow \min_{\underline{\theta}} . \tag{3.11}$$

Eine weitere Möglichkeit ist beispielsweise die Verwendung einer Gleichverteilung

$$p\left(e(i)\right) = \begin{cases} 1/2c & \text{falls } |e(i)| \leq c \\ 0 & \text{falls } |e(i)| > c \end{cases} , \tag{3.12}$$

wobei die Konstante c ein gegebener Schwellwert ist. Für Fehler $|e(i)| > c$ sind diese nicht mehr dem Messrauschen zuzuordnen, sondern werden als Ausreißer mit der Wahrscheinlichkeit Null versehen. Daraus ergibt sich ein Min-Max-Problem, bei dem der *maximal* auftretende Fehler *minimiert* wird [116]:

$$J(\underline{\theta}) = \max\left(e(1), e(2), \ldots, e(N)\right) \rightarrow \min_{\underline{\theta}} . \tag{3.13}$$

Die Ausführungen in dieser Arbeit konzentrieren sich jedoch ausschließlich auf den Least Squares-Fall. Für nähere Informationen zur Bayes'schen Methode sei beispielsweise auf [61], [17] oder [16] verwiesen.

3.3 Gewichtete Least Squares-Schätzung

Bei der lokal gewichteten Least Squares-Schätzung unterscheidet man zwei Vorgehensweisen: die *globale* und die *lokale* Schätzung. Während bei der globalen Schätzung alle Parameter direkt mit Least Squares in einem Schritt geschätzt werden, findet die lokale Schätzung separat für jedes lokale Modell (LM) statt. Dadurch wird die Überlappung benachbarter lokaler Modelle vernachlässigt und das Modell verliert an Flexibilität [26], [113]. In den folgenden Abschnitten wird detailliert auf diese Schätzverfahren eingegangen.

3.3.1 Globale Schätzung

Beim Ansatz der globalen Schätzung werden alle Parameter des Neuro-Fuzzy-Modells *gleichzeitig* in einer einzigen LS-Schätzung optimiert. Der Parametervektor $\underline{w}$ enthält dann alle $M(p+1)$ Parameter der lokalen Modelle, wobei M die Anzahl der lokalen Modelle und p die Anzahl der Eingangsgrößen bedeuten. Die Parametervektor lautet dann

$$\underline{w} = [\underline{w}_1^T \ \underline{w}_2^T \ \cdots \ \underline{w}_M^T]^T \tag{3.14}$$

und hat entsprechend die zugehörige Regressionsmatrix

$$\underline{X}_g = \left[\ \underline{X}_1^{(\mathrm{sub})} \quad \underline{X}_2^{(\mathrm{sub})} \quad \cdots \quad \underline{X}_M^{(\mathrm{sub})} \ \right] \tag{3.15}$$

mit den Untermatrizen

$$\underline{X}_i^{(\mathrm{sub})} = \underline{Q}_i \underline{X}_i . \tag{3.16}$$

Die Gewichtungsmatrix $\underline{Q}_i$ des i-ten lokalen Modells hat eine Diagonalstruktur und die Dimension $(N \times N)$:

$$\underline{Q}_i = \operatorname{diag}\left(\left[\begin{array}{cccc} \Phi_i(\underline{u}(1)) & \Phi_i(\underline{u}(2)) & \cdots & \Phi_i(\underline{u}(N)) \end{array}\right]\right). \tag{3.17}$$

Werden lokal affine Modelle verwendet, lauten die zugehörigen Teil-Regressionsmatrizen $\underline{X}_i$

$$\underline{X}_i = \left[\begin{array}{ccccc} 1 & u_1(1) & u_2(1) & \cdots & u_p(1) \\ 1 & u_1(2) & u_2(2) & \cdots & u_p(2) \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & u_1(N) & u_2(N) & \cdots & u_p(N) \end{array}\right]. \tag{3.18}$$

Generell können die lokalen Modelle eine beliebige Struktur aufweisen, solange die Parameter linear in den Fehler eingehen. Der Einfachheit halber wird aber oft angenommen, dass alle lokalen Modelle eine identische Polynomstruktur haben. Dann simplifiziert sich die Formulierung des Schätzproblems, weil alle Matrizen $\underline{X}_i$ identisch sind. Der Modellausgang $\underline{\hat{y}} = [\hat{y}(1)\ \hat{y}(2)\ \cdots\ \hat{y}(N)]^T$ ergibt sich allgemein zu $\underline{\hat{y}} = \underline{X}_g \underline{w}$.

Bei der globalen Schätzung lautet die zu minimierende Verlustfunktion:

$$J_{\text{global}}(\underline{w}) = \sum_{k=1}^{N} e^2(k) = \underline{e}^T \underline{e} \longrightarrow \min_{\underline{w}}, \tag{3.19}$$

wobei $e(k) = y(k) - \hat{y}(k)$ den Modellfehler bezüglich des Trainingsdatenpunktes $\{\underline{u}(k), y(k)\}$ repräsentiert. Die quadratische Verlustfunktion J_{global} erlaubt es, die Parameter mit der Least Squares-Methode zu bestimmen:

$$\underline{\hat{w}} = \left(\underline{X}_g^T \underline{X}_g\right)^{-1} \underline{X}_g^T \underline{y} \tag{3.20}$$

wobei der Vektor $\underline{y} = [y(1)\ y(2)\ \cdots\ y(N)]^T$ die Messwerte beinhaltet.

3.3.2 Lokale Schätzung

Das Besondere an der lokalen Schätzung ist, dass im Gegensatz zur globalen Schätzung das Gesamtproblem in kleinere Einzelprobleme aufgegliedert wird, indem man die Parameter jedes lokalen Modells *individuell* optimiert. Das führt dazu, dass nicht alle Parameter in einer einzigen Schätzung optimiert werden, sondern für jedes lokale Modell eine separate Schätzung durchzuführen ist.

Der wesentliche Unterschied zum globalen Ansatz liegt darin, dass die Überlappungen zwischen den lokalen Modellen vernachlässigt werden und damit eine Einschränkung der Flexibilität des globalen Modells in Kauf genommen wird. Dieser Effekt ist oft äußerst wünschenswert, insbesondere dann, wenn sich durch eine zu hohe Modellflexibilität die Gefahr des Overfittings einstellt. Durch die lokale Schätzung wird die Anzahl der effektiven Parameter des Modells umso kleiner, je mehr sich die Gültigkeitsbereiche der lokalen Modelle überlappen.

Dieser Regularisierungseffekt wird z.B. in [115] und [116] detailliert erklärt. Er ist umso stärker ausgeprägt, je mehr benachbarte lokale Modelle ineinander überlappen. Der Grund

dafür liegt in der gewichteten Least Squares-Schätzung. Eine Überschneidung der Gültigkeitsbereiche führt dazu, dass ein Teil der Daten gleichermaßen zur Schätzung mehrerer lokaler Modelle genutzt wird. Je größer diese Überlappungsbereiche sind, desto gleichmäßiger teilen sich die Gewichtungen auf verschiedene lokale Modelle auf.

Wie eingangs erwähnt, müssen bei der lokalen Schätzung nicht alle $M(p+1)$ Parameter simultan geschätzt, sondern können im Gegensatz zur globalen Schätzung sequentiell für alle lokalen Modelle mit jeweils $p+1$ Parametern bestimmt werden. Der i-te Parametervektor der einzelnen LM-Schätzung lautet

$$\underline{w}_i = [w_{i0}\ w_{i1}\ \cdots\ w_{ip}]^T\,, \quad i = 1, \ldots, M\,. \tag{3.21}$$

In welchen Regionen des Eingangsraums ein durch $\hat{y}_i = \underline{X}_i \underline{w}_i$ beschriebenes lokales Modell zum Gesamtmodell beträgt, ist durch die jeweilige Gültigkeitsfunktion $\Phi_i(\cdot)$ festgelegt. Üblicherweise kommt bei lokalen Modellnetzen die sogenannte Einheitspartitionierung zum Einsatz, wonach sich alle Gültigkeitsfunktionen für jeden Eingangswert $\underline{u}$ auf den Wert Eins summieren. Dadurch wird eine Aussage zum prozentualen Anteil jedes lokalen Modells in allen Bereichen des Eingangsraumes möglich. In der Nähe des Zentrums eines lokalen Modells hat die jeweilige Gültigkeitsfunktion ihr Maximum. Datenpunkte in dieser Region sind höchst relevant für die Schätzung der LM-Parameter $\underline{w}_i$.

Je weiter man sich vom Zentrum entfernt, desto geringer ist der Beitrag des lokalen Modells zum Gesamtmodell. Datenpunkte weit außerhalb des LM-Zentrums sind daher eher unwichtig für die Schätzung und bekommen deshalb eine geringe Gewichtung. Somit nimmt die Gültigkeit ab, bis sie nahezu verschwindet. Die Gültigkeit bezüglich anderer, benachbarter lokaler Modelle wird entsprechend umso größer.

Die Gültigkeitsfunktionen der lokalen Modelle können durch die Verwendung einer gewichteten Least Squares-Schätzung bei der Optimierung berücksichtigt werden. Die Gewichtung erfolgt dann mit der jeweiligen Gültigkeitsfunktion und es ergibt sich die Verlustfunktion:

$$J_{i,\text{lokal}}(\underline{w}_i) = \sum_{k=1}^{N} \Phi_i(\underline{u}(k))\, e^2(k) \;\longrightarrow\; \min_{\underline{w}_i}\,. \tag{3.22}$$

Die Lösung des gewichteten Least Squares-Problems für das i-te lokale Modell lautet dann:

$$\underline{\hat{w}}_i = \left(\underline{X}_i^T \underline{Q}_i \underline{X}_i\right)^{-1} \underline{X}_i^T \underline{Q}_i \underline{y}\,. \tag{3.23}$$

Diese Schätzung muss sukzessive für alle $i = 1, \ldots, M$ lokalen Modelle durchgeführt werden. Für die praktischen Umsetzung ist die sequentielle Vorgehensweise wegen der kompakt gehaltenen Inversen $\left(\underline{X}_i^T \underline{Q}_i \underline{X}_i\right)^{-1}$ sinnvoll. Für einige mathematische Betrachtungen, beispielsweise im Zusammenhang mit Regularisierung kann jedoch die Realisierung einer lokalen Schätzung *in einem Schritt* sinnvoll sein, siehe z.B. [150]. Dabei werden alle LM-Parameter ähnlich der globalen Schätzung gleichzeitig in einem Schritt bestimmt. Dies ermöglicht auch eine regularisierte Schätzvariante, die einen Kompromiss zwischen globaler und lokaler Schätzung erlaubt. Ein entsprechender Regularisierungsoperator ist in Kapitel 3.4.3 beschrieben.

3.3.3 Implizite Regularisierung durch lokale Schätzung

Ein großer Vorteil der lokalen Schätzung ist der geringere Rechenaufwand. Im Gegensatz zur globalen Schätzung müssen hierbei nur $p+1$ Parameter auf einmal mit Least Squares optimiert werden. Da jedes lokale Modell separat geschätzt wird, steigt der Rechenaufwand lediglich *linear* mit der Anzahl der Neuronen an. Damit gilt für die Ordnung $\mathcal{O}\left(M(p+1)^3\right)$.

Hier erkennt man den signifikant niedrigeren Rechenaufwand, vergleicht man mit der globalen Schätzung, die sich in einem kubischen Anstieg des Rechenaufwands, d.h. mit $\mathcal{O}\left(M^3(p+1)^3\right) \approx \mathcal{O}(M^3p^3)$, niederschlägt [116]. Zunächst nachteilig an diesem Verfahren erscheint jedoch ein zusätzlich eingebrachter Fehler, der durch die Vernachlässigung der Interaktion zwischen den lokalen Modellen bei der Schätzung zustande kommt.

Wie in [113] und [115] gezeigt, führt die lokale Schätzung auf eine Erhöhung des *Biasfehlers*[1]. Als Konsequenz reduziert sich damit die Flexibilität des Modells. Allerdings tritt andererseits ein weiterer, äußerst erwünschter Effekt dadurch auf: Durch die reduzierte Flexibilität verringert sich der *Varianzfehler* des Modells, was eine Reduzierung der Anzahl der *effektiven Parameter* n_{eff} des Modells bedeutet. Die *nominelle* Parameteranzahl bleibt zwar davon unberührt, jedoch wird das Modell robuster bezüglich Overfitting.

Die effektive Parameteranzahl steht eng im Zusammenhang mit der Flexibilität des Modells. Mathematisch kann man die Flexibilität anhand der Glättungsmatrix $\underline{S}$ spezifizieren. Die Glättungsmatrix beschreibt die Abbildung der Messdaten $\underline{y}$ auf den Modellausgang $\hat{\underline{y}}$. Wertet man das Modell an den Messpunkten aus, berechnet sich der gewichtete Modellausgang des i-ten lokalen Modells zu:

$$\underline{Q}_i\hat{\underline{y}}_i = \underline{Q}_i\underline{X}_i\underline{w}_i = \underline{Q}_i\underline{X}_i\left(\underline{X}_i^T\underline{Q}_i\underline{X}_i\right)^{-1}\underline{X}_i^T\underline{Q}_i\underline{y} = \underline{S}_i\underline{y}\,, \tag{3.24}$$

wobei für letztere Umformung die Least Squares-Formel aus Gl. (3.23) eingesetzt wurde. Die Glättungsmatrix $\underline{S}$ des globalen Modells, d.h. mit $\hat{\underline{y}} = \underline{S}\underline{y}$, ergibt sich aus der Summe über alle lokalen Modelle:

$$\underline{S} = \sum_{i=1}^{M}\underline{S}_i\,. \tag{3.25}$$

Nach [115] ist die Anzahl der effektiven Parameter bei Anwendung einer lokalen Schätzung:

$$n_{\text{eff}} = \text{spur}(\underline{S}\underline{S}^T)\,. \tag{3.26}$$

Darüber hinaus ergibt sich die Gesamtzahl der effektiven Parameter n_{eff} aus der Summe der effektiven Parameter jedes einzelnen lokalen Modells $n_{\text{eff},i}$ [115]:

$$n_{\text{eff}} = \sum_{i=1}^{M} n_{\text{eff},i}\,. \tag{3.27}$$

Wenn die Anzahl der effektiven Parameter kleiner als die Anzahl der nominellen Parameter ausfällt, deutet das auf eine Regularisierungstechnik hin. Deshalb kann der lokalen Schätzung ein *impliziter Regularisierungseffekt* zugeschrieben werden [116]. Je größer die Spanne

[1] Eine umfangreiche Diskussion zum Bias-Varianz-Kompromiss findet sich in Kapitel 3.5.

zwischen nomineller und effektiver Parameteranzahl ist, desto stärker ist die Schätzung regularisiert. Dies wird durch die Gewichtungsmatrix $\underline{Q}_i$ in Gl. (3.24) beeinflusst. Daraus ergibt sich, dass die Anzahl der effektiven Parameter entscheidend vom Grad der Überlappung benachbarter lokaler Modelle abhängt. Zwei Grenzfälle lassen sich bei der lokalen Schätzung unterscheiden:

1. *Keine Überlappung:* Die Gültigkeitsfunktionen trennen die Bereiche zwischen den lokalen Modellen sehr scharf voneinander ab und sorgen für sprungförmiges Umschalten. Unter diesen Umständen hat das lokal geschätzte Modell maximale Flexibilität und die Anzahl der effektiven Parameter gleicht der Gesamtzahl an Modellparametern, d.h. $n_{\text{eff}} = n$.
2. *Starke Überlappung:* Alle lokalen Modelle tragen in jedem Bereich des Eingangsraums gleichermaßen zum Gesamtmodell bei. Dabei reduzieren sich die Gültigkeitsfunktionen auf einen konstanten Faktor $\Phi_i(\cdot) = 1/M$. Dieser Extremfall entspräche wegen der identischen Gewichtungsmatrix dann lediglich der Flexibilität eines einzigen lokalen Modells, d.h. $n_{\text{eff}} = n/M$.

Je mehr sich demnach die lokalen Modellgebiete überschneiden, desto geringer fällt die Anzahl der effektiven Parameter und damit die Flexibilität des Modells aus.

Eine weitere übliche Vorgehensweise bei der Schätzung lokaler Modelle ist die Kombination der globalen Schätzung mit einer *expliziten Regularisierungstechnik*, welche ebenfalls die Flexibilität des Modells reduziert, aber auf anderen Glattheitsbedingungen beruht als die implizite Regularisierung bei der lokalen Schätzung.

3.4 Explizite Regularisierung

Der Einsatz einer Regularisierungstechnik ist immer dann sinnvoll, wenn hochdimensionale Modelle mit wenigen Messdaten erzeugt werden sollen. Viele Eingangsgrößen verlangen größtmögliche Flexibilität des Modells. Problematisch dabei ist allerdings, dass sich mit zu hoher Flexibilität Overfitting einstellt. Die Herausforderung bei der Modellbildung besteht also darin, das richtige Maß an Flexibilität zu erreichen. Gerade bei derartigen Problemstellungen führt daher die globale Schätzung meist nicht zum gewünschten Ziel. Im Unterschied zur implizit regularisierten lokalen Schätzung kann das global geschätzte Modell noch gezielter geglättet werden, wenn eine Glättungsvorschrift basierend auf Vorwissen vorgegeben wird. Je nach Anwendung kann diese Vorschrift sehr unterschiedlich ausfallen. Dabei steht jedoch immer im Vordergrund, die richtige Balance zwischen Bias und Varianz der Schätzung zu ermitteln.

Mathematisch betrachtet führen viele Eingangsgrößen bei der globalen Schätzung auf eine schlecht konditionierte Hesse-Matrix. Gründe dafür sind: Viele Eingänge bedeuten, dass auch viele Parameter geschätzt werden müssen. Außerdem dehnt sich der Raum, in dem die Daten liegen, exponentiell mit jeder Eingangsgröße aus (sog. *Fluch der Dimensionalität*), sodass die Messdaten im Hochdimensionalen nur sehr spärlich verteilt sind. Beispielsweise besitzt ein Datengitter mit fünf Punkten pro Achse im fünfdimensionalen $5^5 = 3125$ Punkte. Das gleiche Datengitter hätte im zehndimensionalen Raum $5^{10} = 5^5 \cdot 5^5 \approx 10$ Millionen Punkte. Eine weitere Ursache der schlechten Konditionierung liegt zudem darin, dass Gültigkeitsfunktionen typischerweise lokal gewählt werden, d.h. in großen Bereichen eine

Gewichtung nahe Null aufweisen. Daher ist es im Zusammenhang mit lokalen Modellnetzen üblich, die globale Schätzung durch eine explizite Regularisierungstechnik zu stabilisieren. Dazu wird ein zusätzlicher Strafterm $\Omega(\underline{w}) \geq 0$, gewichtet mit dem Regularisierungsparameter $\alpha > 0$, auf die Verlustfunktion bei der linearen Parameteroptimierung in Gl. (3.19) hinzu addiert:

$$J_{\text{reg}}(\underline{w}, \alpha) = \sum_{k=1}^{N} e^2(k) + \alpha\Omega(\underline{w})\,. \tag{3.28}$$

Eine quadratische Formulierung des Strafterms $\Omega(\underline{w})$ ist sehr vorteilhaft, weil dann das lineare quadratische Optimierungsproblem der globalen Schätzung auf sinnvolle Art derart erweitert wird, dass die lineare Beschaffenheit des Schätzproblems weiterhin bestehen bleibt. Üblicherweise verwendet man dazu folgende Struktur (siehe z.B. [116]):

$$J_{\text{reg}}(\underline{w}, \alpha) = \frac{1}{2}\left(\underline{e}^T\underline{e} + \alpha\underline{w}^T\underline{K}\,\underline{w}\right) \longrightarrow \min_{\underline{w}}\,, \tag{3.29}$$

wobei $\underline{e} = \underline{y} - \underline{\hat{y}}$ der Modellfehler und $\underline{K}$ die Regularisierungsmatrix sind.

Die Lösung des regularisierten Schätzproblems lautet dann:

$$\underline{\hat{w}} = \left(\underline{X}^T\underline{X} + \alpha\underline{K}\right)^{-1}\underline{X}^T\underline{y}\,. \tag{3.30}$$

Das Ziel der Regularisierung wird durch die Wahl der Matrix $\underline{K}$ bestimmt. Die zwei wichtigsten Methoden zur Regularisierung sind zum einen die *Ridge Regression* [64], auch unter dem Namen *Weight Decay* [17] bekannt, zum anderen die *Tikhonov-Regularisierung zweiter Ordnung* [149, 84, 83].

3.4.1 Ridge Regression

Die zuvor beschriebene globale Schätzung liefert Modelle, die ohne explizite Regularisierung oft zum Overfitting neigen. Ein Ausweg ist die Regularisierung mit Ridge Regression. Wählt man die Regularisierungsmatrix als Einheitsmatrix $\underline{K} = \underline{I}$, resultiert daraus das Ridge Regression-Verfahren. Damit ergibt sich der Strafterm $\Omega(\underline{w})$ zu:

$$\Omega(\underline{w}) = \|\underline{w}\|_2^2\,. \tag{3.31}$$

Der Grundgedanke dabei ist, betragsmäßig zu große Parameter w_{ij} und damit eine schlechte Konditionierung der Schätzung vermeiden zu wollen. Das führt zu einer Verschiebung der weniger relevanten Parameter gegen Null, um die Verlustfunktion möglichst klein zu gestalten. In [116] ist beispielsweise illustriert, wie sich die Höhenlinien eines Schätzproblems mit zwei Parametern mit zunehmendem Strafterm $\alpha\Omega(\underline{w})$ immer kreisförmiger gestalten und sich gleichzeitig das Optimum in Richtung des Ursprungs verschiebt. Für $\alpha \to 0$ ergibt sich die unregularisierte globale Least Squares-Schätzung. Steigt hingegen die Gewichtung des Strafterms, d.h. $\alpha \to \infty$, führt das auf die triviale Lösung, dass alle Parameter den Wert Null annehmen. Dies entspricht dann der Optimierung ausschließlich bezüglich des Strafterms $\|\underline{w}\|_2^2$.

Aufgrund dieser Eigenschaften fällt die Ridge Regression unter die Kategorie der sog. *Shrinkage*-Methoden und ist eng verwandt mit dem *Lasso*-Verfahren [61] bzw. der effizienten

Erweiterung der *Least Angle Regression (LARS)* [32]. Diese Form der Regularisierung wird zur Parameterselektion verwendet, indem man beobachtet, wie schnell sich die Parameter in Richtung Null bewegen, wenn der Faktor α immer größer wird.

Ein großer Vorteil der Ridge Regression ist, dass der Faktor α effizient mit Hilfe des Leave-One-Out-Fehlers (siehe Kapitel 3.5.2) optimiert werden kann. Darüber hinaus lässt sich das Optimierungsproblem geschickt in eine numerisch effiziente Form umrechnen, indem die Regressionsmatrix mit Hilfe einer Singulärwertzerlegung (*Singular Value Decomposition*, SVD) transformiert wird. Eine entsprechende Herleitung im Zusammenhang mit kernelbasierten Methoden ist in [129] gegeben. An dieser Stelle findet die Herleitung für den Spezialfall einer gewichteten globalen LS-Schätzung statt.

Die Parameterschätzung bei Ridge Regression lautet:

$$\underline{\hat{w}} = \left(\underline{X}_g^T \underline{X}_g + \alpha \underline{I}\right)^{-1} \underline{X}_g^T \underline{y} \,. \tag{3.32}$$

In MATLAB lässt sich eine SVD sehr einfach durchführen. Dann kann die Regressionsmatrix umgeschrieben werden zu: $\underline{X}_g = \underline{U}\,\underline{\Sigma}\,\underline{V}^T$. Die Matrizen $\underline{U} \in \mathbb{R}^{N\times p}$, $\underline{\Sigma} \in \mathbb{R}^{p\times p}$ und $\underline{V} \in \mathbb{R}^{p\times p}$ haben folgende wichtige Eigenschaften: $\underline{U}^T\underline{U} = \underline{V}^T\underline{V} = \underline{V}\,\underline{V}^T = \underline{I}$ (aber: $\underline{U}\,\underline{U}^T \neq \underline{I}$). Darüber hinaus ist $\underline{\Sigma}$ eine positiv-definite Diagonalmatrix [129].

Der Ausdruck $\underline{X}_g^T \underline{X}_g$ ergibt sich dann wie folgt:

$$\underline{X}_g^T \underline{X}_g = (\underline{U}\,\underline{\Sigma}\,\underline{V}^T)^T(\underline{U}\,\underline{\Sigma}\,\underline{V}^T) = \underline{V}\,\underline{\Sigma}\,\underbrace{\underline{U}^T\underline{U}}_{=\underline{I}}\,\underline{\Sigma}\,\underline{V}^T = \underline{V}\,\underline{\Sigma}^2\underline{V}^T \,. \tag{3.33}$$

Das Argument zur Inversion lässt sich folgendermaßen vereinfachen:

$$\underline{X}_g^T \underline{X}_g + \alpha\,\underline{I} = \underline{V}\,\underline{\Sigma}^2\underline{V}^T + \alpha\,\underline{I} = \underline{V}\,\underline{\Sigma}^2\underline{V}^T + \alpha\,\underline{V}\,\underline{V}^T = \underline{V}\,\left(\underline{\Sigma}^2 + \alpha\,\underline{I}\right)\underline{V}^T \,. \tag{3.34}$$

Dank dieser Vereinfachung ist die Inversion trivial, da nur noch eine Diagonalmatrix zu invertieren ist:

$$\left(\underline{X}_g^T \underline{X}_g + \alpha\,\underline{I}\right)^{-1} = \left(\underline{V}\,\left(\underline{\Sigma}^2 + \alpha\,\underline{I}\right)\underline{V}^T\right)^{-1} = \underline{V}\,(\underbrace{\underline{\Sigma}^2 + \alpha\,\underline{I}}_{diagonal})^{-1}\underline{V}^T \,. \tag{3.35}$$

Einsetzen in Gl. (3.32) ergibt:

$$\begin{aligned} \underline{\hat{w}} &= \underline{V}\,\left(\underline{\Sigma}^2 + \alpha\,\underline{I}\right)^{-1}\underline{V}^T\underline{X}_g^T\underline{y} = \underline{V}\,\left(\underline{\Sigma}^2 + \alpha\,\underline{I}\right)^{-1}\underline{V}^T\,(\underline{V}\,\underline{\Sigma}\,\underline{U}^T)\underline{y} \\ &= \underline{V}\,\left(\underline{\Sigma}^2 + \alpha\,\underline{I}\right)^{-1}\underline{\Sigma}\,\underline{U}^T\underline{y} \end{aligned} \tag{3.36}$$

und damit als Endergebnis:

$$\underline{\hat{w}} = \underline{V}\,\underline{\Sigma}\,\left(\underline{\Sigma}^2 + \alpha\,\underline{I}\right)^{-1}\underline{U}^T\underline{y} \,. \tag{3.37}$$

Die hergeleitete Gleichung erlaubt es, die Parameter $\underline{\hat{w}}$ für verschiedene α-Werte sehr effizient auszurechnen, weil jeweils nur eine Diagonalmatrix invertiert werden muss. Die Frage ist nun, wie der Parameter α zu wählen ist.

Eine gute Abschätzung der Generalisierungsleistung eines Modells liefert der Leave-One-Out-Fehler bzw. der LOOCV-Wert (siehe dazu Gl. (3.52) in Kapitel 3.5.2). Zur Berechnung

dieses Wertes benötigt man die Glättungsmatrix $\underline{S}$, wobei $\hat{\underline{y}} = \underline{S}\underline{y}$. Auch hier kann man die SVD dazu nutzen, um eine numerisch geschickte Formulierung zu finden:

$$\hat{\underline{y}} = \underline{S}\underline{y} = \underline{X}_g\,\hat{\underline{w}} = \underline{X}_g\left(\underline{X}_g^T\underline{X}_g + \alpha\underline{I}\right)^{-1}\underline{X}_g^T\underline{y}\,. \tag{3.38}$$

$$\underline{S} = (\underline{U}\,\underline{\Sigma}\,\underline{V}^T)\,\underline{V}\,\underline{\Sigma}\left(\underline{\Sigma}^2 + \alpha\,\underline{I}\right)^{-1}\underline{U}^T = \underline{U}\,\underline{\Sigma}^2\left(\underline{\Sigma}^2 + \alpha\,\underline{I}\right)^{-1}\underline{U}^T\,. \tag{3.39}$$

Dieser Ausdruck kann für $\alpha \neq 0$ noch weiter vereinfacht werden, um den Aufwand einer Matrix-Multiplikation zu sparen:

$$\begin{aligned}
\underline{S} &= \underline{U}\left(\underline{\Sigma}^2 + \alpha\,\underline{I} - \alpha\,\underline{I}\right)\left(\underline{\Sigma}^2 + \alpha\,\underline{I}\right)^{-1}\underline{U}^T\,,\\
&= \underline{U}\,\underline{U}^T - \alpha\,\underline{U}\left(\underline{\Sigma}^2 + \alpha\,\underline{I}\right)^{-1}\underline{U}^T\,,\\
&= \alpha\,\underline{U}\left(\frac{1}{\alpha}\,\underline{I}\right)\underline{U}^T - \alpha\,\underline{U}\left(\underline{\Sigma}^2 + \alpha\,\underline{I}\right)^{-1}\underline{U}^T
\end{aligned}$$

und damit:

$$\underline{S} = \begin{cases} \underline{U}\,\underline{U}^T, & \text{falls } \alpha = 0\,,\\ -\alpha\,\underline{U}\left[\left(\underline{\Sigma}^2 + \alpha\,\underline{I}\right)^{-1} - \frac{1}{\alpha}\,\underline{I}\right]\underline{U}^T, & \text{sonst.}\end{cases} \tag{3.40}$$

Man sieht, dass zur Berechnung der Glättungsmatrix ebenfalls nur eine Diagonalmatrix invertiert werden muss.

Zur Illustration der Ridge Regression findet in Bild 3.2 ein beispielhafter Vergleich zwischen lokaler und globaler Schätzung und Ridge Regression statt. Der eindimensionale Prozess y lautet:

$$\begin{aligned}
y &= y_1 y_3 + y_2(1 - y_3)\,, \qquad (3.41)\\
\text{mit}\quad y_1 &= 1 + \sin(4\pi u - \frac{\pi}{8}), \quad y_2 = \frac{0.1}{0.1 + u}, \quad y_3 = \left(1 + e^{\frac{u-0.65}{60}}\right)^{-1}\,.
\end{aligned}$$

Der verrauschte Trainingsdatensatz des Beispielprozesses aus Gl. (3.41) besteht aus 30 Datenpunkten. Beim Eingang u handelt es sich um gleichverteilte Zufallszahlen im Intervall $[0, 1]$. Der Ausgang wurde mit weißem, gaussverteiltem Rauschen und Standardabweichung $\sigma_n = 0.2$ gestört. Zum Testen des Modells dienen die unverrauschten Funktionswerte auf einem äquidistanten Gitter mit 100 Stützstellen, ebenfalls im Intervall $[0, 1]$.

Mit HILOMOT (Kapitel 4.3.2) wurde ein lokales Modellnetz mit 10 lokalen Modellen trainiert. Standardmäßig wird bei HILOMOT eine lokale Schätzung durchgeführt. Für diesen Vergleich wurden die Parameter der lokalen Modelle bei identischer Partitionierung nachträglich global geschätzt. Darüber hinaus fand die Ridge Regression Anwendung, wobei der Parameter α bezüglich des Leave-One-Out-Fehlers auf Trainingsdaten optimiert wurde mit dem Ergebnis $\alpha_{opt} = 0.0013$.

Tabelle 3.1 zeigt die Fehlerwerte auf Trainings- und unverrauschten Testdaten. Am Beispiel erkennt man das Potential der impliziten Regularisierung durch die lokale Schätzung. Im Gegensatz dazu führt die globale Schätzung ohne Regularisierung zu Overfitting. Dank der Ridge Regression kann dem Modell so viel Flexibilität entzogen werden, dass sogar eine bessere Generalisierungsleistung erzeugt wird als bei der lokalen Schätzung.

Tabelle 3.1: RMSE für verschiedenen Schätzverfahren (unverrauschte Testdaten).

Schätzung	Training	Test
lokal	0.1979	0.1481
global	0.1830	0.2178
Ridge Reg.	0.1954	0.1239

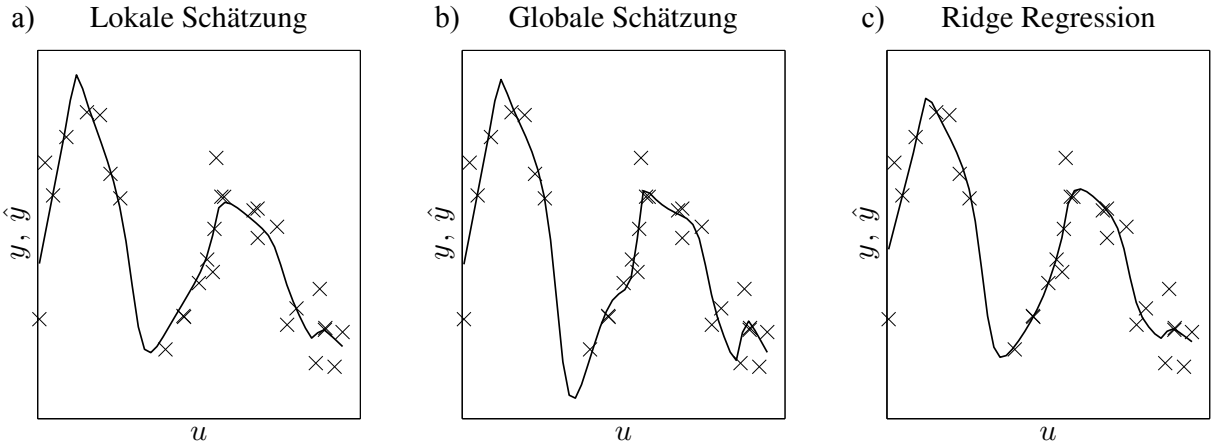

Bild 3.2: Beispiel zur Ridge Regression. Vergleich zwischen a) lokaler Schätzung, b) globaler Schätzung und c) Ridge Regression mit $\alpha_{opt} = 0.0013$. Minimiert wurde der Leave-One-Out-Fehler auf den Trainingsdaten.

3.4.2 Approximative Tikhonov-Regularisierung zweiter Ordnung

Im Rahmen dieser Dissertation wurde eine approximative Tikhonov-Regularisierung, wie in [84] vorgeschlagen, als explizites Regularisierungsverfahren speziell für die Anwendung auf hierarchische lokale Modellnetze entwickelt und implementiert.

Als motivierendes Beispiel für einen sinnvollen Einsatz der Tikhonov-Regularisierung dient das folgende einfache Experiment mit der eindimensionalen nichtlinearen Funktion

$$y = \frac{0.1}{0.1 + u}\,. \tag{3.42}$$

Die Eingangsdaten $\underline{u}$ liegen im Intervall $[0, 1]$. Das Schätzproblem hat 100 Datenpunkte auf einem äquidistanten Gitter mit einer Datenlücke zwischen $u = 0.25$ und $u = 0.75$. Der Beispielprozess ist mit normalverteiltem, weißem Rauschen überlagert, wobei die Standardabweichung $\sigma_n = 0.1$ beträgt.

Insgesamt sind die Parameter für zehn gleichmäßig im Eingangsraum verteilte lokal affine Modelle zu schätzen. Wie in Bild 3.3 gezeigt, wird die Schätzung innerhalb des Bereichs der Datenlücke problematisch, da für dort gelegene lokale Modelle nur Datenpunkte weit außerhalb des jeweiligen Gültigkeitsbereichs und damit sehr kleiner Gewichtung in Frage kommen. Auch wenn die Datenmenge von hundert Punkten bei diesem Beispiel zunächst ausreichend erscheint, kann alleine durch die unvorteilhafte Verteilung der Daten ein schlechtes Modell resultieren. Die globale Schätzung etwa zeigt deutliches Overfitting.

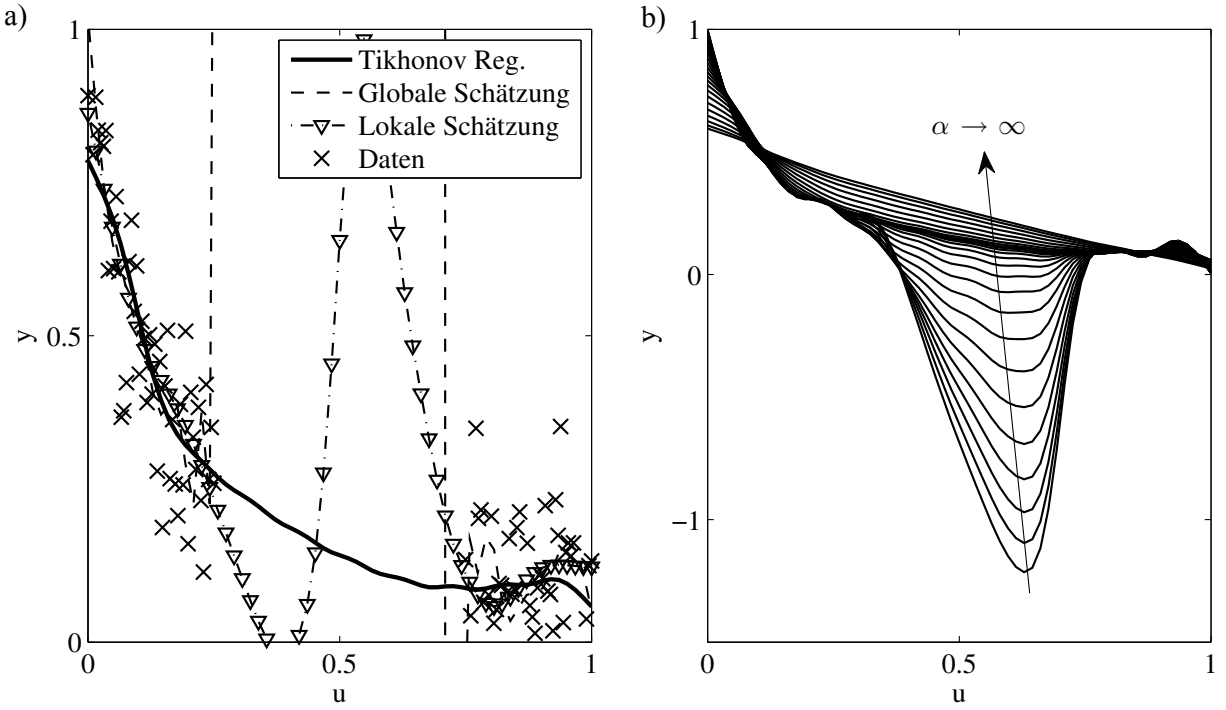

Bild 3.3: a) Gibt es Bereiche im Eingangsraum, die nicht genügend Datenpunkte haben, kann dies auf ein schlecht gestelltes Problem führen. Sowohl globale als auch lokale Schätzung zeigen in diesem Beispiel starkes Overfitting. Mit einer expliziten Tikhonov-Regularisierung wird die Schätzung stabilisiert.
b) Je nach Wahl des Parameters α fällt die Regularisierung stärker oder schwächer aus.

Bemerkenswert ist auch, dass selbst bei lokaler Schätzung das Schätzproblem offensichtlich zu schlecht gestellt ist, um auf einen sinnvollen Modellverlauf zu führen.

Abhilfe schafft die Verwendung einer Tikhonov-Regularisierung zusammen mit der globalen Schätzung. Bild 3.3 zeigt deutlich, dass dadurch eine gute Stabilisierung des Modellverlaufs erlangt werden kann. Der Grad der Modellglättung wird mit dem Faktor α festgelegt. Je höher die Bestrafung ausfällt, desto eher tendiert das Modell zu einem global linearen Verlauf.

Bei der Tikhonov-Regularisierung werden nicht große Parameter wie bei der Ridge Regression bestraft, sondern das Quadrat der zweiten Ableitung des Modellausgangs. Dieser Strafterm soll vermeiden, dass zu große Sprünge beim Gradienten des Modellausgangs auftreten.

Bei Verwendung lokal linearer Modelle liefert die Interpretierbarkeit der Parameter als Offset und Steigung den Vorteil, dass die zweite Ableitung des Modellausgangs durch ein *approximatives* Maß ersetzt werden kann. Laut [83] genügt meist die Annahme, dass man das Modell als *glatt* ansehen darf, wenn die Änderung der Parameter benachbarter lokaler Modelle klein ist. Das bedeutet, dass die approximative Tikhonov-Regularisierung die Pa-

rameterdifferenzen benachbarter lokaler Modelle bestrafen muss. Dies lässt sich mit dem angenäherten Regularisierungsstrafterm

$$\Omega(\underline{w})_{\mathrm{Tik}}^{\mathrm{approx}} = \sum_{j=1}^{\mathrm{M}} \sum_{m=1}^{j-1} (\underline{w}_j - \underline{w}_m)^{\mathrm{T}} (\underline{w}_j - \underline{w}_m)\ \omega_{jm} \tag{3.43}$$

realisieren, wobei die Parameteränderung mit einem Maß ω_{jm} für den Abstand des j-ten und m-ten Teilmodells gewichtet wird. Das in [83] und [159] vorgeschlagene Gewichtungsmaß ist auf Gauß'sche Zugehörigkeitsfunktionen zugeschnitten. Da bei hierarchischen lokalen Modellnetzen *sigmoidale* Gültigkeitsfunktionen zum Einsatz kommen, ist dieses Maß nicht auf die hierarchische Modellstruktur übertragbar. Deshalb wird dafür ein neues Gewichtungsmaß eingeführt, siehe auch [55]:

$$\omega_{jm} = \sqrt{\Phi_j(\underline{c}_m)\Phi_m(\underline{c}_j)}\,. \tag{3.44}$$

Zur Veranschaulichung des neuen Maßes sind in Bild 3.4 zwei sigmoidale Gültigkeitsfunktionen mit deren Zentren c_j und c_m gezeigt. Das Maß gibt an, wie viel Einfluss das lokale Modell j auf das lokale Modell m an dessen Zentrum hat und umgekehrt.

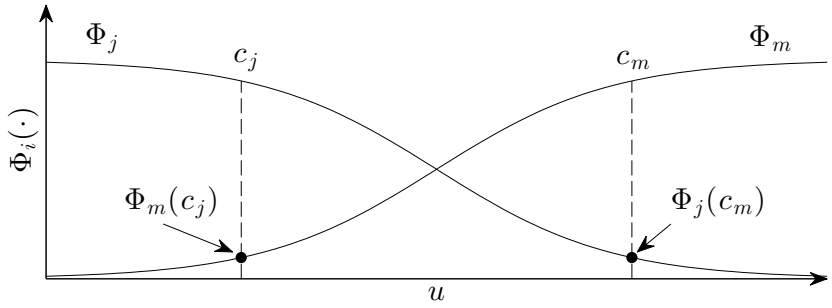

Bild 3.4: Zur Berechnung des Gewichtungsmaßes ω_{jm} werden benachbarte Gültigkeitsfunktionen an den Zentren ausgewertet.

Der Regularisierungsstrafterm in Gl. (3.43) hat eine quadratische Form und kann gemäß [159] in die symmetrische Regularisierungsmatrix

$$\underline{K}_{\mathrm{Tik}}^{\mathrm{approx}} = \begin{pmatrix} \sum_{j=1}^{M} \omega_{1j} & -\omega_{12} & \cdots & -\omega_{1M} \\ -\omega_{12} & \sum_{j=1}^{M} \omega_{2j} & \ddots & \vdots \\ \vdots & \ddots & \ddots & -\omega_{(M-1)M} \\ -\omega_{1M} & \cdots & -\omega_{(M-1)M} & \sum_{j=1}^{M} \omega_{Mj} \end{pmatrix} \tag{3.45}$$

umgerechnet werden.

Zur Berechnung der geometrischen Zentren der Gültigkeitsfunktionen kann man zwei Möglichkeiten in Betracht ziehen. Die eine Variante ist, die Gültigkeitsfunktionen an gitterförmig verteilten Stützstellen auszuwerten. Eine andere Variante sieht vor, die Zentren mit

einem Monte-Carlo-Ansatz, d.h. mit Hilfe von zufällig erzeugten Stützstellen, zu berechnen. Bild 3.5 zeigt das Ergebnis eines Vergleichs dieser zwei Methoden. Als Beispiel für den Methodenvergleich dient der Hyperbel-Prozess

$$y = \frac{1}{1 + \sum_{i=1}^{p} \frac{1-u_i}{p}}\,, \tag{3.46}$$

mit p als Dimension des Eingangsraums. Für diesen Prozess wird für unterschiedliche Anzahl an Eingangsgrößen jeweils ein lokales Modellnetz mit dem LOLIMOT-Algorithmus [116] trainiert. Abbruchkriterium ist das Erreichen von zehn lokalen Modellen. Bei LOLIMOT sind die Teilungen achsenorthogonal und die Zentren der lokalen Modelle exakt bekannt, was einen Vergleich der berechneten mit den LOLIMOT-Zentren möglich macht.

Für den Vergleich wurde zum einen ein Gitter mit jeweils ca. 10^4 Stützstellen erzeugt, um daraus den mit den Gültigkeitsfunktionen gewichteten Schwerpunkt als Zentrum zu berechnen. Zum anderen fand die Berechnung der Zentren mit einem Monte-Carlo-Ansatz statt. Das heißt, es wurden 100 Zentren mit Zufallszahlen ermittelt. Die Anzahl der zufälligen Stützstellen war identisch zur Anzahl der Stützstellen beim Gitteransatz.

In Bild 3.5 sind die Fehler der Zentrumsberechnungen nach der Gitter-Methode im Vergleich zur Monte-Carlo-Methode dargestellt. Wegen des stochastischen Monte-Carlo-Ansatzes sind dabei jeweils Mittelwert, Stichprobenwert (50. Zentrums-Fehlerwert) und größter und kleinster Fehlerwert gezeigt. Aus dieser Untersuchung geht hervor, dass die Zentrumsberechnung nach der Monte-Carlo-Methode im Hochdimensionalen verglichen zur Gitter-Methode deutlich im Vorteil ist. Daher werden bei der approximativen Tikhonov-Regularisierung für hierarchische lokale Modellnetze die Zentren nach der Monte-Carlo-Methode berechnet.

Ein großer Vorteil des quadratischen Regularisierungsstrafterms $\Omega_{\mathrm{Tik}}^{\mathrm{approx}}$ ist, dass sich damit ein lineares Schätzproblem für die Parameter der lokalen Modelle aufstellen lässt:

$$\underline{\hat{w}}_{\mathrm{Reg}} = (\underline{X}^{\mathrm{T}}\underline{X} + \alpha\underline{\widetilde{K}}_{\mathrm{Tik}}^{\mathrm{approx}})^{-1}\underline{X}^{\mathrm{T}}\underline{y}\,. \tag{3.47}$$

Die Wahl von α birgt jedoch die Schwierigkeit, dass der Optimalwert je nach Problemstellung sehr unterschiedliche Größenordnungen einnehmen kann. Dieses Problem wird abgeschwächt, indem die Regularisierungsmatrix $\underline{K}_{\mathrm{Tik}}^{\mathrm{approx}}$ in Gl. (3.47) nach der Frobeniusnorm[2] normiert wird:

$$\underline{\widetilde{K}}_{\mathrm{Tik}}^{\mathrm{approx}} = \underline{K}_{\mathrm{Tik}}^{\mathrm{approx}} \frac{\|\underline{X}^{\mathrm{T}}\underline{X}\|_F}{\|\underline{K}_{\mathrm{Tik}}^{\mathrm{approx}}\|_F}\,. \tag{3.48}$$

Durch die Normierung soll der Einfluss von Datenmenge und Überlappung der Teilmodelle so klein wie möglich gehalten werden. Das folgende Beispiel zeigt einen zusätzlichen Vorteil der Normierung: Die Robustheit bezüglich verschiedener Eingangsskalierungen. Zur Veranschaulichung dient der Testprozess

$$y = \sin(2\pi u) + 2\,u\,. \tag{3.49}$$

[2]Die Frobeniusnorm einer Matrix entspricht der euklidischen Norm im $(m \times n)$-dimensionalen Raum und ist definiert als $\|\underline{A}\|_F = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n}|a_{ij}|^2}$.

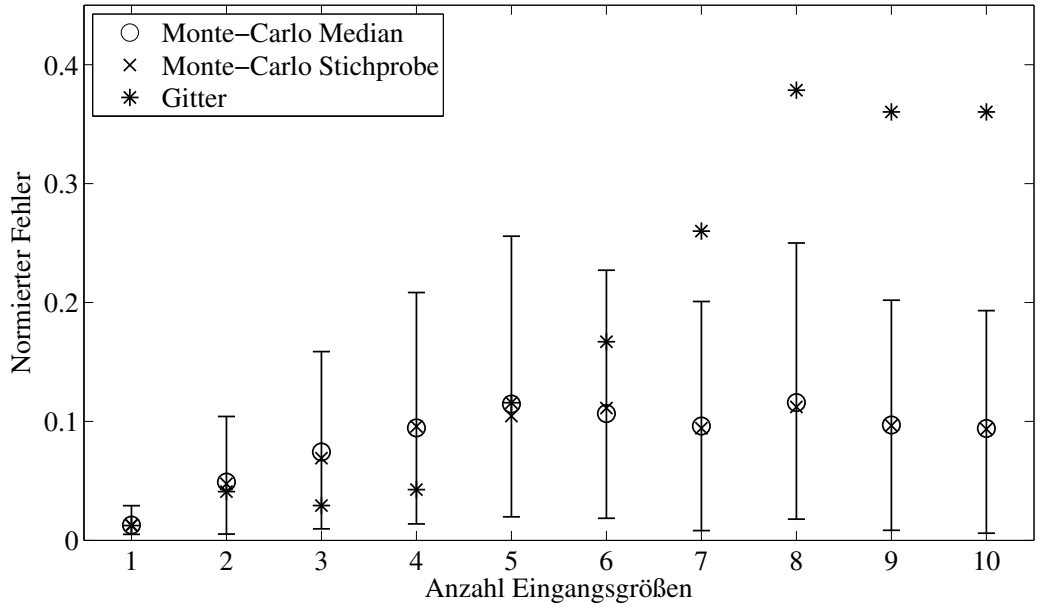

Bild 3.5: Die Monte-Carlo-Methode zur Zentrumsberechnung eignet sich insbesondere im Hochdimensionalen. Ab 7 Eingangsgrößen liefern alle Monte-Carlo-Zentren bessere Fehlerwerte (NRMSE).

Der Trainingsdatensatz des Prozesses besteht aus $N = 50$ Datenpunkten. Die Partitionierung des Modells wird künstlich vorgegeben. Dazu dienen zehn Gültigkeitsfunktionen, die gleichmäßig über den Eingangsraum verteilt sind. Das Schätzproblem wird schlecht gestellt, indem in der Mitte des Eingangsraums eine künstliche Lücke erzeugt wird. Die Mitte des Eingangsraums berechnet sich mit $u_{Mitte} = (u_{max} - u_{min})/2 + u_{min}$. Die Lücke wird jeweils so gewählt, dass 40% des Eingangsraums nicht befüllt sind. Für die lokal linearen Modelle, die in der Region der Datenlücke platziert sind, ergibt sich daher ein schlecht konditioniertes Schätzproblem. Das Experiment wurde zum einen für verschiedene Eingangsskalierungen durchgeführt. Der Eingang u liegt zwischen 0 und 1 und wird mit vier verschiedenen Faktoren $1, 10, 100$ und 1000 unterschiedlich skaliert. Die Eingangswerte liegen, abgesehen von der Datenlücke, auf einem äquidistanten Gitter. Zum anderen wurde der Prozess einmal ohne und einmal mit weißem, normalverteiltem Rauschen trainiert ($\sigma_n = 0.05$).

Bild 3.6 zeigt die Fehlerverläufe auf Testdaten für den unverrauschten Prozess. Ohne die Normierung der Regularisierungsmatrix ergeben sich Verläufe, die sehr unterschiedliche Optima im Wertebereich von 10^{-5} bis 10^5 aufweisen. Im rechten Teilbild erkennt man, dass die optimalen α-Werte unabhängig von der Eingangsskalierung durch die Normierung alle in der gleichen Region liegen, egal, wie die Daten skaliert wurden. Bild 3.7 zeigt die Fehlerverläufe zum verrauschten Prozess. Qualitativ ergibt sich die gleiche Schlussfolgerung: Die Optimierung des α-Wertes vereinfacht sich durch die Normierung in einen einheitlichen Wertebereich erheblich.

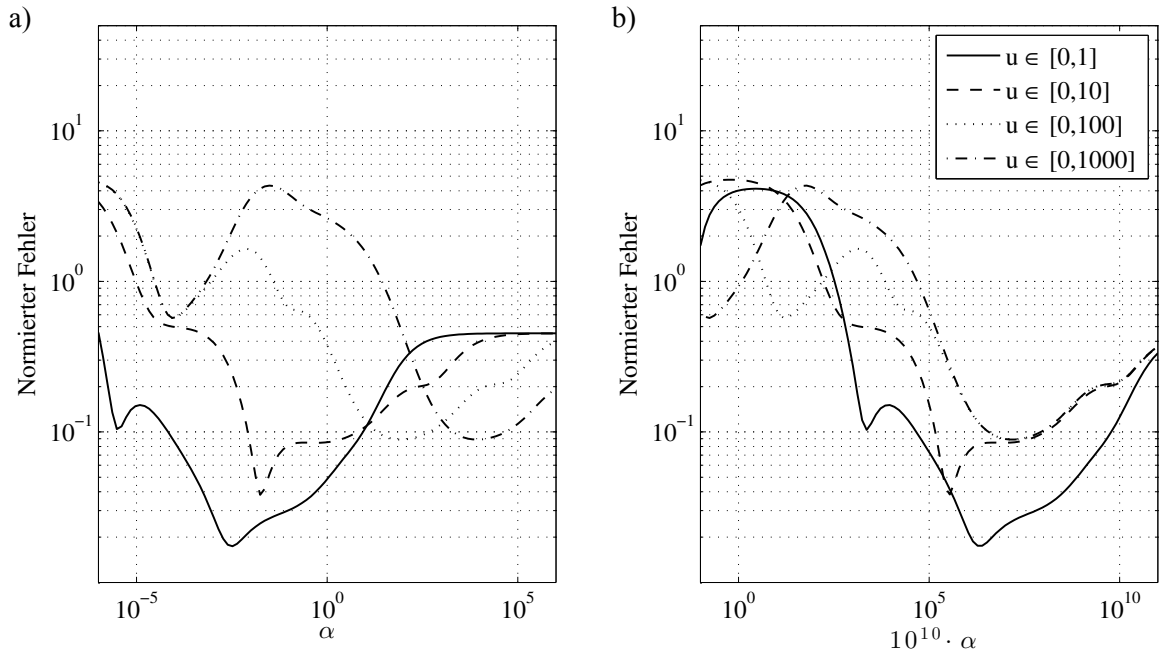

Bild 3.6: Die Normierung des Regularisierungsoperators reduziert den Aufwand zum Einstellen des optimalen α-Werts signifikant. a) Ohne Normierung variieren die optimalen α-Werte stark. b) Mit Normierung wird dieses Problem behoben und die optimalen α-Werte liegen in ähnlichen Wertebereichen.

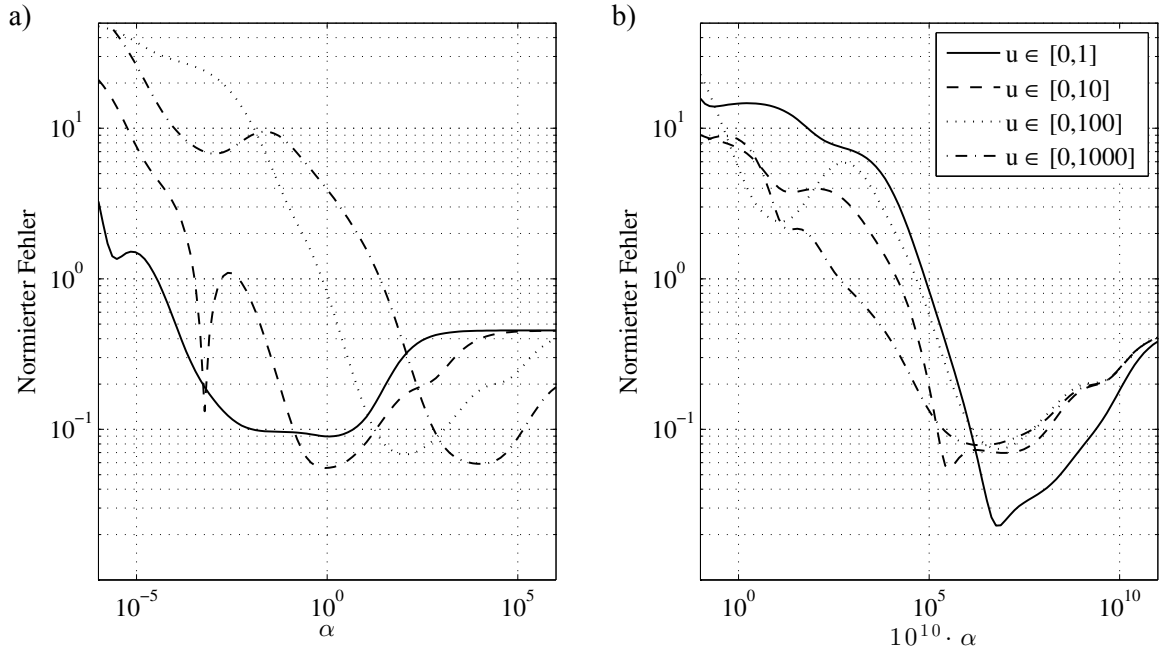

Bild 3.7: Betrachtet wird der gleiche Prozess wie in Bild 3.6, jedoch mit zusätzlichem Rauschen ($\sigma_n = 0.05$). Qualitativ zeigen sich die gleichen Ergebnisse wie im rauschfreien Fall.

3.4.3 Global-lokale Regularisierung

Die globale Schätzung kann durch die geschickte Wahl des Regularisierungsoperators $\underline{K}$ in eine lokale Schätzung überführt werden. Dazu ist in [159] ein Regularisierungsoperator speziell für lokal lineare Modelle hergeleitet worden. Die Gleichung hierfür lautet:

$$\underline{K}_{\text{global/lokal}} = \begin{pmatrix} \underline{X}^T(\underline{Q}_1 - \underline{Q}_1^2)\underline{X} & -\underline{X}^T\underline{Q}_1\underline{Q}_2\underline{X} & \cdots & -\underline{X}^T\underline{Q}_1\underline{Q}_M\underline{X} \\ -\underline{X}^T\underline{Q}_1\underline{Q}_2\underline{X} & \underline{X}^T(\underline{Q}_2 - \underline{Q}_2^2)\underline{X} & \ddots & \vdots \\ \vdots & \ddots & \ddots & -\underline{X}^T\underline{Q}_{M-1}\underline{Q}_M\underline{X} \\ -\underline{X}^T\underline{Q}_1\underline{Q}_M\underline{X} & \cdots & -\underline{X}^T\underline{Q}_{M-1}\underline{Q}_M\underline{X} & \underline{X}^T(\underline{Q}_M - \underline{Q}_M^2)\underline{X} \end{pmatrix}.$$

Insbesondere bei der Modellbildung dynamischer Prozesse ist diese Schätzmethode sehr vorteilhaft, da die lokale Schätzung in einem einzigen Schritt erfolgen kann. Dadurch ist es möglich, die Parameter der lokalen Modelle zwar in der selben Optimierung anzupassen, jedoch trotzdem die guten Eigenschaften der lokalen Schätzung beizubehalten. Insbesondere wenn es sich um ein dynamisches Modell handelt und in paralleler Anordnung simuliert werden soll (NOE-Konfiguration), ist dies äußerst günstig. In der parallelen Anordnung enthält die Regressionsmatrix $\underline{X}$ nicht die gemessenen Ausgangsgrößen $\underline{y}$, sondern die simulierten Modellausgänge $\underline{\hat{y}}$. Dies führt auf ein implizites Problem und kann daher nicht mehr mit linearen Verfahren geschätzt werden. Üblicherweise findet daher die Optimierung mit nichtlinearen Least Squares-Verfahren wie dem Levenberg-Marquardt-Algorithmus statt.

3.5 Optimierung der Modellkomplexität

Eine interessante Herausforderung bei der Entwicklung automatischer Trainingsalgorithmen ist die Frage, wie flexibel ein Modell sich an einen gegebenen Trainingsdatensatz anpassen darf. Grundlage hierfür ist der sogenannte Bias-Varianz-Kompromiss. Darauf basierend lässt sich die richtige Modellkomplexität eines Modells mit Hilfe statistischer Überlegungen bestimmen.

3.5.1 Bias-Varianz-Dilemma

Das *Bias-Varianz-Dilemma* besagt, dass sich der Fehler eines Modells einerseits aus einer *Bias*komponente, andererseits aus einer *Varianz*komponente zusammensetzt, siehe Bild 3.9. Bei allen datenbasierten Modellbildungsverfahren ist ein Zielkonflikt bezüglich dieser Komponenten zu lösen [61]. Der *Bias* beschreibt den *systematischen* und die *Varianz* den *stochastischen* Anteil des Modellfehlers [116, 159]:

$$\underbrace{\mathrm{E}\left\{(y_u - \hat{y})^2\right\}}_{\text{Modellfehler}^2} = \underbrace{\left(y_u - \mathrm{E}\{\hat{y}\}\right)^2}_{\text{Bias}^2} + \underbrace{\mathrm{E}\left\{\left(\hat{y} - \mathrm{E}\{\hat{y}\}\right)^2\right\}}_{\text{Varianz}}, \tag{3.50}$$

wobei y_u der wahre, ungestörte Prozessausgang, $\hat{y}$ der Modellausgang und $\mathrm{E}\{\cdot\}$ der Erwartungswert ist.

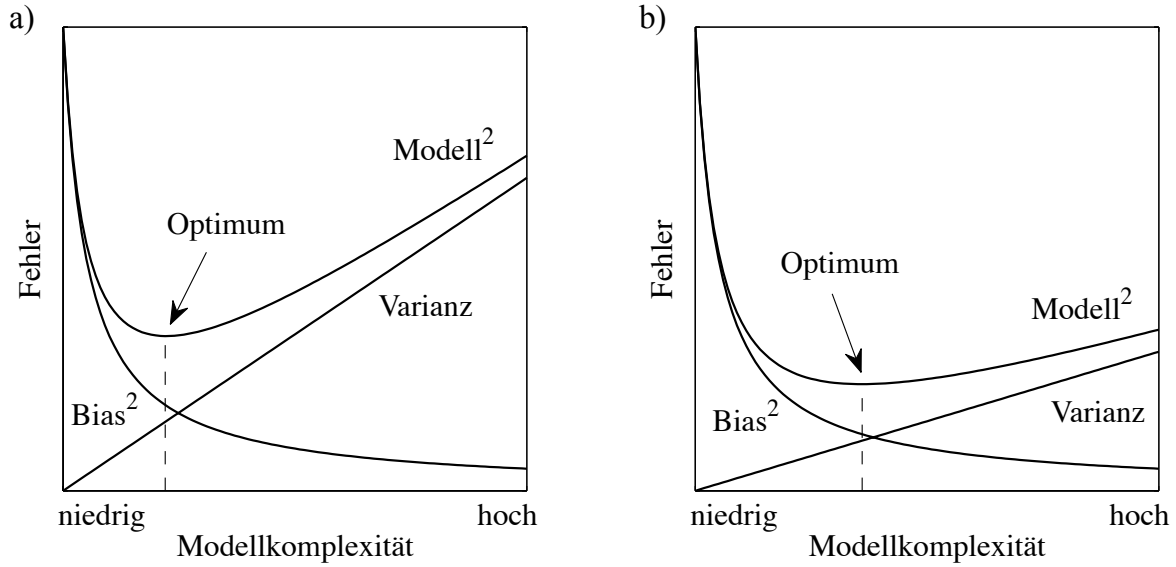

Bild 3.8: Modellfehler^2 = Bias^2 + Varianz. Das Optimum der Modellkomplexität ergibt sich aus dem Kompromiss zwischen Bias und Varianz. Bei starkem Rauschen (a) muss das Modell weniger flexibel gewählt werden als bei geringem Rauschen (b).

Es besteht die Gefahr, für die Beschreibung eines Prozesses durch ein Modell zu viele Freiheitsgrade zu verwenden, das heißt die Komplexität des Modells ist zu groß. Dann hat das Modell eine erhöhte Varianz und es entsteht *Overfitting*. Das Modell wird zu flexibel und beschreibt neben dem Prozess das Rauschen der Daten. Wenn andererseits zu wenige Freiheitsgrade verwendet werden, reduziert sich die Varianz, aber der Biasfehler erhöht sich. Daraus ergibt sich aufgrund von *Underfitting* ebenfalls eine schlechte Abbildung des Prozesses. Die Komplexität des Modells reicht nicht aus, um den Prozess zu beschreiben. Erstrebenswert ist ein Kompromiss zwischen diesen beiden Extremen, auch Bias-Varianz-Kompromiss genannt.

Wie in Bild 3.8 gezeigt, ergibt sich je nach Stärke des Rauschens ein anderes Optimum für die Modellkomplexität, weil der Varianzfehler und damit auch der Modellfehler proportional zur Varianz des Prozessrauschens ansteigt [116], d.h.:

$$\text{Varianzfehler} \;\sim\; \sigma_n^2 \frac{n}{N} \;\widehat{=}\; \text{Rauschvarianz} \cdot \frac{\text{Anzahl Modellparameter}}{\text{Anzahl Datenpunkte}} \,.$$

Eine kleinere Rauschvarianz σ_n^2 erlaubt folglich bei gleicher Datenmenge N eine höhere Parameteranzahl n des Modells. Bild 3.9 zeigt exemplarisch anhand eines gegebenen Datensatzes, wie unterschiedlich sich die Wahl der Komplexität auf ein Modell niederschlägt.

3.5.2 Datengetriebene Validierung

Die gängigste Methode zur Abschätzung der richtigen Modellkomplexität ist die Aufteilung der Daten in einen Trainingsdatensatz einerseits und in einen Validierungsdatensatz andererseits. Will man die Generalisierungsleistung des Modells auf neuen Daten bewerten, muss darüber hinaus ein weiterer Testdatensatz vorgehalten werden. Eine Standardaufteilung für Training : Validierung : Test ist zu Anteilen 70% : 15% : 15% üblich. Bei

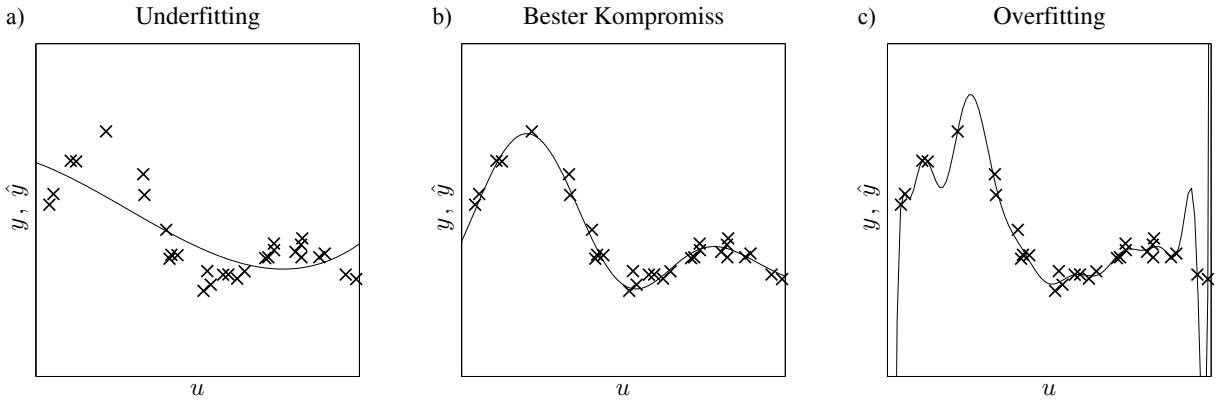

Bild 3.9: Für eine gegebene Datenmenge ist die optimale Modellkomplexität festzulegen. Einem Modell mit zu wenigen Parametern (a) fehlt die nötige Flexibilität, die Daten hinreichend genau zu beschreiben (Underfitting). Zu viele Parameter (c) können dazu führen, dass ein Modell fälschlicherweise das Rauschen auf den Messdaten mit abbildet (Overfitting). Zwischen diesen Extremen ist der beste Kompromiss zu finden (b).

vielen Anwendungen ist dies aber problematisch, weil nur wenige Daten vorhanden sind bzw. die Erhebung der Daten teuer ist. Einen guten Ausweg bietet die Abschätzung der Komplexität mittels *Kreuzvalidierung*.

Kreuzvalidierung

Bei der Kreuzvalidierung (englisch: *cross-validation*, Abk.: CV) wird die Vorhersagequalität eines Modells bewertet, indem *alle* Daten nacheinander einmal zum Training und einmal zum Validieren des Modells verwendet werden. Besonders bei sehr wenigen verfügbaren Messungen kommt die Methode zum Einsatz, da mit ihr eine bessere Ausnutzung der vorhandenen Daten ermöglicht wird. Allerdings hat dies einen erhöhten Berechnungsaufwand zur Folge.

Die vorhandenen Daten unterteilt man in k möglichst gleich große Gruppen. Typischerweise findet die Unterteilung zufällig statt, aber auch andere Strategien sind möglich, um z.B. eine gleichförmige Verteilung der Validierungsdatensätze zu erreichen. Anschließend wird in jedem Iterationsschritt das Training eines Modells mit $k-1$ Datengruppen durchgeführt und die Validierung erfolgt mit der verbleibenden Datengruppe. In jedem der k Iterationsschritte dient also eine andere Datengruppe zur Validierung des Modells, wie schematisch in Bild 3.10 dargestellt.
Daher spricht man im Allgemeinen auch von einer k-fach Kreuzvalidierung, um anzudeuten, dass die Anzahl der Datengruppen frei gewählt werden kann. Nach [61] lässt sich die Kreuzvalidierung mathematisch folgendermaßen beschreiben: Man führt eine Index-Funktion $\kappa_j : \{1, \ldots, N\} \mapsto \{1, \ldots, N_k\}_j$ ein, die jeden Datenpunkt i zur j-ten Gruppe

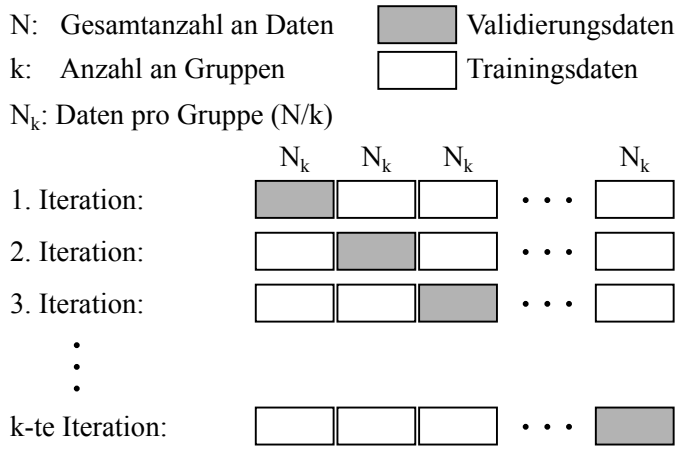

Bild 3.10: Schematischer Ablauf der k-fach Kreuzvalidierung.

zuordnet. Üblicherweise findet diese Zuordnung zufällig für alle Datenpunkte statt. Der Kreuzvalidierungsfehler kann dann mit

$$\mathrm{CV}(\hat{y}) = \frac{1}{N} \sum_{j=1}^{k} \sum_{i \in \kappa_j} \left(y(i) - \hat{y}^{(-j)}(\underline{u}(i)) \right)^2 \tag{3.51}$$

ausgerechnet werden. Dabei bedeutet $\hat{y}^{(-j)}(\underline{u})$, dass beim Modelltraining die j-te Datengruppe weggelassen wurde. Ein gängiger Wert für die Anzahl an Datengruppen ist $k = 10$. Generell gilt, dass der Berechnungsaufwand linear mit der Anzahl der Gruppen ansteigt. Für k Gruppen müssen folgerichtig k Modelle trainiert werden. Ausführliche Informationen zur Wahl der Gruppengröße sind z.B. bei [61] zu finden. Ein beliebter Spezialfall ist die Wahl von $k = N$ Gruppen, auch wenn dies zunächst als aufwändigste Methode und damit paradox erscheint. Dazu wird jeweils nur ein einziger Punkt vom Training ausgeschlossen. Deshalb hat diese Methode im Englischen auch die Bezeichnung *leave-one-out cross-validation* oder kurz *LOOCV*[3]. Formal ergibt sich die LOOCV, wenn in Gl. (3.51) der Index $\kappa(i) = i$ gesetzt wird. Dann ergibt sich der LOOCV-Wert zu:

$$\mathrm{LOOCV}(\hat{y}) = \frac{1}{N} \sum_{i=1}^{N} \left(y(i) - \hat{y}^{(-i)}(\underline{u}(i)) \right)^2 , \tag{3.52}$$

siehe z.B. [61]. Grund für die Beliebtheit des LOOCV-Wertes ist, dass sich diese Art der Kreuzvalidierung im Zusammenhang *linearer* Schätzverfahren sehr effizient berechnen lässt. Hat man einmal die Inverse der Hesse-Matrix $(\underline{X}^T \underline{X})^{-1}$ zur Schätzung berechnet, ist die Berechnung des LOOCV-Wertes ohne großen Aufwand möglich.

Für lineare Modellierungsverfahren lässt sich der Modellausgang mit Hilfe der Glättungsmatrix $\underline{S}$ durch $\underline{\hat{y}} = \underline{S}\underline{y}$ ausdrücken. Eine Herleitung des LOOCV-Werts im Zusammenhang regularisierter Least Squares-Methoden ist z.B. in [129] gegeben. An dieser Stelle wird die Herleitung speziell für lokale Modellnetze durchgeführt, bei denen die Parameter der lokalen Modelle lokal mit der gewichteten LS Methode (WLS) geschätzt werden. Zur kompakteren Schreibweise wird folgende Konvention verwendet: $S_{ij,k}$ bedeutet: Eintrag der Matrix $\underline{S}_k$

[3] Wird u.a. auch als *PRESS* für *Prediction Sum of Squares* bezeichnet.

des k-ten lokalen Modells in Zeile i und Spalte j. $\underline{X}_{i,k}$ bedeutet: i-ter Zeilenvektor der Matrix $\underline{X}_k$ des k-ten lokalen Modells.

Der Modellausgang für den i-ten Datenpunkt (hier: $\underline{\hat{y}}(i) = \underline{\hat{y}}_i$) lautet:

$$\begin{aligned} \hat{y}_i &= \sum_{k=1}^{M} \Phi_{ik}\, \underline{X}_{i,k} \underline{w}_k = \sum_{k=1}^{M} \Phi_{ik}\, \underline{X}_{i,k} \left(\underline{X}_k^T \underline{Q}_k \underline{X}_k \right)^{-1} \underline{X}_k^T \underline{Q}_k \underline{y}\,, \\ &= \sum_{k=1}^{M} \underline{S}_{i,k}\, \underline{y} = \sum_{k=1}^{M} \sum_{j=1}^{N} S_{ij,k}\, y_j \end{aligned} \tag{3.53}$$

Der Modellausgang, trainiert ohne den i-ten Messwert, ausgewertet für den gleichen Eingangswert $u(i) = u_i$ liefert formal:

$$\hat{y}_i^{(-i)} = \sum_{k=1}^{M} \underline{S}_{i,k}\, \underline{y}^{(-i)} = \sum_{k=1}^{M} \sum_{j=1}^{N} S_{ij,k}\, y_j^{(-i)}\,. \tag{3.54}$$

Diese Formulierung ist möglich, indem man den i-ten Messwert in $\underline{y}$ durch den entsprechenden Modellausgangswert $\hat{y}_i^{(-i)}$ ersetzt:

$$y_j^{(-i)} = \begin{cases} y_j, & \text{für } i \neq j\,, \\ \hat{y}_i^{(-i)}, & \text{für } i = j\,. \end{cases} \tag{3.55}$$

Der Unterschied zwischen den Modellen mit und ohne dem i-ten Messwert lautet:

$$\hat{y}_i^{(-i)} - \hat{y}_i = \sum_{k=1}^{M} \sum_{j=1}^{N} S_{ij,k} (y_j^{(-i)} - y_j)\,. \tag{3.56}$$

Durch die in Gl. (3.55) getroffene Voraussetzung ergeben alle Differenzen für $i \neq j$ Null. Dann folgt:

$$\hat{y}_i^{(-i)} - \hat{y}_i = \sum_{k=1}^{M} S_{ii,k} (\hat{y}_i^{(-i)} - y_i) = \sum_{k=1}^{M} S_{ii,k}\, \hat{y}_i^{(-i)} - \sum_{k=1}^{M} S_{ii,k}\, y_i\,. \tag{3.57}$$

Diese Gleichung lässt sich nach $\hat{y}_i^{(-i)}$ wie folgt auflösen:

$$\hat{y}_i^{(-i)} = \frac{\hat{y}_i - y_i \sum_{k=1}^{M} S_{ii,k}}{1 - \sum_{k=1}^{M} S_{ii,k}} = \frac{\sum_{k=1}^{M} \underline{S}_{i,k}\, \underline{y} - y_i \sum_{k=1}^{M} S_{ii,k}}{1 - \sum_{k=1}^{M} S_{ii,k}}\,. \tag{3.58}$$

Der Leave-One-Out-Modellvektor für alle N Messwerte lautet schließlich:

$$\underline{\hat{y}}^{(-i)} = \left(\sum_{k=1}^{M} \underline{S}_k\, \underline{y} - \sum_{k=1}^{M} \mathrm{diag}(\underline{S}_k)\, \underline{y} \right) \left(\mathrm{diag} \left(\underline{I} - \sum_{k=1}^{M} \underline{S}_k \right) \right)^{-1}\,. \tag{3.59}$$

Daraus folgt der Leave-One-Out-Fehlervektor:

$$\begin{aligned}
\underline{e}^{(-i)} = \underline{y} - \underline{\hat{y}}^{(-i)} &= y - \left(\sum_{k=1}^{M} \underline{S}_k\,\underline{y} - \sum_{k=1}^{M} \operatorname{diag}(\underline{S}_k)\,\underline{y}\right)\left(\operatorname{diag}\left(\underline{I} - \sum_{k=1}^{M} S_k\right)\right)^{-1} \\
&= \left(\operatorname{diag}\left(\underline{I} - \sum_{k=1}^{M} \underline{S}_k\right)\underline{y} - \sum_{k=1}^{M} \underline{S}_k\,\underline{y} + \sum_{k=1}^{M} \operatorname{diag}(\underline{S}_k)\,\underline{y}\right)\ldots \\
&\qquad \left(\operatorname{diag}\left(\underline{I} - \sum_{k=1}^{M} \underline{S}_k\right)\right)^{-1} \\
&= \left(\underline{y} - \sum_{k=1}^{M} \underline{S}_k\,\underline{y}\right)\left(\operatorname{diag}\left(\underline{I} - \sum_{k=1}^{M} \underline{S}_k\right)\right)^{-1} \\
&= \left(\underline{y} - \underline{\hat{y}}\right)\left(\operatorname{diag}\left(\underline{I} - \sum_{k=1}^{M} \underline{S}_k\right)\right)^{-1} \qquad (3.60)
\end{aligned}$$

und damit für den einzelnen Messwert j:

$$e_j^{(-i)} = \hat{y}_j - \hat{y}_j^{(-i)} = \frac{y_j - \hat{y}_j}{1 - \sum_{k=1}^{M} S_{jj,k}}\,. \qquad (3.61)$$

Als Endergebnis lautet dann die äquivalente Beschreibung des LOOCV-Werts in Gl. (3.52):

$$\mathrm{LOOCV}(\hat{y}) = \sum_{j=1}^{N} \frac{y_j - \hat{y}_j}{1 - \sum_{k=1}^{M} S_{jj,k}}\,. \qquad (3.62)$$

Man erkennt, dass lediglich die Diagonalelemente S_{jj} der Glättungsmatrix $\underline{S}$ benötigt werden. Obwohl für die LOOCV eigentlich N Modelle trainiert werden müssten, reicht in diesem Fall der Aufwand einer einzigen LS-Schätzung aus.

Die Methode kann sehr effizient unter Verwendung einer reduzierten QR-Zerlegung implementiert werden. Zunächst überführt man die gewichtete in eine gewöhnliche LS-Schätzung, indem man die Regressionsmatrix des k-ten lokalen Modells $\underline{X}_k$ und den Ausgangsvektor $\underline{y}$ transformiert [116]. Die diagonale Gewichtungsmatrix wird dazu umgeschrieben in $\underline{Q}_k = \operatorname{diag}(\underline{\Phi}_k) = \sqrt{\underline{Q}_k}\sqrt{\underline{Q}_k}$. Die Transformation lautet dann $\underline{\widetilde{X}}_k = \sqrt{\underline{Q}_k}\underline{X}_k$ und $\underline{\tilde{y}} = \sqrt{\underline{Q}_k}\underline{y}$. Der Modellausgang berechnet sich mit den transformierten Größen folgendermaßen:

$$\begin{aligned}
\underline{\hat{y}} &= \sum_{k=1}^{M} \underline{Q}_k\,\underline{\hat{y}}_k = \sum_{k=1}^{M} \sqrt{\underline{Q}_k}\sqrt{\underline{Q}_k}\,\underline{\hat{y}}_k = \sum_{k=1}^{M} \sqrt{\underline{Q}_k}\,\underline{\hat{\tilde{y}}}_k = \sum_{k=1}^{M} \sqrt{\underline{Q}_k}\,\underline{\widetilde{X}}_k\,\underline{\hat{w}}_k \\
&= \sum_{k=1}^{M} \sqrt{\underline{Q}_k}\,\underline{\widetilde{X}}_k \left(\underline{\widetilde{X}}_k^T \underline{\widetilde{X}}_k\right)^{-1} \underline{\widetilde{X}}_k^T \underline{\tilde{y}} \\
&= \sum_{k=1}^{M} \underbrace{\sqrt{\underline{Q}_k}\,\underline{\widetilde{X}}_k \left(\underline{\widetilde{X}}_k^T \underline{\widetilde{X}}_k\right)^{-1} \underline{\widetilde{X}}_k^T \sqrt{\underline{Q}_k}}_{\underline{S}_k}\,\underline{y}\,. \qquad (3.63)
\end{aligned}$$

Die reduzierte QR-Zerlegung liefert $\underline{\widetilde{X}}_k = \underline{Q}_{\mathrm{QR}}\underline{R}_{\mathrm{QR}}$. Die Berechnung der Glättungsmatrix $\underline{S}_k$ in Gl. (3.63) mit den Matrizen aus der QR-Zerlegung ergibt:

$$\begin{aligned}
\underline{S}_k &= \sqrt{\underline{Q}_k}\,\underline{Q}_{\mathrm{QR}}\underline{R}_{\mathrm{QR}}\left(\underline{R}_{\mathrm{QR}}^T\underbrace{\underline{Q}_{\mathrm{QR}}^T\underline{Q}_{\mathrm{QR}}}_{=\underline{I}}\underline{R}_{\mathrm{QR}}\right)^{-1}\underline{R}_{\mathrm{QR}}^T\underline{Q}_{\mathrm{QR}}^T\sqrt{\underline{Q}_k} \\
&= \sqrt{\underline{Q}_k}\,\underline{Q}_{\mathrm{QR}}\underbrace{\underline{R}_{\mathrm{QR}}\left(\underline{R}_{\mathrm{QR}}^T\,\underline{R}_{\mathrm{QR}}\right)^{-1}\underline{R}_{\mathrm{QR}}^T}_{=\underline{I}}\underline{Q}_{\mathrm{QR}}^T\sqrt{\underline{Q}_k} \\
&= \sqrt{\underline{Q}_k}\,\underline{Q}_{\mathrm{QR}}\underline{Q}_{\mathrm{QR}}^T\sqrt{\underline{Q}_k} \\
&= \sqrt{\underline{\Phi}_k\,\underline{\Phi}_k^T}\circ\underline{Q}_{\mathrm{QR}}\underline{Q}_{\mathrm{QR}}^T\,. \qquad (3.64)
\end{aligned}$$

wobei es sich beim letzten Ausdruck um das Hadamard-Produkt[4] handelt. Zur Bestimmung des LOOCV-Werts benötigt man lediglich die Diagonalelemente. Diese werden im Englischen auch als *leverage* bezeichnet. Damit vereinfacht sich die Berechnung zu:

$$\begin{aligned}
\underline{l}_k &= \mathrm{diag}(\underline{S}_k) \\
l_{i,k} &= \Phi_{i,k}\sum_{j=1}^{nx} Q_{\mathrm{QR}_{ij}}^2\,. \qquad (3.65)
\end{aligned}$$

Dabei ist nx die Anzahl der Spalten von $\underline{\widetilde{X}}_k$ bzw. $\underline{Q}_{\mathrm{QR}}$. Mit der gezeigten Herleitung kann der Leave-One-Out-Fehler mit folgender Gleichung rechenzeitsparend ausgerechnet werden:

$$\mathrm{LOOCV}(\hat{y}) = \sum_{j=1}^{N}\frac{y_j-\hat{y}_j}{1-\sum_{k=1}^{M} l_{j,k}}\,. \qquad (3.66)$$

Die vorangegangen Rechnungen setzen voraus, dass es sich um linear geschätzte Parameter handelt. Die hier behandelten lokalen Modellnetze gehören allerdings nicht zur Klasse der linear parametrierten Modelle. Die Gültigkeitsfunktionen $\underline{\Phi}_i$ sind durch die heuristischen Verfahren LOLIMOT oder HILOMOT erzeugt worden und beinhalten Parameter, die nur mit nichtlinearen Verfahren geschätzt werden können. Allerdings ist der Modellausgang im Wesentlichen durch die linear geschätzten Parameter der lokalen Modelle $\hat{y}_i$ bestimmt, sofern die durch den Teilungsbaum erzeugten Gültigkeitsfunktionen nicht mehr verändert werden. Trifft man die Annahme, dass der Generalisierungsfehler des Modells im Wesentlichen durch die linear geschätzten Parameter der lokalen Modelle erfasst wird, kann die effiziente Berechnung des LOOCV in Gl. (3.66) auch bei lokalen Modellnetzen genutzt werden. Der Einfluss der Partitionierung bezüglich des Overfittings wird dabei vernachlässigt.

Für andere Modellklassen wie beispielsweise Gaussprozessmodelle wird der LOOCV-Fehler analog berechnet, siehe z.B. [85]. Die optimal eingestellten Metaparameter (Standardabweichung in jeder Raumrichtung und die Potenz im Abstandsmaß in der Exponentialfunktion in jeder Raumrichtung) werden einmalig auf allen Daten optimiert und dann für die

[4]Beim Hadamard-Produkt werden zwei Matrizen elementweise multipliziert: $(A\circ B)_{ij} = A_{ij}\cdot B_{ij}$.

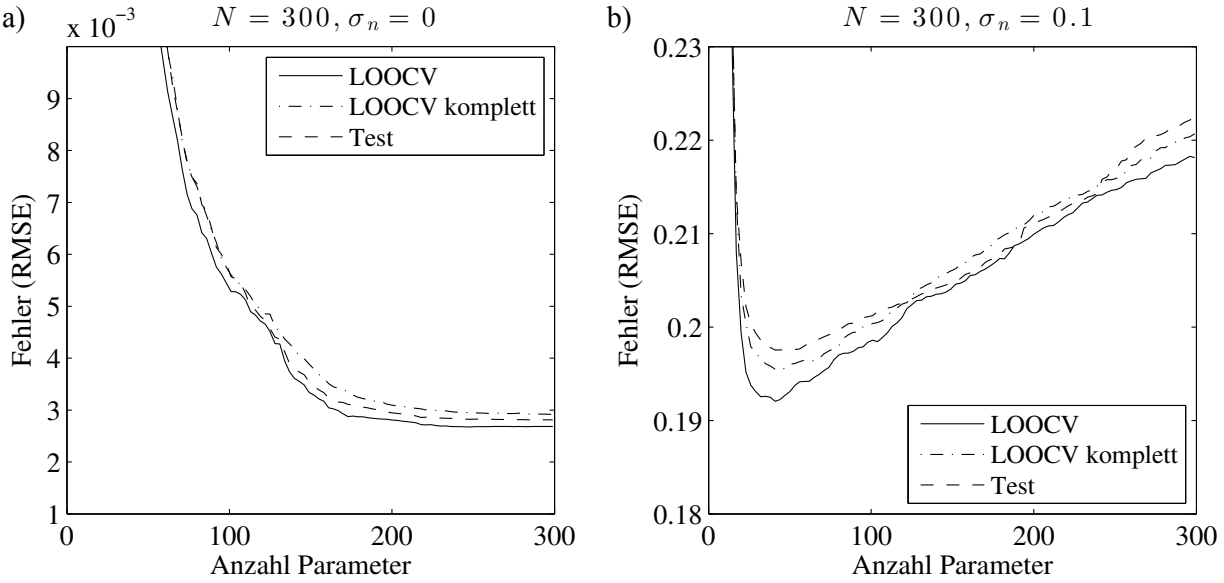

Bild 3.11: Testprozess (Gl. (3.41)) mit $N = 300$ Punkten. a) Rauschfreier Fall. b) Modellierung des rauschbehafteten Prozesses. Man erkennt die gute Übereinstimmung von Testdatenfehler, der linearen LOOCV und der komplett durchgeführten LOOCV, welche den Einfluss der Partitionierung mit berücksichtigt.

LOOCV festgehalten. Im Unterschied zum hier vorgestellten Vorgehen bei lokalen Modellnetzen ist es bei Gaussprozessmodellen allerdings noch viel wichtiger, die nichtlinearen Parameter zu vernachlässigen, weil sonst der Rechenaufwand u.U. zu hoch für die praktische Anwendung wäre.

In Experimenten sich zeigt sich, dass die Approximation des LOOCV-Fehlers bei lokalen Modellnetzen sehr gut mit der komplett durchgeführten LOOCV übereinstimmt. In Bild 3.11 ist die Ähnlichkeit zwischen Testdatenfehler, LOOCV und einer komplett durchgeführten LOOCV, d.h. inklusive Partitionierungseinfluss, exemplarisch anhand des Beispiels aus Gl. (3.41) gezeigt. Die Ähnlichkeit ist bemerkenswert, da dadurch der Rechenaufwand beträchtlich reduziert wird und der LOOCV optional als Zielgröße für die nichtlineare Schnittoptimierung mit HILOMOT verwendet werden kann, siehe Kapitel 4.3.2.

3.5.3 Statistische Komplexitätsabschätzung

Die Kreuzvalidierung hat den Nachteil, dass sie unter Umständen sehr rechenzeitintensiv sein kann. Benutzt man z.B. eine Aufteilung des Datensatzes in 100 Gruppen, müssen dementsprechend auch 100 Modelle trainiert werden. Dies kann in der Anwendung oft unpraktikabel sein. Ein Ausweg liefert die rein statistische Auswertung des Datensatzes. Sogenannte *Informationskriterien* bieten die Möglichkeit, auf der Basis statistischer Annahmen über den Prozess, die Gefahr des Overfittings abzuschätzen.

Ein Blick auf das Thema Modellselektion offenbart schnell, dass die zugehörige Theorie sehr ausführlich in der Literatur Beachtung findet. An dieser Stelle wird allerdings lediglich eine der wichtigsten Erkenntnisse angerissen: das AIC-Kriterium. Akaike [3] hat das *Akaike Informationskriterium* (kurz: AIC) entwickelt und definiert damit eine Abschätzung der

erwarteten relativen Distanz zwischen dem Modell und dem wahren Prozess, mit dem die Messdaten erzeugt wurden. Es basiert auf der Abschätzung der sog. Kullback-Leibler-Information [22]. Die Frage dabei ist: Wie viel Information geht durch die Modellbildung im Vergleich zum wahren Prozess verloren?

Kullback-Leibler-Information

Die Kullback-Leibler-Information (KL-Information) basiert auf der Tatsache, dass mit einer Approximation immer auch ein Informationsverlust einhergeht [21]. Interessant ist, dass die Grundlagen zu dieser Hypothese u.a. bis auf Boltzmann's Konzept zur *Entropie* (1877) und dem damit verbundenen zweiten Hauptsatz der Thermodynamik zurückreichen. Mit der folgenden Formel lässt sich die „Distanz" zwischen der wahren Wahrscheinlichkeitsverteilungsdichte $p_0(y)$ zur approximierten Verteilungsdichte $\hat{p}(y)$ beschreiben:

$$\begin{aligned} D(p_0,\hat{p}) &= \int_\Omega p_0(y)\ln\left(\frac{p_0(y)}{\hat{p}(y)}\right)dy = \int_\Omega p_0(y)\ln\left(p_0(y)\right)dy - \int_\Omega p_0(y)\ln\left(\hat{p}(y)\right)dy \\ &= \mathrm{E}_0\{\ln p_0(y)\} - \underbrace{\mathrm{E}_0\{\ln\hat{p}(y)\}}_{\text{modellabhängig}} . \end{aligned} \tag{3.67}$$

Die KL-Information wurde von Hirotsugu Akaike 1973 als Basis genutzt, um das Akaike Informationskriterium (AIC) herzuleiten, welches einen formalen Zusammenhang mit der Maximum Likelihood-Methode herstellt, siehe [3]. Das Ziel bei der Modellbildung ist, die Informationsverluste $D(p_0,\hat{p})$ so gering wie möglich zu halten. Leider ist allerdings die *wahre* Verteilungsdichte $p_0(y)$ nicht bekannt. Außerdem kennt man den exakten Parametervektor $\underline{\theta}$ nicht, sondern lediglich den Schätzer $\underline{\hat{\theta}}(y)$, welcher wiederum eine Zufallsvariable darstellt, weil die Realisierung y aus $p_0(y)$ erzeugt wurde.

Die Gleichung (3.67) zeigt, dass sich die KL-Information in zwei Terme aufteilen lässt. Der zweite Term wird auch als relative KL-Information bezeichnet. Im Gegensatz zum ersten Term lässt sich dieser durch die Abhängigkeit vom Modell beeinflussen. Außerdem kann man herleiten, dass die KL-Distanz $D(p_0,\hat{p})$ immer größer oder gleich Null sein muss (siehe z.B. [141]). Daher ist die Minimierung der KL-Distanz gleichbedeutend mit der Maximierung der relativen KL-Information:

$$I(p_0,\hat{p}) = \mathrm{E}_0\{\ln\hat{p}(y)\} . \tag{3.68}$$

Leider muss man zur Maximierung dieses Maßes folgendes Problem lösen: Um ein Modell mit diesem Maß zu bewerten, würde man für $\hat{p}(y)$ im Idealfall die *bedingte* Verteilungsdichte $\hat{p}(y) = p(y|\underline{\theta})$ verwenden. Leider ist die Verteilungsdichte $p(y|\underline{\theta})$ i.A. aber nicht bekannt. Darüber hinaus lässt sich der Erwartungswert $\mathrm{E}_0\{\ln\hat{p}(y)\}$ schlichtweg nicht berechnen, weil man dazu die wahre Verteilungsdichte der Messungen $p_0(y)$ kennen müsste.

Akaike's Informationskriterium

Wegen der vorangehend genannten Schwierigkeiten besteht lediglich die Möglichkeit, die relative KL-Information $\mathrm{E}_0\{\ln\hat{p}(y)\}$ zu *approximieren*. In [4] wird dazu die Anwendung

eines Ansatzes vorgeschlagen, der im Sinne einer Kreuzvalidierung interpretiert werden kann, siehe z.B. [22] oder [141]. Der Trick dabei ist, dass man eine fiktive Hilfsgröße x einführt, die zwar unabhängig von y ist, aber die gleiche Anzahl an Datenpunkten N und die gleiche Verteilungsdichtefunktion besitzt. Damit wird

$$\ln \hat{p}(y) = \mathrm{E}_x\{\ln p(y|\hat{\underline{\theta}}_x)\} \tag{3.69}$$

und die relative KL-Information bekommt folgenden Ausdruck:

$$\hat{I} = \mathrm{E}_y\left\{\mathrm{E}_x\{\ln p(y|\hat{\underline{\theta}}_x)\}\right\} . \tag{3.70}$$

Hierbei sind $\mathrm{E}_x\{\cdot\}$ und $\mathrm{E}_y\{\cdot\}$ die Erwartungswerte bezüglich der Verteilungsdichten von x und y. $\hat{\underline{\theta}}_x$ bezeichnet die ML-Schätzung der Modellparameter, wenn x zur Verfügung stünde. Den Ausdruck $\ln p(y|\hat{\underline{\theta}}_x)$ benutzt man nun dazu, um die unbekannte Verteilungsdichte mit Hilfe einer Taylorreihenentwicklung zweiter Ordnung zu approximieren:

$$\begin{aligned}\ln p(y|\hat{\underline{\theta}}_x) \;\approx\; & \ln p(y|\hat{\underline{\theta}}_y) + (\hat{\underline{\theta}}_x - \hat{\underline{\theta}}_y)^T \left(\left.\frac{\partial \ln p(y|\underline{\theta})}{\partial \underline{\theta}}\right|_{\underline{\theta}=\hat{\underline{\theta}}_y} \right) \\ & + \frac{1}{2}(\hat{\underline{\theta}}_x - \hat{\underline{\theta}}_y)^T \left(\left.\frac{\partial^2 \ln p(y|\underline{\theta})}{\partial \underline{\theta}\partial \underline{\theta}^T}\right|_{\underline{\theta}=\hat{\underline{\theta}}_y} \right) (\hat{\underline{\theta}}_x - \hat{\underline{\theta}}_y) .\end{aligned} \tag{3.71}$$

Diese Gleichung lässt sich noch weiter vereinfachen. Setzt man die ML-Schätzung voraus, so ergibt sich für ein Maximum der Log-Likelihood-Funktion $\ln p(y|\underline{\theta})$, dass die erste Ableitung nach den Parametern $\underline{\theta}$ verschwindet und die zweite Ableitung negativ ist. Darüber hinaus lässt sich, wie z.B. in [141] hergeleitet, die zweite Ableitung explizit ausrechnen.

Letztendlich ist das Hauptergebnis von Akaike [4], dass der folgende Ausdruck einer erwartungstreuen Schätzung der relativen KL-Information entspricht:

$$\hat{I} \approx \ln \mathcal{L}(\hat{\underline{\theta}}|y) - n . \tag{3.72}$$

Durch Multiplikation der Gleichung (3.72) mit -2 ergibt sich das *Akaike Information Criterion* (AIC)

$$\mathrm{AIC} = -2 \ln \mathcal{L}(\hat{\underline{\theta}}|y) + 2n , \tag{3.73}$$

welches eine Abschätzung der erwarteten relativen Distanz zwischen dem Modell und dem wahren Prozess erlaubt, mit dem die Messdaten erzeugt wurden [22]. Der häufigste Anwendungsfall für das AIC-Kriterium steht im Zusammenhang mit der (nichtlinearen) Least Squares-Schätzung. Der zusätzliche Faktor -2 ist dadurch motiviert, dass in diesem Fall die zugehörige gauß'sche Likelihood-Funktion (siehe z.B. Gl. (3.5) in Kapitel 3.2) die einfache Gestalt annimmt (siehe auch [22, 141]):

$$-2 \ln \mathcal{L}(\hat{\underline{\theta}}|y) = N \ln \hat{\sigma}^2 = N \ln \left(\frac{1}{N} \sum_{i=1}^{N} (y(i) - \hat{y}(\underline{u}(i)))^2 \right) . \tag{3.74}$$

Korrigiertes Akaike Informationskriterium AIC_c

Für Probleme mit relativ wenigen Daten und vielen Parametern ist die Erweiterung des AIC-Kriteriums um einen Bias-Korrekturterm für wenige Daten sinnvoll. Laut [22] ist dies der Fall für $N/n < 40$ und findet daher auch in allen in dieser Arbeit vorgestellten Algorithmen Anwendung. Das um den zweiten Bias-Term korrigierte AIC-Kriterium lautet dann:

$$\begin{aligned} \mathrm{AIC}_c &= N \ln \left(\frac{1}{N} \sum_{i=1}^{N} (y(i) - \hat{y}(\underline{u}(i)))^2 \right) + 2n + \frac{2n(n+1)}{N-n-1} \\ &= \mathrm{AIC} + \frac{2n(n+1)}{N-n-1} \,. \end{aligned} \tag{3.75}$$

Für große Datenmengen sind AIC und AIC_c äquivalent. Die unterschiedliche Bestrafung eines Modells ohne und mit Korrekturterm ist in Bild 3.12 gezeigt. Man sieht, dass mit AIC_c eine deutlich größere Bestrafung stattfindet, sobald die Anzahl der Parameter in die Größenordnung der Anzahl der Datenpunkte kommt.

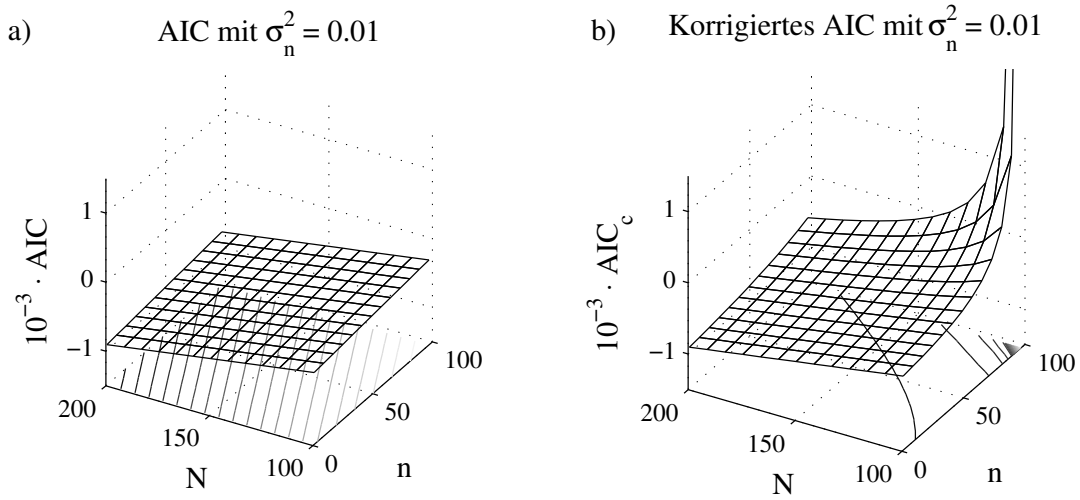

Bild 3.12: Vergleich des AIC-Kriteriums (a) mit dem korrigierten Kriterium AIC_c (b). Je weniger Datenpunkte N und je mehr Parameter n das Modell hat, desto stärker ist die Bestrafung durch AIC und AIC_c. Bei (b) ist die zusätzliche Bestrafung erkennbar.

3.6 Strukturselektion bei gewichteter lokaler Schätzung

Bei der Strukturselektion geht man davon aus, dass dem Schätzproblem mehr Regressoren zur Verfügung stehen als es für den zugrunde liegenden Prozess nötig wäre. In diesem Fall könnte man einen oder mehrere redundante Regressoren einfach weglassen, ohne dass es Einfluss auf das Modell hat. Ein gängiges Vorgehen bei der Strukturselektion ist es, einen

statistischen Signifikanztest durchzuführen, d.h. die Wahrscheinlichkeit auszurechnen, mit der ein Modellparameter zu Null gesetzt werden darf. Dies ist gleichbedeutend mit dem Weglassen eines redundanten Regressors.

Die generelle Vorgehensweise ist laut [31] folgendermaßen:

1. Schätze den Parameter $\hat{\theta}$ und die Standardabweichung der Parameterschätzung $\hat{\sigma}_\theta$. Damit lässt sich das Konfidenzintervall des Parameters θ wie folgt aufstellen:

$$\hat{\theta} \pm \hat{\sigma}_\theta \cdot t_{\nu,1-\alpha/2}\,. \tag{3.76}$$

   Der Parameter liegt demnach mit einer Wahrscheinlichkeit von $1-\alpha$ innerhalb dieses Intervalls. Der Faktor $t_{\nu,1-\alpha/2}$ gibt an, mit welchem Wert die Standardabweichung $\hat{\sigma}_\theta$ multipliziert werden muss, um mit ν Freiheitsgraden die Wahrscheinlichkeit $1-\alpha$ bezüglich einer t-Verteilung zu gewährleisten.
2. Zum Testen der Hypothese, dass ein Parameter zu Null gesetzt werden kann, d.h. $\theta = 0$, lässt sich ein t-Wert berechnen:

$$t = \frac{\hat{\theta}}{\hat{\sigma}_\theta}\,. \tag{3.77}$$

   Mit diesem t-Wert und der gegebenen Anzahl an Freiheitsgraden ν kann man auf die Wahrscheinlichkeit p zurückrechnen, mit der beurteilt werden kann, ob dieser Parameter tatsächlich zu Null gesetzt werden darf, ohne signifikanten Einfluss auf das Modell zu haben. Für $p < \alpha$ ist die Hypothese $\theta = 0$ unwahrscheinlich und der Parameter ist daher signifikant. Umgekehrt für $p > \alpha$ kann der Parameter zu Null gesetzt werden, ohne das Modell signifikant zu beeinflussen.

Im Rahmen dieser Arbeit wurde die Strukturselektion speziell für die gewichtete Least Squares Schätzung entwickelt, die bei lokalen Modellnetzen Verwendung findet. Schwierigkeit hierbei war, dass sich durch die Gewichtung bei der Least Squares-Schätzung Unterschiede zur klassischen Vorgehensweise bei Polynommodellen ergeben. Dies betrifft sowohl die Berechnung der Freiheitsgrade zur Schätzung als auch die Bestimmung der Parametervarianzen. Nähere Details dazu befinden sich in Anhang B.

Prinzipiell unterscheidet man zwischen Rückwärtseliminierung und Vorwärtsselektion. Bei der Rückwärtseliminierung geht man von einem bestehenden Modell mit allen möglichen Regressoren aus und stellt sich davon ausgehend die Frage, ob Regressoren beim Modell herausgelassen werden können. Im Gegensatz dazu selektiert man bei der Vorwärtsselektion, ausgehend von einem initialen Regressorensatz, nacheinander die signifikantesten Regressoren solange, bis keine signifikanten Regressoren mehr gefunden werden. Oft werden diese beiden Selektionsverfahren miteinander kombiniert, um einen möglicherweise noch besseren Regressorensatz zu finden, als dies bei reiner Vorwärtsselektion oder Rückwärtseliminierung der Fall wäre. Diese Arbeit beschränkt sich an dieser Stelle aber auf diese beiden Verfahren.

Im Folgenden wird die Vorgehensweise zur Berechnung der t-Werte detailliert beschrieben. Hierzu wird ein lokales Modell mit Index k und der Gültigkeit $\underline{Q}_k = \mathrm{diag}(\underline{\Phi}_k(\cdot))$ betrachtet.

3.6.1 Rückwärtseliminierung

1. Gewichtung von $\underline{X}$ und $\underline{y}$ mit $\sqrt{\underline{Q}_k}$ ergibt: $\underline{\widetilde{X}} = \sqrt{\underline{Q}_k}\underline{X}$ und $\underline{\widetilde{y}} = \sqrt{\underline{Q}_k}\underline{y}$.

2. Aufteilung der Regressionsmatrix $\underline{\widetilde{X}}$ in $\underline{X}_1$ und $\underline{X}_2$:
   a) $\underline{X}_1$: Regressoren aus $\underline{\widetilde{X}}$ die noch zum Modell gehören.
   b) $\underline{X}_2$: Regressoren die bereits als nicht signifikant verworfen wurden.
3. Auswertung des Modells mit den aktuellen Regressoren $\underline{X}_1$:
   a) Schätze die Parameter des aktuellen Modells:

$$\underline{\hat{\theta}}_1 = (\underline{X}_1^T \underline{X}_1)^{-1} \underline{X}_1^T \underline{\widetilde{y}} \,. \tag{3.78}$$

   b) Berechne die Anzahl der effektiven Parameter:

$$n_{\text{eff}} = \text{spur}(\underline{S}) = \text{spur}\left(\sqrt{\underline{Q}_k}\underline{X}_1(\underline{X}_1^T\underline{X}_1)^{-1}\underline{X}_1^T\sqrt{\underline{Q}_k}\right) . \tag{3.79}$$

   .
   c) Berechne den Modellfehlervektor und schätze die Rauschvarianz:

$$\underline{\widetilde{e}}_1 = \sqrt{\underline{Q}_k}\left(\underline{y} - \underline{X}_1\underline{\hat{\theta}}_1\right) . \tag{3.80}$$

$$\hat{\sigma}_1^2 = \frac{\underline{\widetilde{e}}_1^T\,\underline{\widetilde{e}}_1}{N_k - n_{\text{eff}}} \,, \quad \text{mit } N_k = \sum_{i=1}^{N} \Phi_k(\underline{u}(i)) \,. \tag{3.81}$$

4. Berechne die Parametervarianzen:

$$\hat{\sigma}^2_{\underline{\theta}_1} = \text{diag}\left((\underline{X}_1^T\underline{X}_1)^{-1}\underline{X}_1^T\underline{Q}_k\underline{X}_1(\underline{X}_1^T\underline{X}_1)^{-1}\right)\cdot\hat{\sigma}_1^2 \,. \tag{3.82}$$

5. Bestimme t-Werte zu jedem Kandidatenregressor j aus $\underline{X}_1$:

$$t_j = \frac{\hat{\theta}_{1,j}}{\hat{\sigma}_{\underline{\theta}_{1,j}}} \,. \tag{3.83}$$

6. Berechne p-Werte und treffe Auswahl:
   Zu jedem t-Wert wird eine Wahrscheinlichkeit p berechnet. Der Regressor mit dem größten p-Wert kommt für die Eliminierung in Betracht, weil hier die Wahrscheinlichkeit zum Nullsetzen des zugehörigen Parameters am größten ist. Sofern das Signifikanzniveau α überschritten ist, d.h. $p > \alpha$, wird der Regressor fürs Modell verworfen.
7. Falls $p < \alpha$ oder die geforderte minimale Anzahl an Regressoren erreicht wurde, Abbruch. Sonst gehe zu Schritt 2.

3.6.2 Vorwärtsselektion

1. Gewichtung von $\underline{X}$ und $\underline{y}$ mit $\sqrt{\underline{Q}_k}$ ergibt: $\underline{\widetilde{X}} = \sqrt{\underline{Q}_k}\underline{X}$ und $\underline{\widetilde{y}} = \sqrt{\underline{Q}_k}\underline{y}$.
2. Aufteilung der Regressionsmatrix $\underline{\widetilde{X}}$ in $\underline{X}_1$ und $\underline{X}_2$:
   a) $\underline{X}_1$: Regressoren aus $\underline{\widetilde{X}}$ die bereits zum Modell gehören.
   b) $\underline{X}_2$: Regressoren die als nächste Kandidaten zur Selektion in Frage kommen.
3. Auswertung des Modells mit den bereits selektierten Regressoren $\underline{X}_1$:
   a) Schätze die Parameter des aktuellen Modells:

$$\underline{\hat{\theta}}_1 = (\underline{X}_1^T \underline{X}_1)^{-1} \underline{X}_1^T \underline{\widetilde{y}} \,. \tag{3.84}$$

b) Berechne die Anzahl der effektiven Parameter:

$$n_{\text{eff}} = \text{spur}(\underline{S}) = \text{spur}\left(\sqrt{\underline{Q}_k}\underline{X}_1(\underline{X}_1^T\underline{X}_1)^{-1}\underline{X}_1^T\sqrt{\underline{Q}_k}\right) . \tag{3.85}$$

.

c) Berechne den Modellfehlervektor:

$$\widetilde{\underline{e}}_1 = \sqrt{\underline{Q}_k}\left(\underline{y} - \underline{X}_1\hat{\underline{\theta}}_1\right) . \tag{3.86}$$

4. Bestimme Parametervarianzen zu jedem Kandidatenregressor:
   a) Orthogonalisierung[5] von $\underline{X}_2$ zu $\underline{X}_1$ liefert $\underline{X}_{21}$:

$$\underline{X}_{21} = \underline{X}_2 - \underline{X}_1(\underline{X}_1^T\underline{X}_1)^{-1}\underline{X}_1^T\underline{X}_2 . \tag{3.87}$$

   b) Schätze die Parameter zu allen Kandidatenregressoren $\underline{x}_{21,j}$ (entspricht der j-ten Spalte von $\underline{X}_{21}$):

$$\hat{\theta}_{21,j} = \frac{\underline{x}_{21,j}^T\,\widetilde{\underline{e}}_1}{\underline{x}_{21,j}^T\,\underline{x}_{21,j}} . \tag{3.88}$$

   c) Berechne den Modellfehlervektor zu jedem Kandidatenregressor und schätze mit dem getesteten Regressor die Rauschvarianz:

$$\widetilde{\underline{e}}_2 = \widetilde{\underline{e}}_1 - \underline{x}_{21,j}\,\hat{\theta}_{21,j} . \tag{3.89}$$

$$\hat{\sigma}_{2,j}^2 = \frac{\widetilde{\underline{e}}_2^T\,\widetilde{\underline{e}}_2}{N_k - n_{\text{eff}} - 1} , \quad \text{mit } N_k = \sum_{i=1}^{N}\Phi_k(\underline{u}(i)) . \tag{3.90}$$

   d) Berechnung der Parametervarianz der zu selektierenden Kandidatenregressoren:

$$\hat{\sigma}_{\theta_{21,j}}^2 = \frac{\underline{x}_{21,j}^T\,\underline{Q}_k\,\underline{x}_{21,j}}{\left(\underline{x}_{21,j}^T\,\underline{x}_{21,j}\right)^2}\cdot\hat{\sigma}_{2,j}^2 . \tag{3.91}$$

5. Bestimme t-Werte zu den Kandidatenregressoren:

$$t_j = \frac{\hat{\theta}_{21,j}}{\hat{\sigma}_{\theta_{21,j}}} . \tag{3.92}$$

6. Berechne p-Werte und treffe Auswahl:
   Zu jedem t-Wert wird eine Wahrscheinlichkeit p berechnet. Der Regressor mit dem kleinsten p-Wert kommt für die Selektion in Betracht, weil hier die Wahrscheinlichkeit zum Nullsetzen des zugehörigen Parameters am kleinsten ist. Sofern das Signifikanzniveau α unterschritten ist, d.h. $p < \alpha$, wird der Regressor fürs Modell selektiert.
7. Falls $p > \alpha$ oder die geforderte maximale Anzahl an Regressoren erreicht wurde, Abbruch. Sonst gehe zu Schritt 2.

Zur Veranschaulichung der Schätzung mit und ohne orthogonale Regressoren ist in Bild 3.13 ein Illustrationsbeispiel gezeigt. Zunächst wird, wie im Fall a) gezeigt, der Parameter θ_1 mit dem Regressor $\underline{x}_1$ geschätzt. Darauf basierend schätzt man den Parameter θ_{21}, dessen zugehöriger Regressor $\underline{x}_{21}$ orthogonal zum Regressor $\underline{x}_1$ steht. Entscheidet man sich daraufhin, den Regressor $\underline{x}_2$ mit in das Modell aufzunehmen, so müssen beide Parameter neu geschätzt werden, weil sich die Parameterverhältnisse im nicht-orthogonalen Fall b) ändern.

[5]Die Matrix $\underline{X}_{21}$ ist orthogonal zu $\underline{X}_1$, sodass $\underline{X}_{21}^T\underline{X}_1 = \underline{0}$. Für weitere Details zur Orthogonalisierung von Matrizen siehe z.B. [31].

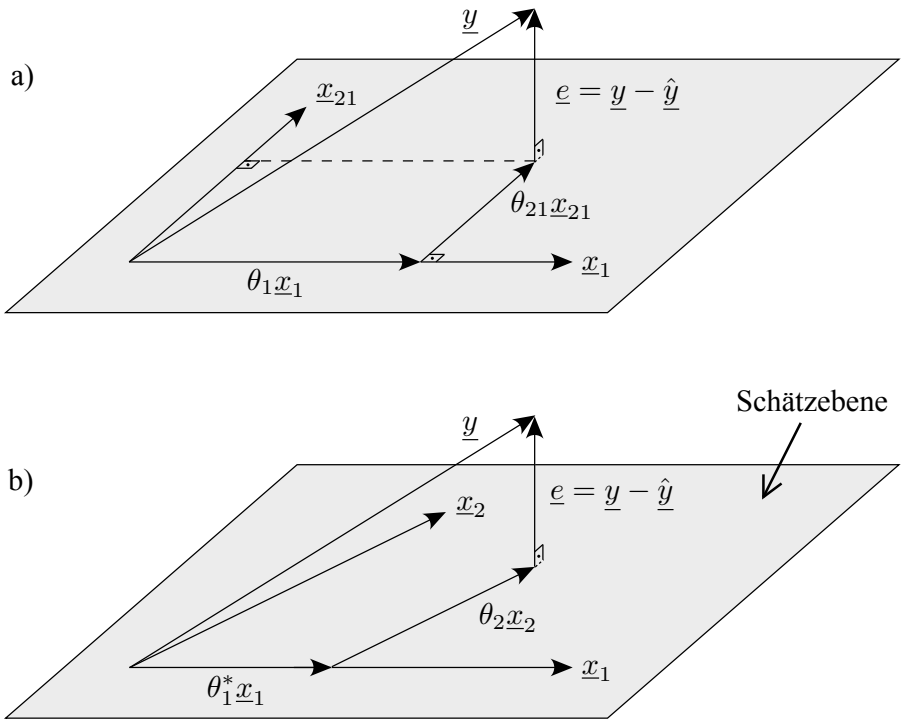

Bild 3.13: Beispiel zum Vergleich der Schätzung mit a) orthogonalen Regressoren und b) nicht-orthogonalen Regressoren. Zu beachten ist, dass der Parameter zum Regressor $\underline{x}_1$ sich von Fall a) zu Fall b) unterscheidet, d.h. $\theta_1 \neq \theta_1^*$.

Hinweise zur schnellen Implementierung

Da die Vorwärtsselektion innerhalb eines Trainingsalgorithmus für lokale Modellnetze verwendet wird, ist es sinnvoll, eine QR-Zerlegung der Regressionsmatrix $\underline{X}_1$ durchzuführen, sodass $\underline{X}_1 = \underline{Q}_{\mathrm{QR},1}\underline{R}_{\mathrm{QR},1}$.

Die Berechnung der Anzahl der effektiven Parameter in Gl. (3.85) vereinfacht sich dann zu:

$$n_{\mathrm{eff}} = \sum_{i=1}^{N}\sum_{j=1}^{nx_1} \Phi_k(\underline{u}(i))\, \underline{Q}^2_{\mathrm{QR},1\,ij}\,, \tag{3.93}$$

wobei nx_1 die Anzahl der Regressoren beziehungsweise die Anzahl der Spalten von $\underline{X}_1$ ist.

Darüber hinaus vereinfacht sich die Orthogonalisierung in Gl. (3.87) zu:

$$\begin{aligned}
\underline{X}_{21} &= \underline{X}_2 - \underline{Q}_{\mathrm{QR},1}\underline{R}_{\mathrm{QR},1}(\underline{R}^T_{\mathrm{QR},1}\underbrace{\underline{Q}^T_{\mathrm{QR},1}\underline{Q}_{\mathrm{QR},1}}_{\underline{I}}\underline{R}_{\mathrm{QR},1})^{-1}\underline{R}^T_{\mathrm{QR},1}\underline{Q}^T_{\mathrm{QR},1}\underline{X}_2 \\
&= \underline{X}_2 - \underline{Q}_{\mathrm{QR},1}\underbrace{\underline{R}_{\mathrm{QR},1}(\underline{R}^T_{\mathrm{QR},1}\underline{R}_{\mathrm{QR},1})^{-1}\underline{R}^T_{\mathrm{QR},1}}_{\underline{I}}\underline{Q}^T_{\mathrm{QR},1}\underline{X}_2 \\
&= \underline{X}_2 - \underline{Q}_{\mathrm{QR},1}\left(\underline{Q}^T_{\mathrm{QR},1}\underline{X}_2\right)\,.
\end{aligned} \tag{3.94}$$

3.7 Zusammenfassung

Das Kapitel hat zunächst die wichtigsten statistischen Grundlagen zur Schätzung lokal gewichteter Modelle dargelegt. Dazu gehören die Methode von Bayes, das Maximum Likelihood-Verfahren und die globale und lokale Schätzung bei lokalen Modellnetzen.

Neben der impliziten Regularisierung, die aus der lokalen Schätzung resultiert, bieten explizite Regularisierungsverfahren die Möglichkeit, die Flexibilität der globalen Schätzung zugunsten einer besseren Generalisierungsleistung des Modells gesteuert einzuschränken. In diesem Kapitel wird daher die Ridge Regression speziell für die globale Schätzung vorgestellt und darüber hinaus ein Verfahren, welches durch den Einsatz einer Singulärwertzerlegung die effiziente Optimierung des Reglurisierungsparameters ermöglicht. Abgesehen von der Ridge Regression ist die Tikhonov-Regularisierung zweiter Ordnung ein wichtiges Verfahren, bei dem die Krümmung des Modellausgangs bestraft wird. Ein aus der Literatur bekanntes approximatives Verfahren wurde in dieser Arbeit eigens für die Verwendung bei hierarchischen und achsenschrägen Partitionierungen entwickelt, bei denen die Schwierigkeit besteht, dass die Geometrie der Gültigkeitsfunktionen mathematisch nur äußerst schwierig zu beschreiben ist. Außerdem wird kurz die global-lokale Regularisierung nach [159] aufgeführt.

Das nächste wichtige Thema ist die Optimierung der Modellkomplexität. Basis des Abschnitts ist der Bias-Varianz-Kompromiss und wie man dies bezüglich für ein lokales Modellnetz das Optimum durch verschiedene Methoden finden kann. Klassischerweise wird die Validierung datengetrieben durchgeführt. Falls keine expliziten Validierungsdaten vorliegen, ist die Kreuzvalidierung das Mittel der Wahl. Ein wichtiger Spezialfall ist die Leave-One-Out-Kreuzvalidierung (LOOCV), weil man sie für linear geschätzte Modelle sehr effizient berechnen kann. In diesem Abschnitt wird hergeleitet und gezeigt, wie man unter Ausnutzung einer QR-Zerlegung die LOOCV numerisch günstig auf die lokale Schätzung anwenden kann. Obwohl der nichtlineare Einfluss der Partitionierungsparameter dabei vernachlässigt wird, zeigt das Verfahren eine sehr gute Übereinstimmung mit einer vollständig durchgeführten Kreuzvalidierung, welche um Größenordnungen aufwändiger und damit langsamer ist. Neben der datengetriebenen Validierung bietet sich der Einsatz von Informationskriterien an. Das korrigierte AIC-Kriterium hat sich hier als sehr nützlich erwiesen und wird daher ausführlich erläutert.

Zum Abschluss des Kapitels rückt die Thematik der Strukturselektion in den Vordergrund. Sowohl die Vorwärtsselektion als auch die Rückwärtseliminierung wurden für die Anwendung bei lokal geschätzten Polynommodellen entwickelt. Zur Umsetzung bei lokalen Modellnetzen mussten einige Anpassungen im Vergleich zur Standardmethode bei ungewichteten Polynommodellen vorgenommen werden. Für die Einschätzung der Regressoren nach Signifikanz dient ein sequentieller t-Test, mit dem zu jedem Regressor eine Wahrscheinlichkeit zur Bestätigung der Null-Hypothese ausgerechnet werden kann. Zur numerisch günstigen Implementierung wird auch hier eine QR-Faktorisierung eingesetzt.

4 Nichtlineare Optimierung der Struktur lokaler Modellnetze

Das vorangegangene Kapitel hat sich im Speziellen mit linearen Optimierungsverfahren für lokale Modellnetze (LMN) beschäftigt. Dieses Kapitel konzentriert sich nun im Unterschied dazu auf die *nichtlineare* Optimierung der Gültigkeitsfunktionen lokaler Modellnetze.

Prinzipiell lassen sich zwei Arten von Architekturen für lokale Modellnetze unterscheiden. Die wohl gängigste ist die parallele Modellstruktur, wobei sich der Ausdruck „parallel“ auf die Eigenschaft bezieht, die Gültigkeitsfunktionen eines LMN parallel ausrechnen zu können. Im Unterschied dazu können LMN auch eine hierarchische Architektur aufweisen, bei der eine parallelisierte Berechnung der Partitionierung bzw. der Gültigkeitsfunktionen nicht mehr möglich ist. Dafür ergeben sich neue begünstigende Vorteile, die eine Realisierung von effizienten, achsenschrägen Partitionierungsalgorithmen ermöglichen und in diesem Kapitel eine tragende Rolle spielen.

Das Kapitel ist wie folgt gegliedert: Zunächst gibt Abschnitt 4.1 einen Überblick zu existierenden Partitionierungsverfahren für lokale Modellnetze. Daraufhin beschreibt Abschnitt 4.2 die Optimierung paralleler Modellstrukturen, wo neben dem prinzipiellen Aufbau paralleler Architekturen und unerwünschten Effekten, die durch die erforderliche Normierung entstehen können, die zwei Trainingsverfahren LOLIMOT und SUHICLUST vorgestellt werden. Kapitel 4.3 beschäftigt sich mit der Optimierung hierarchischer Modellstrukturen. Dazu wird zunächst das Prinzip der hierarchischen Struktur erklärt, woraufhin das Trainingsverfahren HILOMOT mit der zugehörigen Schnittoptimierung detailliert beschrieben wird. Mit der richtigen Einstellung der Modellkomplexität bei LMN beschäftigt sich Abschnitt 4.4, woraufhin die Strategie der Strukturabwägung für LMN in Abschnitt 4.5 vorgestellt wird. Das Kapitel endet mit der Zusammenfassung der wichtigsten Erkenntnisse in Abschnitt 4.7.

4.1 Partitionierungsverfahren – Stand der Forschung

Die Leistungsfähigkeit lokaler Modellansätze hängt im Wesentlichen davon ab, wie die Gültigkeitsbereiche für die einzelnen lokalen Modelle festgelegt werden. Die Form und Art der Unterteilung des Eingangsraums in diese Gültigkeitsbereiche ist für die Eigenschaften sowohl des Modells als auch des Trainingsverfahrens entscheidend.

Die folgenden Ausführungen setzen sich mit den gängigsten Partitionierungsverfahren auseinander, welche beispielsweise in den Veröffentlichungen [117], [47], [119] und [66] zu finden sind. Zur Herausarbeitung der Vor- und Nachteile der verschiedenen Vorgehensweisen werden die prinzipiellen Eigenschaften analysiert, die sich durch theoretische Überlegungen durchführen lassen. Die in der Literatur vorgeschlagenen Partitionierungsverfahren zur Festlegung der Gültigkeitsfunktionen lassen sich in folgende Klassen einteilen:

1. *Gitter:* Eine gitterförmige Verteilung der Gültigkeitsfunktionen ergibt sich in natürlicher Weise aus einem vollständigen Satz von Fuzzy-Regeln, bei der alle Kombinationen der Eingangszugehörigkeitsfunktionen mit „Und“ verknüpft werden (Tensorprodukt) [146]. Diese Vorgehensweise hat den Vorteil, dass sich die Gültigkeitsfunktionen aus den eindimensionalen Zugehörigkeitsfunktionen über die verwendete t-Norm (z.B. den Produkt-Operator zur Realisierung der „Und“-Verknüpfung) direkt ergeben. Dies ermöglicht, hochdimensionale Zusammenhänge auf eindimensionale zurückzuführen. Außerdem stimmt der Gitteransatz exakt mit den weit verbreiteten Rasterkennfeldern überein, bei denen allerdings üblicherweise lineare Interpolation eingesetzt wird, was dreiecksförmigen Zugehörigkeitsfunktionen entspricht.
   Der wesentliche Nachteil des Gitteransatzes besteht im exponentiellen Anwachsen der Anzahl der Neuronen bzw. lokalen Modelle mit der Dimension des Eingangsraums der Regelprämissen nz. Der Ansatz unterliegt daher in vollem Umfang dem „Fluch der Dimensionalität“. Dies beschränkt den Gitteransatz praktisch auf niedrigdimensionale Probleme ($nz \leq 4$). An dieser prinzipiellen Einschränkung ändert sich auch durch die in [143, 75] vorgeschlagenen Verbesserungen und Erweiterungen nichts Wesentliches.
2. *Clustering im Eingangsraum:* Ursprünglich vorgeschlagen für die Festlegung der Zentren von radialen Basisfunktionnetzen [112], wurde die Verwendung von Clustering-Verfahren auch auf Netze mit lokal linearen Modellen erweitert [142]. Dabei wird die Datenverteilung im von $\underline{z}$ aufgespannten Eingangsraum zum Maßstab für die Verteilung der Gültigkeitsfunktionen herangezogen. Der Hauptvorteil einer solchen Vorgehensweise liegt darin, dass die lokalen Modelle nur dorthin platziert werden, wo auch Daten für die Schätzung ihrer Parameter vorhanden sind. Der Hauptnachteil ist die Verteilung der Modellkomplexität nach der Verteilung der Eingangsdaten (d.h. viele lokale Modelle werden dort erzeugt, wo viele Daten liegen) und nicht nach der Komplexität des zu approximierenden Prozesses. Informationen über die Ausgangsgröße y werden für die Partitionierung nicht herangezogen.
   Ein weiterer Nachteil aller Clustering-Ansätze besteht darin, dass die Interpretation in Form von Fuzzy-Regeln größtenteils verloren geht, siehe Bild 4.1 aus [116]. Es entstehen durch die Projektion der Gültigkeitsfunktionen auf die einzelnen Eingangsgrößen für jede Gültigkeitsfunktion jeweils eine Zugehörigkeitsfunktion pro Eingang, siehe Bild 4.1a. Die dadurch entstehende übergroße Zahl an Zugehörigkeitsfunktionen und deren teilweise große Ähnlichkeit führen zum Verlust der Anschaulichkeit. Darüber hinaus lassen sich die Gültigkeitsfunktionen nicht verlustfrei auf die Eingänge projizieren, wenn sie nicht achsenorthogonal ausgerichtet sind, siehe Bild 4.1b. In [7] werden diese Interpretationsprobleme mit Hilfe von geeigneten Vereinfachungsmethoden überwunden, allerdings auf Kosten einer reduzierten Modellgüte.
3. *Clustering im Produktraum:* Hiermit wird der Hauptnachteil des zuvor geschilderten Clustering-Ansatzes vermieden, nämlich das Ignorieren des Prozessausgangs y für die Partitionierung. Voraussetzung hierfür sind identische Eingangsräume für die Regelprämissen und -konklusionen, also $\underline{z} = \underline{x}$. Diese Einschränkung stellt auch gleich einen schwerwiegenden Nachteil dieses Ansatzes dar. Es wird mittels spezieller Clustering-Verfahren nach Hyper-Ellipsoiden im sog. Produktraum, aufgespannt durch die Eingänge und den Ausgang $[z_1 \; z_2 \; \cdots \; z_{nz} \; y]$, gesucht. Dies stellt eine weitere Einschränkung auf Systeme mit nur einem Ausgang dar. Typischerweise wer-

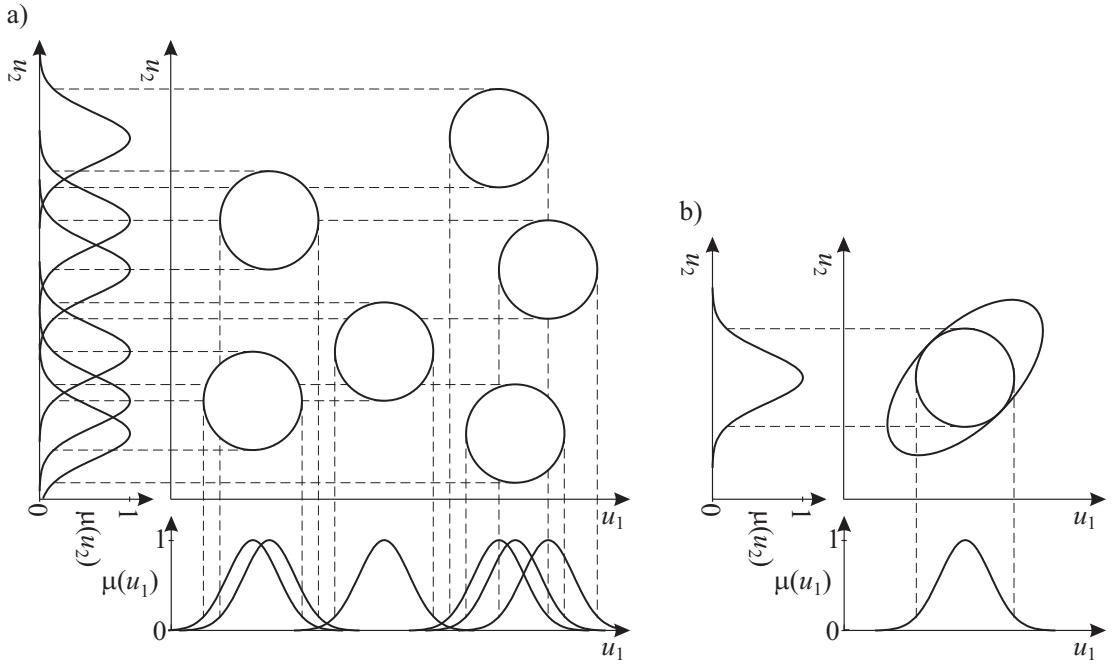

Bild 4.1: Probleme bei der Interpretation von durch Clustering entstandenen mehrdimensionalen Gültigkeitsfunktionen in Form von eindimensionalen Zugehörigekeitsfunktionen: a) Jede Gültigkeitsfunktion korrespondiert mit jeweils einer Zugehörigkeitsfunktion pro Eingang. b) Nicht achsenorthogonal ausgerichtete, mehrdimensionale Gültigkeitsfunktionen können nicht verlustfrei auf eindimensionale Zugehörigkeitsfunktionen projiziert werden [116].

den die Clustering-Verfahren nach Gustafson und Kessel [41] oder Gath und Geva [39] verwendet, die nach Hyper-Ellipsoiden identischen bzw. variablen Volumens suchen. Diese Hyper-Ellipsoiden approximieren Gebiete, in welchen sich der Prozess näherungsweise linear verhält, indem sich die Hauptachse, die senkrecht auf dem linearisierten Prozessverhalten steht, (im Vergleich zu den anderen) extrem verkürzt. Hauptstärke des Ansatzes ist die sehr flexible Ausrichtung der Gültigkeitsfunktionen. Diese werden typischerweise als Gaußfunktionen mit variabler Kovarianzmatrix gewählt und können damit leicht sehr komplexe Prozessgebiete beschreiben. Das Clustering-Verfahren sorgt automatisch dafür, dass in Bereichen stark nichtlinearen Prozessverhaltens genügend Cluster und damit lokal lineare Modelle generiert werden, wenn die Cluster-Anzahl genügend hoch gewählt wurde. Allerdings erzeugt das Verfahren nach Gustafson-Kessel Hyper-Ellipsoiden identischen Volumens, sodass in näherungsweise linearen Bereichen zu viele Cluster mit sehr ähnlichen linearen Modellen erzeugt werden. Diese können im Nachhinein zusammengefasst werden oder es kann der komplexere Gath-Geva-Algorithmus zum Einsatz kommen, bei dem man allerdings oft mit Konvergenz in schlechte lokale Optima rechnen muss [6].
Neben dem bereits erwähnten Nachteil bezüglich identischer Eingangsräume und der Einschränkung auf Systeme mit einem Ausgang, teilt das Produktraum-Clustering auch die im vorherigen Punkt schon diskutierten interpretatorischen Schwierigkeiten als Fuzzy-Regeln. Allerdings gibt es einige neuere Forschungsergebnisse, in welchen die bekannten Clustering-Algorithmen erweitert werden, um durch Nebenbedingun-

gen eine bessere Interpretierbarkeit zu erreichen [2]. Außerdem sind die Standardalgorithmen nur für die Suche von lokal *linearen* Modellen geeignet; andere lokale Modelle (z.B. quadratische) können nicht (direkt) verwendet werden.
Darüber hinaus ist die praktische Anwendung von Clustering-Algorithmen mit zwei grundlegenden Problemen behaftet: Zum einen sind diese Algorithmen stark von der jeweiligen Initialisierung der Cluster-Zentren abhängig; unterschiedliche Anfangswerte liefern verschiedene Resultate. Zum anderen ist vorab nicht bekannt, *wie viele* Clusters trainiert werden müssen. In der Vergangenheit wurden daher große Anstrengungen unternommen, um eine systematische Vorgehensweise zur Identifikation von Modellen basierend auf Fuzzy-Clustering zu ermöglichen, siehe z.B. [38], [137], [100] und [87].
An dieser Stelle wird eine grobe Übersicht gegeben, welche Verfahren in der gegenwärtigen Literatur vorkommen: Ein iteratives Verfahren zum regressionsbasierten Fuzzy-C-means-Clustering im Produktraum ist in [91] vorgestellt. Zur Erstellung der Fuzzy-Regeln werden die gefundenen Cluster auf die jeweilige Eingangsachse projiziert. Bei [144] wird zunächst eine geeignete Anzahl an Cluster mittels Fuzzy-C-means-Clustering im Ausgangsraum gesucht. Anschließend wird die Fuzzy-Partition durch Projektion der Cluster im Eingangsraum beschrieben. Problematisch bei diesem Ansatz ist, dass die Beziehungen zwischen Eingangs- und Ausgangsraum injektiv sind, d.h. dass Mehrdeutigkeiten zwischen Eingang und Ausgang entstehen können. Eine Anwendung zur Modellierung der Nitrat-Konzentration im Grundwasser ist beispielsweise in [153] zu finden, wo Fuzzy-C-means-Clustering in Verbindung mit Gauß'schen Zugehörigkeitsfunktionen zur lokalen Least Squares-Schätzung zum Einsatz kommen. Weiterhin stellt z.B. [33] eine systematische Vorgehensweise zur Modellierung komplexer Systeme unter Verwendung von Fuzzy-C-means-Clustering vor. Zur Begegnung höherdimensionaler Zusammenhänge wird in [71] die systematische Zerlegung eines komplexen Fuzzy-Systems in mehrere Subsysteme vorgeschlagen. In diesem Zusammenhang ist z.B. in [156] ein hierarchischer Clustering-Ansatz vorgestellt. Ein weiteres Verfahren, das mit hierarchischem Clustering arbeitet, ist in [152] zu finden. Dabei kommt ein gewichteter Fuzzy-C-means-Clustering-Algorithmus zum Einsatz. [99] stellt einen Trainingsalgorithmus vor, der automatisch die Parameter der Partitionierung und der lokalen Modelle eines Neuro-Fuzzy-Modells optimiert. Dabei wird eine rekursive Singulärwertzerlegung kombiniert mit einem Gradientenverfahren verwendet. Die Methode des *Substractive Clusterings* zur Identifikation von Fuzzy-Modellen ist in [24] vorgestellt.

4. *Datenpunktbasierte Methoden:* Ähnlich wie die Clustering-Ansätze orientieren sich diese Methoden an der Datenverteilung. Meist werden alle (oder eine Untermenge aller) Datenpunkte als mögliche Kandidaten für die Zentren der Gültigkeitsfunktionen betrachtet. Die Ausdehnung der Gültigkeitsfunktionen (z.B. die Standardabweichungen bei Gaußfunktionen) werden typischerweise heuristisch aus einem Nächste-Nachbar-Prinzip oder einer Kovarianzmatrizenschätzung bestimmt.
In der Literatur sind Lösungsansätze zu finden, die fälschlicherweise die Interpretierbarkeit als Fuzzy-System nicht berücksichtigen. Aus der großen Menge an potentiellen Gültigkeitsfunktionen werden dann die für eine hohe Modellgüte wichtigsten herausgesucht. Dies ist ein Strukturselektionsproblem, das sich z.B. mit einem orthogonalen Least Squares-Verfahren (OLS) durch Vorwärts-Selektion effizient aber nur subop-

timal lösen lässt [155, 69, 105]. Hierbei wird allerdings außer Acht gelassen, dass die potentiellen Gültigkeitsfunktionen von den unbekannten, noch mittels OLS zu selektierenden, Gültigkeitsfunktionen über die Normierung abhängen, siehe Bild 4.2:

$$\Phi_i(\underline{z}) = \frac{\mu_i(\underline{z})}{\sum_{j=1}^{M} \mu_j(\underline{z})} \tag{4.1}$$

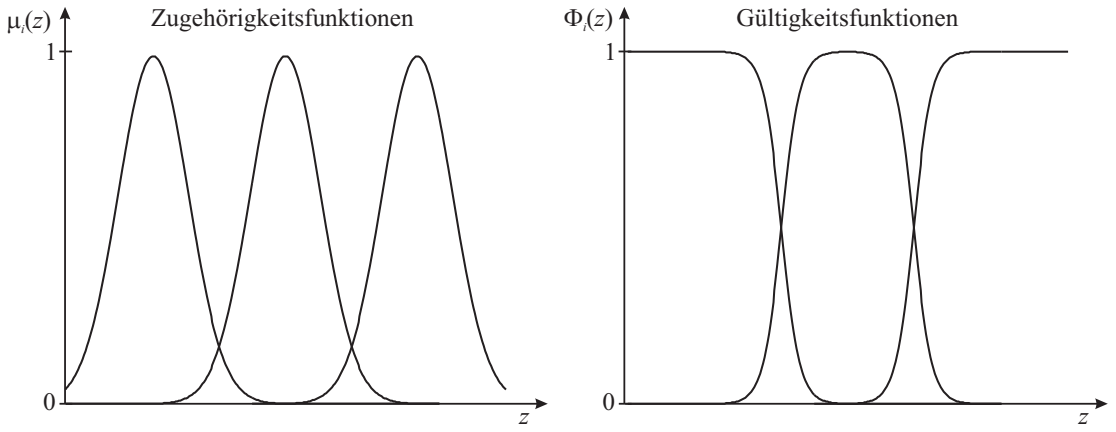

Bild 4.2: Die Normierung in (4.1) wandelt die Zugehörigkeits- in Gültigkeitsfunktionen um [116].

Diese Normierung berechnet aus den Zugehörigkeitsfunktionen $\mu_i(\underline{z})$ die Gültigkeitsfunktionen $\Phi_i(\underline{z})$ und stellt damit die „Partition of Unity“ her. Um die zur Auswahl stehenden Gültigkeitsfunktionen richtig zu berechnen, müsste man also schon vor der Strukturselektion wissen, welche M Gültigkeitsfunktionen selektiert werden. Um dieses Problem zu umgehen, wurden in [155, 69, 105] verschiedene Möglichkeiten vorgeschlagen. Zum einen ist es möglich, die Normierung zunächst wegzulassen, d.h. nicht normierte Zugehörigkeitsfunktionen zu selektieren und dann erst nach der Selektion die Normierung durchzuführen. Zum anderen kann die Normierung stets in Bezug auf alle Gültigkeitsfunktionskandidaten durchgeführt werden. Die erste Lösungsidee scheitert, da sich die Form der Funktionen durch die Normierung signifikant ändert (wie ein Vergleich zwischen den Zugehörigkeits- und Gültigkeitsfunktionen in Bild 4.2 zeigt) und daher eine nachträgliche Normierung die zuvor durchgeführte Strukturselektion wert- und sinnlos werden lässt. Der zweite Vorschlag bezieht in die Berechnung einer Gültigkeitsfunktion immer alle potentiellen Gültigkeitsfunktionen durch die Normierung mit ein. Dies ist aus Gründen des Rechenaufwands äußerst unvorteilhaft, denn die Anzahl der potentiellen Gültigkeitsfunktionen entspricht der Anzahl an Datenpunkten. Außerdem verletzt dies die Definition eines Fuzzy-Systems, weil dabei nur mit den Aktivitäten der selektierten Regeln normiert werden darf und die Bedingung zur Einhaltung der „Partition of Unity“ verletzt wird. Kurzum sind diese Ansätze aus interpretatorischer Sicht untauglich, selbst wenn einige numerisch gute Ergebnisse liefern können.

Wesentlich praxistauglicher sind die in [114] und [77] vorgeschlagenen Verfahren, die eine inkrementelle Vorgehensweise beinhalten. Schrittweise wird die Komplexität des Netzes durch Einfügen zusätzlicher, lokaler Modelle erweitert. Diese Erweiterungen

werden jeweils dort vorgenommen, wo ein lokales Fehlermaß am größten ist, also das existierende Netz den größten Verbesserungsbedarf aufweist. Der größte Vorteil dieser Ansätze ist die sehr flexible und sowohl dem Prozessverhalten als auch der Datenverteilung entsprechende Partitionierung. Nachteilig wirkt sich die hohe Empfindlichkeit bezüglich einzelner Datenpunkte aus, sodass sie eine mangelnde Robustheit in Bezug auf Ausreißer oder stark verrauschte Daten aufweisen.

5. *Nichtlineare Optimierung:* Der direkteste Ansatz zum Training ist, schlichtweg alle Parameter des Modells in einer einzigen Optimierung anzupassen. Sowohl linear als auch nichtlinear einfließende Parameter passt man dabei gleichzeitig an. Ein Nachteil dieser Vorgehensweise ist, dass die Modellkomplexität, d.h. die Anzahl der lokalen Modelle, a priori festgelegt sein muss. Darüber hinaus wächst die Anzahl der Parameter stark mit der Dimensionalität. Das führt unter Umständen zu sehr vielen Variablen, die zu optimieren sind. Dementsprechend ist die Wahl des Optimierungsalgorithmus von entscheidender Bedeutung. Das Hauptproblem hierbei ist jedoch, dass die Modellerstellung mit großem Rechenaufwand verbunden ist, was zu erheblichen Problemen bezüglich der Anwendbarkeit in der Praxis führen kann [2].
   Da eine nichtlineare Optimierung der Parameter der Gültigkeitsfunktionen sinnvolle Initialwerte benötigt, kann jedes der anderen beschriebenen Verfahren zur Erzeugung der Initialwerte verwendet werden. Eine nichtlineare Optimierung kann dann als Verfeinerungsschritt angesehen werden. Durch die Fuzzy-Toolbox von Matlab [122] ist z.B. das ANFIS-Verfahren (*adaptive neuro-fuzzy inference system*) [78, 79] bekannt geworden, welches auf einer Gitter-Partitionierung beruht. Dies bietet den Vorteil, dass durch die Gitterstruktur die Anzahl der Parameter relativ klein gehalten wird. Allerdings gelten die oben genannten Nachteile der Gitter-Partitionierung weiterhin. Geht man von den datenpunktbasierten oder Clustering-Ansätzen aus, liegt die Anzahl der zu optimierenden Parameter weitaus höher, z.B. bei $2pM$ für p-dimensionale Gaußfunktionen mit achsenorthogonaler Ausrichtung (eine Zentrumskoordinate und eine Standardabweichung je Eingangsgröße und Neuron). Außerdem müssten für die Bewertung der Güte die jeweiligen optimalen Parameter der lokalen Modelle geschätzt werden. Offensichtlich steht daher der für eine nichtlineare Optimierung nötige Aufwand in keinem Verhältnis zu der zu erwartenden Modellverbesserung. Daher wird dieser Weg nur sehr selten beschritten. Dies gilt sowohl für lokale Suchverfahren z.B. auf Gradientenbasis, als auch in noch verstärkter Weise für globale Suchverfahren wie evolutionäre Algorithmen o.ä.
   In [137], [158], [81], [131] und [70] findet die Suche nach einer geeigneten Modellstruktur und die Parameteroptimierung der lokalen Modelle mit Hilfe genetischer Algorithmen statt. Ein Ansatz basierend auf evolutionärer Programmierung wird in [87] vorgestellt. Ein heuristischer Optimierungsalgorithmus basierend auf einer Tabu-Suche ist in [29] vorgestellt. Darüber hinaus lassen sich Beispiele dieser Verfahrensklasse in [103], [66], [65] und [67] finden. Sogenannte *Evolving Neuro-Fuzzy Systeme* sind darüber hinaus online einsetzbar und ebenfalls in der Lage, die Regelprämissen und -konklusionen eines Takagi-Sugeno Fuzzy-Systems gleichzeitig zu adaptieren. Bekannte Vertreter in der gegenwärtigen Literatur sind: DENFIS [88], eTS [5], FLEXFIS [102], SONFIN [86] und SOFMLS [80].
6. *Heuristische Baumkonstruktionsverfahren:* Die in den CART-Algorithmus (*classification and regression tree*) [20] eingeflossenen Ideen wurden in verschiedenen Varia-

tionen auf Netze mit lokalen Modellen übertragen. Neben den Baumkonstruktionsverfahren aus [143, 82] hat sich der LOLIMOT-Algorithmus (*local linear model tree*) aus [120] recht weit verbreitet. Die Grundidee dieser Konstruktionsverfahren ist es, den Eingangsraum durch immer weitere Unterteilungen in seine Gültigkeitsbereiche zu partitionieren. Um die Anzahl an möglichen Unterteilungen überschaubar zu halten, werden üblicherweise nur achsenorthogonale Unterteilungen untersucht. Diese Vorgehensweise bietet den Vorteil, dass sich die entstehenden Gültigkeitsbereiche verlustfrei auf die Eingangsachsen projizieren lassen, was eine einfache Interpretationsmöglichkeit im Sinne einer Fuzzy-Logik sicherstellt.
Diese Ansätze kombinieren viele Vorteile. Sie liefern mittels Fuzzy-Logik gut interpretierbare Modelle, erlauben unterschiedliche Eingangsräume für Regelprämissen und -konklusionen und verschiedene lokale Modelltypen. Sie lassen sich auch für Netze mit mehreren Ausgängen erweitern, zudem vereinfacht die inkrementelle Natur der Algorithmen die Wahl der optimalen Modellkomplexität. Die Partitionierung richtet sich nach dem Prozessverhalten und erzeugt dort viele lokale Modelle, wo ein nichtlineares Verhalten besonders stark ausgeprägt ist. Außerdem benötigen die Algorithmen sehr wenig Rechenaufwand und ihre Komplexität wächst nur moderat mit der Dimension des Eingangsraums und der Anzahl der Neuronen, d.h. sie sind für (prinzipiell) hochdimensionale und sehr komplizierte Modellierungsaufgaben geeignet. Der Hauptnachteil dieser Verfahren liegt in der Beschränkung auf achsenorthogonale Unterteilungen. Diese Beschränkung ist zwar für eine gute Interpretierbarkeit sinnvoll und macht die Algorithmen schnell und einfach, aber sie kann sehr ineffizient sein. Als Worst-Case-Szenario stelle man sich eine Nichtlinearität vor, die entlang der Diagonale des Eingangsraums verläuft. Diese kann nur sehr unvollkommen durch eine große Anzahl achsenorthogonaler Partitionen beschrieben werden. Diese Ineffizienz wird umso dramatischer, je höherdimensionaler der Eingangsraum (der Regelprämissen) ist. Diese Ansätze leiden also erheblich unter dem Fluch der Dimensionalität.

4.2 Optimierung paralleler Modellstrukturen

Lokale Modellnetze mit paralleler Modellstruktur[1] haben den Vorteil, dass die Gültigkeitsfunktionen *parallel* ausgerechnet werden können. Dies kann insbesondere für industrielle Anwendungen bei entsprechender Hardware von großem Nutzen sein. Die folgenden Ausführungen erklären, wie man mittels Gaußfunktionen eine parallele Architektur konstruiert. Nähere Betrachtung findet die Unterscheidung zwischen achsenorthogonaler und achsenschräger Partitionierung. Diese Differenzierung ist insbesondere bei parallelen Strukturen bedeutsam, weil sich je nach Konfiguration unerwünschte Seiteneffekte durch die Normierung der Zugehörigkeitsfunktionen ergeben können.

4.2.1 Architektur paralleler Modellstrukturen

Üblicherweise setzt man bei parallelen Modellstrukturen Gauß'sche Zugehörigkeitsfunktionen ein. Folglich müssen zwei Komponenten bestimmt werden, um eine Zugehörigkeitsfunk-

[1] Gelgentlich auch als flache Modellstrukturen bezeichnet.

tion eindeutig zu definieren: Einerseits bestimmt das Zentrum $\underline{c}_i$ der Gaußfunktion zugleich das Zentrum des jeweiligen lokalen Modells. Andererseits wird die Skalierung und Ausrichtung der Zugehörigkeitsfunktionen durch die Kovarianzmatrix $\underline{\Sigma}_i$ vorgegeben. Damit ergibt sich die Zugehörigkeitsfunktion $\mu_i(\cdot)$ zu:

$$\mu_i\left(\underline{u}\right) = \exp\left(-\frac{1}{2}||\underline{u} - \underline{c}_i||^2_{\underline{\Sigma}_i}\right) = \exp\left(\sqrt{\left(\underline{u} - \underline{c}_i\right)^T \underline{\Sigma}_i^{-1}\left(\underline{u} - \underline{c}_i\right)}\right) . \tag{4.2}$$

Sobald alle Zugehörigkeitsfunktionen ausgerechnet sind, können diese durch eine Normierung mit der Summe aller Zugehörigkeitsfunktionen auf eine sog. Einheitspartitionierung bzw. eine *partition of unity* überführt werden:

$$\Phi_i(\underline{u}) = \frac{\mu_i(\underline{u})}{\sum\limits_{j=1}^{M} \mu_j(\underline{u})} . \tag{4.3}$$

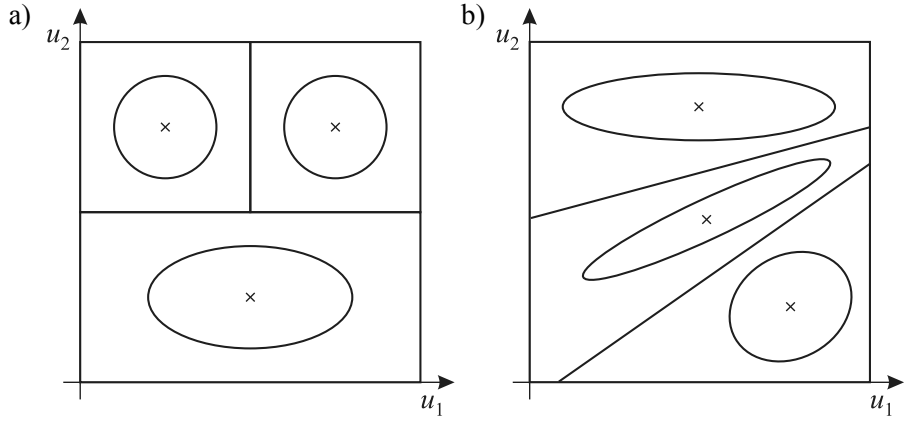

Bild 4.3: Die Kreise und Ellipsen entsprechen den Höhenlinien der mehrdimensionalen Zugehörigkeitsfunktionen μ_i, bevor diese normiert werden. Im Falle einer lediglich diagonal besetzten Kovarianzmatrix ergibt sich eine achsenorthogonale Partitionierung (a). Andernfalls erzeugt eine voll besetzte, symmetrische Kovarianzmatrix eine achsenschräge Partitionierung (b).

Je nach Wahl der Kovarianzmatrizen lassen sich zwei Fälle für die Partitionierung des Eingangsraums unterscheiden, siehe Bild 4.3:

1. *Achsenorthogonale Partitionierung:* Bei einer Unterteilung des Eingangsraums in (Hyper-)Rechtecke spricht man von einer achsenorthogonalen Partitionierung. Die Besonderheit dabei ist, dass die Kovarianzmatrix lediglich auf der Diagonalen mit den Varianzen $\sigma^2_{i,1}, \ldots, \sigma^2_{i,p}$ der jeweiligen Raumrichtung besetzt ist:

$$\underline{\Sigma}_i = \begin{bmatrix} \sigma^2_{i,1} & 0 & \cdots & 0 \\ 0 & \sigma^2_{i,2} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \sigma^2_{i,p} \end{bmatrix} . \tag{4.4}$$

   Charakterisierend für diese Wahl der Zugehörigkeitsfunktionen ist, dass sie sich dann komplett durch Vorgabe von p Komponenten des Zentrumsvektors $c_{i,j}$ und p Varianzen $\sigma^2_{i,j}$ beschreiben lassen, d.h. mit insgesamt $2p$ Parametern, wenn p die Anzahl

der Eingangsgrößen ist. Bei neuronalen Netzen wird oft das Tensorprodukt mit eindimensionalen Zugehörigkeitsfunktionen als Konstruktionsprinzip genutzt:

$$\mu(u_1) \cdot \mu(u_2) \cdot \ldots \cdot \mu(u_p) = \mu(u_1, u_2, \ldots, u_p)\,. \tag{4.5}$$

Daraus resultiert dann eine Partitionierung mit Gitterstruktur.
Ein großer Vorteil der achsenorthogonalen Formulierung ist die Interpretierbarkeit als Fuzzy-System. Vom Konstruktionsprinzip herrührend lassen sich die Gültigkeitsregionen auf die eindimensionalen Achsen ohne Informationsverlust projizieren und im Zuge dessen als Fuzzy-Regeln interpretieren. Darüber hinaus können bei dieser Art der Partitionierung unerwünschte Normierungseffekte und das Extrapolationsverhalten des Modells durch geeignete Strategien verbessert werden [138, 116]. Bei Clustering oder datenpunktbasierten Methoden entfällt dieser Vorteil. Bekannte Strategien, die achsenorthogonale Partitionierungen verwenden, sind beispielsweise CART [20], MARS [37] und LOLIMOT [120], siehe auch [143, 82].

2. *Achsenschräge Partitionierung:* Mit einer voll besetzten, symmetrischen Kovarianzmatrix $\underline{\Sigma}_i$ lassen sich achsenschräge Unterteilungen realisieren. Bekannte Partitionierungsstrategien sind z.B. Gustafson-Kessel Clustering [41] und Gath-Geva Clustering [39]. Fuzzy-Modelle mit achsenschräger Partitionierung zeichnen sich durch eine hohe Flexibilität aus, wodurch die Anzahl der benötigten Cluster beziehungsweise lokalen Modelle gegebenenfalls stark reduziert werden kann. In diesem Fall geht jedoch die Interpretierbarkeit als Fuzzy-System verloren, da sich die beliebig ausgerichteten Gültigkeitsfunktionen nicht verlustfrei auf die Eingangsachsen projizieren lassen.

4.2.2 Seiteneffekte der Normierung

Die Normierung in (4.3) kann zu unerwarteten und unerwünschten Effekten führen, die beispielsweise in [138] und [116] aufgezeigt werden. Im Falle einer diagonalen Kovarianzmatrix tritt dieser Effekt *nicht* auf, wenn alle Zugehörigkeitsfunktionen die gleiche Standardabweichung haben, d.h.: $\sigma_{1j} = \sigma_{2j} = \ldots = \sigma_{Mj}$. Die unerwünschten Seiteneffekte kommen zustande, wenn die Aktivität der Zugehörigkeitsfunktion mit der größten Standardabweichung für große Eingangswerte alle anderen Zugehörigkeitsfunktionen übertrifft, siehe Bild 4.4. Dann führt die Normierung dazu, dass die breiteste Zugehörigkeitsfunktion in unerwarteten Bereichen *reaktiviert*. Dadurch verliert die Gültigkeitsfunktion ihre Lokalität und das entsprechende lokale Modell reaktiviert bei Eingangswerten, die ursprünglich anderen lokalen Modellen zugeordnet wurden. Dieser Effekt führt zu Eigenschaften, die man nicht erwartet und bei lokalen Modellnetzen unerwünscht sind.

Im höherdimensionalen Fall verschärfen sich die Probleme durch die Normierung. Bei Verwendung einer achsenorthogonalen Partitionierung lassen sich zwar die Normierungseffekte vermeiden, indem beispielsweise die Projektionen auf die Eingangsachsen analysiert werden. Wenn es sich jedoch um skalierte und rotierte Zugehörigkeitsfunktionen handelt, kann die Normierung zu äußerst undurchsichtigen Reaktivierungen im Eingangsraum führen. Die Seiteneffekte der Normierung führen typischerweise nicht zu einer signifikanten Verschlechterung des Modells, allerdings sind sie sehr störend bezüglich der Interpretierbarkeit der Partitionierung. Bild 4.5 zeigt einen Vergleich zwischen normierten Gaußfunktionen und Sigmoiden im Zweidimensionalen und unterstreicht die Problematik der Normierung. Es

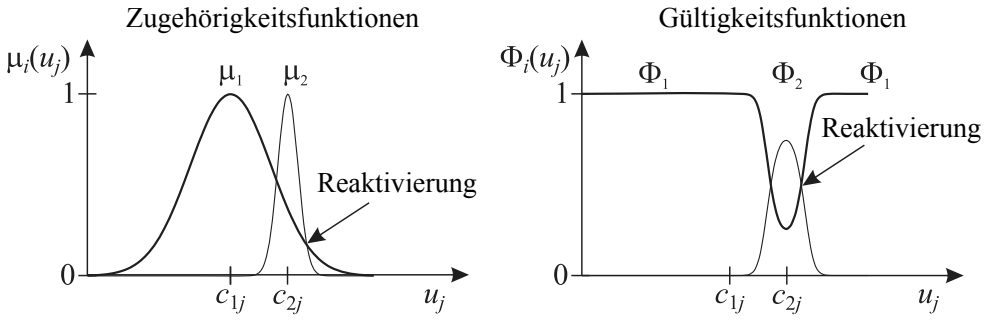

Bild 4.4: Wenn benachbarte Gaußfunktionen mit unterschiedlicher Standardabweichung normiert werden, kann dies zu einer Reaktivierung der breiteren Gaußfunktion führen.

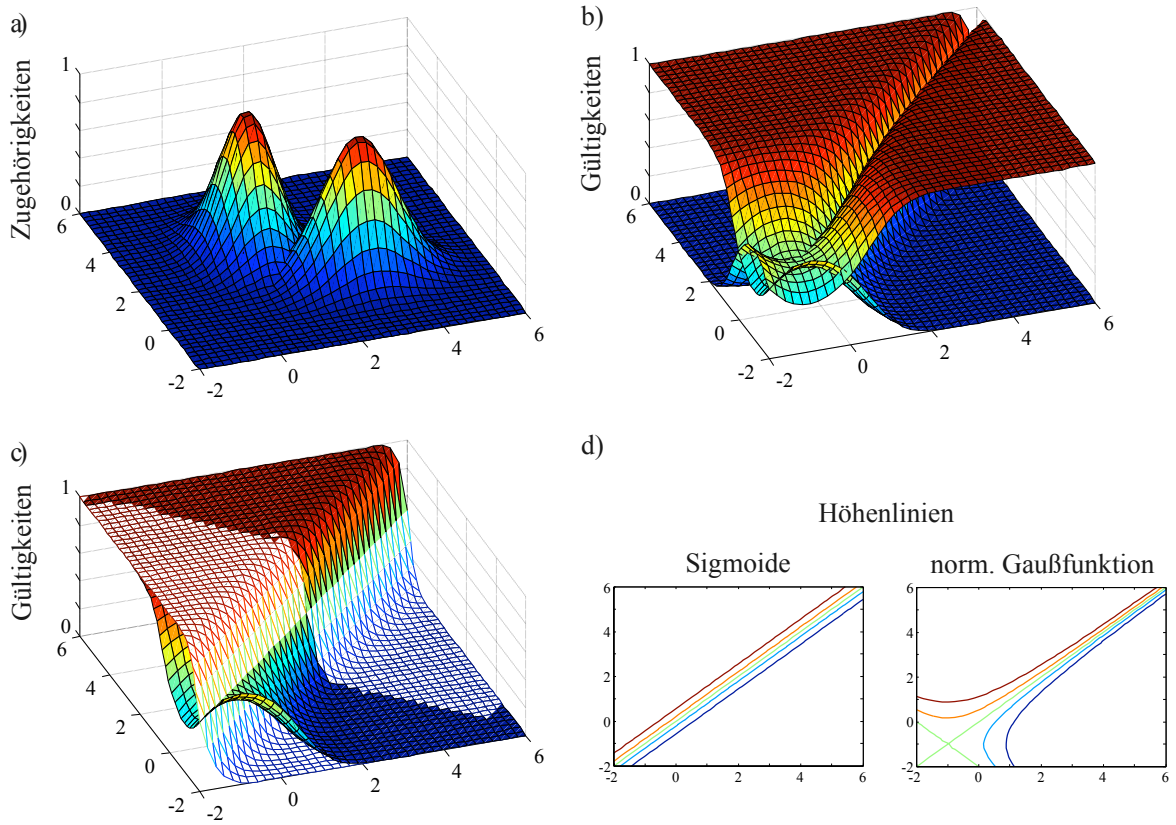

Bild 4.5: a) Zwei gaußförmige Zugehörigkeitsfunktionen. b) Durch Normierung ergeben sich die Gültigkeitsfunktionen $\Phi_i(\cdot)$. c) Vergleich der Gültigkeitsfunktionen in normierter Gauß- bzw. Sigmoid-Form. d) Deren Höhenlinien. Durch ungewünschte Reakivierungseffekte sind die normierten Gaußfunktionen schlechter geeignet.

bietet sich daher an, jede Teilung des Eingangsraums z.B. mit einer Sigmoiden zu beschreiben und, wie in Kapitel 4.3.1 erläutert wird, die endgültigen Gültigkeitsfunktionen hierarchisch aufzubauen.

4.2.3 Training paralleler Modellstrukturen

An dieser Stelle werden Trainingsalgorithmen für lokale Modellnetze vorgestellt, welche Gauß'sche Zugehörigkeitsfunktionen verwenden. Die resultierende Struktur der Modelle ist demnach parallel. Auch wenn die hier behandelten Baumkonstruktionsverfahren eine hierarchische Teilungsstrategie verfolgen, bleibt das Konstruktionsprinzip der Partitionierung parallel. Zunächst wird der sogenannte LOLIMOT-Algorithmus vorgestellt. Dieses Verfahren bildet die Basis der in dieser Arbeit durchgeführten Weiterentwicklungen. Die Einfachheit dieses Ansatzes durch die achsenorthogonale Partitionierung ist gleichzeitig der größte Nachteil des Verfahrens, wenn die Modellierungsprobleme hochdimensionaler werden. Der in dieser Arbeit entwickelte SUHICLUST-Algorithmus bringt deutlich mehr Flexibilität in die Partitionierung und ermöglich daher effizientere Modelle.

LOLIMOT-Algorithmus

Die Abkürzung LOLIMOT steht für „*LOcal LInear MOdel Tree*". Der LOLIMOT-Algorithmus erzeugt lokal lineare Neuro-Fuzzy-Modelle, sowohl für statische als auch für dynamische Prozesse. Die ausführliche Beschreibung des Verfahrens ist in [116] zu finden. Hier wird lediglich die Grundidee der Methode angesprochen.

Die Funktionsweise ist anhand vier beispielhafter Iterationen von LOLIMOT in Bild 4.6 illustriert. Statt die Lage und Ausrichtung der Gültigkeitsfunktionen mit nichtlinearen Verfahren explizit zu optimieren, arbeitet das Verfahren nach einem rein heuristischen „teile und herrsche"-Prinzip. In jeder Iteration wird dem Gesamtmodell durch Teilung des schlechtesten lokalen Modells (LM) ein LM hinzugefügt und die Parameter der zwei neu entstandenen LM mit einer gewichteten lokalen LS-Schätzung optimiert. Dies geschieht nach folgendem Schema:

1. *Initialisierung:* Zunächst startet LOLIMOT mit einem global linearen Modell.
2. *Bestimme schlechtestes lokales Modell:* Für jedes LM wird die gewichtete Summe der Fehlerquadrate als lokale Verlustfunktion berechnet, d.h.:

$$J_{i,\text{lokal}} = \sum_{k=1}^{N} \Phi_i(\underline{z}(k))\, e^2(k)\,. \tag{4.6}$$

   Anhand der lokalen Verlustfunktionen wird das schlechteste lokale Modell mit $\max(J_{i,\text{lokal}})$ bestimmt. Dem schlechtesten lokalen Modell weist man den Index l zu.
3. *Teste Teilungen in jeder Eingangsdimension:* Das schlechteste LM l wird in zwei Hälften geteilt. Der Schnitt verläuft dabei immer achsenorthogonal durch das Zentrum des lokalen Modells. Für alle Raumrichtungen $dim = 1, \ldots, p$ findet eine Teilung und die Berechnung des globalen Modellfehlers mit dem jeweiligen Schnitt statt.
4. *Finde besten Schnitt:* Von den p durchgeführten Teilungsalternativen wird diejenige für das Modell übernommen, welche auf den kleinsten globalen Modellfehler führt. Dadurch erhöht sich die Anzahl der LM um Eins, d.h.: $M \leftarrow M + 1$.

5. *Überprüfe Abbruchbedingung:* Wenn die Abbruchbedingung erfüllt ist, findet keine weitere Teilung mehr statt und der Algorithmus ist beendet. Ansonsten gehe zu Schritt 2.

Zum Abbruch des Algorithmus sind mehrere alternative Kriterien möglich, beispielsweise die maximale LM-Anzahl, ein geforderter Modellfehler, verschiedene Validierungstechniken oder statistische Informationskriterien. Eine ausführliche Diskussion dazu findet sich in den Kapiteln 3.5 und 4.4.

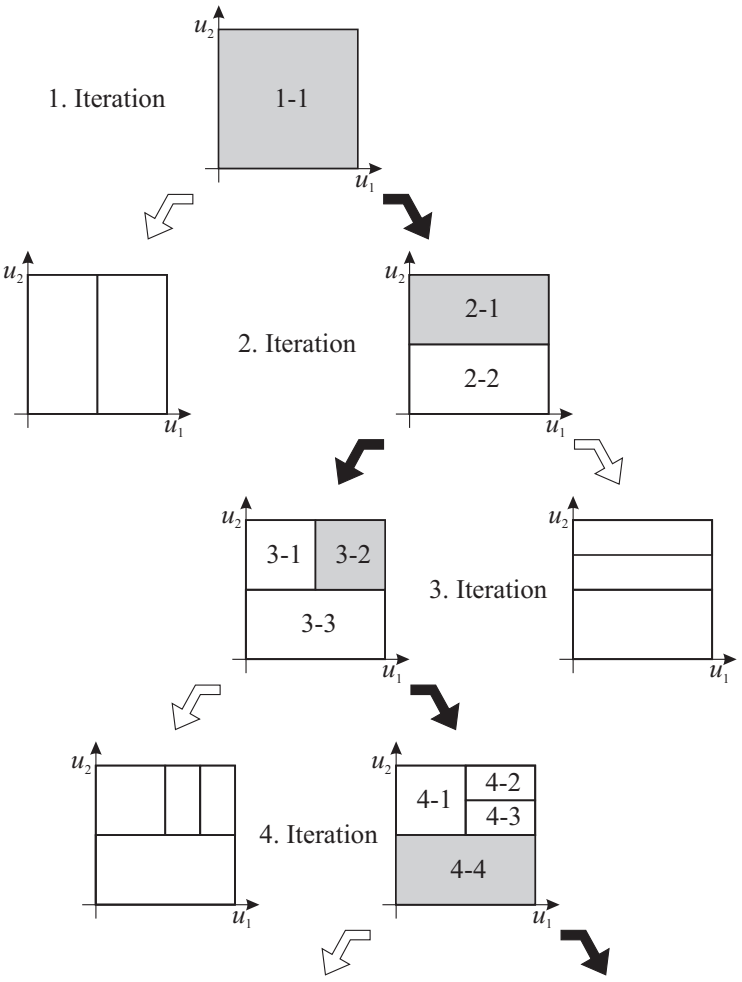

Bild 4.6: Vier mögliche Iterationen von LOLIMOT [116].

Die Gültigkeitsbereiche jedes lokalen Modells werden durch normierte Gauß'sche Zugehörigkeitsfunktionen μ_i definiert. Diese sind achsenorthogonal ausgerichtet und durch folgende nichtlineare Parameter eindeutig bestimmt: den Zentrumsvektor $\underline{c}_i$ und die Kovarianzmatrix $\underline{\Sigma}_i$. Die Kovarianzmatrix $\underline{\Sigma}_i$ ist wegen der Orthogonalität nur auf der Diagonalen mit den Standardabweichungen σ_{ij} des i-ten LM in die j-te Raumrichtung besetzt. Wie in Bild 4.7 dargestellt, ist die Breite der LM mit dem Parameter Δ_{ij} festgelegt. Um die Glattheit der Modellübergänge einstellen zu können, werden die Δ_{ij} mit einem Proportionalitätsfaktor k_σ multipliziert, der standardmäßig den Wert $k_\sigma = 1/3$ annimmt. Dann ist $\sigma_{ij} = k_\sigma \cdot \Delta_{ij}$.

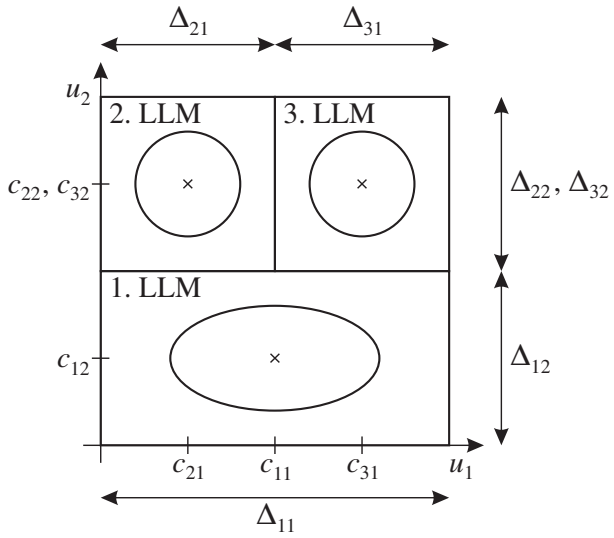

Bild 4.7: Festlegung der Parameter einer zweidimensionalen Partitionierung. Die Kreise und Ellipsen repräsentieren die Höhenlinien der Zugehörigkeitsfunktionen μ_i [116] (LLM = Lokal Lineares Modell).

SUHICLUST-Algorithmus

Die Abkürzung SUHICLUST steht für „*SUpervised HIerarchical CLUSTering*“. Die Idee zur Entwicklung und Implementierung des neuen SUHICLUST-Algorithmus entstand im Rahmen dieser Arbeit als Folge eines Vergleichs zwischen HILOMOT (siehe Kapitel 4.3.2), LOLIMOT und Produktraum-Clustering. Wie in Bild 4.8 gezeigt, handelt es sich bei SUHICLUST um die Kombination aus heuristischem Baumkonstruktionsverfahren und Produktraum-Clustering.

Die Entwicklung des Verfahrens wurde durch Folgendes motiviert: LOLIMOT hat den Vorteil eines überwachten Lernverfahrens, allerdings den Nachteil von weniger flexiblen Gültigkeitsfunktionen infolge der achsenorthogonalen Teilungen. Produktraum-Clustering ist ein unüberwachtes Lernverfahren. Nachteilig hierbei ist, dass die Ergebnisse nicht reproduzierbar sind und von zufälligen Initialisierungen abhängen. Allerdings bietet Produktraum-Clustering mit achsenschräger Projektion der Cluster vom Produktraum in den Eingangsraum den Vorteil einer sehr flexiblen Partitionierung, welche insbesondere bei hochdimensionalen Zusammenhängen nötig ist, um den „Fluch der Dimensionalität“ zu umgehen.

In den Publikationen [47, 147, 60, 59] findet sich eine detaillierte Beschreibung und Analyse des SUHICLUST-Algorithmus'. Das Grundprinzip ist das gleiche wie bei LOLIMOT. In jeder Iteration von SUHICLUST wird dem Gesamtmodell lediglich ein einzelnes lokales Modell hinzugefügt. Dabei wird jeweils das lokale Teilmodell mit dem höchsten lokalen Fehlermaß weiter unterteilt. Die Besonderheit des Verfahrens ist, dass hierbei die Unterteilung durch Gustafson-Kessel Produktraum-Clustering erfolgt. Die ersten drei Iterationen von SUHICLUST sind schematisch in Bild 4.9 illustriert.

Die Bilder 4.10 und 4.11 zeigen ein Beispiel, welches insbesondere die Flexibilität des neuen Algorithmus' hervorhebt. Die Schnitte erfolgen orthogonal zur Richtung der Nichtlinearität des Beispielprozesses. Dieser Vorteil spielt vor allem im Hochdimensionalen eine entscheidende Rolle, wie Bild 4.12 veranschaulicht. Während LOLIMOT im 4D-Fall 59 lokal lineare

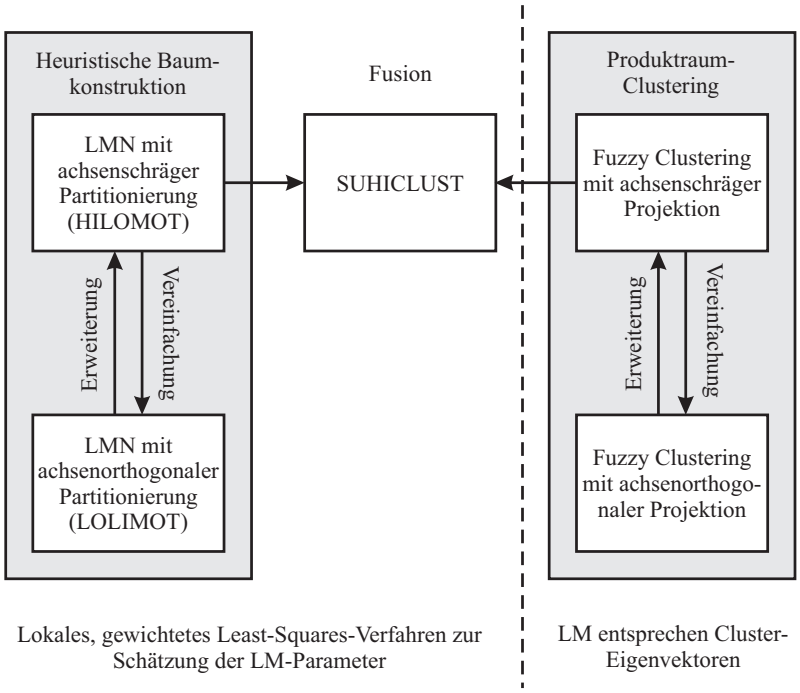

Bild 4.8: SUHICLUST ist die Kombination aus heuristischem Baumkonstruktionsverfahren und Produktraum-Clustering. Die LM-Parameter werden mit einem gewichteten Least Squares-Verfahren (WLS) geschätzt.

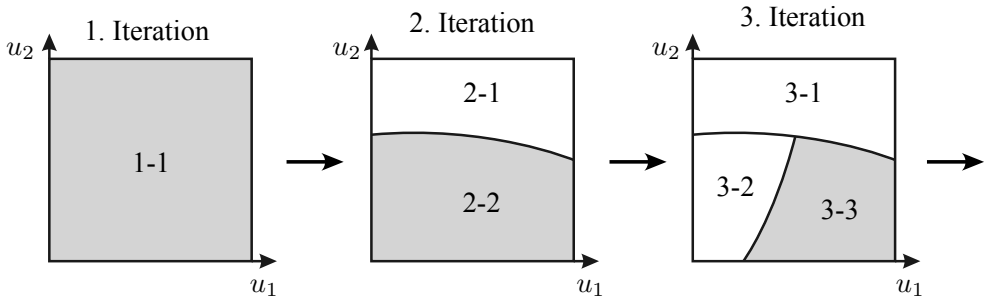

Bild 4.9: Arbeitsweise des Struktur-Suchalgorithmus' SUHICLUST in den ersten drei Iterationen für einen zweidimensionalen Eingangsraum.

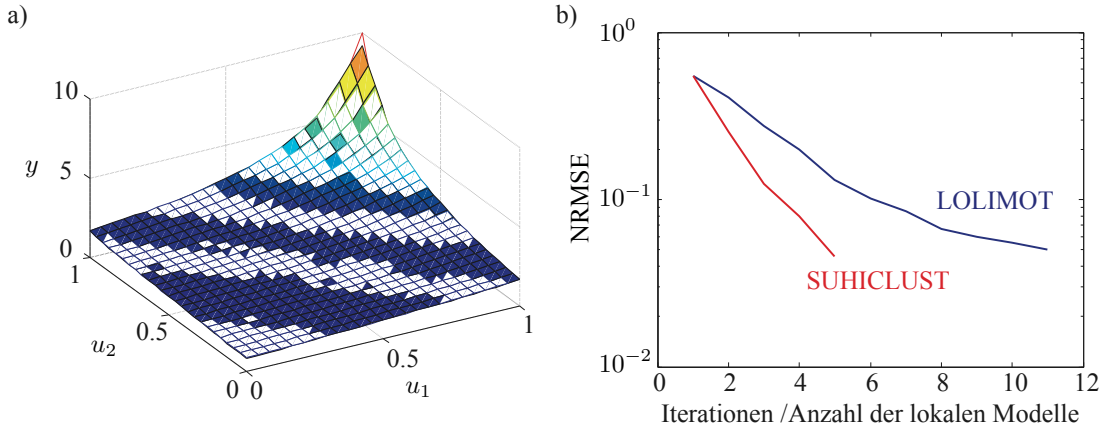

Bild 4.10: a) Prozess (hell) im Vergleich zum Modellausgang erzeugt durch SUHICLUST (dunkel). b) Konvergenzverhalten von LOLIMOT und SUHICLUST.

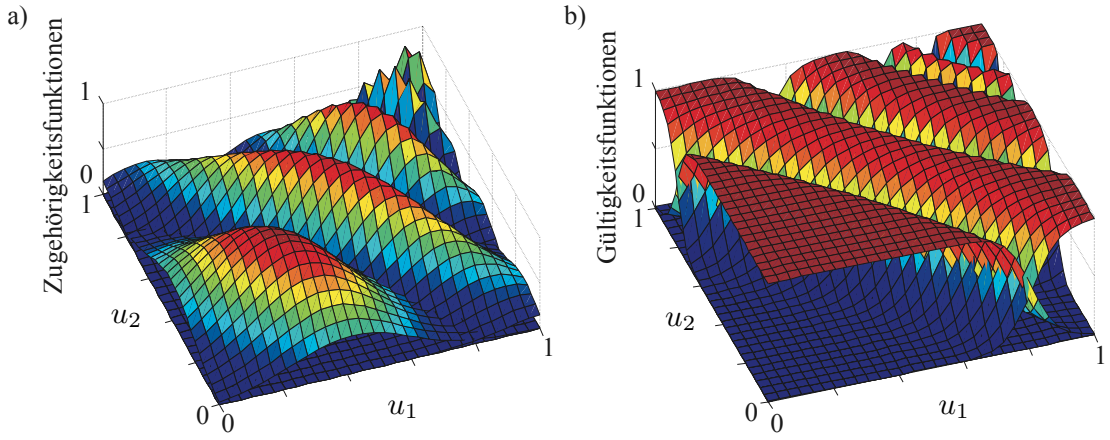

Bild 4.11: Zugehörigkeitsfunktionen (a) und Gültigkeitsfunktionen (b) des SUHICLUST-Modells.

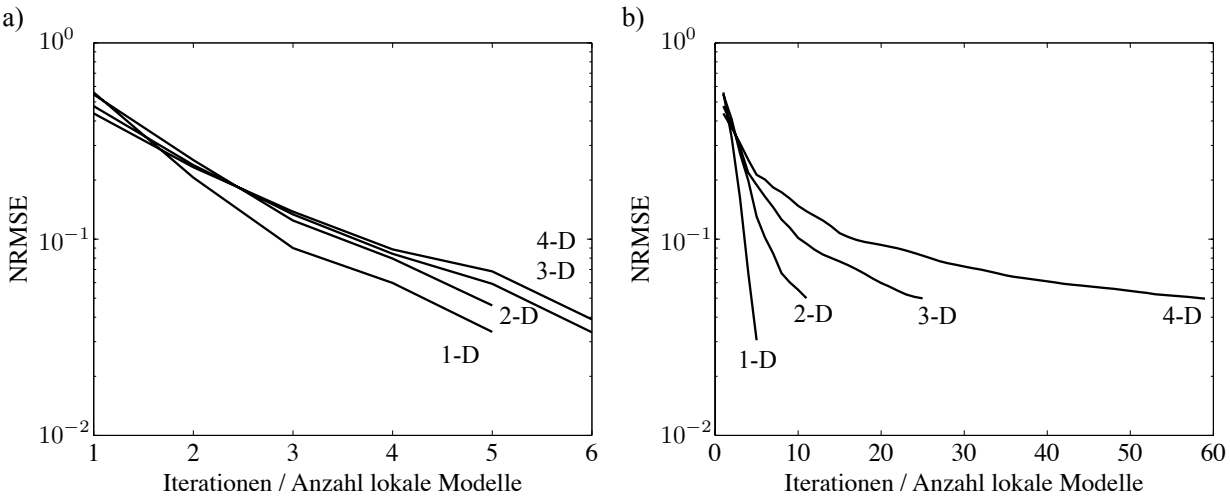

Bild 4.12: Konvergenzverhalten für ein 1-, 2-, 3- und 4-dimensionales Approximationsproblem, einmal für eine achsenschräge (a) und einmal für eine achsenorthogonale Partitionierungsstrategie (b).

Modelle benötigt, um einen Trainingsdatenfehler unter 5% zu erzielen, reichen beim Training mit SUHICLUST 6 lokale Modelle zur Erfüllung des gleichen Ziels aus.

Jedoch ergibt sich bei dem Verfahren die folgende Schwierigkeit: Durch die voll besetzte symmetrische Kovarianzmatrix ist eine hohe Anzahl an Parametern zu schätzen. Daher muss für eine gut konditionierte Schätzung gewährleistet sein, dass eine ausreichende Anzahl an Datenpunkten im lokalen Modell vorhanden ist. Entsprechende Strategien müssen daher im Algorithmus berücksichtigt werden.

4.3 Optimierung hierarchischer Modellstrukturen

4.3.1 Hierarchisches Konstruktionsprinzip

Ein wesentlicher Grund für die Effizienz der Baumkonstruktionsverfahren bezüglich ihres Rechenaufwands liegt in der Ausnutzung der Lokalitätseigenschaften des Modells. So bleiben z.B. bei LOLIMOT in jeder Iteration alle bereits vorgenommenen Unterteilungen bestehen. Nur für die neu durchzuführende Unterteilung ist die beste Alternative zu finden. Da bei LOLIMOT jede Unterteilung achsenorthogonal und halbierend (also im Verhältnis 1:1) durchgeführt wird, verzerren sich die Gültigkeitsfunktionen der anderen, unveränderten lokalen Modelle durch die neu durchzuführende Normierung in (4.3) nur unwesentlich. Bei einer optimierten achsenschrägen Unterteilung ist dies leider nicht mehr der Fall. Durch die in Kapitel 4.2.2 beschriebenen unerwünschten Seiteneffekte der Normierung [138] besteht die Gefahr, dass sich die Gültigkeitsfunktionen der anderen lokalen Modelle signifikant ändern. So kann beispielsweise vorkommen, dass eine Gültigkeitsfunktion mit besonders großem Gültigkeitsbereich reaktiviert wird. Der ausschließlich lokale Einfluss der lokalen Modelle ist dann nicht mehr sicher gestellt.

Da bei einer Optimierung der achsenschrägen Unterteilung solche Effekte verstärkt auftreten, muss von dem Konzept abgegangen werden, die „Partition of Unity“ mittels der Normierung in (4.3) herzustellen. Ein alternatives Konzept zur Vermeidung der genannten Schwierigkeiten, das bislang in der Literatur nur selten verfolgt wird, ist die Verwendung hierarchischer Modellstrukturen. In diesem Kapitel wird eine solche hierarchische Modellstruktur vorgestellt und mit der konventionellen parallelen (flachen) Struktur verglichen.

Baumstruktur der Partitionierung

Bei einer hierarchischen Modellstruktur wird nicht nur der Konstruktionsalgorithmus, sondern auch das Modell selbst durch einen Baum repräsentiert, siehe Bild 4.13. Jeder Verzweigungsknoten i des Baums symbolisiert eine Unterteilung des Eingangsraums in zwei Bereiche, welche durch die Teilungsfunktionen $\Psi_i(\cdot)$ und $\widetilde{\Psi}_i(\cdot)$ beschrieben werden. Diese komplementären Teilungsfunktionen ergänzen sich jeweils zu Eins:

$$\Psi_i(\underline{z}) + \widetilde{\Psi}_i(\underline{z}) = 1\,. \tag{4.7}$$

Die Bereiche werden jeweils solange weiter unterteilt, bis ein Blatt des Baums erreicht ist. Jedes Blatt repräsentiert ein lokales Modell und dessen Beitrag zum Gesamtmodell,

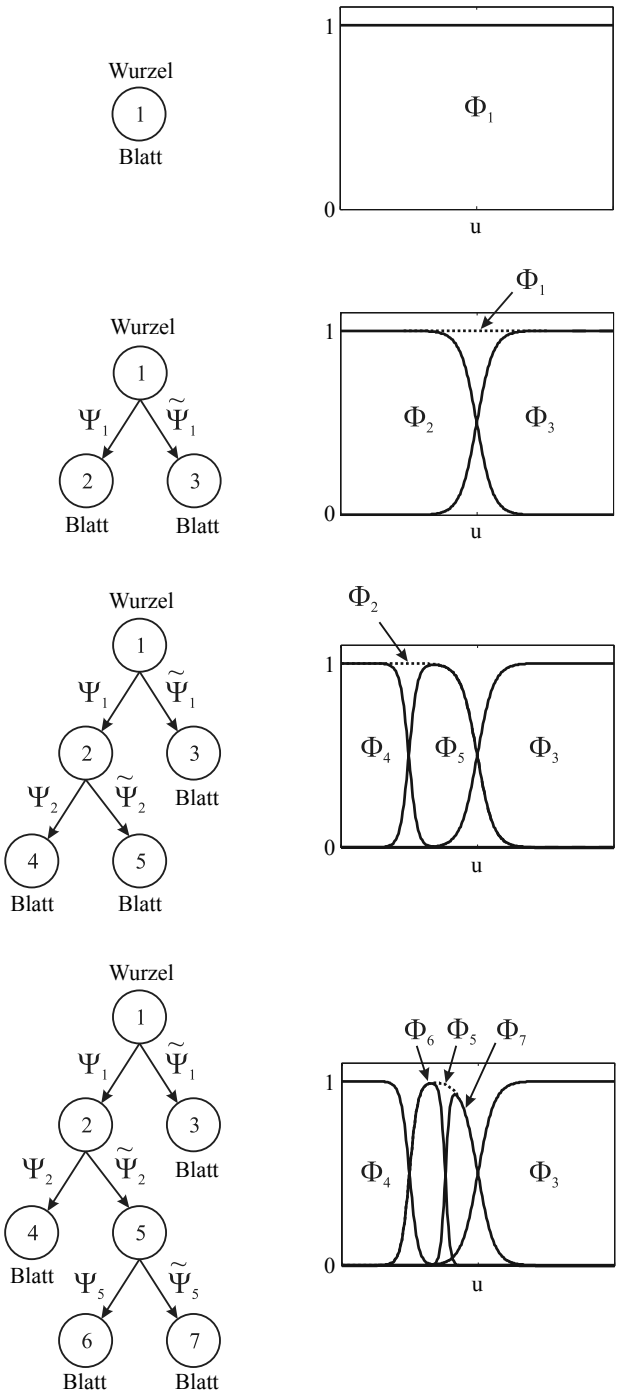

Bild 4.13: Fünf mögliche Iterationen zur Konstruktion eines hierarchischen Teilungsbaums (links). Die Gültigkeitsfunktionen Φ_i (rechts) entstehen, indem die Teilungsfunktionen Ψ_i bzw. $\widetilde{\Psi}_i$ bis zum jeweiligen Blatt miteinander multipliziert werden.

beschrieben durch dessen Gültigkeitsfunktion, lässt sich aus der Multiplikation aller Teilungsfunktionen zwischen der Wurzel und dem Blatt berechnen. Für den fertig konstruierten Baum in Bild 4.13 (unten) bedeutet dies beispielsweise:

$$\begin{aligned}
\Phi_3(\underline{z}) &= \widetilde{\Psi}_1(\underline{u}), \\
\Phi_4(\underline{z}) &= \Psi_1(\underline{z})\Psi_2(\underline{z}), \\
\Phi_6(\underline{z}) &= \Psi_1(\underline{z})\widetilde{\Psi}_2(\underline{z})\Psi_5(\underline{z}), \\
\Phi_7(\underline{z}) &= \Psi_1(\underline{u})\widetilde{\Psi}_2(\underline{z})\widetilde{\Psi}_5(\underline{z}).
\end{aligned}$$

Mit dieser Methode kann die hierarchische Modellstruktur in eine äquivalente parallele Modellstruktur umgerechnet werden. Durch die Bedingung (4.7) wird automatisch eine „Partition of Unity“ erzeugt, ohne dass eine Normierung notwendig ist. Damit ist garantiert, dass sich eine Unterteilung in einem Unterbaum nicht auf den restlichen Baum auswirkt, wodurch die Lokalität sichergestellt ist.

Sobald die Gültigkeitsfunktionen auf diese Art und Weise berechnet sind, kann der Ausgang des Gesamtmodells als gewichtete Summe der lokalen Modelle formuliert werden:

$$\hat{y} = \sum_{i \in \mathcal{L}} \hat{y}_i(\underline{x})\Phi_i(\underline{z})\,, \tag{4.8}$$

wobei $\mathcal{L}$ die Menge der Blatt-Knoten des Teilungsbaums darstellt. Für das Beispiel in Bild 4.13 ergibt sich $\mathcal{L} = \{3, 4, 6, 7\}$.

Vergleich mit parallelen Modellstrukturen

Die wichtigsten Vorteile einer parallelen Modellstruktur sind:

- Zur Realisierung einer parallelen Modellstruktur in der Praxis ist es sehr vorteilhaft, dass in diesem Fall die Gültigkeitsfunktionen parallel ausgerechnet werden können. Somit kann eine parallele Rechnerarchitektur voll ausgenutzt werden.
- Ein paralleles Modell ist hinsichtlich einer „Fuzzy Logik“ gut interpretierbar, vorausgesetzt, es handelt sich um eine achsenorthogonale Partitionierung. Die Gauß'schen Zugehörigkeitsfunktionen können ohne Informationsverlust auf die Eingangsachsen projiziert werden, sodass eine regelbasierte Interpretation möglich ist.
- Ein weiterer Vorteil ist, dass die Partitionierung unter Verwendung einer parallelen Modellstruktur unabhängig von der Reihenfolge der Teilung ist.

Allerdings birgt die parallele Struktur auch Nachteile:

- Jede Unterteilung des Eingangsraums hat Auswirkung auf alle existierenden Gültigkeitsfunktionen. Das bedeutet, dass streng genommen alle lokalen Modelle nach jeder Unterteilung neu geschätzt werden müssten. Bild 4.14a greift diesen Effekt auf. Nach jeder Iteration des Partitionierungsalgorithmus' verändert sich die Form der rechten Gültigkeitsfunktion, obwohl die eigentliche Teilung in einer anderen Region des Eingangsbereiches vorgenommen wurde. Die Unterschiede zwischen paralleler und hierarchischer Partitionierung sind in Bild 4.14b gezeigt. Die Glattheit wurde so eingestellt, dass die Gültigkeitsfunktionen von paralleler und hierarchischer Struktur nach dem ersten Schnitt miteinander möglichst gut übereinstimmen.

- Außerdem müssen die Zugehörigkeitsfunktionen aufgrund der parallelen Struktur *normiert* werden. Daher kann es insbesondere bei achsenschrägen Teilungen zu unerwünschten Seiteneffekten der Normierung kommen. Dazu gehören die Reaktivierung von Gültigkeitsfunktionen weit außerhalb des ursprünglichen Gültigkeitsbereichs eines Teilmodells sowie die Verzerrung der Form und die Verschiebung der Maxima gegenüber der nicht normierten Aktivierungsfunktion. Eine ausführliche Beschreibung dieser Problematik findet man in [138].
- Ein weiterer Nachteil ist, dass für achsenschräge Teilungen eine voll-symmetrische Kovarianzmatrix erforderlich ist. Daher sind viele Parameter, d.h. Varianzen und Kovarianzen, zu schätzen, falls keine achsenorthogonale Partitionierung vorgenommen wird.

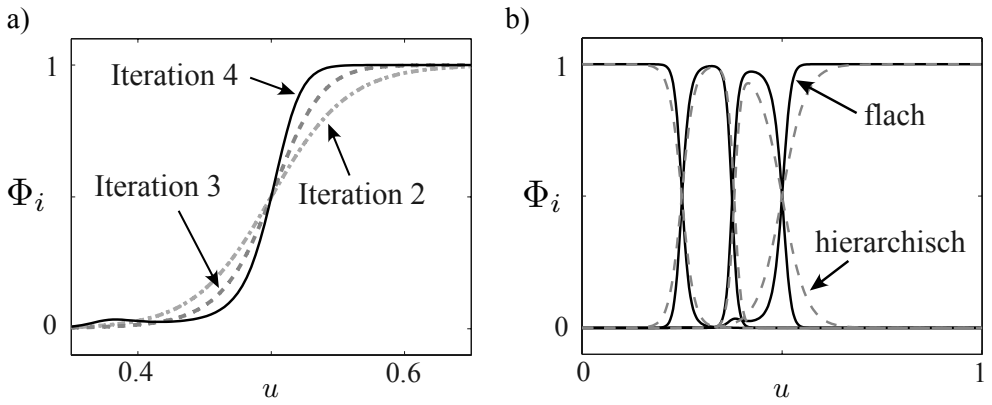

Bild 4.14: a) Im Falle einer parallelen (bzw. flachen) Modellstruktur verändert sich die Form der rechten Gültigkeitsfunktion, obwohl die Unterteilungen in einer anderen Region des Eingangsraums stattfinden. b) Vergleich der Parititionierung mit paralleler und mit hierarchischer Modellstruktur.

Bei einer hierarchischen Modellstruktur ergeben sich hingegen folgende Vorteile:

- Der hierarchische Ansatz erlaubt eine Partitionierung ohne Normierung. Die „Partition of Unity“ $\sum_{i=1}^{M} \Phi_i(\underline{z}) = 1$ ist automatisch erfüllt, wenn die Bedingung $\Psi_i(\underline{z}) + \widetilde{\Psi}_i(\underline{z}) = 1$ bei der hierarchischen Struktur eingehalten wird. Daraus folgt, dass alle angesprochenen negativen Seiteneffekte infolge der Normierung vermieden werden, da die Normierung nicht mehr notwendig ist [138, 116].
- Weiterhin wird durch die Einsparung der Normierung die Rechenzeit verkürzt und numerischen Problemen aus dem Weg gegangen. Sigmoidale Funktionen erfüllen alle erforderlichen Eigenschaften als Unterteilungsfunktionen und sind daher gut geeignet für achsenschräge Partitionierungsansätze.
- Die Anzahl der Parameter ist im Vergleich zu normierten Gaußfunktionen reduziert.
- Bei hierarchischen Gültigkeitsfunktionen ist die Lokalität garantiert. Das bedeutet, dass ganze Modell-Unterbäume verändert oder ausgetauscht werden können, ohne einen Effekt auf andere Bereiche des Baumes zu haben.

Die Lösung der Probleme einer parallelen Struktur führt allerdings auf neue Herausforderungen bei der Anwendung der hierarchischen Modellstruktur:

- Eine parallelisierte Berechnung des Modells ist wegen des hierarchischen Baumprinzips nicht möglich.
- Zudem wird die Partitionierung von der Reihenfolge der Teilungen beeinflusst. Je nachdem, in welcher Reihenfolge geteilt wird, ändert sich auch die Partitionierung.
- Die richtige Einstellung der Interpolation zwischen den lokalen Modellen stellt eine Herausforderung dar. Die Steilheit der Teilungsfunktionen muss idealerweise die Hierarchiestufe im Teilungsbaum mit berücksichtigen. Diese ist jedoch vorab nicht bekannt. In Anhang C wird dieser Sachverhalt ausführlich diskutiert und eine Lösung dies bezüglich vorgeschlagen.

Der Vergleich der Modellstrukturen und die positiven Erfahrungen mit dem hierarchischen Ansatz, die in dieser Arbeit insbesondere bei hochdimensionalen Problemen und achsenschräger Partitionierung gemacht wurden, zeigen jedoch, dass die Anwendung der hierarchischen Struktur bei achsenschrägen Partitionierungsansätzen sehr vorteilhaft ist.

4.3.2 HILOMOT-Algorithmus

Im Rahmen dieser Arbeit ist die MATLAB-Toolbox HILOMOT entstanden. HILOMOT steht für „*HIerarchical LOcal MOdel Tree*“. Es handelt sich dabei um einen konstruktiven Strukturoptimierungsalgorithmus zur automatischen Generierung von lokalen Modellnetzen. HILOMOT basiert auf dem LOLIMOT-Algorithmus, der in Kapitel 4.2.3 beschrieben wird. Die Grundidee von HILOMOT ist, den Eingangsraum wie bei LOLIMOT durch immer weitere Unterteilungen in seine Gültigkeitsbereiche zu unterteilen. Anders als beim LOLIMOT-Algorithmus werden bei HILOMOT nicht nur achsenorthogonale, sondern auch achsenschräge Schnitte vorgenommen. Dies hat den Vorteil, ein wesentlich flexibleres Modell generieren zu können. Insbesondere bei hochdimensionalen Problemen, welche in der Praxis zunehmend an Bedeutung gewinnen, kommt dieser Vorteil zur Geltung.

Motivation und Hintergrund zu achsenschrägen Partitionierungen

In [13] wurde gezeigt, dass bei vorgegebenen Modellfehlergrenzen ein notwendiges exponentielles Anwachsen der Modellkomplexität mit der Dimension des Eingangsraums nur verhindert werden kann, wenn die Parameter der verdeckten Schicht (also der Gültigkeitsfunktionen) optimiert werden. Das ist konsistent mit allen Erfahrungen, die übereinstimmend belegen, dass das Multilayer Perzeptron (MLP) von allen neuronalen Netzarchitekturen am besten für hochdimensionale Eingangsräume geeignet ist. Der Mechanismus, mit dem das MLP dem „Fluch der Dimensionalität“ entkommt, ist die Optimierung der räumlichen Ausrichtung und Positionierung der Basisfunktionen. Dies entspricht bei Netzen mit lokalen Modellen einer Optimierung der Ausrichtung und Position der Gültigkeitsfunktionen. Dieser Zusammenhang erklärt theoretisch, wieso durch eine optimierte achsenschräge Partitionierung der „Fluch der Dimensionalität“ überwunden wird. Ein evtl. hochdimensionales Problem wird dadurch auf die wirkliche Dimension der Nichtlinearität reduziert. Diese Richtungsoptimierung ist daher der Schlüssel zum Erfolg bei Problemen mit vielen Eingangsgrößen, allerdings erfordern solche Algorithmen nichtlineare Optimierungsverfahren. Hauptnachteile von MLP-Netzen sind zum einen die äußerst schlechte Interpretierbarkeit und zum anderen der große Rechenaufwand, weil die Optimierung aller Parameter

des MLP-Netzes gleichzeitig stattfindet. Ein weiterer schwerwiegender Nachteil ist, dass die Modellierungsergebnisse stark von der Initialisierung der Parameter abhängen. Meist ist die Initialisierung zufällig; die Reproduzierbarkeit der Ergebnisse ist damit sehr unwahrscheinlich. Das in dieser Arbeit vorgestellte Verfahren mit optimierter achsenschräger Partitionierung soll die Vorteile des MLPs bewahren, aber gleichzeitig die Nachteile umgehen, indem die lokale Modellnetzstruktur ausgenutzt wird.

Ein erster Schritt zur Anwendung der Idee der Richtungsoptimierung auf lokale Modellnetze wurde in [19] mit den sogenannten *Hinging Hyperplanes* getätigt. Hinging Hyperplanes erinnern an ein teilweise geöffnetes Buch, da man zwei Ebenen so miteinander kombiniert, dass die Schnittlinie der zwei Ebenen als Begrenzung zwischen den lokal linearen Modellen dient. Bildlich gesprochen entspräche die Schnittlinie dem Buchrücken. Die Lage und Richtung dieser Schnittlinie, die man im Englischen als *Hinge* bezeichnet, wird dann optimiert. Der nächste Entwicklungsschritt war es, wie in [127] gezeigt, die stückweise linearen Modelle mit Interpolationsfunktionen zu glätten. Allerdings gab es zu diesem Zeitpunkt noch die Einschränkung, dass die Eingangsräume für die lokal linearen Modelle und die Gültigkeitsfunktionen identisch konstruiert sein mussten. Damit sind die Parameter der lokalen Modelle direkt mit den Hinge-Parametern gekoppelt. Diese Eigenschaft ermöglichte zwar die Entwicklung eines effizienten Trainingsalgorithmus', aber im Kontext lokaler Modellnetze stellt die Kopplung eine starke Einschränkung dar. Darüber hinaus erlauben diese Ansätze lediglich die Verwendung lokal *linearer* Modelle. Diese Einschränkungen wurden schließlich in [34] durch die Einführung *generalisierter Hinging Hyperplanes* aufgehoben. Hierbei ist es möglich, die Eingangsräume von lokalen Modellen und Gültigkeitsfunktionen unabhängig voneinander zu betrachten. Auch in der gegenwärtigen Literatur werden diese Ansätze stetig weiterentwickelt. Beispielsweise findet in [45] ein Vergleich zwischen achsenschräg partitionierten lokalen Modellnetze und MLP-Netzen statt. Darüber hinaus werden in [44] Ansätze zur Verwendung von Expectation Maximization-Algorithmen für die Optimierung der Partitionierung und generalisierte Total Least Squares-Verfahren zur Optimierung der lokalen Modelle untersucht.

Achsenschräger Konstruktionsalgorithmus

HILOMOT ist ein Baumkonstruktionsalgorithmus, der mit dem Ansatz der Hinging Hyperplanes aus [19] sehr verwandt ist und in [34, 150, 117, 53, 50] weiterentwickelt wurde. In jeder Iteration wird zum Gesamtmodell eine neue Regel bzw. ein neues lokales Modell (LM) hinzugefügt. Daher gehört HILOMOT zu der Klasse der *inkrementellen* oder *wachsenden* Algorithmen. Die Parameter der lokalen Modelle können in einem einzigen Berechnungsschritt mit einer gewichteten Least Squares-Schätzung bestimmt werden. Die Optimierung der Parameter zur Bestimmung der achsenschrägen Schnitte ist allerdings nichtlinear und muss daher iterativ bezüglich einer geeigneten Verlustfunktion optimiert werden. Näheres zur nichtlinearen Optimierung wird in Kapitel 4.3.3 diskutiert.

Bevor der Algorithmus startet, findet eine Normierung der Eingangsgrößen auf einheitliche Standardabweichung und eine Mittelwertbefreiung statt. Generell wird bei jedem Schnitt, der während des Verfahrens getestet wird, eine lokale Schätzung der LM-Parameter mit dem gewichteten Least Squares-Verfahren durchgeführt. Der Ablauf des HILOMOT-Algorithmus ist wie folgt, siehe Bild 4.15:

1. *Initialisierung:* Starte mit einem einzigen LM. Setze die Anzahl der LM auf $M = 1$. Das LM ist gleichzeitig das globale Modell, da der Wert der Gültigkeitsfunktion über dem kompletten Eingangsraum Eins ist: $\Phi_1(\underline{z}) = 1$.
2. *Bestimme schlechtestes lokales Modell:* Berechnung der lokalen Verlustfunktion $J_{i,\text{lokal}}$ für alle $i = 1, \ldots, M$ lokalen Modelle. Als lokale Verlustfunktion dient der gewichtete, quadratische Modellfehler:

$$J_{i,\text{lokal}} = \sum_{k=1}^{N} \Phi_i(\underline{z}(k))\, e^2(k)\,. \tag{4.9}$$

   Das LM mit dem größten Wert der Verlustfunktion, d.h. $\max(J_{i,\text{lokal}})$, wird als schlechtestes LM mit dem Index l markiert.
3. *Teste verschiedene Teilungsrichtungen zur Initialisierung:* Das LM l soll nun weiter verfeinert und in zwei neue LM unterteilt werden. In jeder Eingangsraumdimension wird ein achsenorthogonaler Schnitt durch das LM-Zentrum durchgeführt und die jeweilige globale Verlustfunktion für das gesamte Modell berechnet. Falls bereits mehr als ein LM vorhanden ist, wird zusätzlich die Teilungsrichtung des Elternknotens getestet. Die beste Teilung dient als Initialisierung der nachfolgenden Optimierung.
4. *Nichtlineare Schnittoptimierung:* Beginnend beim Initialschnitt aus Schritt 3 werden Position und Richtung des neuen Schnitts bezüglich der folgenden Verlustfunktion des globalen Modells nichtlinearer optimiert (Wurzel des normierten quadratischen Fehlers, NRMSE):

$$J = \sqrt{\frac{\sum_{i=1}^{N} \left(y(i) - \hat{y}(i)\right)^2}{\sum_{j=1}^{N} \left(y(j) - \bar{y}\right)^2}}\,, \tag{4.10}$$

   wobei $\bar{y}$ dem Mittelwert der Messdaten entspricht. Einzig der neu hinzugefügte Schnitt wird optimiert; alle anderen Schnitte bleiben unverändert. Die Gesamtzahl der LM erhöht sich um Eins zu $M \to M + 1$. Die Parameter der zwei neu erzeugten LM werden innerhalb jeder Berechnung der Verlustfunktion zur nichtlinearen Optimierung durch eine gewichtete Least Squares-Optimierung neu geschätzt.
5. *Überprüfe Abbruchbedingung:* Wenn die Abbruchbedingung erfüllt ist, findet keine weitere Teilung mehr statt und der Algorithmus ist beendet. Ansonsten gehe zu Schritt 2.

Für das Abbrechen des Algorithmus' gelten die gleichen Kriterien wie beim LOLIMOT-Algorithmus. Kriterien zur Optimierung der Modellkomplexität sind ausführlich in Kapitel 3.5 diskutiert. Standardmäßig wird das korrigierte AIC-Kriterium verwendet, auf welches ausführlicher in Kapitel 4.4 eingegangen wird.

Bild 4.15 veranschaulicht die Funktionsweise von HILOMOT in den ersten vier Iterationen bei einem zweidimensionalen Eingangsraum. HILOMOT startet mit dem globalen Modell 1-1. In der zweiten Iteration wird sowohl für die u_1- als auch für die u_2-Richtung jeweils der achsenorthogonale, hälftige Schnitt durchgeführt. Eine Auswertung des globalen Modellfehlers selektiert dann den besseren der beiden Schnitte als Initialisierung für die nichtlineare Schnittoptimierung. Nach Durchführung des Schnittes wird für beide LM 2-1 und 2-2 der lokale Modellfehler berechnet. Das schlechtere LM, im Beispiel LM 2-2 (grau markiert), wird nun weiter unterteilt und dem Gesamtmodell ein weiteres LM hinzugefügt. Der Algorithmus schreitet dann solange fort, bis ein Abbruchkriterium erreicht wurde.

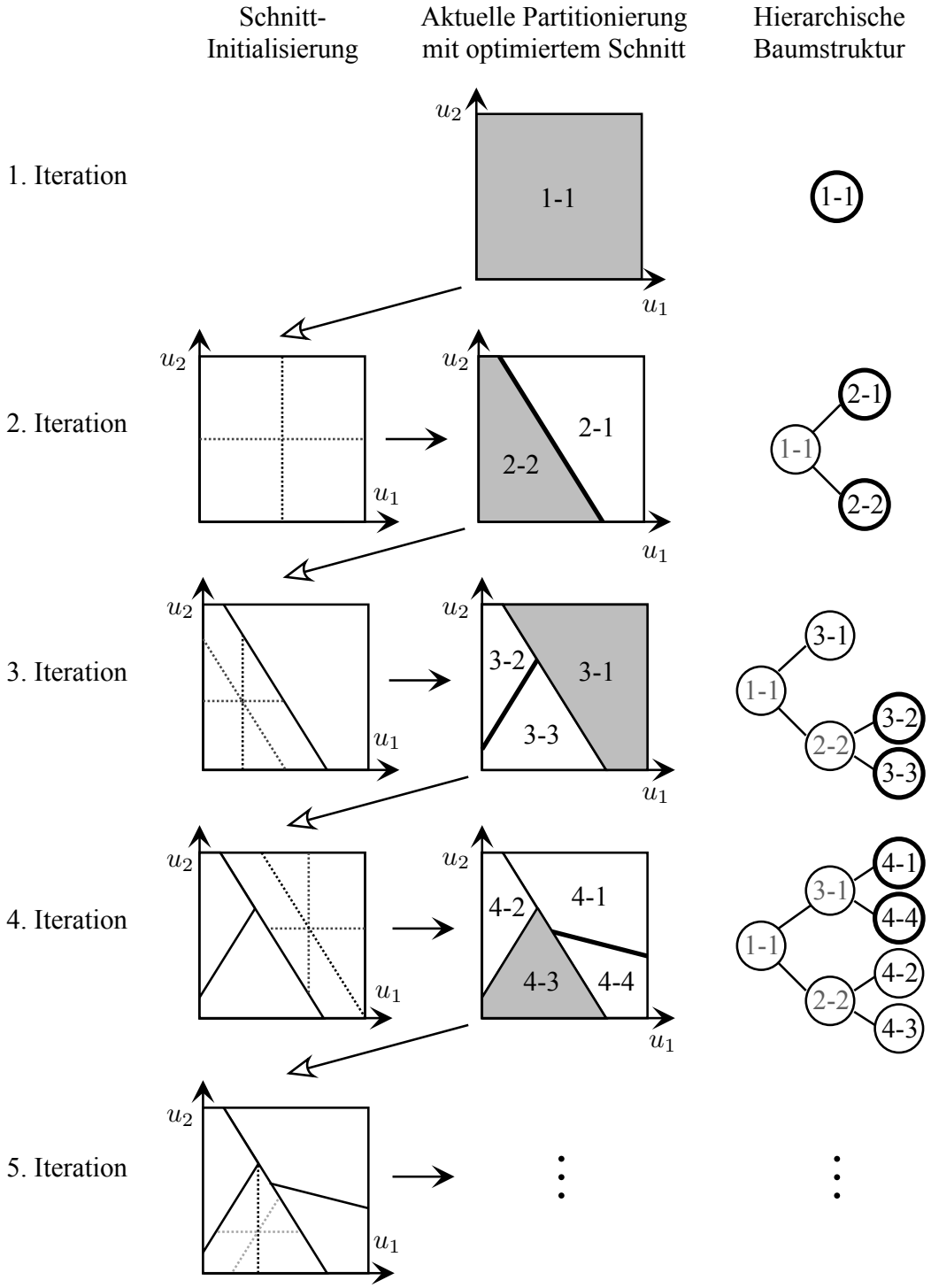

Bild 4.15: Fünf mögliche Iterationen von HILOMOT. Das schlechteste LM ist jeweils grau markiert.

4.3.3 Nichtlineare Schnittoptimierung

Zur Realisierung achsenschräger Teilungen eignen sich insbesondere Sigmoidfunktionen, wie sie beispielsweise auch bei MLP-Netzen Verwendung finden. Im Gegensatz zu Gaußfunktionen, wo eine voll besetzte Kovarianzmatrix vorgegeben werden muss, ist die Sigmoid-Funktion lediglich durch den Parametervektor $\underline{\widetilde{v}} = [v_0\ v_1\ \ldots\ v_{nz}]^T$ definiert:

$$\Psi(\underline{z}) = \frac{1}{1+\exp(\alpha)}\,. \tag{4.11}$$

mit $\alpha = -\kappa\,[1\ \underline{z}]^T\,\underline{\widetilde{v}} = -\kappa\,(v_0 + v_1 z_1 + \ldots + v_{nz} z_{nz})$ und $\underline{z} = [z_1\ z_2\ \ldots\ z_{nz}]$. Damit die Steilheit der Sigmoidfunktion unabhängig von der Wahl des Parametervektors $\underline{\widetilde{v}}$ bleibt, lässt sich die Konstante κ folgendermaßen einführen:

$$\kappa = \frac{\lambda}{\|\underline{\widetilde{v}}\|}\,. \tag{4.12}$$

Bild 4.16 zeigt eine Sigmoidfunktion für verschiedene Werte von λ. Die Steilheit ist proportional zu λ.

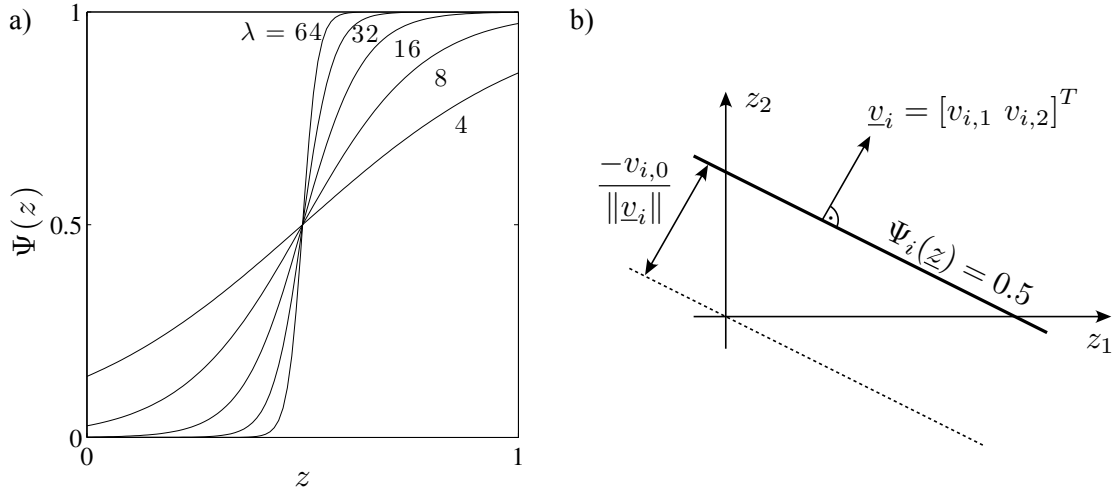

Bild 4.16: a) Der Parameter λ beeinflusst die Steilheit der Sigmoide mit $\kappa = \frac{\lambda}{\|\underline{\widetilde{v}}\|}$.
b) Teilungsfunktion des i-ten LM. Der Gleichanteil $v_{i,0}$ und die Parameter $\underline{v}_i$ definieren die Lage und Richtung der Schnittlinie $\Psi_i(\underline{z}) = 0.5$.

Die Parameter in $\underline{\widetilde{v}}$ bestimmen die Lage und Richtung der Sigmoidfunktion. In Bild 4.16 ist dies für den zweidimensionalen Fall illustriert (vgl. auch [44]). Der Index i deutet an, dass es sich um die Teilungsfunktion des i-ten lokalen Modells handelt. Charakterisierend für die Teilungsfunktion ist die Höhenlinie $\Psi_i(\underline{z}) = 0.5$. Sie kann als Schnittlinie interpretiert werden, welche die Grenze zwischen zwei LM bildet.

Der reduzierte Vektor $\underline{v}_i = [v_{i,1}\ v_{i,2}\ \cdots\ v_{i,nz}]^T$ wird mit den Eingangsgrößen $z_1, z_2, \ldots, z_{nz}$ multipliziert und bestimmt die Richtung des weichen Schnitts. Der Gleichanteil $v_{i,0}$ bestimmt die Position des Schnittes (den senkrechten Abstand vom Ursprung) und der redundante Parameter κ_i bestimmt die Glattheit des Schnittes. Streng betrachtet wird der

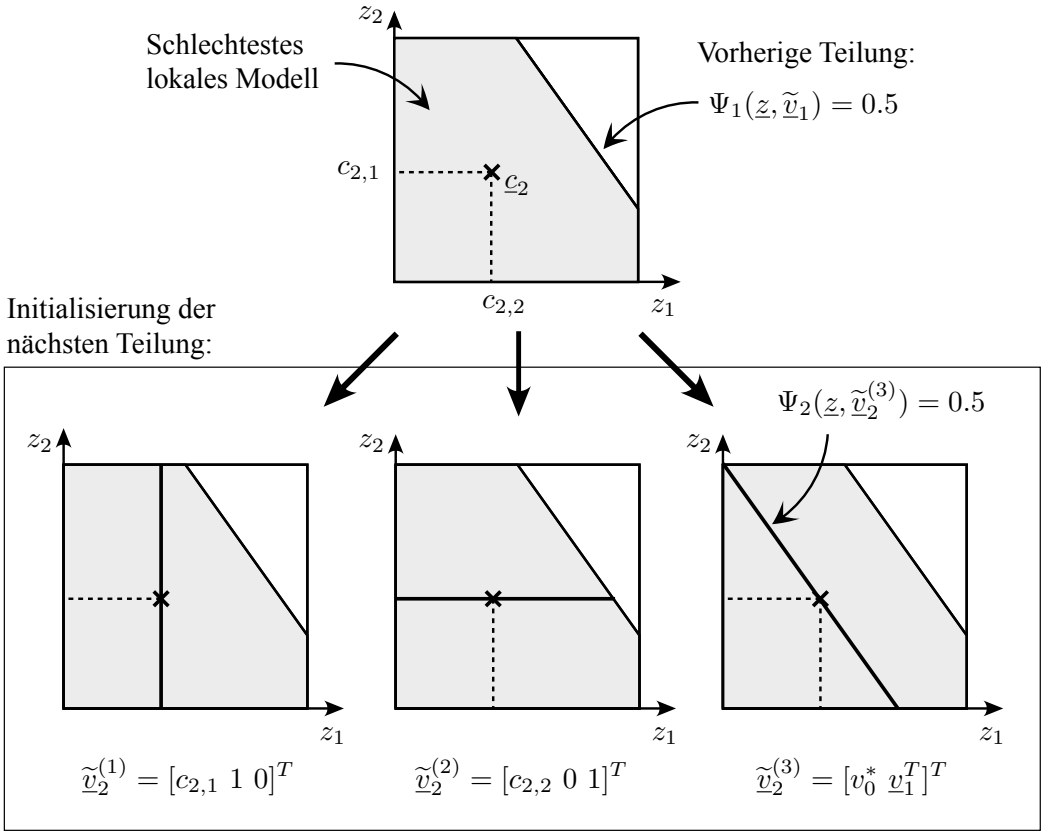

Bild 4.17: Teilungsfunktion des i-ten LM. Der Gleichanteil $v_{i,0}$ und die Parameter $\underline{v}_i$ definieren die Lage und Richtung der Schnittlinie $\Psi_i(\underline{z}) = 0.5$.

Parameter κ_i nicht benötigt, da die Glattheitsinformation schon in der Größe $\|\underline{v}_i\|$ enthalten ist. Allerdings vereinfacht die explizite Einführung des Parameters κ_i die Schnittoptimierung. Die Redundanz wird aufgehoben, indem der Richtungsvektor $\underline{v}_i$ auf Einheitslänge beschränkt wird. Dies erreicht man durch Anwendung der folgenden Operation in (4.11): $v_{i,j} \leftarrow v_{i,j}/\|\underline{v}_i\|$ für $j = 0, 1, \ldots, nz$. Darüber hinaus genügt es dann für die Optimierung, den Gleichanteil $v_{i,0}$ auf einem Initialwert festzuhalten, denn sowohl die Schnittrichtung als auch die Schnittposition können alleinig mit den Parametern $\underline{v}_i$ eingestellt werden. Damit ist die Anzahl der zu optimierenden Parameter nz.

Die Festlegung des Gleichanteils $v_{i,0}$ hat den großen Vorteil, dass die Schnittinitialisierung geometrisch interpretierbar ist und man dadurch den Initialschnitt deterministisch festlegen kann. Bild 4.17 zeigt die Vorgehensweise bei der Initialisierung der Schnittoptimierung. Die achsenorthogonale Initialisierung funktioniert unproblematisch, indem der Gleichwert auf die Komponente des Zentrumsvektors gesetzt wird, in welcher der Schnitt erfolgen soll. Die Richtungsparameter setzt man alle Null, bis auf den Parameter, der zur orthogonalen Schnittrichtung gehört. Soll z.B. in Richtung z_2 geteilt werden, ergeben sich die Parameter zu $\underline{v} = [0\ 1\ 0\ \cdots\ 0]^T$. Zur Initialisierung mit der Teilungsrichtung des Elternknotens kann man die zugehörigen Richtungsparameter direkt übernehmen, in Bild 4.17 z.B. der Vektor $\underline{v}_1$. Allerdings muss der Gleichanteil v_0^* so ausgerechnet werden, dass der Initialschnitt zwar die gleiche Richtung hat, aber im Unterschied zum Schnitt des Elternknotens durch das neue Zentrum des zu teilenden LM verläuft, im Beispiel $\underline{c}_2$. Anhand folgender Überlegung lässt sich v_0^* schließlich berechnen: Man weiß, dass die Teilungsfunktion

$\Psi_2(\underline{z}, \widetilde{\underline{v}})$, ausgewertet am Zentrum $\underline{c}_2$, den Funktionswert 0.5 hat. Das bedeutet, der Exponent α in Gl. (4.11) muss Null sein. Daraus folgt:

$$v_0^* = -\underline{c}_2^T \underline{v}_1 \tag{4.13}$$

Die vorgestellte deterministische Vorgabe der Initialschnitte hat den Vorteil, dass die initialen Parameter immer bereits nahe am Optimum zu vermuten und daher einer zufälligen Parameterinitialisierung weit überlegen sind.

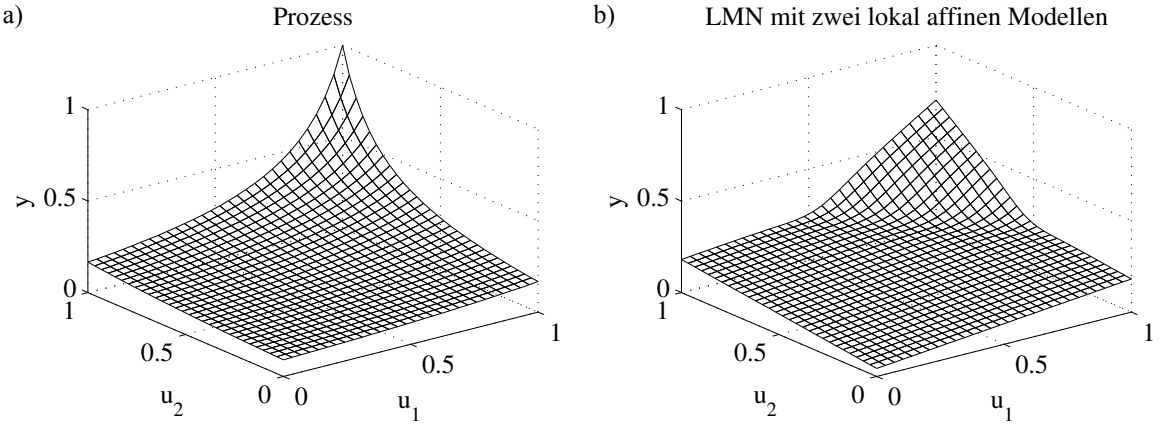

Bild 4.18: Vergleich des wahren Prozesses (a) mit dem HILOMOT-Modell nach der ersten Iteration (b). Die achsenschräge Optimierung liefert eine Schnittrichtung, die orthogonal zur Prozessnichtlinearität liegt.

Zur Illustration der Optimierung mit HILOMOT zeigen die Bilder 4.18 und 4.19 ein zweidimensionales Beispiel. Man erkennt, dass die Schnitte nach der Optimierung orthogonal zur nichtlinearen Richtung des Prozesses liegen.

Die Interpolationsglattheit wird heuristisch gewählt. Hauptidee dabei ist es, eine Glattheit zu wählen, die ungefähr proportional zur Größe der Gültigkeitsregionen ist. Daher wird zwischen großen Regionen eine weichere Interpolation durchgeführt als zwischen kleinen Gebieten. Dieses Konzept liefert die Eigenschaft einer adaptiven Auflösung und ermöglicht es, beliebig kleine Details im Prozessverhalten abzubilden. Die Zentren der Gültigkeitsfunktionen werden durch den Mittelwert der Datenpunkte approximiert, welche mit deren Gültigkeitswerten gewichtet sind:

$$\underline{c}_i = \frac{1}{N} \sum_{k=1}^{N} \Phi_i(\underline{z}(k)) \cdot \underline{z}(k)\,, \tag{4.14}$$

mit N als Anzahl der Datenpunkte. Damit kann κ_i gewählt werden zu

$$\kappa_i = \frac{1}{k_\sigma \|\underline{c}_i - \underline{c}_j\|}\,, \quad k_\sigma > 0\,, \tag{4.15}$$

wobei j der Nachbar von Knoten i im Baum ist und k_σ der Glattheitsfaktor für das gesamte Modell. Der Faktor k_σ kann wie beim LOLIMOT-Algorithmus vom Benutzer eingestellt werden. Je größer er gewählt wird, desto weicher sind die Übergänge. Im Zusammenhang mit LOLIMOT hat sich gezeigt, dass die Modelleigenschaften im Wesentlichen nicht von der

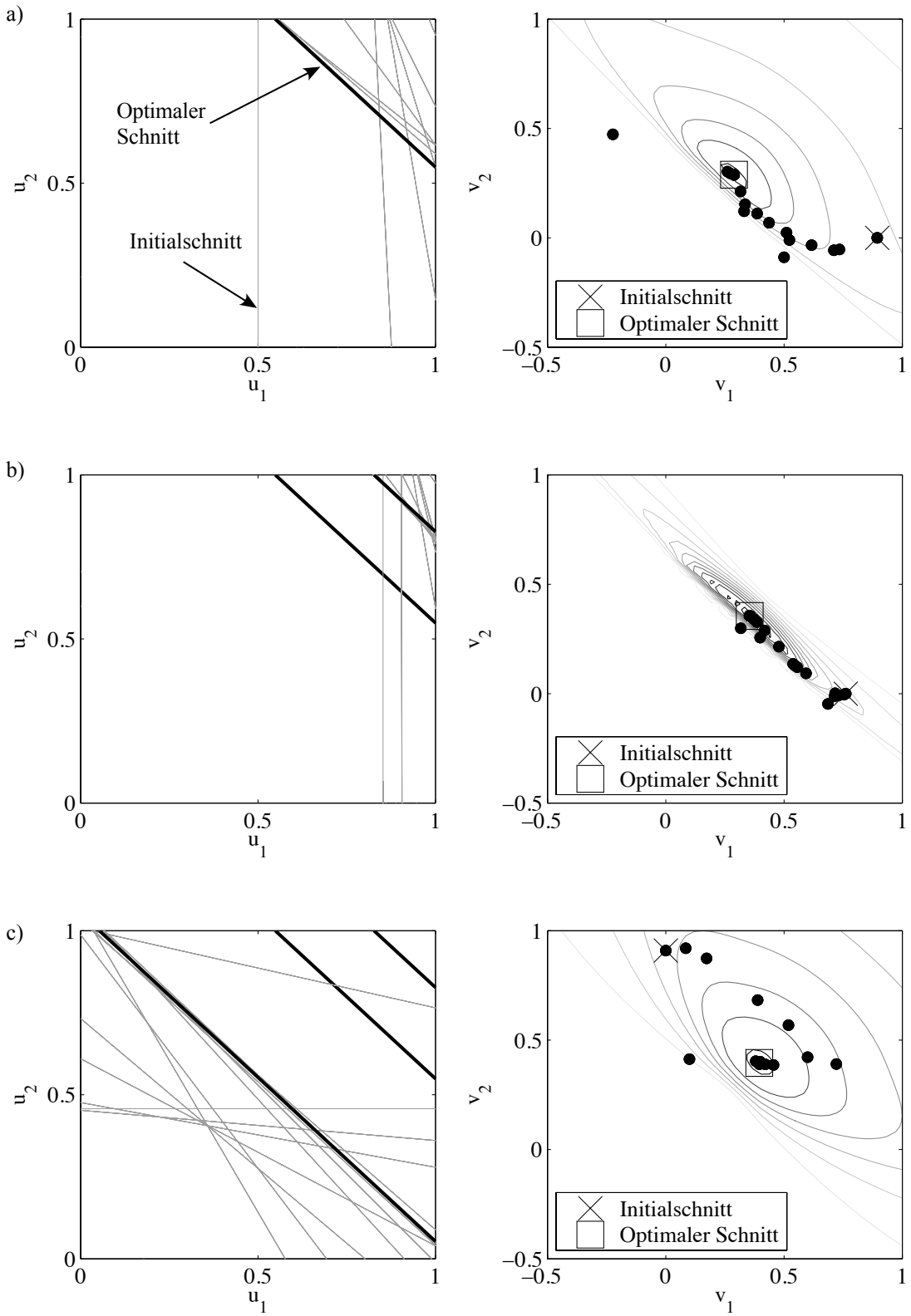

Bild 4.19: a)-c) Illustration der ersten drei Iterationen von HILOMOT anhand des in Bild 4.18 dargestellten Beispielprozesses. Links: Iterationen zur Partitionierung. Die Schnittlinien entsprechen den Höhenlinien der Gültigkeitsfunktionen bei $\Phi_i(\underline{u}) = 0.5$. Rechts: Iterationen der Optimierung $(\cdot)$ im Parameterraum. v_1 und v_2 bestimmen die Lage und Richtung des Schnitts.

exakten Wahl dieses Parameters abhängen [116]. Jedoch muss angebracht werden, dass die hierarchische Konstruktion der Partitionierung unter Umständen eine a priori-Einstellung des Glattheitsfaktors k_σ problematisch gestalten kann, weil die resultierende Tiefe des Teilungsbaums mit der Interpolationsglattheit des Modells zusammenhängt. Daher wird im Anhang C zusätzlich eine Methode zur automatischen Einstellung der Glattheit während des Trainings vorgestellt.

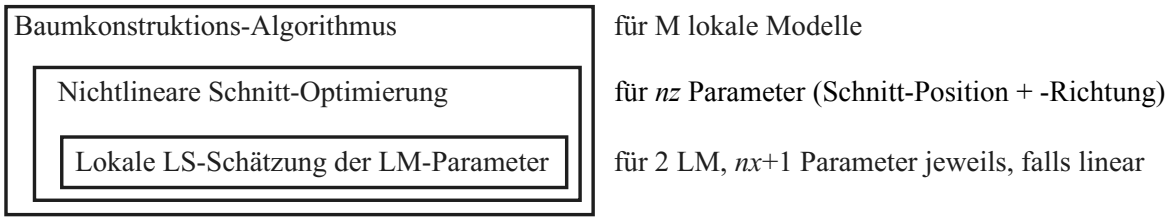

Bild 4.20: Verschachtelung der Optimierung bei HILOMOT.

Wie in Bild 4.20 dargestellt, sind der Baumkonstruktionsalgorithmus, die nichtlineare Schnittoptimierung und die lokale LS-Schätzung ineinander verschachtelt. Der Rechenaufwand lässt sich wie folgt beziffern: Der Baumkonstruktionsalgorithmus muss für M LM durchgeführt werden. Zur achsenschrägen Unterteilung ist die Optimierung von nz Parametern nötig, wobei nz die Anzahl der Regressoren zur Berechnung der Gültigkeitsfunktionen ist. Die lokale Least Squares-Schätzung optimiert für zwei LM jeweils $nx + 1$ Parameter, wobei nx die Anzahl der Regressoren zur LM-Schätzung ist.

Zur Analyse der Robustheit des vorgestellten Verfahrens sind in Anhang D weiterführende Untersuchungen gezeigt. Im Anhang D.1 wird der Frage nachgegangen, wie stark die heuristisch gewählten Initialwerte für die Schnittoptimierung das Ergebnis der Optimierung beeinflussen. Darüber hinaus ist die Frage interessant, wie viel Verbesserungspotential die gleichzeitige Optimierung aller Schnitte in jeder Iteration von HILOMOT verspricht. Dies wird in Anhang D.2 empirisch untersucht.

4.4 Komplexitätsoptimierung für Baumkonstruktionsverfahren

Wesentlicher Bestandteil eines Baumkonstruktionsverfahrens ist es, die Modellkomplexität, d.h. die Anzahl der lokalen Modelle, festzulegen. Dabei ist der Kompromiss zwischen Modellflexibilität und Robustheit bezüglich des Overfittings zu finden. Das in Kapitel 3.5.3 vorgestellte korrigierte AIC-Kriterium bietet eine sinnvolle Alternative zur numerisch wesentlich aufwändigeren Kreuzvalidierung, falls keine expliziten Validierungsdaten zum Prozess vorliegen. Es ergibt sich aber bei der praktischen Anwendung des AIC_c-Kriteriums die Schwierigkeit, dass sich speziell bei Baumkonstruktionsverfahren die Parameteranzahl nur schwer beziffern lässt. Bei LOLIMOT geht man der Einfachheit halber davon aus, dass lediglich die linear geschätzten LM-Parameter ins AIC_c-Kriterium einfließen. Bei HILOMOT muss jedoch zusätzlich berücksichtigt werden, dass die nichtlineare Schnittoptimierung im Gegensatz zur heuristischen Festlegung der Schnitte erkennbaren Einfluss bezüglich des Overfittings haben kann.

Daher setzt sich die Parameteranzahl in Gl. (3.75) aus mehreren Komponenten zusammen:

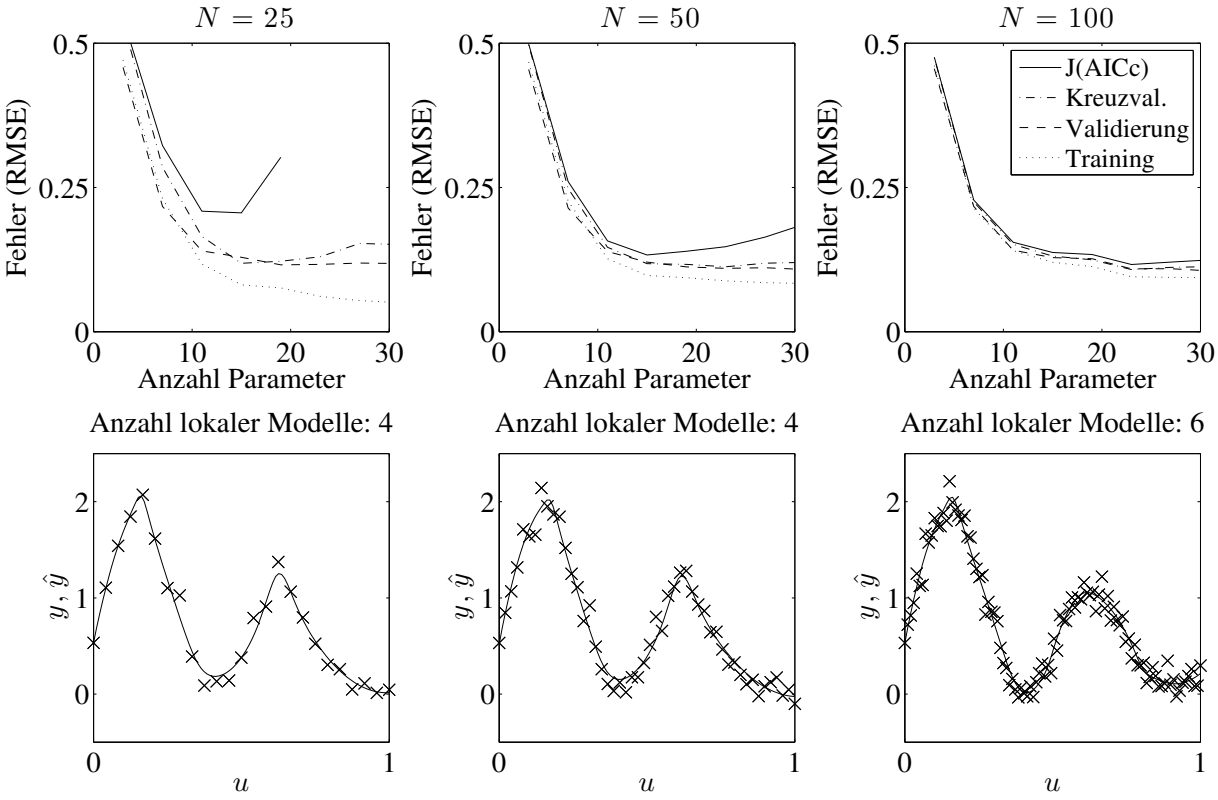

Bild 4.21: Testprozess mit $N = 25, 50, 100$ Punkten. Oben: Fehlerverläufe auf Trainings- und Validierungsdaten in Abhängigkeit von der Anzahl der Modellparameter. Außerdem sind der Kreuzvalidierungsfehler und die auf RMSE umgerechneten AIC_c-Werte dargestellt. Unten: Modelle $\hat{y}$ mit minimalem AIC_c im Vergleich zu den Messdaten y (markiert mit $\times$).

- Zunächst wird ein Freiheitsgrad verwendet, um die Varianz zu schätzen.
- Darüber hinaus werden die effektiven Parameter der linearen lokalen Schätzung hinzu addiert mit spur($\underline{S}$). Hierbei handelt es sich um die effektiven Freiheitsgrade der Regression und nicht um die effektiven Freiheitsgrade der Residuen wie in Gl. (3.26), also spur($\underline{S}\underline{S}^T$). Der große Vorteil bei Verwendung von spur($\underline{S}$) gegenüber spur($\underline{S}\underline{S}^T$) ist, dass der Rechenaufwand deutlich geringer ist.
- Bei Anwendung des HILOMOT-Algorithmus wird zusätzlich ein Term addiert, der die nichtlineare Teilungsoptimierung bestraft. Ein lokales Modellnetz mit M lokalen Modellen hat $M-1$ Teilungen. Für jede Teilung werden dem Modell nz Sigmoid-Parameter hinzugefügt.

Die Parameteranzahl wird also insgesamt mit $n = 1 + \text{spur}(\underline{S}) + (M-1)nz$ berechnet. Bei LOLIMOT wird der letzte Term entsprechend weggelassen. Die Abschätzung der Teilungsparameter mit $(M-1)nz$ begünstigt eine eher konservative Auslegung der Modellkomplexität, da beim Training die Teilungsparameter nicht *gleichzeitig*, sondern *nacheinander* optimiert werden. Dies führt zu einem höheren Biasfehler des Modells, was generell wünschenswert ist, um Overfitting entgegenzuwirken.

Zur Veranschaulichung der dargelegten Kriterien sind diese in Bild 4.21 an einem Beispielprozess (Gl. (3.41)) für drei unterschiedliche Datenmengen miteinander verglichen. Zu beachten ist, dass prinzipiell sogar mehr Parameter geschätzt werden können, als Datenpunkte vorhanden sind, da durch die lokale Schätzung ein erheblicher Regularisierungseffekt vorhanden ist. Je mehr Datenpunkte dem Schätzproblem zur Verfügung stehen, desto ähnlicher sind sowohl die Fehlerverläufe auf Trainings- und Validierungsdaten als auch der Kreuzvalidierungsfehler und das korrigierte AIC-Kriterium. Zur automatischen Komplexitätsauswahl ist das korrigierte AIC-Kriterium am besten geeignet, sofern keine Validierungsdaten vorhanden sind.

4.5 Strategie der Strukturabwägung

Bei lokalen Modellnetzen stellt der Einsatz lokal linearer Modelle einen guten Kompromiss zwischen Flexibilität und Interpretierbarkeit dar. Oftmals ist allerdings der Einsatz lokaler Polynome höheren Grades vorteilhaft. Durch den Einsatz von Strukturselektionsverfahren lässt sich zudem der Grad eines Polynoms erhöhen, ohne zwangsläufig alle möglichen Regressoren des vollständigen Polynoms verwenden zu müssen. Dieser Abschnitt stellt einen neuen Ansatz vor, der das Einbinden von Strukturselektion in das Training lokaler Modellnetze erlaubt. Dazu wird in jeder Iteration des Trainings die Teilung mit einer Komplexitätserhöhung des lokalen Modells verglichen. Daraus resultiert die Strategie der Strukturabwägung. Die nichtlineare Schnittoptimierung in Verbindung mit einer Strukturabwägung führt auf ein sehr flexibles Partitionierungsverfahren, welches mit moderatem Mehraufwand beim Training auf deutlich sparsamere Modelle führt.

4.5.1 Lokal polynomiale Modelle

Globale Polynome gehören zu den einfachsten nichtlinearen Modellen. Durch ihre lineare Parametrierung sind sie effizient schätzbar. Allerdings weisen sie schwerwiegende Nachteile

bezüglich ihres Interpolations- und Extrapolationsverhaltens auf [116]. Außerdem wächst die Anzahl der Regressoren und damit der zu schätzenden Parameter sehr stark mit der Eingangsraumdimension p und dem Polynomgrad l an:

$$\frac{(l+p)!}{l!\,p!}\,. \tag{4.16}$$

Daher werden Polynome, insbesondere wenn p und/oder l größer werden, meist nicht in ihrer vollständigen Form eingesetzt, sondern die signifikanten Regressoren mit Hilfe eines Strukturselektionsverfahrens [106] ausgesucht. Wegen der linearen Zusammenhänge kann dies recht effizient geschehen.

Polynome können auch im Rahmen lokaler Modellnetze eingesetzt werden, indem ihre Regressoren fest vorgegeben [12] oder mittels Strukturselektion ausgesucht [136] werden. Die klassischen Fälle für die Wahl der lokalen Polynommodelle LM_i sind:

- 0. Grad: Lokal konstante Modelle führen auf ein normiertes radiales Basisfunktionennetz bzw. ein Neuro-Fuzzy-System mit Singletons [116]. Das Modellverhalten wird von der exakten Form der Gültigkeitsfunktionen dominiert (Position und Breite). Für eine sinnvolle lokale Optimierung sind die lokalen Modelle zu primitiv; die vernachlässigte Überlappung spielt die Hauptrolle.
- 1. Grad: Lokal lineare Modelle führen auf Takagi-Sugeno Neuro-Fuzzy-Systeme. Deren Vorteile sind u.a. (i) ihre Robustheit gegenüber der exakten Form der Gültigkeitsfunktionen, (ii) die Möglichkeit, sinnvoll eine effiziente lokale Optimierung einzusetzen und (iii) ein nur linearer Anstieg der Komplexität mit der Eingangsdimension.
- 2. Grad: Lokal quadratische Modelle können einen signifikanten Vorteil bringen, wenn das Gesamtmodell später bezüglich seiner Eingangsgrößen optimiert werden soll. Dies ist beispielsweise bei der Erzeugung optimaler Steuerkennfelder bei Motorsteuerungen der Fall. Kombiniert mit einer nichtlinearen Optimierung kann dies als ein erweitertes Newton-Verfahren aufgefasst werden. Allerdings besteht der Nachteil in einer quadratisch anwachsenden Komplexität mit der Eingangsdimension.
- Höherer Grad: Lokale Modelle noch höheren Grades kommen nur für niedrigdimensionale Probleme in Frage. Typischerweise werden sie in Kombination mit einem Strukturselektionsverfahren eingesetzt [106].

Je mehr Vorwissen vorhanden ist, desto eher ist eine Vorgabe der Struktur der lokalen Modelle möglich und sinnvoll. Im Black-Box-Fall ist aber ein vollautomatisches Verfahren wünschenswert.

Auch im Zusammenhang lokaler Modellnetze können Strukturselektionsverfahren sinnvoll zur Auswahl der Regressoren des jeweiligen lokalen Modells eingesetzt werden. Zur Strukturselektion im Zusammenhang mit einer gewichteten Least Squares-Schätzung bieten sich insbesondere die in Kapitel 3.6 vorgeschlagenen Methoden zur Vorwärts- und Rückwärtsselektion an, bei denen ein sequentieller t-Test zur Auswahl der signifikanten Regressoren eingesetzt wird. Verwendet man zudem lokale Koordinatensysteme für die lokalen Modelle, bei denen die Regressoren auf das Zentrum des LM transformiert werden, so können die selektierten Terme in vielen Anwendungen wertvolle Rückschlüsse auf das Prozessverhalten liefern.

4.5.2 Komplexitätserhöhung durch Strukturabwägung

Wie aus dem obigen Abschnitt hervorgeht, können je nach Anwendung sowohl lokale Polynome niedrigen als auch hohen Grades sinnvoll sein. Auch innerhalb eines Prozesses können in verschiedenen Arbeitsbereichen verschiedene Ansätze wünschenswert sein. Der in [11] erstmals vorgeschlagene und z.B. in [10] weiter untersuchte Algorithmus erfüllt diese Anforderungen durch eine Erweiterung des LOLIMOT-Verfahrens. Im Unterschied zu [136] soll nicht nur die Least Squares-Schätzung der Parameter der lokalen Modelle durch eine Strukturselektion lokaler Polynome ersetzt werden. Die Polynomselektion soll in den Partitionierungsalgorithmus eingebunden werden.

In jeder Iteration von LOLIMOT wird ein lokales Modell unterteilt. Es kommt also ein lokales Modell mit $p+1$ Parametern hinzu. Durch die lokale Schätzung ist deren effektive Anzahl allerdings kleiner [115]. In jeder Iteration werden p alternative Teilungen des lokal schlechtesten Modells entlang jedes Eingangs verglichen. In LOLIMOT mit Strukturabwägung tritt als zusätzliche Alternative eine Komplexitätserhöhung des lokalen Modells ohne Teilung hinzu, siehe Bild 4.22. Dies geschieht, indem $p+1$ Regressoren zusätzlich mittels der in Kapitel 3.6.2 vorgestellten Vorwärtsselektion dem Modell hinzugefügt werden. Da hierdurch das Modell einen globaleren Charakter bekommt und mehr Parameter gleichzeitig geschätzt werden, ist es sinnvoll, besser $C \cdot (p+1)$ Regressoren zusätzlich zu selektieren, wobei C einen relativen Strafterm darstellt, der typischerweise kleiner 1 ist.

So wird in jeder Iteration zwischen zwei Arten der Komplexitätserhöhung abgewogen: (i) Teilung eines Modells unter Hinzufügung von $p+1$ Parametern und (ii) Erweiterung des lokalen Polynoms eines Modells durch Hinzufügen von $C \cdot (p+1)$ Parametern. In beiden Fällen können nicht ganze Zahlen für die gewünschte Regressorenanzahl entstehen, die für die Strukturselektion gerundet werden. Für das weitere Fortschreiten des Verfahrens bleiben die exakten Zahlen aber bestehen. Allerdings kann es vorkommen (insbesondere bei dafür konstruierten Beispielen), dass ein lokales Modell den Prozess schon perfekt beschreibt und die Strukturselektion keine sinnvollen weiteren Regressoren findet. Solche Fälle führen auf eine „Zwangsteilung“, welche die beste Teilungsalternative erzwingt.

Der Metaparameter C ermöglicht dem Benutzer eine Bevorzugung der Teilungen oder der Steigerung der Polynomkomplexität: $C \to 0$ führt auf LOLIMOT, $C \to \infty$ führt auf ein globales Polynom mit Strukturselektion, da in diesem Fall nie geteilt werden wird.

4.5.3 Strukturabwägung bei achsenschräger Unterteilung

Eine große Schwäche von LOLIMOT ist die Einschränkung auf achsenorthogonale Partitionierungen. Je höherdimensionaler der zu modellierende Prozess ist, desto extremer wirkt sich dies als Nachteil aus. Um so wichtiger ist dann das Auffinden der „Richtungen“ der Nichtlinearität. Da diese Richtungen aber notwendigerweise innerhalb eines nichtlinearen Zusammenhangs liegen, ist deren Optimierung immer ein nichtlineares Problem. Daher muss bei LOLIMOT die sehr hohe Geschwindigkeit durch den rein linearen Optimierungsansatz aufgegeben werden. Der in dieser Arbeit vorgestellte HILOMOT-Algorithmus bietet dafür ein sinnvolles Verfahren, welches hier mit dem Prinzip der Strukturabwägung kombiniert wird ([56]), siehe Bild 4.23.

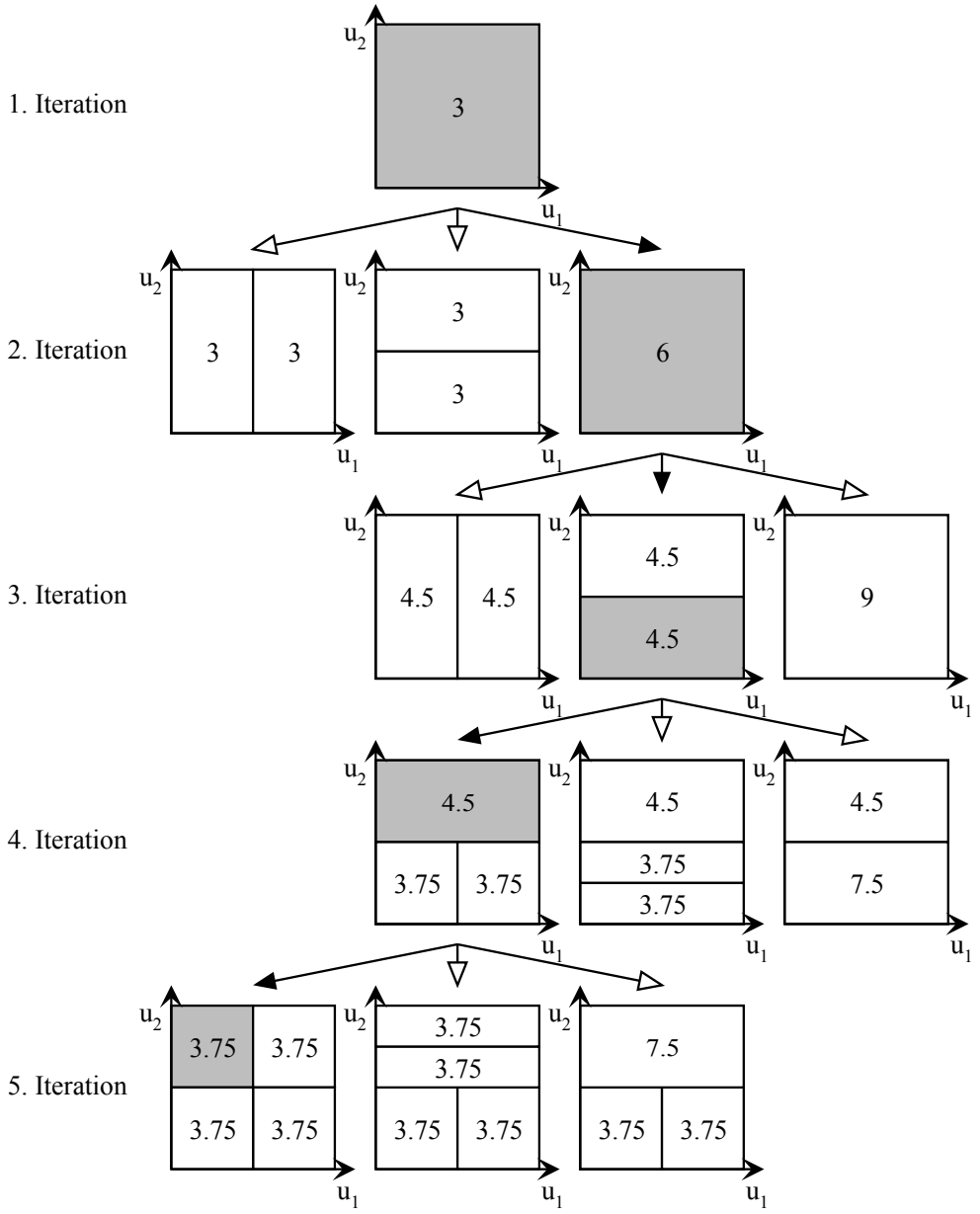

Bild 4.22: Fünf mögliche Iterationen von LOLIMOT mit Strukturabwägung (kurz: LOLIMOT+). Die Zahlen geben die gewünschte Regressorenanzahl für jedes lokale Modell an. Die relative Bestrafung C für die Erhöhung der Polynomkomplexität ist in diesem Beispiel gleich 1.

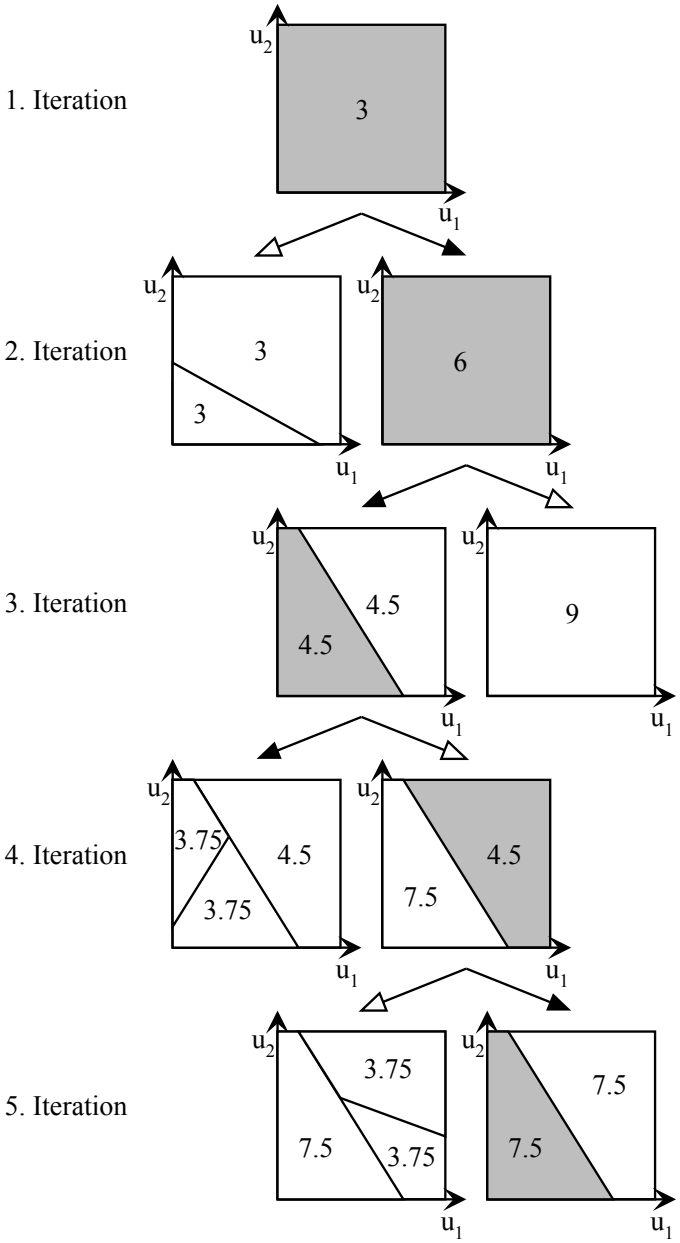

Bild 4.23: Fünf mögliche Iterationen von HILOMOT mit Strukturabwägung (kurz: HILOMOT+).

Der Vorschlag ist an dieser Stelle, nicht in jeder Iteration des unterlagerten nichtlinearen Optimierungsverfahrens die lineare Strukturselektion einzusetzen. Vielmehr ist es völlig hinreichend, bei der Schnittinitialisierung einmal mit der Strukturselektion für die beiden neuen lokalen Modelle die signifikanten Regressoren zu selektieren. Während der iterativen, nichtlinearen Optimierung werden diese eingefroren und die zugehörigen Parameter nur noch mittels Least Squares neu geschätzt. Damit ist das HILOMOT-Verfahren ohne und mit Strukturabwägung nahezu gleich schnell.

Ein Vergleich der Bilder 4.22 und 4.23 zeigt, dass in letzterem immer zwei Alternativen miteinander verglichen werden, während dies in ersterem mit $p+1$ Alternativen dimensionsabhängig ist. Dieser Zusammenhang wird sehr wichtig, wenn künftig daran gedacht wird, beim Teilungsbaum k Schritte in die Zukunft zu optimieren. Dann steht ein Rechenaufwand von $K_1 \cdot 2^k$ einem von $K_2 \cdot (p+1)^k$ gegenüber. Dabei ist K_1 der Aufwand für eine nichtlineare Optimierung inklusive der darin verschachtelten linearen Strukturselektionen, während K_2 nur den Aufwand für die linearen Strukturselektionen beschreibt, also $K_1 \gg K_2$. Für eine Optimierung von genügend vielen Schritten k in die Zukunft wäre daher die HILOMOT-Variante sogar weniger aufwändig.

Die Eigenschaften und Funktionsweise von HILOMOT mit Strukturabwägung (kurz: HILOMOT+) werden in den Bildern 4.24 und 4.25 anhand von zwei statischen Beispielprozessen illustriert. Für jedes lokale Modell steht für das Training mit HILOMOT+ ein Polynom dritten Grades zur Auswahl ($C = 0.9$). Zu jeder Gültigkeitsfunktion ist jeweils die zugehörige Anzahl der selektierten Regressoren angegeben.

Als erstes Beispiel zeigt Bild 4.24 einen eindimensionalen Prozess, der einmal ohne und einmal mit Rauschen modelliert wird. Hierzu kommt weißes Rauschen mit $\sigma_n = 0.1 \cdot (y_{max} - y_{min})$ zur Anwendung. Dargestellt ist jeweils die erste, dritte und fünfte Iteration von HILOMOT+. Interessant ist, dass im rauschbehafteten Fall mit drei LM mehr geteilt und weniger Regressoren selektiert werden, wohingegen im rauschfreien Fall eher ein höherer Polynomgrad gewählt, dafür aber nur zwei LM erzeugt werden, siehe Bild 4.24 rechts.

Das zweite Beispiel in Bild 4.25 vergleicht HILOMOT und HILOMOT+. Als Abbruchkriterium ist ein Modellfehler von NRMSE = 0.05 zu erreichen. Das Beispiel zeigt, dass durch die Strukturabwägung bei HILOMOT+ der nichtlineare Prozess deutlich effizienter nachgebildet werden kann. HILOMOT erreicht das verlangte Fehlermaß mit 23 lokal linearen Modellen und 113 Parametern (inklusive der nichtlinearen Schnittparameter[2]), wohingegen HILOMOT+ mit 5 lokal polynomialen Modellen und 36 Parametern auskam, die entsprechend durch die Strukturselektion an den Prozess angepasst wurden.

Anhand dieser zwei einfachen Beispiele wird schnell deutlich, dass die Kombination aus inkrementellem Baumkonstruktionsverfahren, nichtlinearer Schnittoptimierung und der Strategie der Strukturabwägung eine flexible und effiziente Modellierungsmethode hervorbringt.

[2] Anzahl Parameter = Anzahl aller LM-Parameter + $(M-1)\cdot$ Anzahl Schnittparameter

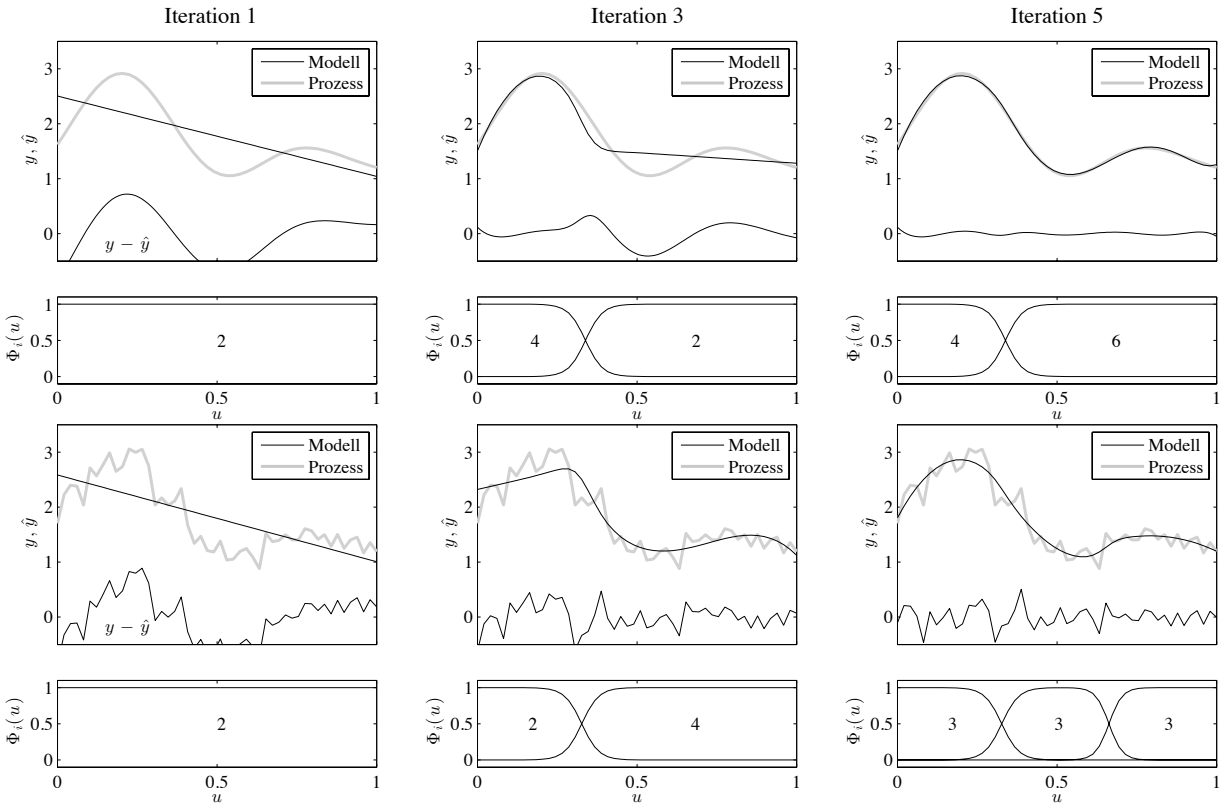

Bild 4.24: Modellierung einer eindimensionalen Testfunktion ohne Rauschen (oben) und mit Rauschen (unten).

4.6 Vergleich der Partitionierungsverfahren

Zur Übersicht sind in Bild 4.26 die Unterschiede der vier vorgestellten Varianten zum Training lokaler Modellnetze zusammengefasst. Zunächst unterscheidet sich die parallele Struktur mit achsenorthogonalen, hälftigen Teilungen mittels LOLIMOT von der hierarchischen Struktur bei HILOMOT, welche achsenschräge Teilungen ermöglicht. Darauf basierend kann man lokal lineare Modelle von selektierten, durch Strukturabwägung erzeugte, polynomiale Modelle unterscheiden.

Für einen Vergleich der vorgestellten Verfahren bietet sich als Testfunktion die Überlagerung einer Hyperbel- mit einer Gaußfunktion an (siehe Bild 4.27):

$$y(k) = \frac{1}{1 + 10 \cdot \sum_{j=1}^{p} \frac{1}{p}(1 - u_j(k))} + \frac{29}{44} \exp\left(-\frac{1}{2} \sum_{i=1}^{p} \left(\frac{u_i(k)}{0.25} \right)^2 \right), \qquad (4.17)$$

wobei p die Anzahl der Eingangsgrößen u_i ist, für $k = 1, \ldots, N$ Datenpunkte.

Diese Funktion kann als „best case“-Szenario für HILOMOT und HILOMOT+ angesehen werden, weil die achsenschräge Teilung bei der Modellbildung dieses Prozesses deutliche Vorteile gegenüber dem weniger flexiblen LOLIMOT-Verfahren aufweist.

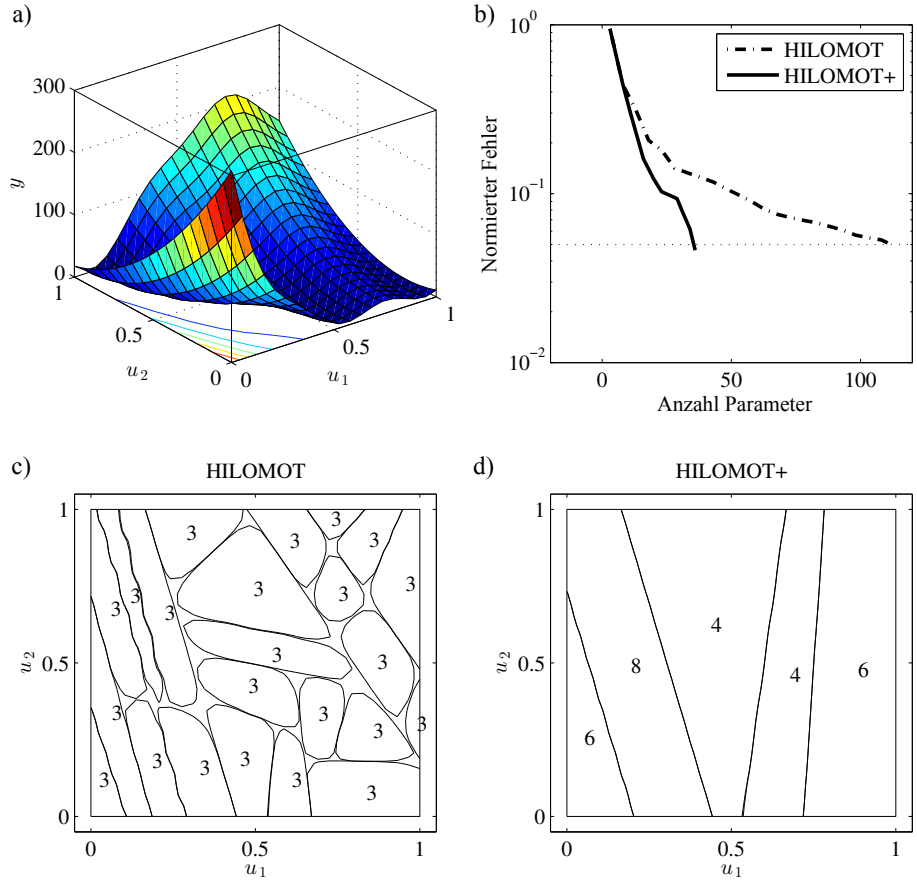

Bild 4.25: Modellierungsbeispiel mit zwei Eingangsgrößen. Ziel ist es, den Prozess a) mit einem normierten Fehler von 0.05 abzubilden. Der Konvergenzverlauf b) zeigt deutlich den Vorteil von HILOMOT+ gegenüber HILOMOT. Die Partitionierungen zu beiden Modellen sind in c) und d) gezeigt für $\Phi_i(\cdot) = 0.5$. Die Zahlen in den LM entsprechen der Anzahl der gewählten Regressoren.

Bei allen Untersuchungen wird ein normierter Fehler[3] kleiner als 0.05 gefordert und $5^5 = 3125$ unverrauschte Datenpunkte zum Training verwendet. Dies entspricht der Punkteanzahl eines fünfdimensionalen Datengitters. Allerdings werden die Trainingsdaten zur Modellbildung nicht auf einem Gitter, sondern raumfüllend[4] verteilt. Zum Vergleich der prinzipiellen Eigenschaften der Verfahren beschränken sich die Untersuchungen auf die Minimierung des Biasfehlers. Das Finden eines guten Bias-Varianz-Kompromisses hängt natürlich von der richtigen Wahl der Modellkomplexität mit Hilfe von Informationskriterien (z.B. AIC_c) und/oder dem Einsatz geeigneter Validierungsdaten ab. Zudem hat die verwendete lokale Schätzung dank inhärenter Regularisierung gute Robustheitseigenschaften.

Interessant ist zunächst die Betrachtung der Partitionierung des Modells für zwei Eingangsgrößen, siehe Bild 4.28. Vergleicht man LOLIMOT (20 LM) mit HILOMOT (15 LM), lässt

[3] $J = \sqrt{\frac{\sum_{k=1}^{N}(y(k)-\hat{y}(k))^2}{\sum_{i=1}^{N}(y(i)-\bar{y})^2}}$, $\bar{y} = \frac{1}{N}\sum_{k=1}^{N} y(k)$

[4] Die Punkte wurden sequentiell mit maximalem Nearest-Neighbor-Abstand zu allen gegenwärtig existierenden Punkten erzeugt.

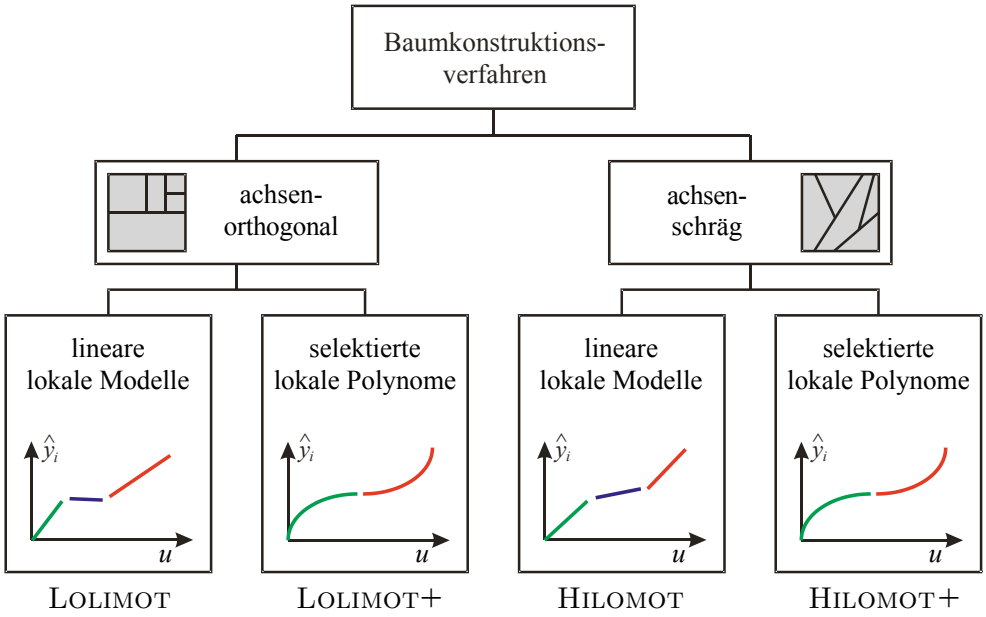

Bild 4.26: Übersicht der Baumkonstruktionsverfahren ohne und mit Strukturabwägung.

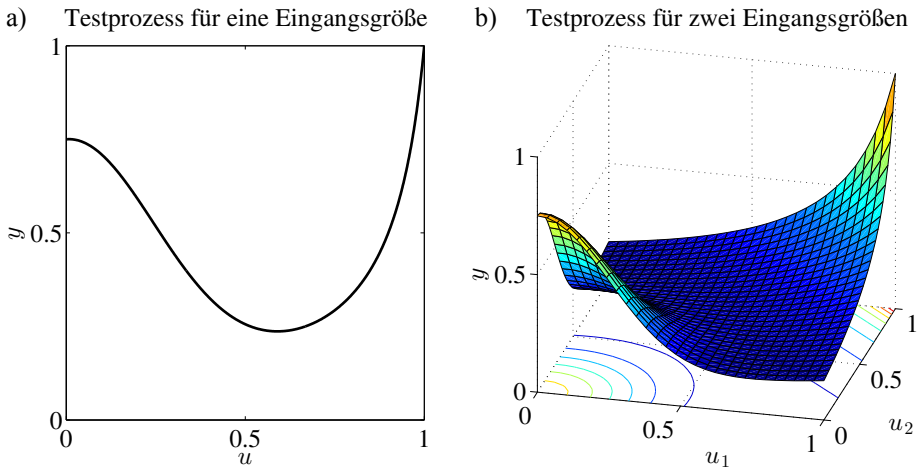

Bild 4.27: Ausgewählter Testprozess, exemplarisch für einen a) ein- und b) zweidimensionalen Eingangsraum.

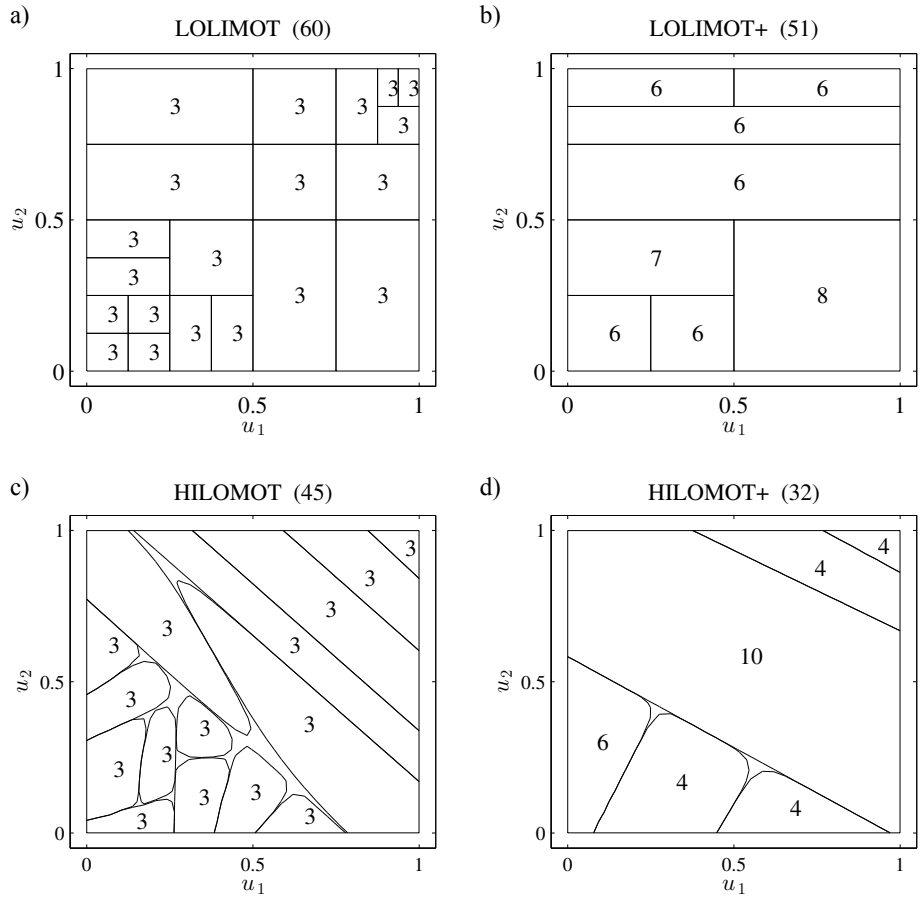

Bild 4.28: a)-d) 2D-Partitionierungen für die vier verschiedenen Methoden bei $\Phi_i(\cdot) = 0.5$. Die Zahlen entsprechen der Anzahl der LM-Parameter, in den Bildüberschriften steht die Summe aller LM-Parameter.

sich eine Einsparung von 5 lokalen Modellen alleine durch den Einsatz einer nichtlinearen Schnittoptimierung erreichen. Setzt man zusätzlich die Strukturabwägung ein ($C = 0.9$), lässt sich die Anzahl der lokalen Modelle noch stärker reduzieren (6 LM bei HILOMOT+). Die Zahl der ausgewählten Regressoren pro lokalem Modell wird zwar generell größer, aber im Endergebnis ist die Summe aller LM-Parameter kleiner. HILOMOT+ ist im Gegensatz zu LOLIMOT+ darüber hinaus in der Lage, die Schnitte an die Prozessnichtlinearität anzupassen.

Je mehr Eingangsgrößen der Prozess hat, desto effizienter werden sowohl Schnittoptimierung als auch Strukturabwägung. Vergleich man, wie in Bild 4.29 gezeigt, die Konvergenzverläufe von HILOMOT+ und LOLIMOT, ergibt sich mit der Anzahl der Eingangsgrößen ein deutlicher Anstieg des Modellbildungsaufwands bei LOLIMOT. Die Anzahl der Modellparameter bei HILOMOT+ steigt hingegen nur geringfügig an. Ein Grund dafür ist, dass für die Gesamtzahl der Modellparameter, abgetragen auf der Abszisse, auch die nichtlinear optimierten Schnittparameter mit der Faustformel $(M-1)\cdot p$ zur Gesamtzahl der LM-Parameter hinzuaddiert werden, um diese für einen fairen Vergleich berücksichtigen zu können.

Bild 4.30 zeigt die Gesamtzahl der Modellparameter in Abhängigkeit von der Anzahl der Eingangsgrößen für alle vier Verfahren. Außerdem sind die Anzahl der lokalen Modelle

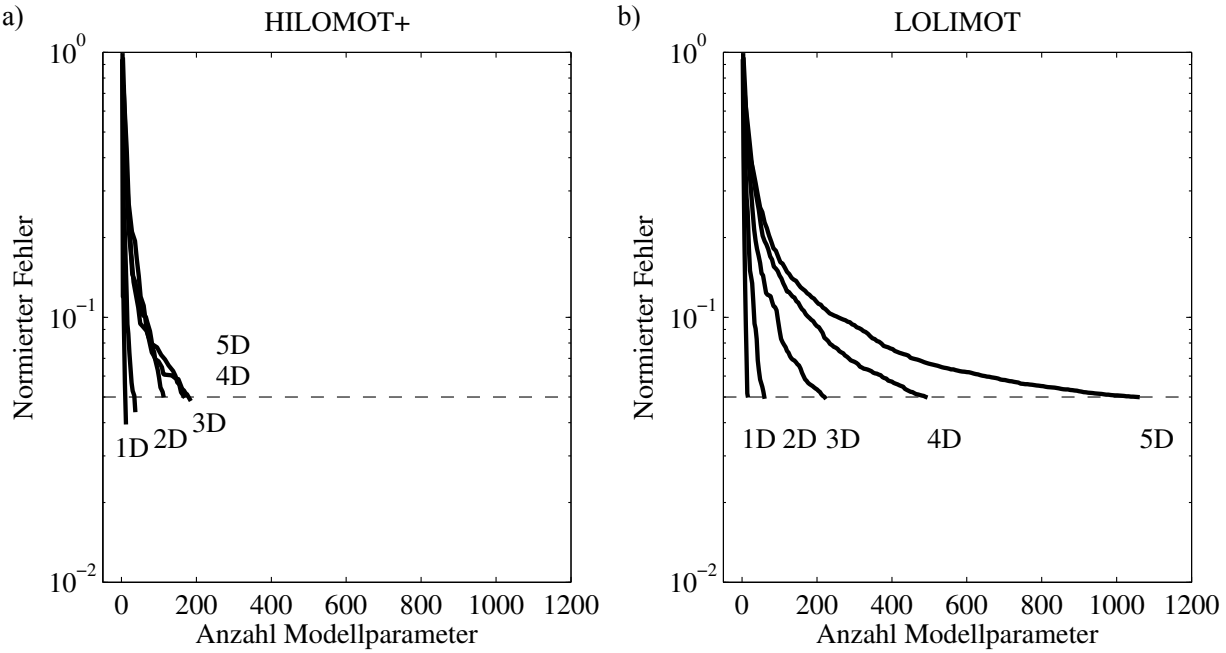

Bild 4.29: Konvergenzverläufe für einen ein- bis fünfdimensionalen Eingangsraum. Vergleich des flexibelsten Verfahrens HILOMOT+ (a) mit dem unflexibelsten Verfahren LOLIMOT (b).

(obere Zahlenreihe) und die benötigte Trainingszeit[5] (untere Zahlenreihe) angegeben. Für wenige Eingänge sind die Unterschiede eher gering. Je höher allerdings die Dimension des Eingangsraums ist, desto deutlicher treten die Vorteile einer höheren Modellflexibilität zum Vorschein. Im 5D-Fall benötigt HILOMOT+ mit 12 lokalen Modellen 165 lokale Modelle weniger als LOLIMOT und weniger als ein Viertel der Rechenzeit, um auf die gleiche Modellgüte zu kommen.

4.7 Zusammenfassung

Um den Aufwand durch nichtlineare Optimierungsverfahren so gering wie möglich zu halten und das Training lokaler Modellnetze möglichst effizient zu gestalten, wurden in diesem Kapitel heuristische Verfahren vorgestellt und untersucht, die in der Lage sind, eine Optimierung der Modellstruktur automatisiert durchzuführen. Ausgehend vom LOLIMOT-Algorithmus wurden Verfahren entwickelt, die eine deutliche Verbesserung der Modellflexibilität durch den Einsatz achsenschräger Partitionierungen erlauben. Das neue Verfahren SUHICLUST kombiniert dazu die Vorteile der Baumkonstruktion mit der Flexibilität von Produktraum-Clusteringverfahren. Diese Modelle verwenden eine parallele Modellstruktur.

Ein weiteres Verfahren basiert auf einer hierarchischen Struktur, welche in dieser Arbeit umgesetzt wurde. Nach der Vorstellung der hierarchischen Partitionierung bot sich zunächst der Vergleich mit der parallelen Struktur an, wobei sich viele Vorteile der hierarchischen Struktur für ein achsenschräges Partitionierungsverfahren herauskristallisierten.

[5] Apple MacBook Pro, 2.53GHz Intel Core 2 Duo, 4GB 1067MHz DDR3.

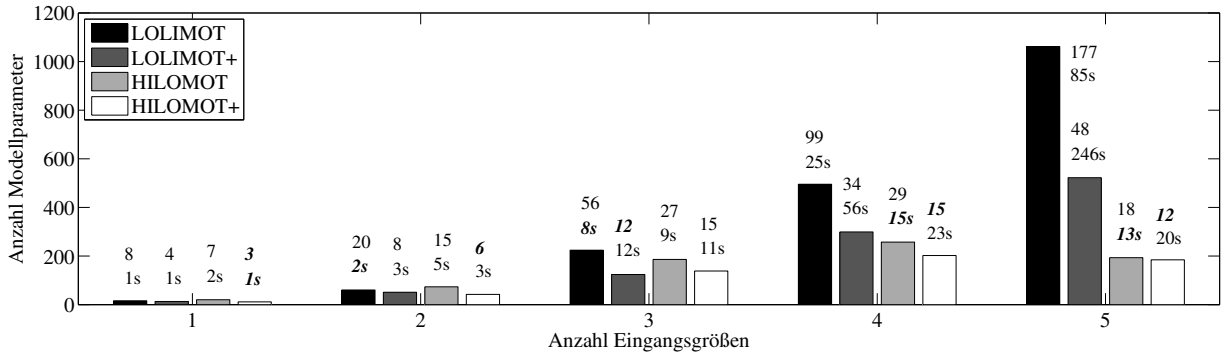

Bild 4.30: Vergleich der Methoden anhand des Testbeispiels für verschiedene Eingangsraumdimensionen. Die Balkenhöhen stehen für die benötigte Gesamtzahl an Modellparametern. Die obere Zahlenreihe zeigt die Anzahl der benötigten lokalen Modelle (LM). Die untere Zahlenreihe gibt die Trainingszeit an. Die jeweilig besten Zahlenwerte sind kursiv und fett hervorgehoben.

Der resultierende HILOMOT-Algorithmus basiert auf der Idee von Hinging Hyperplane Tree-Modellen (HHT) und verwendet verallgemeinerte Hinge-Funktionen [150], die unterschiedliche Eingangsräume für die Partitionierung und die lokalen Modelle erlauben. Zur Umsetzung der achsenschrägen Teilungen kommen Sigmoid-Funktionen zum Einsatz, die durch eine geschickte Normierung des Parametervektors ein adaptives Auflösungsvermögen der Partitionierung erlauben. Die Steilheit der Übergänge zwischen den lokalen Modellen ist umso steiler, je mehr sich die benachbarten LM-Zentren annähern.

Ein wichtiges Thema für praktische Anwendungen ist die Wahl der Modellkomplexität, da oft nur wenige Trainingsdaten zur Verfügung stehen und eine Validierung ohne separate Daten durchzuführen ist. Neben der rechnerisch aufwändigen Kreuzvalidierung bietet sich die Anwendung eines Informationskriteriums an. Die Spezifikation des korrigierten AIC-Kriteriums für lokale Modellnetze war daher ebenfalls Gegenstand dieses Kapitels.

Durch den Einsatz linearer Strukturselektionsverfahren eröffnet sich eine neue Perspektive für das Training lokaler Modellnetze. Die hier vorgestellte Methode der Strukturabwägung erlaubt es, während des Trainings mit HILOMOT nicht nur die wichtigsten Regressoren jedes Teilmodells zu selektieren, sondern darüber hinaus die Polynomkomplexität der LM individuell an verschiedene Regionen des Prozesses anzupassen. Dies erlaubt die Reduktion des Modells auf wenige Teilgebiete, die aber wiederum bezüglich der Regressoren so auf den Prozess abgestimmt sind, dass man sie zur lokalen Beschreibung des Prozessverhaltens sinnvoll nutzen kann.

Zum Abschluss fand ein Vergleich der vorgestellten Partitionierungsverfahren anhand eines analytischen Beispiels statt, der den Zugewinn an Effizienz bei der Modellbildung mit den neuen Verfahren untermauert.

5 Passive und aktive Messdatengewinnung mit HilomotDoE

Versuchsplanungsverfahren erfreuen sich mittlerweile in der Praxis größter Beliebtheit (siehe z.B. [72]), da durch Einsparung von Messungen mittels innovativer Messmethoden in den meisten technischen Disziplinen Ressourcen und Geld gespart werden können. Das vorliegende Kapitel beschäftigt sich daher mit der Fragestellung, wie man den Messprozess durch passive und aktive Verfahren effizienter gestalten kann. Insbesondere wird untersucht, inwiefern sich lokale Modellnetzansätze speziell im Fall nichtlinearer Prozesse sowohl für die Versuchsplanung als auch für das aktive Lernen eines Prozesses nutzen lassen.

Zunächst bietet Abschnitt 5.1 einen Überblick zu existierenden Ansätzen für die statistische Versuchsplanung. Lässt sich der vorliegende Prozess nicht mehr durch linear parametrierte Modellansätze beschreiben, können keine einfachen optimalen Versuchspläne mehr eingesetzt werden. Neben der Verwendung von geometrischen oder raumfüllenden Versuchsplänen bietet die Anwendung einer aktiven Lernstrategie das Potential, ein nichtlinear parametriertes Modell während der Messung zur aktiven Planung der Messpunkte einzusetzen, siehe Abschnitt 5.2. In Abschnitt 5.3 werden schließlich verschiedene Strategien vorgestellt, wie sich im Speziellen Hilomot-Modelle für die passive und aktive Versuchsplanung einsetzen lassen. Die Verfahren werden unter dem Begriff „HilomotDoE“ zusammengefasst. Ferner wird in Kapitel 5.4 eine Methode zur automatisierten Versuchsbeendigung auf Basis der Modellgüte vorgestellt. Schließlich bietet Abschnitt 5.5 einen Ausblick auf die Verwendung von Komitees mit Hilomot-Modellen für die aktive Versuchsplanung, gefolgt von der Zusammenfassung des Kapitels.

5.1 Statistische Versuchsplanung

Die statistische Versuchsplanung (engl.: *Design of Experiments (DoE)*) stellt als Teilgebiet der Statistik ein Werkzeug zur Generierung von Messdaten mit möglichst geringem Versuchsaufwand dar, bzw. macht eine sinnvolle Vermessung hochdimensionaler Prozesse überhaupt erst möglich. Ihr Ziel ist die systematische Untersuchung von Zusammenhängen zwischen Einflussgrößen und interessierenden Zielgrößen [95]. Prinzipiell lassen sich die DoE-Methoden in zwei Kategorien einteilen [74, 107]:

1. *Klassische, geometrische Versuchspläne*: Die Verteilung der Daten erfolgt nach einem vorgegebenen Muster oder auch raumfüllend im Eingangsraum ohne jede Berücksichtigung des Prozessverhaltens. Bekannte Versuchspläne sind unter den folgenden Stichworten bekannt: *Full* und *Fractional Factorial*, *Central Composite* und *Box-Behnken*, *Latin Hypercube*, usw. [124].
2. *Optimale Versuchspläne*: Hierbei soll das Prozessverhalten mit einem linear parametrierten Modell, meist ein Polynom zweiten oder dritten Grades, beschrieben werden. Die Verteilung der Daten erfolgt dann bezüglich eines Optimalitätskriteriums. Meist

wird hierzu entweder die Varianz der Modellparameter oder die Varianz des Modellausgangs verwendet. Dafür muss eine Modellstruktur angenommen werden, die den Zusammenhang zwischen den Eingangsgrößen und dem Ausgang korrekt beschreibt. Typische Verfahren sind unter den Begriffen *D*-, *A*- und *V-Optimalität* bekannt [123].

Die Verfahren der ersten Kategorie sind universell einsetzbar. Die Daten werden problemunabhängig nach einem bestimmten Schema im Eingangsraum verteilt. Ist der zu vermessende und modellierende Prozess deterministisch oder nahezu deterministisch, bieten sich raumfüllende Versuchspläne an. Hierbei geht man davon aus, dass das Rauschen auf den Daten gering ist und die Reduktion des Biasfehlers im Vordergrund steht. Bias ist der Unterschied zwischen dem geschätzten Modell und dem wahren Prozessverhalten ohne Rauschen. Laut [73] lassen sich die *raumfüllenden* Versuchspläne in folgende Kategorien unterteilen: *Sphere Packing*, *Latin Hypercube*, *Uniform*, *Minimum Potential* und *Maximum Entropy*. Letztendlich haben alle Verfahren das Ziel einer möglichst gleichförmigen Verteilung durch z.B. Maximierung der minimalen Abstände aller Punktpaare. Weil Verfahren der ersten Kategorie keinerlei Strukturinformationen über den (vermuteten) Zusammenhang zwischen den Eingangsgrößen und dem Ausgang in die Versuchsplanung einfließen lassen, können sie u.U. auch nicht besonders effektiv sein. Dieser Nachteil soll durch Verfahren der zweiten Kategorie vermieden werden.

Die optimale Versuchsplanung kann insbesondere im Bereich der Motorenentwicklung als eines der bekanntesten Standardverfahren angesehen werden. Grund dafür ist, dass in der Praxis Polynommodelle sehr etabliert sind und in allen gängigen Auswertungswerkzeugen für Messdaten eingesetzt werden. Deshalb wird das Verfahren hier näher erklärt.

Für ein linear parametriertes Modell mit m Regressoren und N Datenpunkten ergibt sich der Modellausgang $\underline{\hat{y}}$ zu:

$$\underline{\hat{y}} = \underline{X}\underline{w} = [\underline{x}_0\ \underline{x}_1\ \cdots\ \underline{x}_m] \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_m \end{bmatrix} \tag{5.1}$$

mit $\underline{x}_i = [x_i(1)\ x_i(2)\ \cdots\ x_i(N)]^T$, $i = 0, \ldots, m$. Der geschätzte Parametervektor lautet nach der Least Squares-Methode:

$$\underline{\hat{w}} = (\underline{X}^T\underline{X})^{-1}\underline{X}^T\underline{y}\,. \tag{5.2}$$

Die Güte der geschätzten Parameter lässt sich durch die Kovarianzmatrix wie folgt ausdrücken:

$$\mathrm{cov}(\underline{\hat{w}}) = \mathrm{E}\left\{(\underline{\hat{w}} - \mathrm{E}\{\underline{\hat{w}}\})\ (\underline{\hat{w}} - \mathrm{E}\{\underline{\hat{w}}\})^T\right\}\,. \tag{5.3}$$

Nimmt man nun an, dass $\underline{y} - \mathrm{E}\{\underline{y}\}$ die Verteilung $\mathcal{N}(0, \sigma_n^2\underline{I})$ hat, folgt daraus, dass $\underline{\hat{w}} - \mathrm{E}\{\underline{\hat{w}}\}$ ebenfalls mittelwertfrei gaussverteilt ist mit der Kovarianzmatrix:

$$\mathrm{cov}(\underline{\hat{w}}) = \sigma_n^2\ (\underline{X}^T\underline{X})^{-1} = \mathcal{F}^{-1}\,. \tag{5.4}$$

Dabei entspricht σ_n^2 der Varianz des Messrauschens. Die Matrix $\underline{X}^T\underline{X}$ beinhaltet die Information, welche Sensitivität $\underline{\varphi}(k) = \frac{\partial}{\partial \underline{w}}\hat{y}(k,\underline{w}) = [x_0(k)\ x_1(k)\ \dots\ x_m(k)]$ das Modell bezüglich der Parameter besitzt. Daher wird der Ausdruck

$$\mathcal{F} = \frac{1}{\sigma_n^2}(\underline{X}^T\underline{X}) = \frac{1}{\sigma_n^2}\sum_{k=1}^{N}\frac{\partial\hat{\underline{y}}(k,\underline{w})}{\partial\underline{w}}^T\frac{\partial\hat{\underline{y}}(k,\underline{w})}{\partial\underline{w}} = \frac{1}{\sigma_n^2}\sum_{k=1}^{N}\underline{\varphi}(k)^T\underline{\varphi}(k) \tag{5.5}$$

auch als *Fisher Informationsmatrix* bezeichnet, siehe z.B. [35] oder [101]. Wie man an der Gleichung erkennt, hängt die Information zum einen vom Rauschen σ_n^2, zum anderen von der Wahl der Regressionsmatrix $\underline{X}$ ab. Mit den gegebenen Regressoren, z.B. für ein Polynommodell zweiten Grades, lässt sich also die Informationsmatrix lediglich durch die Wahl der Messdatenplatzierung beeinflussen.

Generell möchte man die Messpunkte für ein Experiment so planen, dass der Informationsgehalt für die Parameterschätzung maximal wird. Da die Fisher-Information die Gestalt einer Matrix hat, muss zur konkreten Quantifizierung des Informationsgehalts mit einem skalaren Kriterium gearbeitet werden, nach dem dann vor dem Experiment die Lage der zu vermessenden Eingangswerte optimiert wird. Verschiedene Optimalitätskriterien zur Versuchsplanung sind in der Praxis üblich (siehe z.B. [18]):

- Ein Versuchsplan $\underline{X}$ heißt D-optimal, wenn die Determinante der Kovarianzmatrix minimal, d.h. $|(\underline{X}^T\underline{X})^{-1}| \to \min$ oder umgekehrt die Determinante der Informationsmatrix maximal ist, d.h. $|\underline{X}^T\underline{X}| \to \max$.
- Als A-optimal bezeichnet man den Versuchsplan, wenn die Summe der Diagonalelemente der Kovarianzmatrix minimal ist, d.h. spur $\left((\underline{X}^T\underline{X})^{-1}\right) \to \min$.
- Beim E-optimalen Versuchsplan wird der größte Eigenwert der Kovarianzmatrix minimiert.

Eine zusätzliche Erweiterung der Versuchsplanung ist es, die Varianz des Modellausgangs zu betrachten. Möchte man die Varianz des Modells auf neuen Daten $\widetilde{\underline{X}}$ berechnen, ergibt sich die Kovarianzmatrix des Modellausgangs zu:

$$\begin{aligned}
\mathrm{cov}(\underline{\hat{y}}) &= \mathrm{E}\left\{\left(\underline{\hat{y}} - \mathrm{E}\{\underline{\hat{y}}\}\right)\left(\underline{\hat{y}} - \mathrm{E}\{\underline{\hat{y}}\}\right)^T\right\} \\
&= \mathrm{E}\left\{\left(\widetilde{\underline{X}}\left(\underline{\hat{w}} - \mathrm{E}\{\underline{\hat{w}}\}\right)\right)\left(\widetilde{\underline{X}}\left(\underline{\hat{w}} - \mathrm{E}\{\underline{\hat{w}}\}\right)\right)^T\right\} \\
&= \widetilde{\underline{X}}\ \mathrm{E}\left\{\left(\underline{\hat{w}} - \mathrm{E}\{\underline{\hat{w}}\}\right)\left(\underline{\hat{w}} - \mathrm{E}\{\underline{\hat{w}}\}\right)^T\right\}\widetilde{\underline{X}}^T \\
&= \widetilde{\underline{X}}\mathrm{cov}(\underline{\hat{w}})\widetilde{\underline{X}}^T .
\end{aligned} \tag{5.6}$$

Ziel hierbei ist die Optimierung der Modellgüte bezüglich des Varianzfehlers. Die Versuchsplanung für $\underline{X}$ kann dann z.B. V-optimal erfolgen, sodass die gemittelte Varianz bezüglich ausgewählter Punkte $\widetilde{\underline{X}}$ minimal ist. Weitere Kriterien sind G-optimal, wobei die maximale Varianz minimiert wird oder I-optimal, wenn das Integral über alle möglichen Messwertplatzierungen, d.h. über den gesamten Versuchsraum, minimal wird [126].

Zur Veranschaulichung, wie die optimale Versuchsplanung funktioniert, zeigt Bild 5.1 ein einfaches Beispiel. Möchte man einen Versuch zur Schätzung einer Geraden mit $y(u) = w_0 + w_1 u$ planen, sind die Messpunkte im Versuchsraum $u \in [0\ 1]$ möglichst

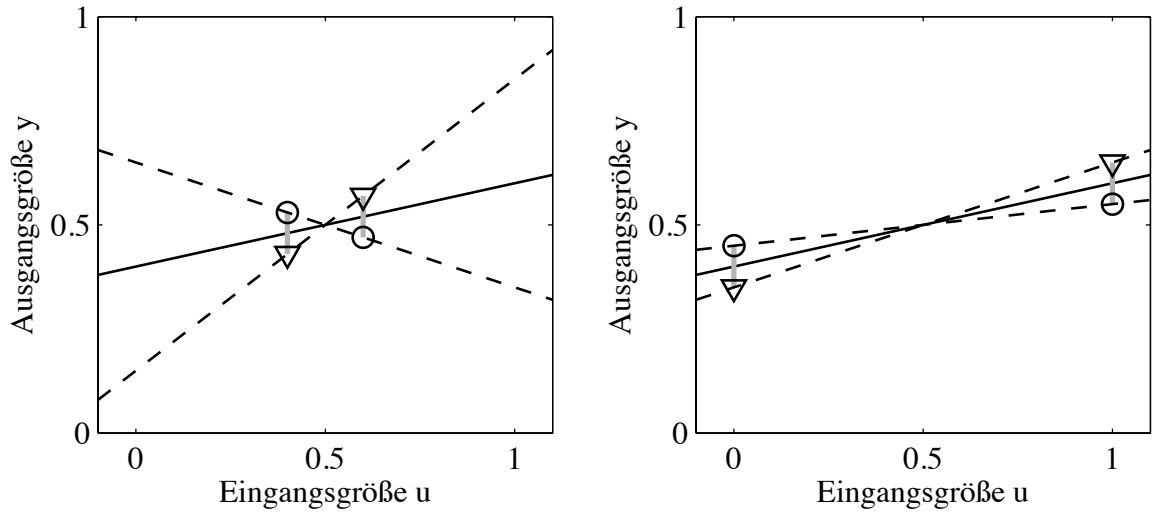

Bild 5.1: Einfluss der Versuchspunkte auf die Varianz der Modellparameter am Beispiel einer 1D-Geraden. Links liegen die Messpunkte bezüglich u dicht beieinander. Hier ist die Varianz der Schätzung hoch und der Informationsgehalt niedrig ($|\underline{X}^T\underline{X}| = 0.16$). Je weiter die Punkte bei diesem Modell auseinander liegen, desto sicherer wird die Schätzung. Rechts ist demzufolge die Varianz gering und der Informationsgehalt hoch ($|\underline{X}^T\underline{X}| = 4$).

weit voneinander entfernt zu planen. Unterliegen die Messwerte y Schwankungen, etwa durch Messrauschen, so wirken sich diese wesentlich stärker aus, wenn die Punkte dicht beieinander platziert werden (Bild 5.1 links), als wenn sie weit auseinander liegen (Bild 5.1 rechts). Dementsprechend wird $|\underline{X}^T\underline{X}|$ immer größer je breiter die Punkte im Versuchsraum platziert werden.

Eine erstaunliche Erkenntnis bei der optimalen Versuchsplanung ist, dass die Güte der Versuchsplanung ausschließlich von den Eingangswerten abhängt. Das bedeutet, dass die Messungen unabhängig von den Messwerten, also *vor* der eigentlichen Messung, geplant werden können. Allerdings muss man sich im Klaren darüber sein, dass dies nur aufgrund einer getroffenen Modellannahme möglich ist. Die Annahme dahinter ist ein linear parametriertes, biasfreies Modell, mit dem sich der zu messende Prozess prinzipiell perfekt beschreiben lässt.

Aus praktischer Sicht bezwecken alle diese Ansätze eine Reduktion des Varianzfehlers. Hieraus leitet sich auch gleichzeitig der größte Nachteil dieser Methode ab, nämlich die Vernachlässigung des Biasfehlers. Wird die Komplexität des Modells so gewählt, dass das Modell zu unflexibel für den wahren Prozess ist, dann ist die Versuchsplanung bezüglich dieses Modells zwar optimal, kann aber trotzdem den echten Prozess nicht gut abbilden. Gleiches gilt, wenn ein *linear* parametriertes Modell verwendet wird, obwohl der Prozess *nichtlinear* ist.

Die genannten Schwierigkeiten bei der statistischen Versuchsplanung mit linear parametrierten Modellen führen dazu, dass die Methode des *aktiven Lernens* immer mehr an Popularität gewinnt.

5.2 Aktives Lernen

Mit den optimalen Versuchsplanungsverfahren im vorangegangenen Kapitel berechnet man *vor* der Messung einen Satz an zu messenden Punkten, um *nach* der Messung Optimalität zu erreichen. Man erkennt, dass hierbei die Messwerte keinerlei Einfluss auf die Optimalitätsbedingungen haben. Mit anderen Worten: Der Informationsgewinn durch das Messen kann nicht zur Verbesserung der Versuchsplanung genutzt werden. Erstaunlicherweise bedeutet dies in der Theorie keinen Nachteil, solange der Prozess dem angenommenen Modellverhalten auch gehorcht.

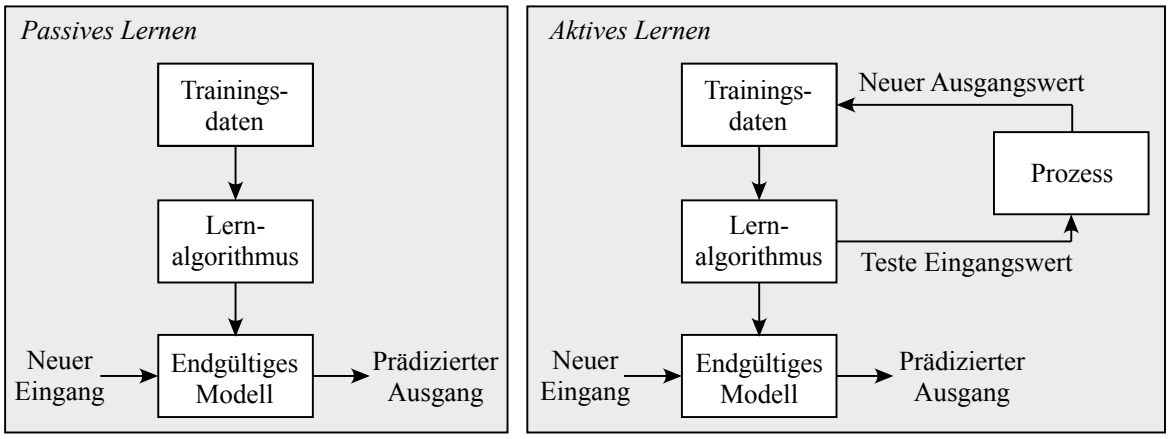

Bild 5.2: Vergleich von passivem und aktivem Lernen. Im Gegensatz zum passiven Lernen erweitert der aktive Lernalgorithmus iterativ den aktuellen Datensatz und platziert neue Messpunkte basierend auf den bis dahin gemessenen Werten. Dieser Ansatz kann die Modellgüte signifikant verbessern, hat aber leider auch den Nachteil eines hohen Rechen- und evtl. Zeitbedarfs [27].

Anders ist das bei Prozessen, die durch nichtlinear parametrierte Modelle zu beschreiben sind. Hierbei liefert jede Messung sehr wohl Information, die zur Verbesserung der Versuchsplanung genutzt werden kann. Dennoch lässt sich bei vielen Anwendungen beobachten, dass der Vorgang des Messens meist komplett vom nachfolgenden Schritt der Modellierung entkoppelt ist. Ansätze zur Modellbildung werden zumeist als passive Empfänger von Daten angesehen. Der Vorteil, den Prozess aktiv durch eine Interaktion zwischen Modell und Prozess beeinflussen zu können, wird vollständig außer Acht gelassen. Wie beispielsweise in [28], [27] oder [132] gezeigt, können aktive Lernstrategien die Modellgüte bei gleicher Anzahl an Messungen deutlich verbessern. Allerdings muss permanent während der Messung ein Modell identifiziert werden, was natürlich sehr rechenintensiv sein kann. Für die praktische Umsetzung des aktiven Lernens ist daher vor allem eine Sache wichtig: Ein flexibles nichtlineares Modell, welches sowohl schnell als auch mit einer hohen Robustheit erstellt werden kann.

Das Gebiet des aktiven Lernens beschäftigt sich mit der Fragestellung, wie man während der Messung eine Interaktion zwischen Prozess und Modell ermöglichen kann. Wie in Bild 5.2 gezeigt, wird im Gegensatz zu passivem Lernen der Trainingsdatensatz während der Messung permanent erweitert, indem das Lernverfahren auf Basis der gegenwärtigen Trainingsdaten neue zu vermessende Eingangswerte, so genannte *Queries*, vom Prozess anfordert, mit denen dann die Trainingsdaten ergänzt werden. Je mehr Daten vorhanden sind,

desto genauer passt sich das Modell mit Hilfe des Lernalgorithmus' an den Prozess an und desto präziser gestaltet sich die Platzierung des neuen Queries bezüglich eines *Querykriteriums*. Einerseits möchte man dabei die Modellgüte verbessern, was typischerweise auch der Fall ist, und andererseits den Rechenaufwand möglichst gering halten. Das Hauptziel ist jedoch, mit so wenigen Messpunkten wie möglich ein gutes Modell zu erstellen [27].

Während der Messung benötigt man demnach eine Strategie, mit der neue Messpunkte geplant werden. Dazu definiert man ein Querykriterium, nach dem eine Menge an Kandidatenpunkten, die für die nächste Messung in Frage kommen, bezüglich des zu erwartenden Informationsgewinns bewertet werden. Das Ziel ist dann, den erwarteten Informationsgewinn mit Hilfe des Querykriteriums zu maximieren. Als Maß für den zu erwartenden Informationsgewinn sind viele Querykriterien denkbar und diese hängen immer vom individuellen Anwendungsfall ab. Meist fällt auch hier, wie bei der optimalen Versuchsplanung, die Wahl auf die Varianz der Parameterschätzung oder die Varianz des Modellausgangs. Darüber hinaus sind Modellkomitees beliebt, bei denen man den Dissens des Komitees berechnet. Dort wo sich die Modelle am meisten widersprechen, definiert man den Punkt maximalen Informationsgewinns.

5.3 Versuchsplanung mit Hilomot-Modellen

Dieser Abschnitt stellt drei verschiedene Strategien vor, die eine Verwendung des Hilomot-Algorithmus' zur Versuchsplanung erlauben. Diese Methoden werden im Kapitel mit der Abkürzung „HilomotDoE“ für „*Hierarchical Local Model Tree for Design of Experiments*“ zusammengefasst. Zunächst wird der einfachste Ansatz vorgestellt. Hier geht es um die Fragestellung, wie man eine achsenschräge Hilomot-Partitionierung dazu nutzen kann, um offline die nächste Messung eines ähnlichen Prozesses zu planen. Danach konzentriert sich der Abschnitt auf die aktive Versuchsplanung, die online während der Messung mit Hilfe des Hilomot-Algorithmus' die aktuellen Messdaten analysiert und davon abhängig die nächsten Messungen plant. Hierbei lässt sich das aktive Lernverfahren HilomotDoE in zwei Kategorien einteilen: Aktives Lernen mit Batch-Query-Strategie und aktives Lernen mit Single-Query-Strategie. Letzteres Verfahren ist das effizienteste, da es nach jeder Messung das Modell neu trainiert und jeweils nur ein Query-Punkt an den Prozess übergibt.

5.3.1 Passive Versuchsplanung mit Hilomot

Wie z.B. in [151] gezeigt, können Partitionierungen, die mit Lolimot erstellt wurden, effizient zur passiven Versuchsplanung verwendet werden. Dabei kann man das Wissen über die Nichtlinearität eines Prozesses im Gegensatz zu einer Gittervermessung zur Platzierung der Messpunkte ausnutzen. Je nach Prozess und der Anzahl an Eingangsgrößen schlägt beim Gitteransatz der „Fluch der Dimensionalität“ voll zu Buche. Um dies durch geschickte Versuchsplanung zu umgehen, kann ein vergleichbarer, in der Vergangenheit vermessener Prozess, für den bereits Messdaten vorliegen, mit lokalen Modellnetzen trainiert und auf Basis der erzeugten Partitionierung ein Versuchsplan erstellt werden. Dahinter steckt die Annahme, dass sich die Nichtlinearitäten des alten und neuen Prozesses nicht wesentlich

unterscheiden. Man erwartet daher auch, dass die Partitionierung zum alten Prozess und die Partitionierung zum neuen Prozess sehr ähnlich sind. Denkbar ist z.B. ein Einsatz des Verfahrens im Bereich der Motorenapplikation, wo zwar verschiedene Motoren zu vermessen sind, aber das grundlegende Prozessverhalten der Motoren ähnlich ist.

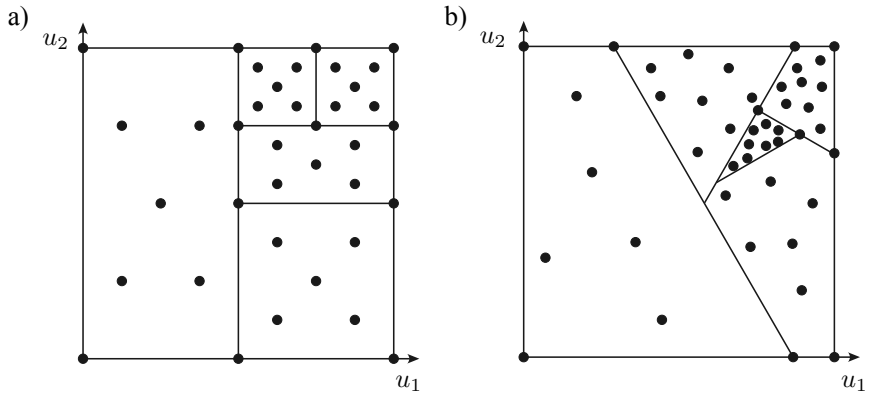

Bild 5.3: a) Geometrische Messpunktplatzierung bei gegebener LOLIMOT-Partitionierung. b) Versuchsplanung bei achsenschräger Partitionierung. Jedem lokalen Modell wird die gleiche Anzahl an Punkten zugewiesen, um die Versuchsplanung dem Prozessverhalten zu adaptieren.

Die Verteilung der stationären Messpunkte gestaltet sich für LOLIMOT-Partitionierungen unproblematisch, weil der LOLIMOT-Algorithmus achsenorthogonale Teilungen erzeugt und sich die lokalen Modelle zu Hyperquadern ergeben. Eine geometrische Beschreibung der Gültigkeitsgebiete ist daher ohne großen Aufwand möglich. Wenn man jedem lokalen Modell ein vorher festgelegtes, geometrisches Versuchsmuster unterlegt, z.B. Messpunkte auf das Zentrum, in jede „Ecke“ und auf die Diagonalen des lokalen Modells (Bild 5.3a), so ist durch die unterschiedlich große Ausdehnung der Gültigkeitsgebiete automatisch eine sinnvolle, globale Verteilung der Messpunkte sichergestellt. Bei diesem Versuchsplan setzt man die Messpunkte automatisch dicht in Regionen, wo der Prozess stark nichtlinear ist. In Regionen, wo nahezu lineares Prozessverhalten herrscht, können die Messungen sparsamer platziert werden.

Dieselbe Idee soll nun auf achsenschräge Partitionierungen angewendet werden, wie dies im Bild 5.3b illustriert ist. Beim HILOMOT-Algorithmus kommt es jedoch zu der Schwierigkeit, dass eine geometrische Verteilung der Datenpunkte aufgrund der achsenschrägen Partitionierung nicht mehr so einfach möglich ist, da sich die Gültigkeitsfunktionen aus der Multiplikation mehrerer Teilungsfunktionen entlang eines Modellbaumes ergeben. Um die geometrische Form der lokalen Gültigkeitsgebiete zu bestimmen, muss eine der hierarchischen Struktur angepasste Alternative gefunden werden, damit eine Messpunkteverteilung unter Voraussetzung achsenschräger Teilungen möglich wird.

Anstelle einer geometrischen Platzierung der Messpunkte wird vorgeschlagen, die Punkte mit einer raumfüllenden Versuchsplanung festzulegen. Das LOLIMOT-Konzept, in jedem lokalen Modell die gleiche Punktzahl zu generieren, wird bei einer vorliegenden HILOMOT-Partitionierung beibehalten, um auch in diesem Fall eine adaptive Punkteverteilung zu ermöglichen. Das Vorgehen ist dann wie folgt:

Für jedes lokale Modell werden separat die Punkte im jeweiligen Gültigkeitsgebiet erzeugt. Die Reihenfolge zur Auswahl der lokalen Modelle ergibt sich durch die Baumstruktur der Partitionierung. Lokale Modelle, die im Baum eine hohe Hierarchiestufe[1] haben, füllt man zuerst mit Versuchspunkten auf. Nacheinander werden dann die Punkte in den lokalen Modellen mit niedrigeren Hierarchiestufen unter Berücksichtigung der bereits vorhandenen Punkte platziert. Die Punktevergabe ist von der Reihenfolge der lokalen Modelle abhängig, da bereits aufgefüllte LM die Platzierung weiterer Punkte in anderen LM-Gebieten beeinflussen.

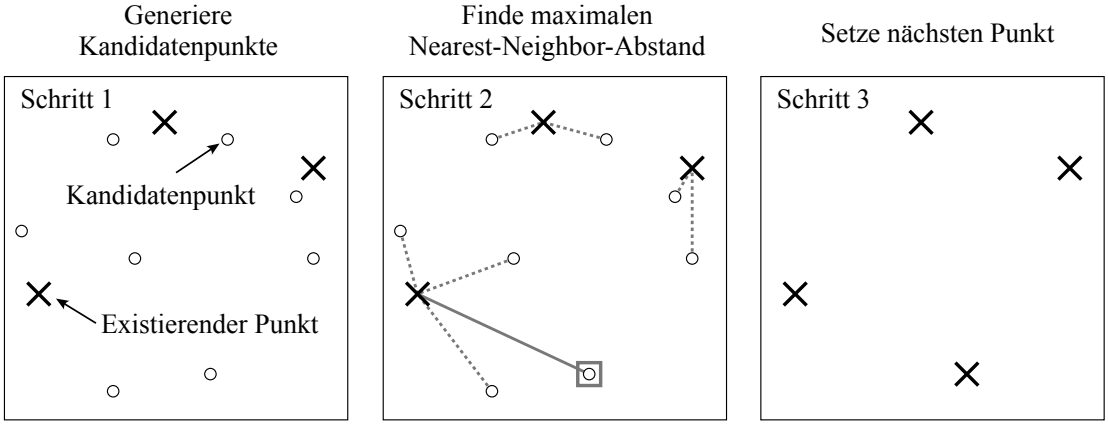

Bild 5.4: Generelle Vorgehensweise der vorgestellten Methode zur raumfüllenden Versuchsplanung. Hierbei ist das Ziel, den Nearest-Neighbor-Abstand zu allen existierenden Punkten zu maximieren.

Die hier vorgestellte Methode zur raumfüllenden Versuchsplanung ist inspiriert durch die so genannten *Sphere Packing*-Versuchspläne [125] und die Erzeugung von Pseudo-Zufallszahlen wie z.B. Halton-Sequenzen [43] oder Sobol-Sequenzen [140], die z.B. zur numerischen Approximation hochdimensionaler Integrale eingesetzt werden. Das Ziel bei der Versuchsplanung ist, eine möglichst homogene Verteilung der Punkte im jeweiligen lokalen Modell unter Berücksichtigung bereits existierender Messpunkte zu gewährleisten. Wie in Bild 5.4 gezeigt, erfolgt die Platzierung in drei Schritten:

1. Generiere einen Satz von Kandidatenpunkten. Damit die Kandidatenpunkte innerhalb eines lokalen Modells liegen, können sie mit der Bedingung $\max_i(\underline{\Phi}_i(\underline{u}_j))$, $i = 1, \ldots, M$ klassifiziert werden. Damit erreicht man, dass ausschließlich im Gültigkeitsgebiet des LM neue Punkte erzeugt werden können.
2. Zu allen Kandidatenpunkten wird der nächste, existierende Nachbarpunkt (engl: *Nearest Neighbor*, Abk.: NN) gesucht und der Abstand berechnet.
3. Der Kandidatenpunkt mit dem größten Nearest-Neighbor-Abstand wird als nächster zu messender Punkt gesetzt.

Ziel der Strategie ist schließlich, in jedem lokalen Modell dieselbe Anzahl an Messpunkten zu platzieren, um damit die Dichte der Messpunkte an das lokale Prozessverhalten zu adaptieren.

[1] Die Bezeichnung „Hierarchiestufe“ meint die Anzahl der Teilungsfunktionen Ψ_i, die zur Berechnung einer Gültigkeitsfunktion Φ_j miteinander multipliziert werden. Die nullte Hierarchiestufe wäre demnach die Wurzel des Teilungsbaums.

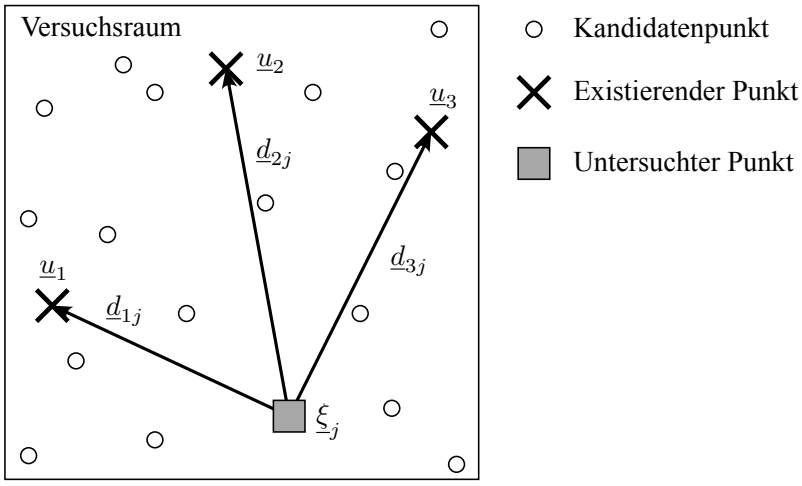

Bild 5.5: Abstandsberechnung im Versuchsraum.

Nachdem das grundlegende Prinzip erklärt wurde, wird nun der konkrete Algorithmus zur raumfüllenden Strategie vorgestellt:

1. Erzeuge einen Kandidatensatz $\mathcal{X} \in \mathbb{R}^p$ von gleichförmig verteilten Zufallspunkten $\{\underline{\xi}_j\}_{j=1}^{N_c}$. Hierbei sind p die Anzahl der Eingangsgrößen und N_c die Anzahl der Kandidatenpunkte. Standardmäßig verwendet das Verfahren eine Sobol-Sequenz zur Erzeugung des Kandidatensatzes. Die Punkte werden so erzeugt, dass sie innerhalb des Versuchsraums und der Gültigkeitsfunktion des lokalen Modells liegen.
2. Berechne die Abstandsmatrix $\underline{D} = \{d_{ij}\} \in \mathbb{R}^{N \times N_C}$ von allen Kandidatenpunkten $\{\underline{\xi}_j\}_{j=1}^{N_c}$ zu allen existierenden Punkten $\{\underline{u}_i\}_{i=1}^{N}$, siehe Bild 5.5:

$$\underline{D} = \begin{bmatrix} d_{11} & d_{12} & \cdots & d_{1N_c} \\ d_{21} & d_{22} & \cdots & d_{2N_c} \\ \vdots & \vdots & \ddots & \vdots \\ d_{N1} & d_{N2} & \cdots & d_{NN_c} \end{bmatrix} . \tag{5.7}$$

   Die Abstände d_{ij}^2 entsprechen der Mahalonobis-Distanz:

$$d_{ij}^2 = ||\underline{u}_i - \underline{\xi}_j||_{\underline{\Sigma}}^2 = (\underline{u}_i - \underline{\xi}_j)^T \underline{\Sigma} (\underline{u}_i - \underline{\xi}_j) \,, \tag{5.8}$$

   mit der Kovarianzmatrix $\underline{\Sigma}$. Bei gleich skalierten Achsen entspricht die Kovarianzmatrix der Einheitsmatrix ($\underline{\Sigma} = \underline{I}$) und die Mahalonobis-Distanz ist gleichbedeutend mit dem euklidischen Abstand.
3. Die j-te Spalte der Abstandsmatrix $\underline{D}$ enthält die Abstände eines Kandidatenpunktes $\underline{\xi}_j$ zu allen existierenden Punkten $\{\underline{u}_i\}_{i=1}^{N}$. Mit der Abstandsmatrix lässt sich dann zu jedem Kandidatenpunkt $\underline{\xi}_j$ der nächste Nachbarpunkt mit dem Abstand $d_{\mathrm{NN,j}}$ bestimmen. Damit erhält man den NN-Abstandsvektor

$$\underline{d}_{\mathrm{NN}} = \left[\min \begin{pmatrix} d_{11} \\ d_{21} \\ \vdots \\ d_{N1} \end{pmatrix}, \min \begin{pmatrix} d_{12} \\ d_{22} \\ \vdots \\ d_{N2} \end{pmatrix}, \cdots, \min \begin{pmatrix} d_{1N_c} \\ d_{2N_c} \\ \vdots \\ d_{NN_c} \end{pmatrix} \right] . \tag{5.9}$$

4. Ist zu jedem Kandidatenpunkt der nächste Nachbarpunkt bestimmt, wird derjenige als nächster Messpunkt übernommen, der den größten Eintrag in $\underline{d}_{\mathrm{NN}}$ hat, d.h. den größten NN-Abstand aufweist.

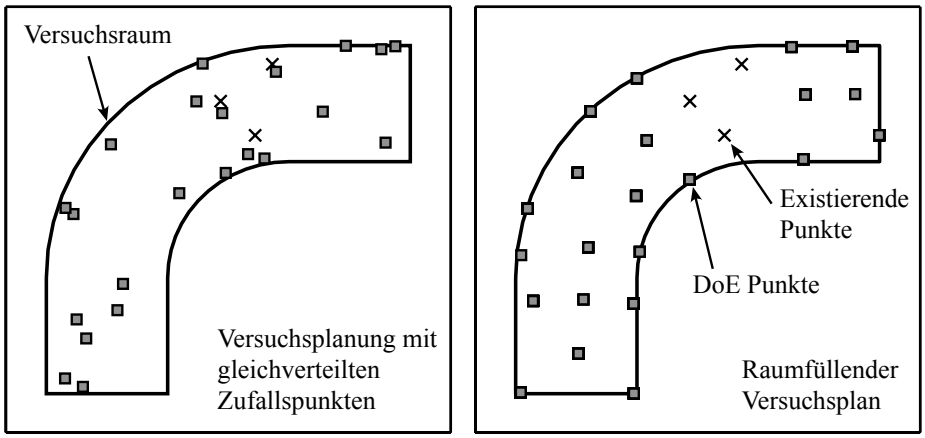

Bild 5.6: Vergleich von Queries, die mit gleichverteilten Zufallszahlen erzeugt wurden (links) mit raumfüllend verteilten Queries (rechts). Der raumfüllende Ansatz erzeugt eine homogene Verteilung unter Berücksichtigung existierender Punkte.

Wie in Bild 5.6 zu erkennen ist, verteilt das Verfahren die Messpunkte deutlich homogener als dies der Fall bei gleichverteilten Zufallspunkten wäre. Außerdem ist es damit möglich, die bereits vermessenen Punkte beim Setzen neuer Punkte zu berücksichtigen, sodass z.B. nicht die Gefahr besteht, an der gleichen Stelle mehrmals zu messen.

Eine zusätzliche wichtige Eigenschaft der vorgestellten Versuchsplanung ist, dass man für praktische Anwendungen a priori-Wissen über die Prozessgrenzen berücksichtigen kann. Dies ist einfach möglich, indem die Kandidatenpunkte durch die Verwendung eines Klassifikators wie z.B. einer konvexen Hülle oder einer Support Vector Machine nur innerhalb des Versuchsraums erzeugt werden.

Bei den in dieser Arbeit getätigten Untersuchungen war die diskrete Maximierung des Nearest-Neighbor-Abstandes mit Hilfe eines Kandidatensatzes immer zielführend. Als Ausblick auf zukünftige Untersuchungen sollte natürlich nicht außer Acht gelassen werden, dass man das Optimierungsproblem auch kontinuierlich mittels nichtlinearer Optimierungsverfahren lösen kann. Eine Einbindung der Prozessgrenzen ist dann allerdings schwieriger und muss über Nebenbedingungen geschehen. Bei der Optimierung ergeben sich neue Herausforderungen bezüglich der Robustheit des Verfahrens. Jedoch bietet gerade bei höherdimensionalen Problemen ein kontinuierlicher Ansatz viel Potential, die Punkte noch besser zu platzieren. Im diskreten Fall können bei vielen Eingangsgrößen große „Lücken“ entstehen, auch wenn sehr viele Punkte als Kandidaten erzeugt werden. Das Optimum bezüglich diskreter Werte kann daher unter Umständen deutlich vom wahren Optimum abweichen.

5.3.2 Aktives Lernen mit Batch-Query-Strategie

Beim aktiven Lernen mit der sogenannten Batch-Query-Strategie laufen zwei Vorgänge simultan ab: Das Training mit HILOMOT und die Berechnung neuer Query-Punkte. Wie bei der passiven Strategie ist die grundlegende Regel, jedem lokalen Modell die gleiche

Punktzahl zuzuweisen. In jeder Iteration des Partitionierungsalgorithmus' verfeinert man das globale Modell, indem ein lokales Modell hinzugefügt wird. Die lokalen Modelle werden auf ihre jeweilige Punktzahl geprüft und mit neuen Queries aufgefüllt, falls sich dort weniger Punkte als vorgegeben befinden. Dadurch erzeugt HILOMOTDOE in jeder Iteration die gleiche Anzahl neu zu messender Queries. Die Verteilung der Query-Punkte erfolgt nach dem raumfüllenden Verfahren, das im vorangegangenen Abschnitt erläutert wurde. Da es sich hierbei um mehrere Punkte gleichzeitig handelt, die dem Prozess pro Iteration übermittelt werden, spricht man von einem *Batch-Query*.

Um die aktive, komplexitätsabhängige Versuchsplanung zu ermöglichen, läuft das Verfahren folgendermaßen ab:

1. *Initialisierung*: Generiere ein initiales Batch-Query $\underline{U}_{ini} \in \mathbb{R}^{N_{ini} \times p}$, welches zuerst am Prozess vermessen wird. Standardmäßig erfolgt die Verteilung der Initialpunkte raumfüllend.
2. *Trainiere Initialmodell*: Nach der Messung wird ein initiales Modell mit HILOMOT trainiert. Das Modell wird so lange geteilt, bis die geforderte Mindestpunktzahl in einem LM unterschritten wurde. Zum Zählen der Anzahl der Messpunkte in jedem LM wird jeder Punkt aus $\{\underline{u}_j\}_{j=1}^{N}$ seinem Gültigkeitswert $\max(\underline{\Phi}_i(\underline{u}_j))$, $i = 1, \ldots, M$ entsprechend klassifiziert.
3. *Bestimme schlechtestes LM*: Finde das LM mit dem größten lokalen Fehlermaß $\max(\hat{\sigma}_i)$, $i = 1, \ldots, M$.
4. *Führe Teilung durch*: Unterteile dieses LM mit der achsenschrägen Schnittoptimierung des HILOMOT-Algorithmus' in zwei neue LM.
5. *Generiere Batch-Query*: Fülle die zwei LM bis zur geforderten Punktzahl N_{LM} mit einer raumfüllenden Versuchsplanung auf. Berücksichtige die bereits gemessenen Punkte beim Setzen neuer Queries. Da die Anzahl der Queries von der im LM vorliegenden Punktzahl abhängt, kommen allgemein N_{query} Punkte hinzu. Sofern allerdings vor der Teilung nicht mehr als N_{LM} Punkte im LM vorlagen, werden insgesamt $N_{query} = N_{LM}$ Punkte neu geplant.
6. *Messung*: Übermittle Batch-Query $\underline{U}_{query} \in \mathbb{R}^{N_{query} \times p}$ an den Prozess und führe die Messung durch.
7. *Schätze LM-Parameter*: Führe eine Schätzung mit den hinzugekommenen Datenpunkten für die Parameter der zwei neu erzeugten LM durch und berechne die lokalen Fehlermaße.
8. *Überprüfe Abbruchbedingung*: Wenn die Abbruchbedingung erfüllt ist, findet keine weitere Messung mehr statt und der Algorithmus ist beendet. Ansonsten gehe zu Schritt 3.

Als Abbruchbedingung sind z.B. die Vorgabe einer maximalen Punktzahl oder einer maximalen Messzeit möglich. Darüber hinaus bietet die Online-Auswertung des Leave-One-Out-Fehlers eine gute Möglichkeit, die Güte des Modells als Kriterium heranzuziehen.

Die Batch-Query-Strategie ist in den Bildern 5.7 und 5.8 anhand eines zweidimensionalen Hyperbelbeispiels mit $y = 0.1\,(0.1 + 0.5(1 - u_1) + 0.5(1 - u_2))^{-1}$ gezeigt. Pro LM sind 6 Messpunkte gefordert. Bild 5.7 zeigt vier mögliche Iterationen von HILOMOTDOE mit Batch-Query-Strategie. Nach jeder Verfeinerung des Modells werden die Punkte in den LM mit Queries aufgefüllt, falls zu wenige Punkte im LM liegen. Das Beispiel unterstreicht die prozessabhängige Verteilung der Messpunkte durch die adaptive Versuchsplanungsstrate-

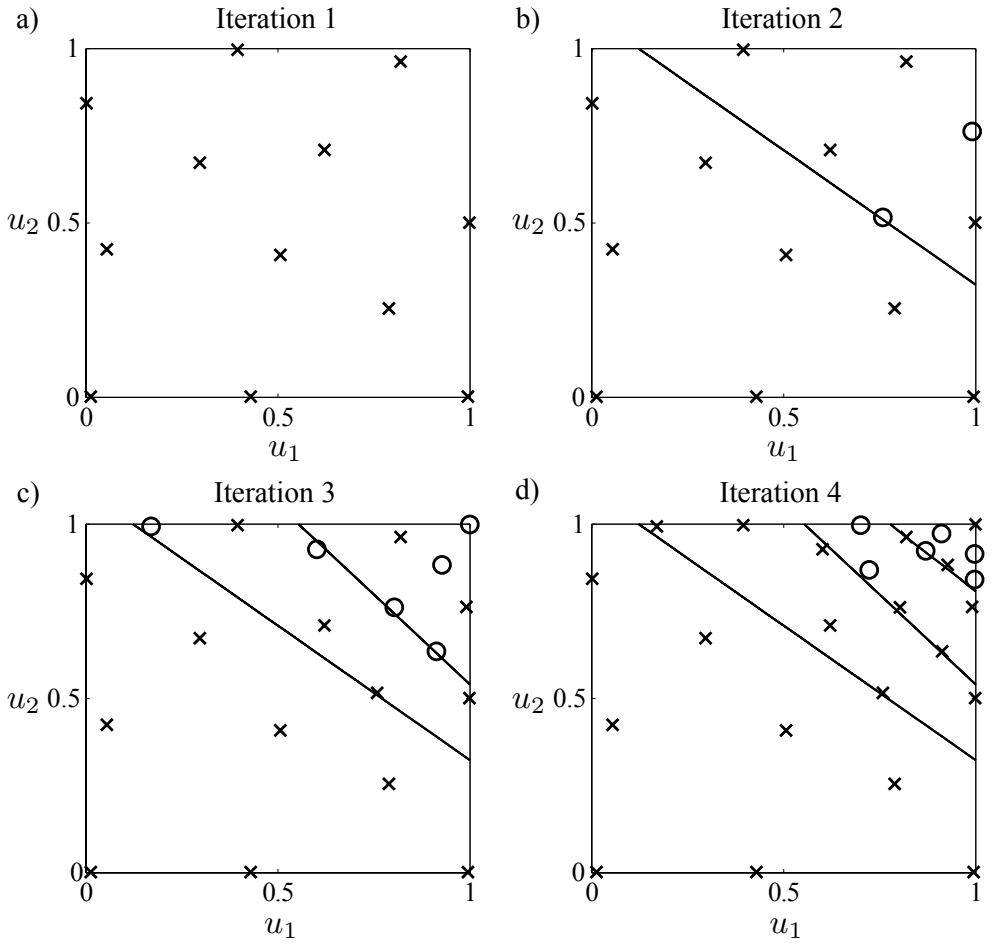

Bild 5.7: a)-d) Vier mögliche Iterationen von HILOMOTDOE mit Batch-Query-Strategie. Nach jedem Hinzufügen eines Schnittes überprüft der Algorithmus, ob in jedem LM genügend Messpunkte (×) vorhanden sind und fügt neue Query-Punkte (o) hinzu, falls die geforderte Punkteanzahl pro LM unterschritten ist. Standardmäßig wird die doppelte Anzahl der polynomialen LM-Parameter als Punkteanzahl gefordert.

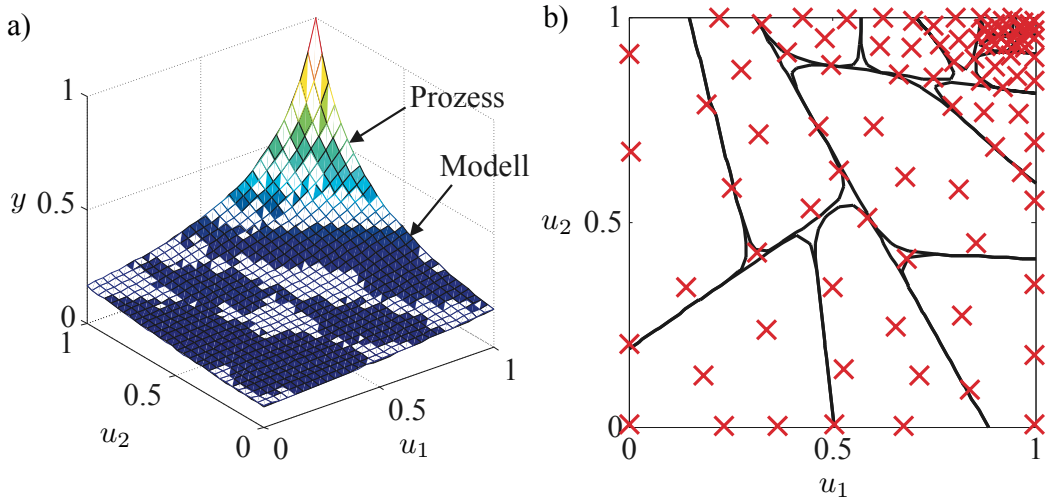

Bild 5.8: a) Vergleich zwischen Modell und Prozess. b) Partitionierung mit 6 Datenpunkten pro lokalem Modell und 90 Datenpunkten insgesamt.

gie. Exemplarisch zeigt Bild 5.8 den Vergleich zwischen dem resultierenden Modell und dem Prozess für die 6-Punkt-Strategie. Die Partitionierung zusammen mit den erzeugten Datenpunkten zeigt deutlich, dass gemäß der komplexitätsabhängigen DoE-Strategie in Bereichen größerer Nichtlinearität auch dichter gemessen wurde als in Bereichen, wo der Prozess annähernd lineares Verhalten aufweist.

Der HILOMOTDOE-Algorithmus mit Batch-Query-Strategie hat zwei Einstellparameter, die die Versuchsplanung beeinflussen: Zum einen die Punktanzahl pro LM N_{LM}, zum anderen die Anzahl der Initialpunkte N_{ini}. Für den Testprozess fand daher zudem eine Versuchsplanung mit drei verschiedenen Datenpunkt-Strategien statt (siehe Bild 5.9a). Bei der Gesamtzahl von 90 erzeugten Messpunkten erfolgte der Abbruch des HILOMOT-DOE-Algorithmus'. Hierbei hat sich herausgestellt, dass eine Strategie mit 6 Messpunkten pro lokalem Modell sowohl besser ist als eine 3-Punkt-Strategie als auch eine 9-Punkt-Strategie. Bild 5.9b zeigt den Zusammenhang der Datenpunkt-Strategie mit unterschiedlicher Rauschvarianz auf dem gemessenen Ausgang. Es zeigt sich, dass auch hier die Ergebnisse von der gewählten Strategie abhängen. Je mehr Datenpunkte pro LM geplant werden, desto robuster wird das Verfahren gegenüber Rauschen. Das liegt auch daran, dass bei gleicher Gesamtpunkteanzahl N weniger LM trainiert werden, wodurch die Versuchsplanung mehr explorativen Charakter hat und die Messwerte eher raumfüllend platziert, statt den Prozess in lokalen Gebieten feiner aufzulösen. Gleicher Zusammenhang wäre zu erkennen, wenn der Anteil der Initialpunkte an der Gesamtpunkteanzahl vergrößert wird. Letztlich lässt sich also der Algorithmus mit den zwei Parametern mehr explorativ und robust oder mehr investigativ und prozessauflösend einstellen.

Die Theorie der statistischen Versuchsplanung basiert auf der Annahme linear parametrierter Modelle. Wenn sich der Prozess durch ein solches Modell gut beschreiben lässt, kann von einer optimalen Versuchsplanung gesprochen werden und die Platzierung weiterer Messpunkte wäre der Theorie nach ohne Vorteile für das Modell. Jedoch kann der Einsatz von Polynommodellen in Verbindung mit statistischer Versuchsplanung unter Umständen gravierende Nachteile mit sich bringen, nämlich wenn es sich um einen nichtlinearen Prozess handelt. Bild 5.10 zeigt zwei künstliche Beispiele, die als Worst-Case-Szenario für die

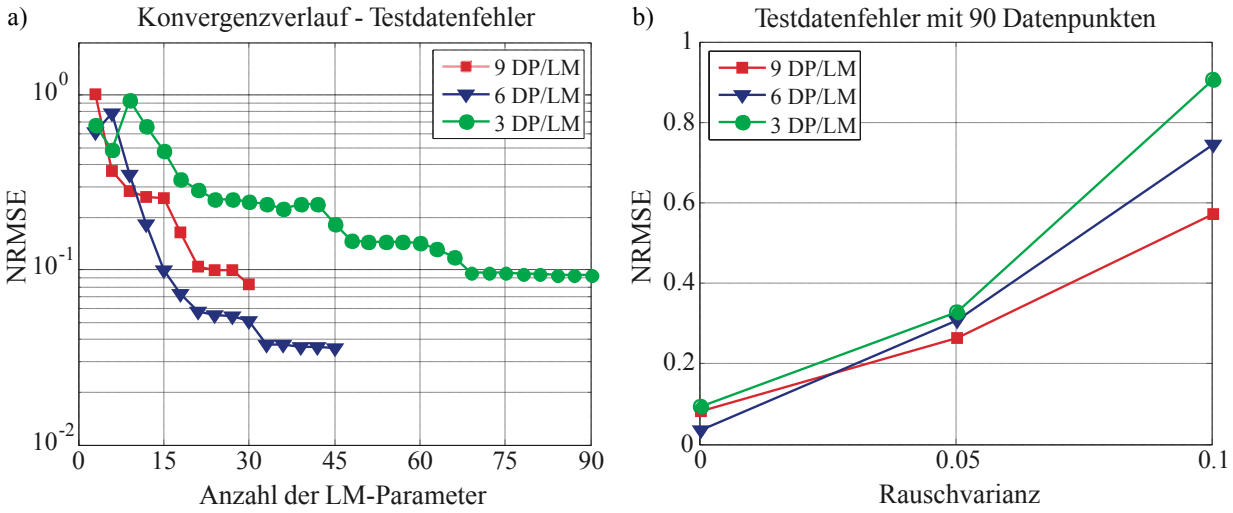

Bild 5.9: a) Konvergenzverhalten von HILOMOT als DoE-Verfahren mit achsenschräger Teilung. b) Resultierende Modellfehler auf Testdaten für verschiedene Punktedichten (jeweils insgesamt 90 Datenpunkte (DP)).

D-optimale Versuchsplanung konstruiert wurden. Einerseits wurden die Punkte D-optimal bezüglich eines Polynoms dritten Grades mit der `candexch`-Funktion[2] in MATLAB geplant und mit dem gleichen Polynomansatz modelliert. Zur Auswahl standen 1000 gleichverteilte Zufallspunkte in $[0, 1]$. HILOMOTDOE mit Batch-Query-Strategie hatte die gleichen Kandidatenpunkte und wurde mit vier Punkten initialisiert. Pro LM waren 3 Punkte gefordert. Man erkennt, dass die getroffene Modellannahme bei der D-optimalen Versuchsplanung im schlimmsten Fall dazu führen kann, dass entscheidende nichtlineare Effekte unter Umständen gar nicht von der Messung erfasst werden.

Die Haupteigenschaften des HILOMOTDOE-Algorithmus' mit Batch-Query-Strategie können wie folgt zusammengefasst werden:

- Versuchsplanung, Messung und Modellbildung laufen simultan ab.
- Das Ziel von HILOMOTDOE ist die Minimierung des globalen Modellfehlers, nicht die Minimierung der Parametervarianzen eines bestimmten vordefinierten Modells, wie dies z.B. bei D-optimaler Versuchsplanung der Fall ist.
- Jedem LM wird die gleiche Anzahl an Messpunkten zugeordnet.
- Der Algorithmus hat zwei Einstellparameter: Die Anzahl der Initialpunkte und die Punktanzahl pro LM. Mit diesen Parametern lässt sich der Explorationsgrad des Verfahrens vorgeben.
- Das Modell wird iterativ verfeinert, indem nach jeder Messung jeweils ein neues LM hinzugefügt und dort neue Query-Punkte platziert werden. Dadurch passt sich das nichtlineare Modell schrittweise dem Prozessverhalten an. Dies entspricht dem Ansatz des *aktiven Lernens*.
- Die Platzierung neuer Query-Punkte erfolgt raumfüllend innerhalb des jeweiligen LM unter Berücksichtigung bereits vermessener Punkte.

[2] Die `candexch`-Funktion in MATLAB erzeugt einen D-optimalen Versuchsplan durch Auswahl von Punkten aus einem Kandidatensatz.

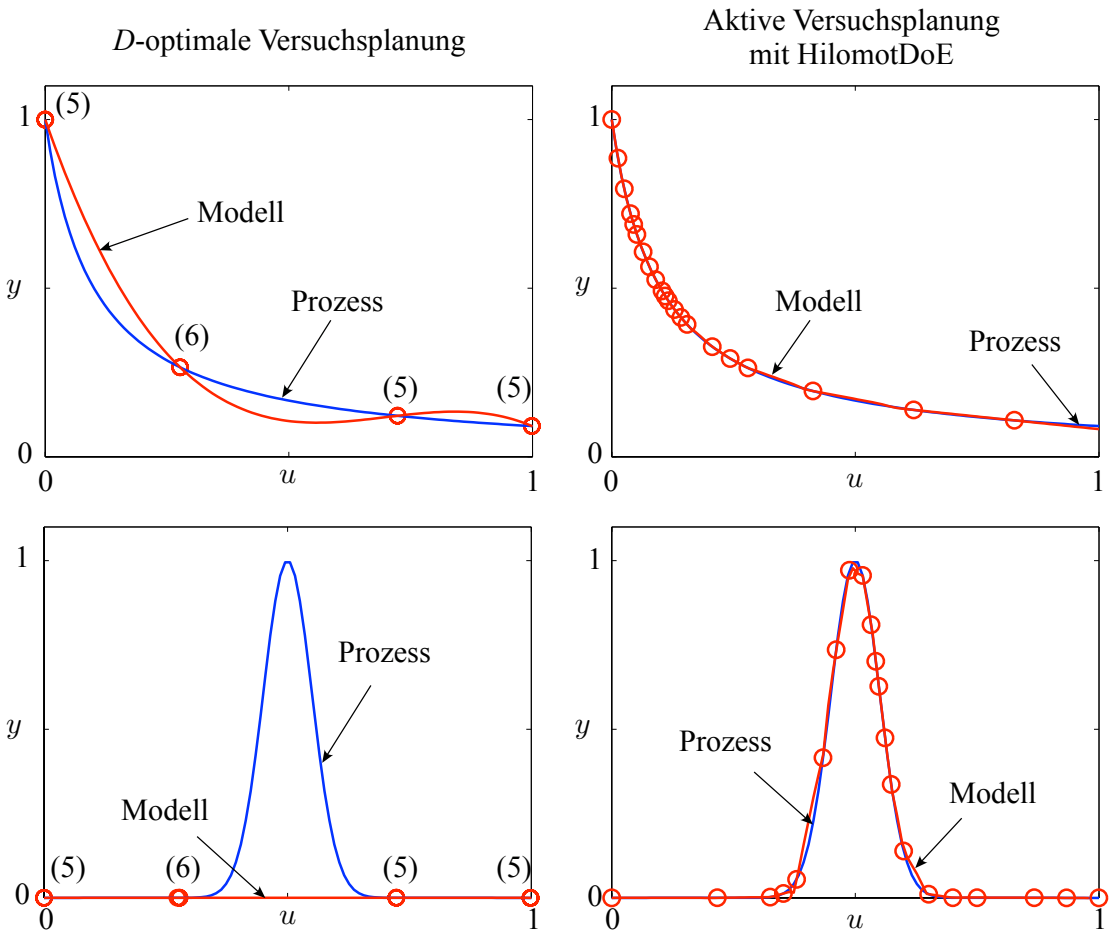

Bild 5.10: Links: Prozess 1 (oben) und Prozess 2 (unten) mit *D*-optimaler Versuchsplanung. Modellierung mit Polynom dritten Grades.
Rechts: Aktive Versuchsplanung für Prozess 1 (oben) und Prozess 2 (unten) mittels HILOMOTDOE mit Batch-Query-Strategie. In allen Fällen wurden 21 Messungen platziert.

- In Regionen im Eingangsraum, wo sich der Prozess stark nichtlinear verhält, findet eine dichtere Platzierung der Punkte statt als in Regionen, in denen der Prozess näherungsweise linear verläuft.
- A priori bekannte Versuchsraumgrenzen können in einfacher Weise durch die Verwendung von Kandidatenpunkten berücksichtigt werden.

5.3.3 Aktives Lernen mit Single-Query-Strategie

Beim aktiven Lernen geht man davon aus, dass sich das Modell nicht wesentlich ändert, wenn die angeforderten Queries vermessen und der Trainingsdatensatz mit den neuen Messpunkten aktualisiert wurde. Wie z.B. in [126] erwähnt und in [132] empirisch gezeigt, ist die beste Strategie beim aktiven Lernen, immer nur ein einzelnes Query an den Prozess zu übergeben, weil dann das Modell nach jeder Messung aktualisiert und dadurch auch die Planung des nächsten Queries genauer wird. Daher verfolgt HILOMOTDOE mit Single-Query-Strategie genau dieses Ziel. Dem Prozess wird hierbei vom Lernalgorithmus immer nur ein einzelner Query-Punkt übermittelt und nach jeder Messung das Modell mit HILOMOT neu trainiert. Bild 5.11 zeigt den schematischen Ablauf dieser Strategie.

Der Algorithmus kann in drei Arbeitsphasen eingeteilt werden:

1. Initiale Offline-Phase: Es wird ein initialer Versuchsplan (VP) erstellt und als Batch-Query an den Prozess übermittelt.
2. Online-Phase: Aktives Lernen mit modellbasierter Query-Optimierung.
3. Offline Nachverarbeitungsphase: Mit dem kompletten Datensatz wird ein Offline-Modell trainiert, welches in nachfolgenden Schritten, wie z.B. einer modellbasierten Stellgrößen-Optimierung, Verwendung findet.

Initiale Offline-Phase

Der Algorithmus startet mit einem initialen Offline-Versuchsplan, der als Batch-Query formuliert wird. Standardmäßig findet die initiale Versuchsplanung raumfüllend statt. Alternativ kann allerdings auch z.B. ein zentral zusammengesetzter Versuchsplan (sog. *CCI*-Versuchsplan) verwendet werden. Das Batch-Query dient schließlich als Initialisierung der nachfolgenden Online-Phase, indem die Queries vermessen und ein erstes Modell mit HILOMOT trainiert wird.

Online-Phase

Das eigentliche aktive Lernen findet während der Online-Phase statt. In dieser Arbeitsphase interagiert HILOMOTDOE permanent mit dem Prozess. Nach jeder Messung wird ein komplett neues lokales Modellnetz trainiert. Basierend auf dem gegenwärtigen Modell kann der nächste Query-Punkt in zwei Schritten optimiert werden:

1. Zunächst findet die Berechnung einer Modellfehlerfunktion statt, die sowohl den Bias- als auch den Varianzfehler des Modells repräsentiert. Im Fall von HILOMOT-Modellen kann von Underfitting ausgegangen werden, d.h. der Fehler kommt hauptsächlich durch die Biaskomponente zustande, weil die Modellkomplexität mittels AIC_c ausgewählt wird und das Modell durch die lokale Schätzung weniger flexibel ist.

2. Daraufhin bestimmt HILOMOTDOE den nächsten Query-Punkt innerhalb des Versuchsraumbereiches, wo der prädizierte Modellfehler am höchsten ist.

Diese zwei Schritte werden im Folgenden näher beschrieben.

Schritt 1: Prädiktion des Modellfehlers

Die wichtigste Frage für die Online-Versuchsplanung lautet: Wo muss die nächste Messung platziert werden, um maximalen Informationsgewinn über den Prozess zu erhalten?

Zur Beantwortung der Frage muss man sich zunächst über das Ziel der Versuchsplanung im Klaren sein, nämlich die Erstellung eines möglichst guten Prozessmodells. Die Güte kann anhand des prädizierten Modellfehlers quantifiziert werden, der sich generell in eine Bias- und eine Varianzkomponente unterteilen lässt, siehe z.B. Kapitel 3.5.1. In der Literatur nehmen die meisten Versuchsplanungsstrategien an, das Modell sei biasfrei und konzentrieren sich deshalb beim Planen neuer Messungen auf die Minimierung des Varianzfehlers, der aus der Parameterschätzung resultiert. Im Falle von HILOMOTDOE hingegen fokussiert sich das Verfahren auf die Minimierung des Modellfehlers als Ganzes, d.h. sowohl auf Bias- als auch auf Varianzfehler. Daher dient der Punkt im Eingangsraum mit dem größten Modellfehlermaß als nächster Query-Punkt. Dort erwartet man bei der nächsten Messung den größtmöglichen Informationsgewinn.

Im Gegensatz zu beispielsweise D- oder V-optimalen Versuchsplänen mit Polynommodellen ist der hier vorgestellte Ansatz nicht eingeschränkt in seiner Flexibilität, sondern ist darüber hinaus in der Lage, die Modellkomplexität an den gegenwärtigen Datensatz zu adaptieren, weil das Verfahren online während des Messprozesses arbeitet. Die Anzahl der lokalen Modelle wird abhängig vom Umfang und der Qualität der Trainingsdaten automatisch angepasst.

Die Varianz des Modellausgangs auf neuen Daten kann bei lokalen Modellnetzen ähnlich wie bei Polynommodellen durch die Berechnung sogenannter „Fehlerbalken“ bzw. „errorbars“ geschätzt werden. Die Kovarianzmatrix der Parameterschätzung lautet für das i-te lokale Modell [116]:

$$\mathrm{cov}\{\underline{\hat{w}}_i\} \;=\; \sigma_n^2 \left(\underline{X}_i^T \underline{Q}_i \underline{X}_i\right)^{-1} \underline{X}^T \underline{Q}_i \underline{Q}_i \underline{X} \left(\underline{X}_i^T \underline{Q}_i \underline{X}_i\right)^{-1} , \tag{5.10}$$

mit σ_n^2 als Varianz des Messrauschens. Bei praktischen Anwendungen bleibt meist nichts anderes übrig, als die Rauschvarianz auf Basis der verfügbaren Daten mit dem Modell zu schätzen, sofern kein anderes Vorwissen existiert. Dann ist die Varianz des Modellausgangs auf einem neuen Datensatz $\widetilde{\underline{X}}$:

$$\mathrm{cov}\{\underline{\hat{y}}\} \;=\; \sum_{i=1}^{M} \underline{Q}_i \mathrm{cov}\{\underline{\hat{y}}_i\} \;,\; \text{mit} \;\; \mathrm{cov}\{\underline{\hat{y}}_i\} \;=\; \widetilde{\underline{X}}\, \mathrm{cov}\{\underline{\hat{w}}_i\} \widetilde{\underline{X}}^T \tag{5.11}$$

und die Fehlerbalken bzw. das Konfidenzintervall berechnen sich dann mit

$$\mathrm{errorbar} = \pm \sqrt{\mathrm{diag}\left(\mathrm{cov}\{\underline{\hat{y}}\}\right)} \times t_{1-\frac{\alpha}{2}} \,, \tag{5.12}$$

wobei der Wert $t_{1-\frac{\alpha}{2}}$ das Konfidenzintervall so korrigiert, dass es einer Wahrscheinlichkeit von $1-\alpha$ entspricht. Üblicherweise wählt man $\alpha = 0.05$, was zu einem Konfidenzniveau von 95% führt.

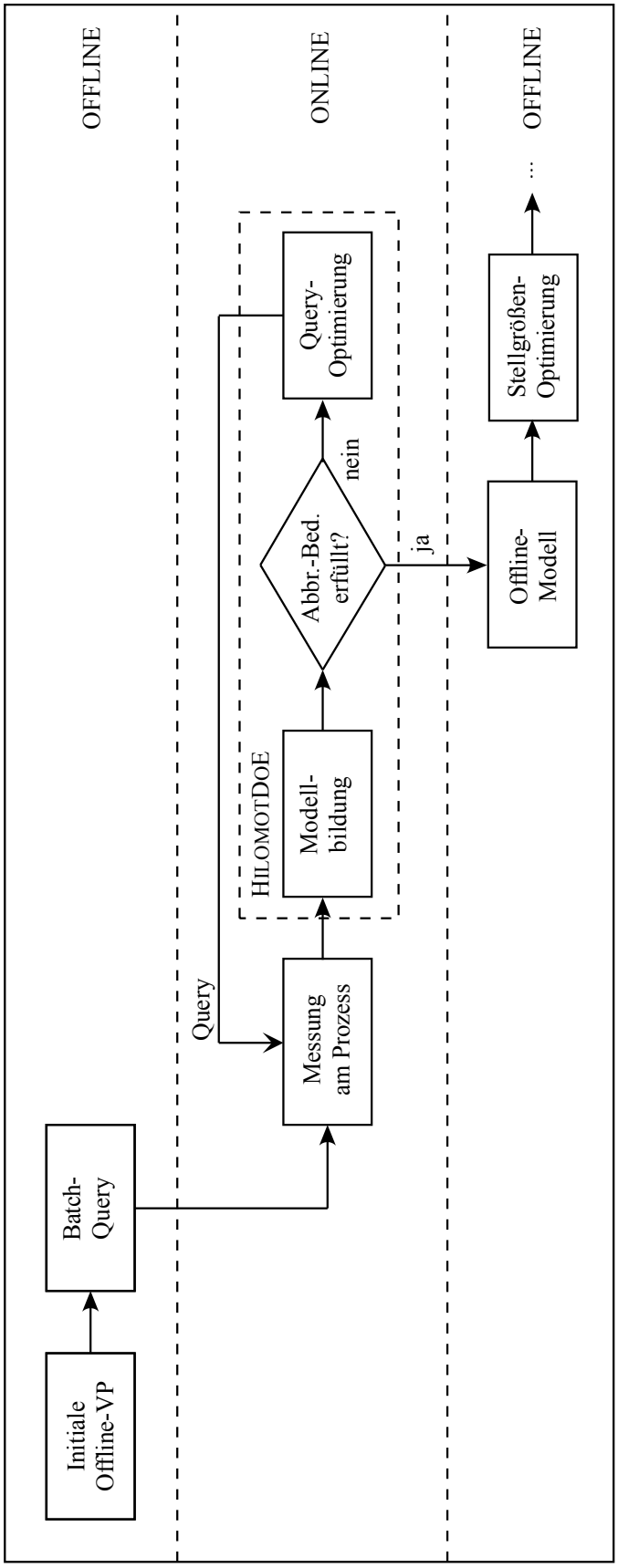

Bild 5.11: Funktionsweise des HilomotDoE-Algorithmus' mit Single-Query-Strategie.

Die Verwendung von Konfidenzintervallen zur Versuchsplanung ist allerdings im Falle von HILOMOT-Modellen aus folgenden Gründen problematisch:

- Einerseits müssen umfangreiche Matrixmultiplikationen zur Berechnung der Konfidenzintervalle durchgeführt werden. Das ist rechen- und zeitintensiv und daher für eine Online-Versuchsplanung sehr nachteilig.
- Andererseits ergeben sich in den Interpolationsregionen zwischen den lokalen Modellen lokale Maxima, siehe Bild 5.12c. Möchte man zur Query-Optimierung das globale Maximum der Fehlerbalken bzw. des Varianzfehlers bestimmen, so ist dies nur schwer möglich, ohne aufwändige globale Optimierungsverfahren einzusetzen.
- Darüber hinaus muss man berücksichtigen, dass die LM mit einer lokal gewichteten LS-Schätzung optimiert werden. Dieses Schätzverfahren reduziert durch implizite Regularisierung den Varianzfehler signifikant, wobei gleichzeitig der Biasfehler ansteigt, siehe z.B. [115]. Daher wäre der alleinige Fokus auf den Varianzfehler in dieser Konfiguration nicht zielführend.
- Konfidenzintervalle werden unter der Voraussetzung berechnet, dass der Biasfehler des Modells Null ist [116]. Solange die Modellstruktur eine biasfreie Schätzung zulässt, kann der Biasfehler tatsächlich vernachlässigt werden. Allerdings weisen nichtlineare Modelle meist einen erheblichen Biasfehler auf. Zur Einstellung eines guten Bias-/Varianzkompromisses, z.B. mittels Informationskriterien, fällt das Modell eher wenig flexibel aus, um mit diesem konservativen Modell möglichst robust bezüglich Overfitting zu sein.

Um diese Probleme zu umgehen, kann die Struktur von lokalen Modellnetzen ausgenutzt werden, indem ein einfaches Fehlermodell erstellt wird, was nicht alleine den Varianzfehler, sondern darüber hinaus insbesondere den Biasfehler im Fokus hat. Für jedes lokale Modell ist das Fehlermaß dann folgendermaßen definiert:

$$\hat{\sigma}_i = \sqrt{\frac{(\underline{y} - \underline{\hat{y}}_i)^T \underline{Q}_i (\underline{y} - \underline{\hat{y}}_i)}{\mathrm{spur}(\underline{Q}_i) - n_{\mathrm{eff},i}}}\,, \tag{5.13}$$

wobei $\underline{Q}_i = \mathrm{diag}(\Phi_i)$ die Gewichtungsmatrix des LM ist und sich die Anzahl der effektiven Parameter wie folgt ausrechnet:

$$n_{\mathrm{eff},i} = \mathrm{spur}(\underline{S}_i) = \mathrm{spur}\left(\underline{Q}_i \underline{X}_i \left(\underline{X}_i^T \underline{Q}_i \underline{X}_i\right)^{-1} \underline{X}_i^T \underline{Q}_i\right)\,. \tag{5.14}$$

Diese Berechnung erfolgt sehr effizient, wenn eine QR-Zerlegung ausgenutzt wird, die bereits durch die Schätzung der LM-Parameter bereitsteht. Die gewichtete Regressionsmatrix lautet in der zerlegten Form: $\sqrt{\underline{Q}_i}\underline{X}_i = \underline{Q}_{\mathrm{QR,i}} \underline{R}_{\mathrm{QR,i}}$. Mit den zerlegten Matrizen vereinfacht sich die Berechnung von $n_{\mathrm{eff},i}$ in Gl. (5.14) zu:

$$n_{\mathrm{eff},i} = \sum_{k=1}^{N} \sum_{l=1}^{nx} \Phi_i(\underline{u}(k))\, Q_{\mathrm{QR,i}}^2(k,l)\,, \tag{5.15}$$

mit nx als Anzahl der Regressoren in $\underline{X}_i$.

Damit kann dann ein kontinuierliches Fehlermodell $\hat{\sigma}^2(\underline{u})$ erstellt werden, indem die Gültigkeitsfunktionen zur Interpolation der lokalen Fehlerwerte verwendet werden:

$$\hat{\sigma}^2(\underline{u}) = \sum_{i=1}^{M} \Phi_i(\underline{u}) \hat{\sigma}_i^2\,. \tag{5.16}$$

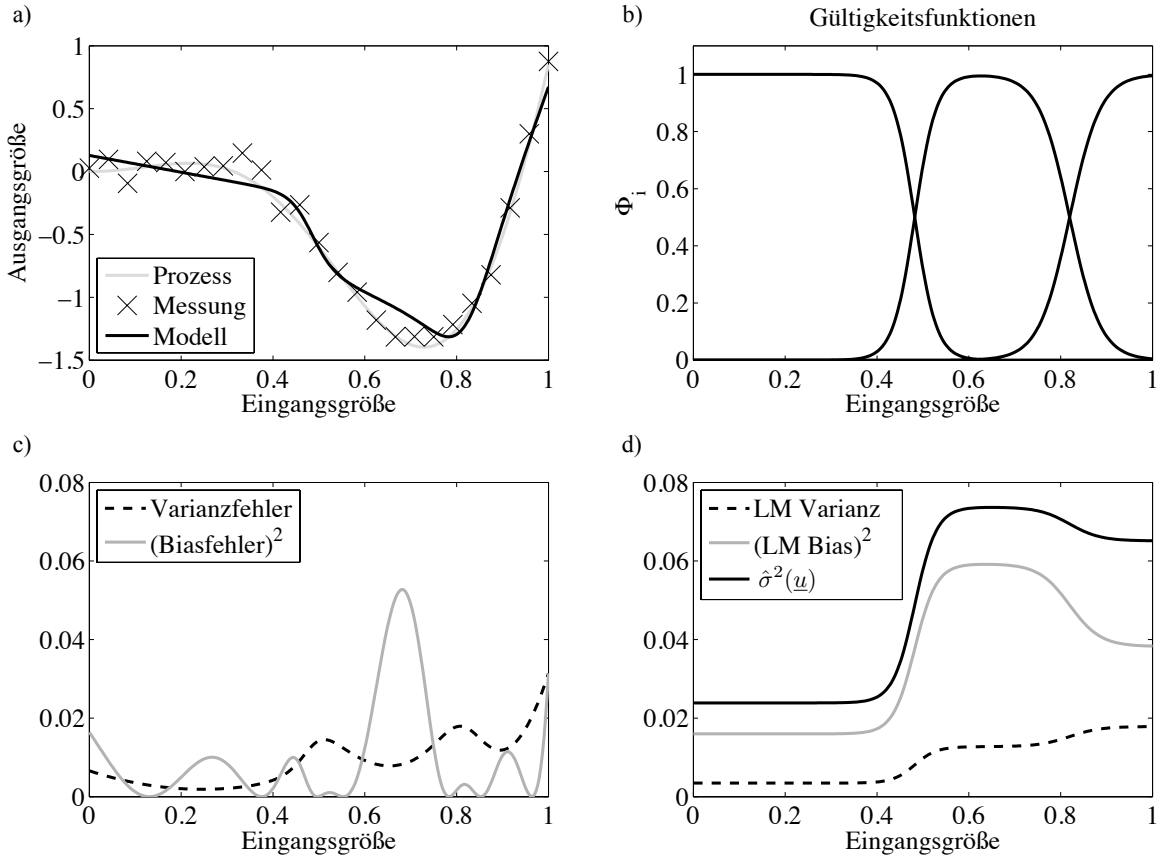

Bild 5.12: a) Lokales Modellnetz mit drei lokal linearen Modellen im Vergleich zum wahren Prozess und den Trainingsdaten. b) Gültigkeitsfunktionen des HILOMOT-Modells. c) Varianzfehler im Vergleich zum quadratischen Biasfehler. d) Vergleich von Varianzfehler, interpoliertem quadratischen Biasfehler und Modellfehler $\hat{\sigma}^2(\underline{u})$ aus Gl. (5.16).

Bild 5.12 zeigt ein Beispiel zum Vergleich von quadratischem Biasfehler und dem Varianzfehler des Modells. Zur Vergleichbarkeit wurde der t-Wert in Gl. (5.12) auf $t_{1-\frac{\alpha}{2}} = 1$ gesetzt, damit das quadrierte Konfidenzintervall dem approximierten Varianzfehler entspricht. Die Prozessgleichung lautet $y(u) = 3\cos(5u)u^2$ und wird mit unkorrelliertem, normalverteiltem und mittelwertfreiem Rauschen mit der Varianz $\sigma_n^2 = 0.05^2$ überlagert. Zum Training stehen $N = 25$ Punkte zur Verfügung, womit der HILOMOT-Algorithmus ein lokales Modellnetz mit drei lokal linearen Modellen erzeugt. Da in diesem Fall die wahre Funktionsgleichung des Prozesses bekannt ist, kann der Biasfehler ausgerechnet werden. Für einen besseren Vergleich mit der vorgeschlagenen Modellfehlerfunktion $\hat{\sigma}^2(\underline{u})$, werden sowohl der Bias- als auch der Varianzfehler für jedes LM analog zu Gl. (5.13) separat ausgerechnet und wie in Gl. (5.16) mit den Gültigkeitsfunktionen interpoliert. Der Vergleich in Bild 5.12 c) und d) zeigt die Dominanz des Biasfehlers gegenüber dem Varianzfehler. In diesem Beispiel würde man den nächsten Query-Punkt besser bezüglich des Bias- als bezüglich des Varianzfehlers optimieren.

Die Dominanz des Biasfehlers schlägt sich auch im Fehlermaß $\hat{\sigma}^2(\underline{u})$ nieder, wie im Bild 5.12d zu erkennen ist. Das Gesamtfehlermaß und der Biasfehler verlaufen qualitativ sehr ähnlich. Der Offset kommt dadurch zustande, dass im Fehlermaß auch die Rauschvari-

anz σ_n^2 enthalten ist. Allerdings findet sich auch die Varianzkomponente im Fehlermaß wieder, wie sich im rechten LM an der relativen Erhöhung des Fehlermaßes im Vergleich zum Biasfehler erkennen lässt. Das einfache Beispiel zeigt, dass bei der Auswahl des Queries mit HILOMOTDOE sowohl Bias- als auch Varianzkomponente des Modellfehlers berücksichtigt werden.

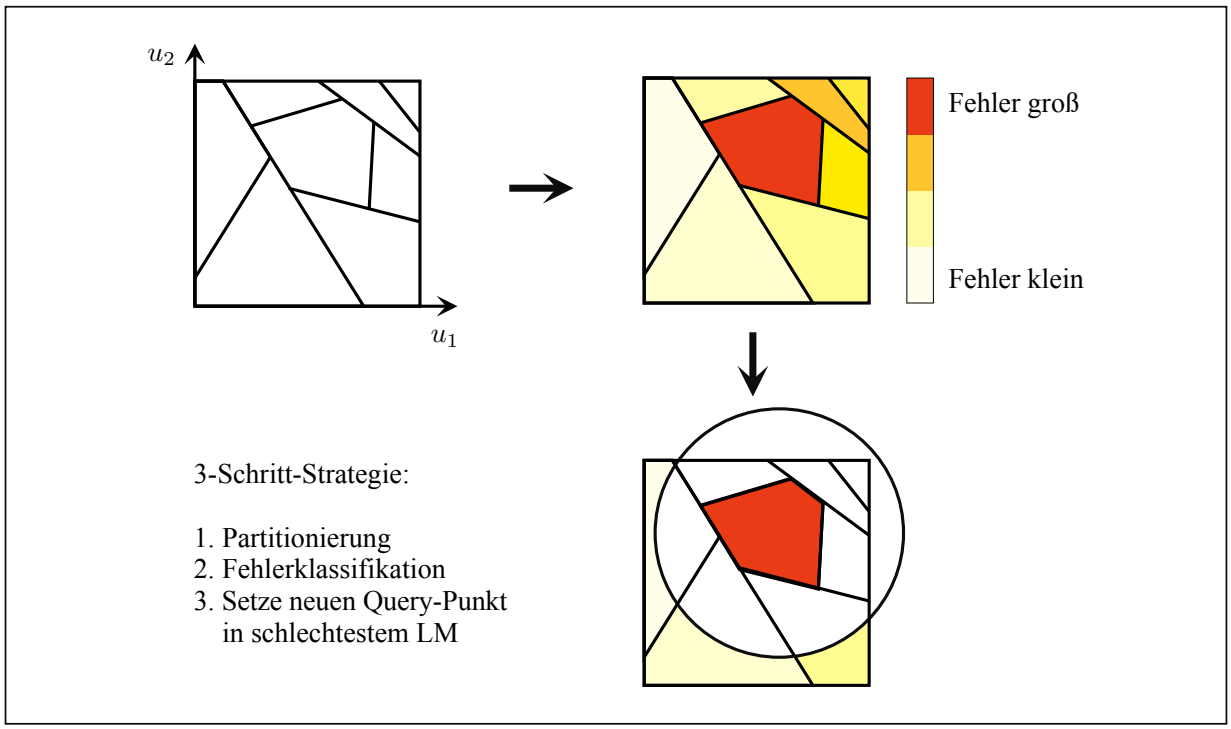

Bild 5.13: 3-Schritt-Strategie zur Berechnung neuer Query-Punkte.

Schritt 2: Query-Berechnung

Nach der Modellbildung ist der Eingangsraum partitioniert und der Prozess in den lokalen Regionen durch lokal affine Modelle beschrieben. Wie in Bild 5.13 gezeigt, werden die lokalen Modellregionen nach ihrem Fehlermaß $\hat{\sigma}_i^2$ (Gl. (5.16)) klassifiziert. Das LM mit dem größten Fehler verspricht den höchsten zu erwartenden Informationsgewinn für die nächste Messung. Daher wird in dieser Region raumfüllend das nächste Query platziert. Die Platzierung ist schematisch in Bild 5.14 erklärt.

Bei der Online-Versuchsplanung mit HILOMOTDOE ist man bestrebt, automatisch einen Kompromiss zwischen raumfüllender und optimaler Versuchsplanung zur Minimierung des Modellfehlers zu finden. Je mehr Daten zum Training zur Verfügung stehen, desto besser wird die Modellqualität und der Modellfehler wird immer zuverlässiger als Kriterium zur Versuchsplanung.

Bild 5.15 illustriert die Abhängigkeit der Versuchsplanung zur Modellkomplexität. Wenn z.B. nur ein einziges globales Modell existiert, ist die Versuchsplanung ausschließlich raumfüllend. Dadurch hat der Algorithmus anfangs ein stark exploratives Verhalten. Je mehr Daten allerdings gemessen werden, desto mehr lokale Modelle hat das HILOMOT-Modell. Daraus ergeben sich immer feiner auflösende Gültigkeitsgebiete, die für die Versuchsplanung herangezogen werden. Für sehr große Datensätze wäre die Versuchsplanung schließlich optimal bezüglich des globalen Modellfehlers.

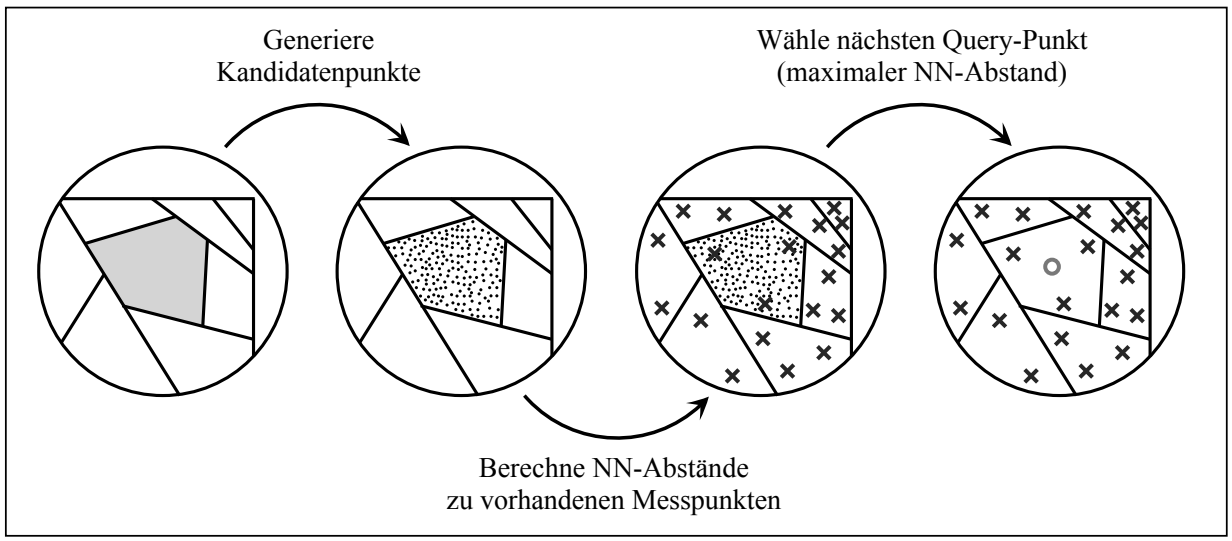

Bild 5.14: Sobald das schlechteste LM ausgesucht ist, kann ein neuer Query-Punkt innerhalb dieser Gültigkeitsregion platziert werden. Im lokalen Modell werden Kandidatenpunkte generiert ($\cdot$) und der Query-Punkt durch Maximierung des Nächste-Nachbar-Abstands zu allen existierenden Punkten ($\times$) bestimmt.

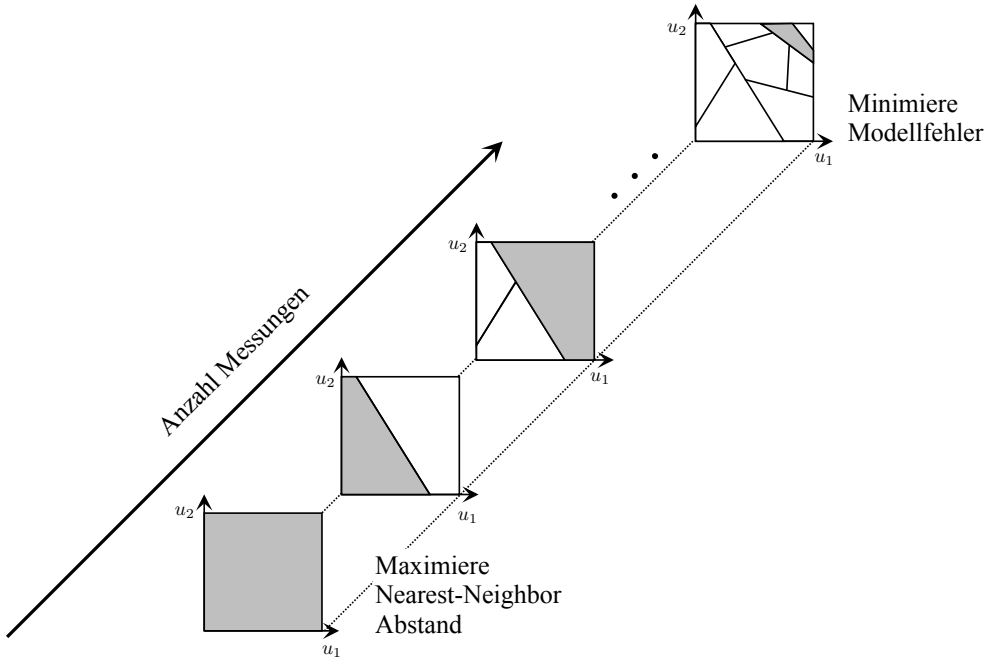

Bild 5.15: Die Idee bei HILOMOTDOE ist es, automatisch einen Kompromiss zwischen raumfüllender und optimaler Versuchsplanung zur Minimierung des Modellfehlers zu finden.

Offline Nachverarbeitungsphase

Die gesammelten Messwerte können schließlich im Nachgang dazu genutzt werden, um ein Offline-Modell zu trainieren. Das Training kann dann beispielsweise mit verschiedenen Modellkonfigurationen erfolgen und das beste Modell anhand separater Validierungsdaten oder basierend auf statistischen Tests ausgewählt werden. Nachfolgende Schritte wie z.B. eine modellbasierte Stellgrößenoptimierung oder die Erzeugung von Kennfeldern können schließlich folgen.

Demonstrationsbeispiel

Die Bilder 5.16 und 5.17 zeigen ein zweidimensionales Beispiel zur Demonstration der Online-Versuchsplanung mit Single-Query-Strategie. Ziel hierbei ist, die Versuchspunkte so zu platzieren, dass damit das Optimum des Prozesses mit Hilfe des Modells zu finden ist. Man erkennt, dass mit zunehmender Messpunktanzahl die Modellqualität verbessert und die Lage des Optimums identifiziert wird. In der Nähe des Optimums werden die Query-Punkte wie zu erwarten dichter platziert.

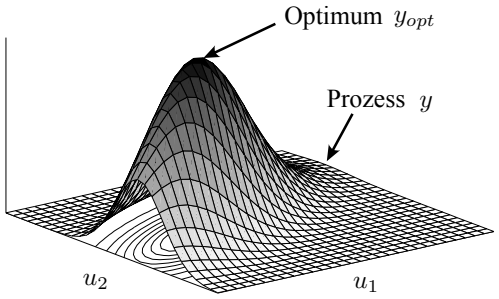

Bild 5.16: Beispielprozess, der mit HILOMOTDOE gelernt werden soll. Ziel ist das Finden des Optimums auf Basis des resultierenden HILOMOT-Modells.

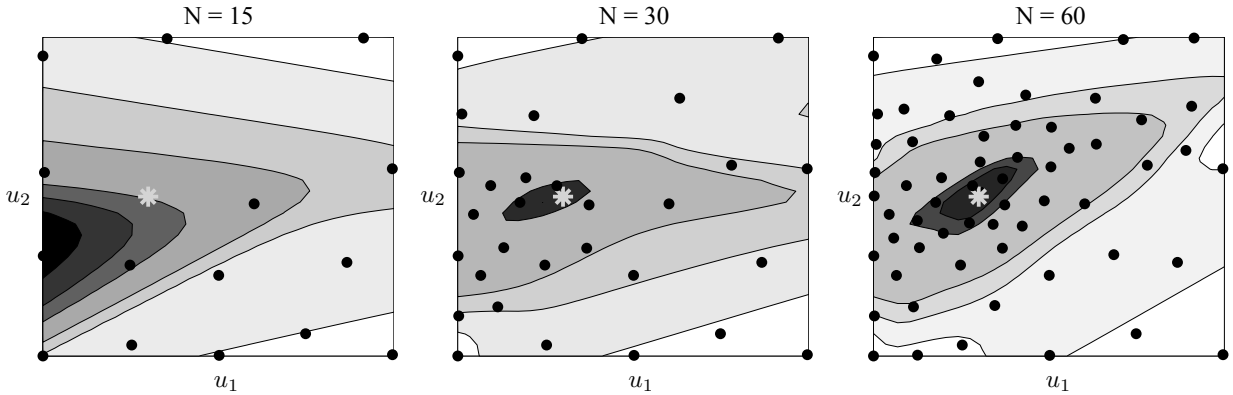

Bild 5.17: Mit zunehmender Anzahl an Messpunkten (·) verbessert sich das Modell (Höhenlinien). Der Testprozess aus Bild 5.16 kann mit $N = 60$ Punkten gut vom Modell wiedergegeben und das Optimum genau gefunden werden.

5.3.4 Empirischer Vergleich

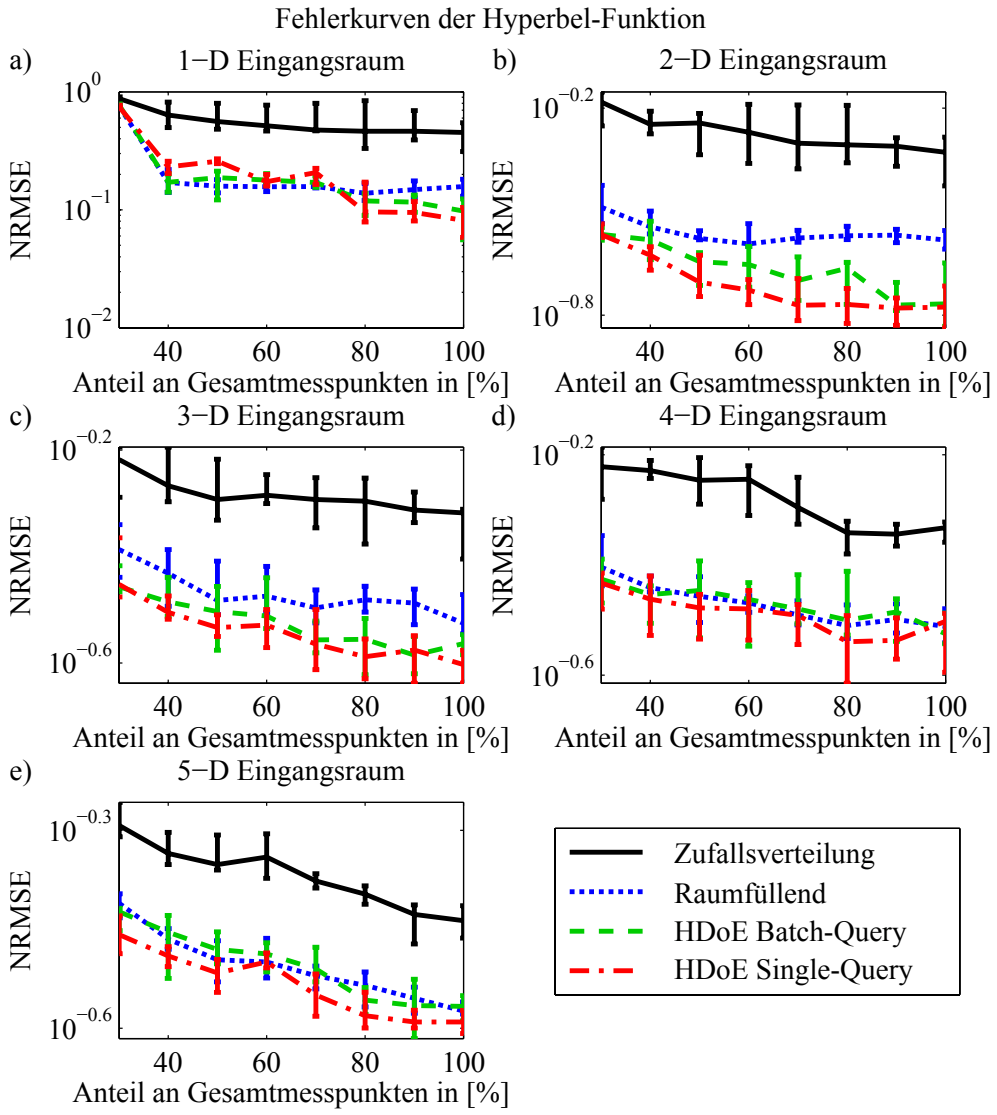

Bild 5.18: Testdatenfehlerverläufe bezüglich der Hyperbel-Funktion für verschiedene Eingangsraumdimensionen.

Zum Abschluss des Abschnitts findet ein empirischer Vergleich der vorgestellten aktiven Lernverfahren statt. Zur Veranschaulichung der prinzipiellen Eigenschaften eignet sich die Hyperbelfunktion $y = 0.1\,(0.1 + \frac{1}{p}\sum_{i=1}^{p}(1-u_i))^{-1}$ in den Dimensionen 1D bis 5D. Mit steigender Eingangsraumdimension wurde die zu planende Punktanzahl höher gewählt, d.h. $N(\dim) = \{20, 100, 150, 200, 400\}$ Messpunkte. Der Ausgang ist mit normalverteiltem, weißem Rauschen überlagert, wobei die Standardabweichung 4% des Wertebereichs beträgt.

Die zwei Varianten von HILOMOTDOE mit Batch-Query- und Single-Query-Strategie werden mit einer global raumfüllenden, passiven Versuchsplanung und einer rein zufälligen Versuchsplanung durch gleichverteilte Messwerte im Eingangsraum verglichen. Die Initialisierung der HILOMOTDOE-Algorithmen erfolgt mit raumfüllend verteilten Versuchspunkten, jeweils mit 25% der vorgegebenen Gesamtpunkteanzahl. Für die Batch-Query-Strategie waren pro LM doppelt so viele Punkte verlangt wie LM-Parameter. Zum Vergleich wurden

nach der simulierten Messung die Daten jeweils mit dem gleichen HILOMOT-Modellansatz trainiert und der Fehler auf gitterförmig verteilten, unverrauschten Testdaten ausgewertet.

Bild 5.18 zeigt die Medianwerte, das 25. und das 75. Perzentil des normierten Modellfehlers (NRMSE) für insgesamt zehn Durchläufe, dargestellt für die Dimensionen 1D bis 5D. Man erkennt, dass sich die Versuchsplanungsstrategien deutlich von einer zufälligen Versuchsplanung abheben. Darüber hinaus wird deutlich, dass sich durch die aktive Versuchsplanung die Modellgüte im Vergleich zur raumfüllenden Versuchsplanung verbessern lässt. Wie zu erwarten, ist die Single-Query-Strategie zudem der Batch-Query-Strategie in allen Dimensionen überlegen.

5.4 Automatisiertes Stoppen einer Messung

Ein wesentlicher Punkt bei der Versuchsplanung ist, die Gesamtanzahl der Messungen als Abbruchbedingung festzulegen. Wie viele Messungen müssen durchgeführt werden, um ein gutes Modell vom Prozess zu generieren? Beim aktiven Lernen muss also der Algorithmus auf Basis des gegenwärtigen Modells entscheiden, ob es „genau genug“ ist. Das kann erreicht werden, indem man den Leave-One-Out-Kreuzvalidierungsfehler $\mathrm{LOOCV}(\hat{y})$ des Modells berechnet, der als approximatives Maß für die Generalisierungsleistung des Modells dient. Im Fall von HILOMOT-Modellen errechnet sich der LOOCV-Wert folgendermaßen: Geht man davon aus, dass sich der Modellausgang mit Hilfe der Glättungsmatrix $\underline{S}$, die die gemessenen Werte auf die Modellausgangswerte abbildet, d.h. $\underline{\hat{y}} = \underline{S}\underline{y}$, ausdrücken lässt, dann folgt:

$$\mathrm{LOOCV}(\hat{y}) = \frac{1}{N}\sum_{i=1}^{N}\left(y(i) - \hat{y}^{-i}(\underline{u}(i))\right)^2 = \frac{1}{N}\sum_{i=1}^{N}\left(\frac{y(i) - \hat{y}(\underline{u}(i))}{1 - S_{ii}}\right)^2 . \tag{5.17}$$

Die ausführliche Herleitung dieses Maßes ist in Kapitel 3.5.2 zu finden. Der Vorteil dieser Methode ist, dass sich die Diagonalelemente S_{ii} sehr effizient berechnen lassen. Man beachte allerdings, dass es sich hier um eine Annäherung des wahren LOOCV handelt, weil der Einfluss der nichtlinearen Modellparameter nicht in die Berechnung mit einfließt. Jedoch kann davon ausgegangen werden, dass auch die Approximation gute Anhaltswerte zur Modellgüte liefert. Die LOOCV-Werte werden im Online-Verfahren mit Gl. (5.17) angenähert, weil für die vollständige Berechnung während der Messung zu viel Rechenaufwand entstünde.

In der Automobilindustrie hat sich das Bestimmtheitsmaß R^2_{PRESS} [31] als Fehlermaß etabliert. Der LOOCV-Wert kann entsprechend umgerechnet werden:

$$R^2_{\mathrm{PRESS}} = 1 - \frac{\mathrm{LOOCV}(\hat{y})}{\frac{1}{N}\sum_{i=1}^{N}\left(y(i) - \bar{y}\right)^2} , \tag{5.18}$$

wobei $\bar{y}$ der Mittelwert der Messungen ist.

Nach jeder Modellanpassung kann durch den aktualisierten Datensatz während der Online-Phase des Lernalgorithmus' das Bestimmtheitsmaß R^2_{PRESS} neu berechnet werden. Sobald neue Messpunkte keine signifikante Verbesserung der Kreuzvalidierungswerte liefern, kann die Messung abgebrochen werden. Zum automatisierten Abbruch der Messung muss also im

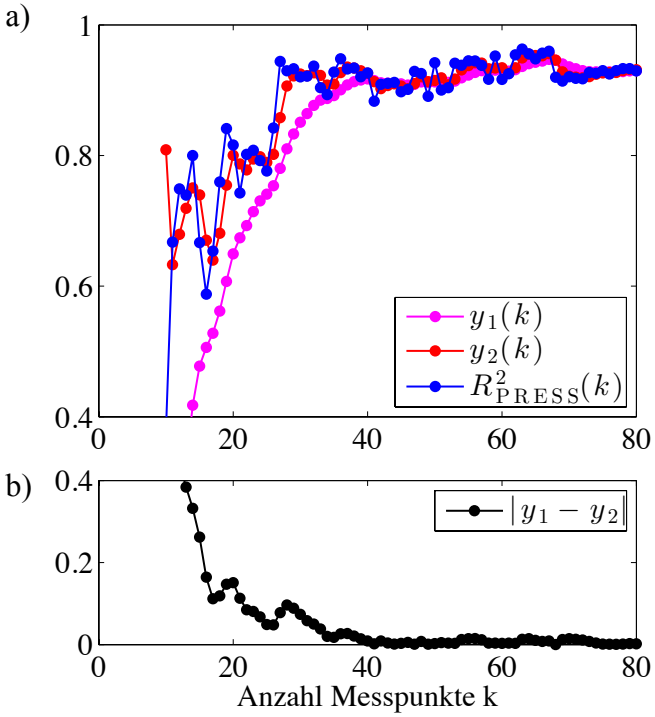

Bild 5.19: a) Verlauf des Bestimmtheitsmaßes R^2_{PRESS} über der Anzahl der Messungen k. b) Die Differenz aus langsamer (y_1) und schneller Filterung (y_2) des diskreten Verlaufs kann zur automatisierten Versuchsbeendigung verwendet werden.

Online-Betrieb erkannt werden, wenn sich die R^2_{PRESS}-Werte nicht mehr wesentlich ändern bzw. sich der R^2_{PRESS}-Verlauf über der Anzahl der Messung in der Sättigung befindet.

Betrachtet man die R^2_{PRESS}-Werte als diskreten Signalverlauf $R^2_{\mathrm{PRESS}}(k)$, wobei k hier die Anzahl der bisher durchgeführten Messungen beschreibt, lässt sich die Sättigung der Fehlerwerte mit der folgenden Methode automatisiert bestimmen: Das Signal $R^2_{\mathrm{PRESS}}(k)$ wird mit einem langsamen und einem schnellen Butterworth-Tiefpassfilter analysiert. Sobald die absolute Differenz der gefilterten Signale klein genug ist, kann man davon ausgehen, dass sich der Fehlerwert nicht mehr signifikant ändert.

Bild 5.19 zeigt diese Idee anhand eines typischen R^2_{PRESS}-Verlaufs. Eine sinnvolle Annahme für das Butterworth-Filter ist die Filterordnung 1. Grades, da der Fehlerverlauf typischerweise der Sprungantwort eines PT_1-Systems sehr nahe kommt. Dann hat die kontinuierlicher Filterübertragungsfunktion lediglich einen Pol und die Verstärkung Eins, d.h. ist von der Form:

$$G(s) = \frac{a}{(s+a)} \ , \tag{5.19}$$

mit dem Laplace-Operator s und der Konstanten a. Die Pollage bestimmt dann die Grenzfrequenz des Filters. Ein einfacher Ansatz besteht darin, die Grenzfrequenz des langsamen Filters grob durch die geschätzte Zeitkonstante T des PT_1-Systems festzulegen. Das Signal wird dann mit der Grenzfrequenz $f_{g1} = 1/T$ tiefpassgefiltert und ergibt $y_1(k)$. Darüber hinaus entwirft man einen schnellen Filter, indem man eine zweite Grenzfrequenz f_{g2} als x_f-faches der Frequenz des langsamen Filters auslegt, also $f_{g2} = x_f \cdot f_{g1}$. Der Filterausgang ist dann $y_2(k)$. Die betragsmäßige Differenz $|y_1(k) - y_2(k)|$ kann dann als Abbruchbedingung für die Messung benutzt werden.

Die folgenden drei Parameter müssen für das des Verfahren festgelegt sein:

1. Die Grenzfrequenz des langsamen Filters f_{g1},
2. der Faktor x_f zur Berechnung der zweiten, höheren Grenzfrequenz f_{g2} und
3. eine Toleranzschwelle ε, um die Messung bei einer Abweichung $|y_1(k) - y_2(k)| < \varepsilon$ abzubrechen.

In Bild 5.19 wurde die erste Grenzfrequenz $f_{g1} = 1/30\, f_0$ und der Faktor $x_f = 4$ gewählt. Zur diskreten Auswertung wird die Grenzfrequenz noch auf die halbe Abtastfrequenz $f_0/2$ normiert. Die Initialisierung war $y_1(0) = 0$ und $y_2(0) = 1$.

Um die Parametrierung anhand eines konkreten Prozesses durchzuführen, wird vorgeschlagen, die Toleranzschwelle auf einen festen Wert $\varepsilon = 0.05$ festzusetzen und sowohl die Grenzfrequenz f_{g1} als auch den Faktor x_f anhand eines vermessenen Datensatzes, der den Prozess repräsentiert, zu optimieren. Als Optimalitätskriterien können beispielsweise die Modellgüte und die Anzahl der durchzuführenden Messungen herangezogen werden.

5.5 Aktives Lernen mit Modellkomitees

Beim vorgestellten HILOMOTDOE-Algorithmus konzentriert sich die Query-Optimierung auf die Reduktion sowohl des Bias- als auch des Varianzfehlers, weil ein HILOMOT-Modell generell so ausgelegt ist, dass die Biaskomponente nicht zu vernachlässigen ist. Dabei ist jedoch zu beachten, dass bei dieser Betrachtung von einer festgelegten Partitionierung ausgegangen wird, deren Einfluss auf den Varianzfehler nicht beziffert wird. Die Verwendung von Modellkomitees ermöglicht es, durch Mittelung vieler Einzelmodelle, den Varianzfehler zu minimieren. Die Varianz der Einzelmodelle kann dann als Query-Kriterium zur aktiven Versuchsplanung sinnvoll genutzt werden. Es sollen zwei Methoden zur Bildung von Komitees mit HILOMOT-Modellen kurz vorgestellt werden. Zum einen die Komiteebildung durch *Bootstrapping*, zum anderen die Komiteebildung mit unterschiedlich konfigurierten, heterogenen HILOMOT-Modellen.

5.5.1 HILOMOT-Komitee durch Bootstrapping

Beim sogenannten *Bootstrapping* [61] wird ein bestehender Trainingsdatensatz durch „Ziehen mit Zurücklegen“ der Datenpunkte randomisiert. Dadurch erzeugt man viele verschiedene Datensätze, die sich durch zufällige Mehrfachauswahl oder Weglassen von Punkten alle ein wenig vom Original unterscheiden. Trainiert man dazu jeweils ein Modell stellt man fest, dass auch die Modellrealisierungen Schwankungen unterworfen sind. Dies ist in Bild 5.20 anhand eines Beispiels illustriert. Die Volatilität der Modelle kann als Maß für den Varianzfehler des Modells interpretiert werden. Um ein Gesamtmodell zu erstellen, bildet man den Mittelwert aller Modelle, die durch das Bootstrapping erzeugt wurden. Diese Methode nennt sich *Bagging* [61]. Wie in [61] theoretisch hergeleitet, wird durch Mittelwertbildung der Einzelmodelle mit zunehmender Anzahl der Realisierungen die Varianzkomponente immer kleiner. Das heißt, im Falle unendlich vieler Realisierungen hätte das gemittelte Bagging-Modell einen verschwindenden Varianzfehler. Im Umkehrschluss kann man mit den Bagging-Realisierungen und dem gemittelten Modell sehr gut die Modellvarianz abschätzen.

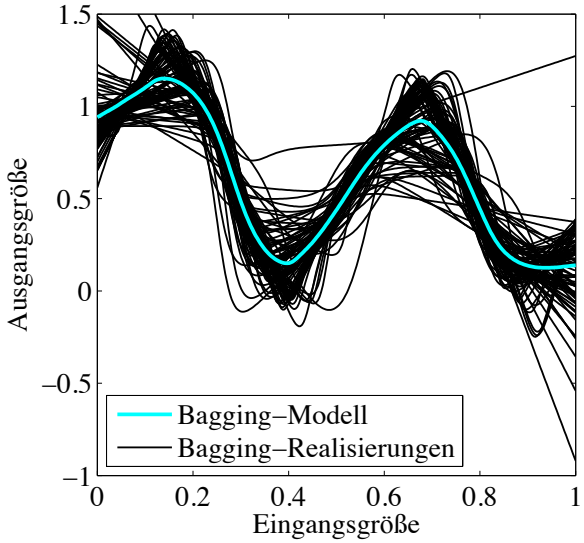

Bild 5.20: *Bagging*: Durch *Bootstrapping* werden die Trainingsdaten randomisiert und jeweils ein Modell mit HILOMOT trainiert. Aus den einzelnen Modellrealisierungen bildet man den Mittelwert. In diesem Beispiel handelt es sich um 100 HILOMOT-Modelle.

Im Fall lokaler Modellnetze kann man sich überlegen, die Methode des Baggings durchzuführen, ohne dabei die Partitionierung neu zu trainieren. Dies spart enorm viel Rechenaufwand. Jedoch erhält man dadurch qualitativ kein anderes Resultat für den Varianzfehler, als wenn man die Kovarianzmatrix des Modellausgangs bzw. die Konfidenzintervalle mit Gl. (5.12) schätzt. Das zeigt auch das Experiment in Bild 5.21. Der gezeigte Prozess (a) wurde mit 1000 verschiedenen Bootstrap-Realisierungen geschätzt, allerdings ohne dabei die Partitionierung zu verändern. Wie bei der lokalen Schätzung zu erwarten, ergibt sich bei einer festen Partitionierung (b) keine nennenswerte Verbesserung des Modells durch das Bagging (c). Erlaubt man allerdings, dass bei jeder Bootstrap-Realisierung ein komplett neues Modell trainiert werden darf, im Beispiel mit 100 Modellen durchgeführt, so ergibt sich eine deutliche Verbesserung des Modells (d). Der Varianzanteil des Modells, verursacht durch die *nichtlinearen* Partitionierungsparameter, wird durch das Bagging minimiert und führt schließlich dazu, dass der Biasfehler des gemittelten nichtlinearen Modells zum Vorschein kommt. Am Bild 5.21 erkennt man, dass in diesem Fall eine Query-Optimierung bezüglich des *Varianz*fehlers zielführend ist.

Bild 5.22 zeigt ein Beispiel, bei dem die Bagging-Methode für das aktive Lernen eines Modells genutzt wurde. Für jeden Messpunkt wurden 100 Modelle mit verschiedenen Bootstrap-Samples trainiert. Der Punkt, an dem die Varianz der Modellrealisierungen am größten war, diente als Query für die nächste Messung. Insgesamt wurden $N = 50$ Punkte geplant. Zum Vergleich zeigt Bild 5.22 auch das Ergebnis mit HILOMOTDOE (Single-Query-Strategie). An der Platzierung der Punkte mit dem Bagging-Ansatz erkennt man wie bei HILOMOTDOE die höhere Datendichte in Bereichen größerer Nichtlinearitäten. In diesem Beispiel führte allerdings die explorative Komponente von HILOMOTDOE zu einem leicht besseren Ergebnis. Zudem muss vor Augen geführt werden, dass der Bagging-Ansatz durch die hohe Anzahl zu trainierender Modelle äußerst rechenlastig ist. Unvorteilhaft ist außerdem der Zufallseinfluss auf die Modellbildung durch die Bootstrap-Samples.

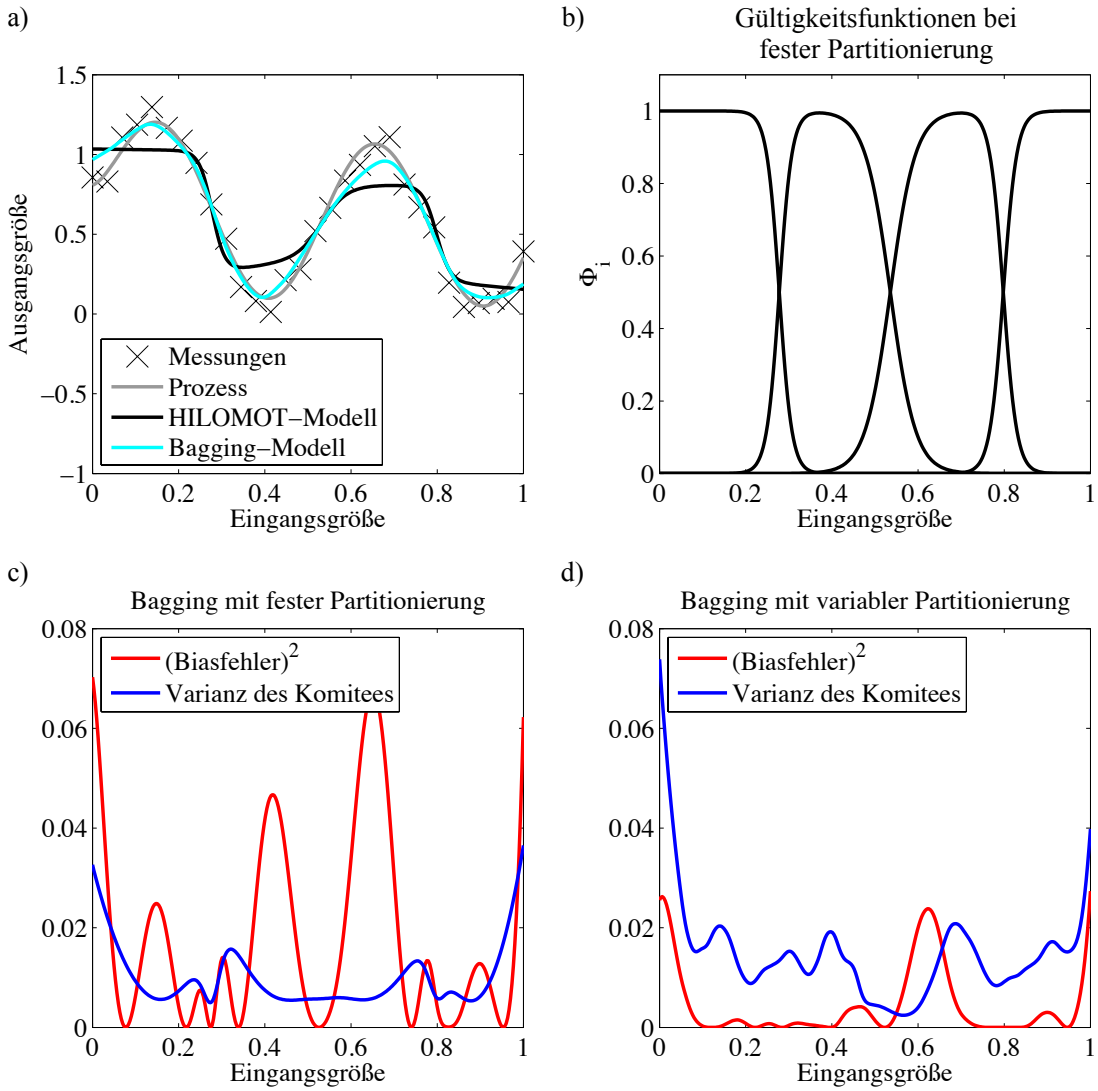

Bild 5.21: a) Modellvergleich. b) Partitionierung des HILOMOT-Modells. c) Biasfehler dominant. d) Varianzfehler dominant.
Bagging ändert im Wesentlichen nur etwas am Modell, wenn auch die Partitionierungsparameter bei den Einzelrealisierungen freigegeben werden. Allerdings zeigt die Methode dann deutliches Potential zur Verbesserung der Modellgüte. Das Bagging mit fester Partitionierung wurde mit 1000 Modellen, das Bagging mit variabler Partitionierung mit 100 Modellen durchgeführt.

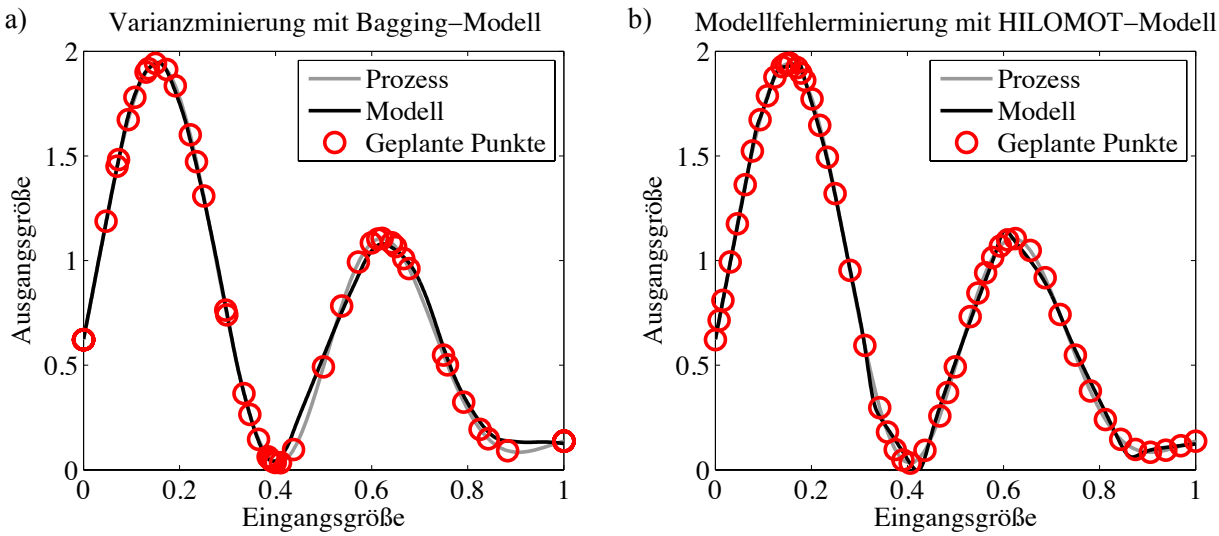

Bild 5.22: a) Aktives Lernen mit Bagging bezüglich einer minimalen Varianz.
b) Aktives Lernen mit HILOMOTDOE (Single-Query-Strategie).

5.5.2 Heterogenes HILOMOT-Komitee

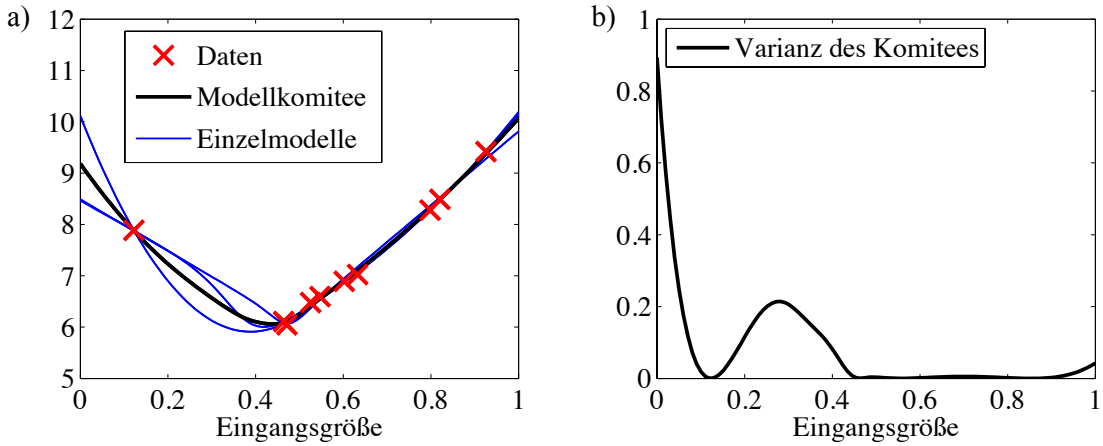

Bild 5.23: a) Modellkomitee mit Einzelmodellen. b) Varianz des Komitees.

Deutlich weniger rechenintensiv ist es, wenn ein Komitee aus wenigen HILOMOT-Modellen, die unterschiedlich konfiguriert sind, erzeugt wird. In Bild 5.23 besteht das Komitee aus HILOMOT mit lokal linearen Modellen, HILOMOT mit lokal quadratischen Modellen und einem HILOMOT+-Modell. Zwar sind die drei Ansätze ähnlich flexibel, jedoch kommt es durch die unterschiedliche Konfiguration zu Unterschieden bei der Modellbildung. Wie beim Bagging kann ein Komitee durch gewichtete Aufsummierung der Modelle erzeugt werden, sodass sich für das Gesamtmodell

$$\hat{y}_{Kom}(\underline{u}) = \sum_{i=1}^{n_{Kom}} \alpha_i \, \hat{y}_{M,i}(\underline{u}) \tag{5.20}$$

ergibt, wobei n_{Kom} die Anzahl der Einzelmodelle $\hat{y}_{M,i}(\underline{u})$ ist und α_i ein Gewichtungsfaktor. Im einfachsten Fall ist $\alpha_i = \frac{1}{n_{Kom}}$, $\forall\, i = 1, \ldots, n_{Kom}$. Zur unterschiedlichen Gewichtung

der Modelle entsprechend Ihrer Güte kann beispielsweise der reziproke LOOCV-Wert jedes einzelnen Modells verwendet werden, sodass

$$\alpha_i = \frac{\frac{1}{\mathrm{LOOCV}_i}}{\sum_{j=1}^{n_{Kom}} \frac{1}{\mathrm{LOOCV}_j}} \,. \tag{5.21}$$

Für das Beispiel in Bild 5.23 wurden die Gewichte $\alpha_1 = 0.293$, $\alpha_2 = 0.433$ und $\alpha_3 = 0.274$ berechnet. Alternativ zum Kreuzvalidierungsfehler können z.B. auch Akaike-Gewichte zur Modellbewertung verwendet werden.

Wie beim Bagging ergibt sich die Varianz bzw. der Dissens der Komitee-Ausgabe zu:

$$\sigma^2_{Kom}(\underline{u}) = \frac{1}{n_{Kom} - 1} \sum_{i=1}^{n_{Kom}} \left(\hat{y}_{M,i}(\underline{u}) - \hat{y}_{Kom}(\underline{u})\right)^2 . \tag{5.22}$$

Dieses Kriterium wird auch auch *Query by Committee (QBC)* bezeichnet [126]. Beim Komitee kann die Modellunsicherheit durch das Varianzkriterium sehr gut beschrieben und fürs aktive Lernen benutzt werden. Allerdings muss auch hier ein höherer Rechenaufwand im Vergleich zu HILOMOTDOE in Kauf genommen werden.

5.6 Zusammenfassung

Zu Anfang des Kapitels wurden die wichtigsten Grundlagen zur statistischen Versuchsplanung dargelegt. Falls sich der zu messende Prozess durch ein linear parametriertes Polynommodell beschreiben lässt, liefert die Statistik eine geschlossene Theorie zur optimalen Platzierung der Messpunkte bezüglich dieses Modells. Dazu kann mittels verschiedener Optimalitätskriterien wie z.B. D- oder V-Optimalität entweder die Varianz der Parameterschätzung oder die Varianz des Modellausgangs minimiert werden. Bemerkenswert ist, dass man der Theorie nach die optimale Platzierung der Punkte vornehmen kann, ohne auch nur eine einzige Messung am Prozess durchführen zu müssen. Dazu muss allerdings die Annahme zutreffen, dass sich formal der Prozess perfekt durch das vom Anwender gewählte Modell beschreiben lässt. Kann man jedoch nicht vorab beurteilen, mit welchem Modell der Prozess nachgebildet werden kann, so bleibt zunächst nichts weiter übrig, als die Punkte nach einem geometrischen Versuchsplan im Eingangsraum zu verteilen. Dazu gibt es verschiedene Verfahren wie z.B. das Sphere Packing oder Latin Hypercube Sampling, die die Punkte möglichst gleichförmig platzieren. Mathematisch lässt sich das durch die Maximierung der minimalen Abstände aller Punktpaare erreichen.

Zur statistischen Versuchsplanung für *nichtlinear* parametrierte Modelle müssen sogenannte aktive Lernverfahren eingesetzt werden. Im Unterschied zum passiven Lernen, ist dann der Lernalgorithmus in der Lage, mit dem Prozess zu *interagieren*. Der Lernalgorithmus kann auf Basis gegenwärtiger Messdaten entscheiden, wo neue Messungen zu platzieren sind. Durch eine größer werdende Datenbasis bildet das Modell den Prozess immer besser ab und die Versuchsplanung verfeinert sich.

Der Hauptteil des Kapitels befasst sich mit der Verwendung von HILOMOT-Modellen zur passiven und aktiven Versuchsplanung. Der Fokus bei diesen Versuchsplanungsmethoden liegt auf der Minimierung des Modellfehlers, der sowohl durch Bias- als auch Varianzfehler

gekennzeichnet ist. Um die Gefahr des Overfittings klein zu halten, ist im Allgemeinen bei dieser nichtlinear parametrierten Modellarchitektur der Bias-Varianz-Kompromiss so eingestellt, dass die Bias- gegenüber der Varianzkomponente des Modellfehlers überwiegt. Aus diesem Grund wäre die Versuchsplanung alleinig bezüglich des Varianzfehlers problematisch. Hier lässt sich die Struktur der lokalen Modellnetze explizit ausnutzen: Der Prozess wurde bereits vom Modellbildungsalgorithmus in lokale Bereiche unterteilt, die lokal betrachtet unterschiedliches Prozessverhalten aufweisen. Es bietet sich daher an, das Wissen über die Gültigkeitsbereiche lokaler Modelle explizit für die Vergabe neuer Messpunkte zu nutzen. Das Kapitel hat dazu drei Algorithmen vorgestellt, denen jeweils ein lokal raumfüllendes Versuchsplanungsverfahren zugrunde liegt. Es setzt einen neuen Punkt so, dass sein Nächster-Nachbar-Abstand zu allen bereits vorhandenen Messpunkten maximal ist. Die Methode bezieht durch die Verwendung von Kandidatenpunkten sowohl die Gültigkeitsgebiete der LM als auch die Grenzen des Versuchsraums bei der Auswahl neuer Punkte mit ein und beschränkt in dieser Weise die raumfüllende Versuchsplanung auf lokale Gebiete.

Das passive oder aktive Lernen eines Prozesses mit Hilomot-Modellen kann dann auf drei verschiedene Arten erfolgen:

1. Mit einer passiven Versuchsplanung, indem das Modell eines ähnlichen Prozesses als Vorwissen über den zukünftig zu messenden Prozess genutzt wird. Die Versuchsplanung findet vor der Messung anhand der achsenschrägen Hilomot-Partitionierung statt. Die Punktedichte im Eingangsraum richtet sich dann nach dem Grad der Prozessnichtlinearitäten.
2. Durch eine aktive Versuchsplanung mit Batch-Query-Strategie: Das Hilomot-Modell wird nach jeder Messung durch Teilung des lokalen Modells mit der größten geschätzten Modellfehlervarianz verfeinert. Anschließend werden die lokalen Modelle mit Query-Punkten so aufgefüllt, dass eine geforderte Mindestpunkteanzahl pro LM eingehalten wird. Der Vorteil dieses Verfahrens ist, dass ein einzelnes Modell iterativ während der Messung verfeinert wird. Dadurch spart das Verfahren Rechenzeit und ist für einen Online-Einsatz auch bei schnell durchzuführenden Messungen gut anwendbar. Prinzipiell eignet sich dieser Ansatz für Anwendungen, bei denen die Rechenzeit während der Messungen im Vordergrund steht.
3. Durch eine aktive Versuchsplanung mit Single-Query-Strategie: Die beste Strategie im Sinne des aktiven Lernens ist es, immer nur einzelne Queries zur Messung an den Prozess zu übermitteln und darüber hinaus nach jedem Erweitern des Datensatzes ein neues Modell zu trainieren. Die für dieses Einsatzfeld zwingend notwendigen Eigenschaften bringt der Hilomot-Trainingsalgorithmus mit sich:
   - Schnelles Training,
   - deterministische Modelle,
   - eine hohe Modellgüte,
   - Robustheit für den Online-Einsatz und
   - eine automatische Komplexitätsauswahl.

   Nach jedem Modelltraining werden die LM nach ihrer Fehlervarianz klassifiziert und schließlich in diesem Gebiet raumfüllend ein neuer Query-Punkt platziert. Der besondere Reiz des Verfahrens liegt darin, dass durch die automatische Komplexitätsbestimmung des Modells je nach Qualität und Umfang der gegenwärtigen Daten gleichzeitig automatisch ein Kompromiss aus global raumfüllender und optimaler Versuchsplanung bezüglich des Modellfehlers gefunden wird.

Wichtig beim aktiven Lernen ist zudem, dass während des Messens beurteilt werden kann, ob das Modell bereits genau genug ist oder ob weitere Messungen zur Verbesserung der Güte notwendig sind. Es wurde dazu ein Verfahren vorgeschlagen, welches anhand des Bestimmtheitsmaßes, ausgewertet mit dem Leave-One-Out-Kreuzvalidierungsfehler, die Modellgüte bewertet. Wenn sich an der Modellgüte nichts mehr signifikant ändert, kommt es zur Versuchsbeendigung. Festgestellt wird die Sättigung des Fehlerverlaufs, indem dieser tiefpassgefiltert wird, einmal mit einer niedrigen Grenzfrequenz und einmal mit einer hohen Grenzfrequenz. Ist die Differenz aus langsam und schnell gefiltertem Signal kleiner als eine Toleranz, bricht die Messung ab.

Als Ausblick wurde der Einsatz von HILOMOT-Modellkomitees motiviert. Die grundlegende Idee dahinter ist, viele Modellvarianten zu trainieren, um dann durch Mittelung der Modelle den Varianzfehler zu minimieren. Gleichzeitig kann die Varianz bzw. der Dissens der Einzelmodelle für die aktive Versuchsplanung genutzt werden, wie dies auch bei in der Praxis etablierten Verfahren, wie z.B. dem *mbminimize*-Algorithmus [126], [145], der Fall ist. Zum einen bietet sich die Methode des Bootstrappings an, um verschiedene Varianten zu generieren. Jedoch muss man hierzu sehr viele Modelle trainieren, um eine ausreichende Randomisierung zu gewährleisten. Alternativ dazu können wenige HILOMOT-Modelle mit unterschiedlicher Konfiguration trainiert werden und diese dann mit geeigneten Gewichtungsfaktoren ins Komitee einfließen. Grundsätzlich sollten hierfür die Modelle eher offensiv ausgelegt sein, d.h. die Komplexität mehr in Richtung Overfitting eingestellt sein. Die höhere Varianz der Einzelmodelle verringert sich durch die Mittelung im Komitee; auf den Biasanteil hat das Komitee jedoch keinen Einfluss. Die geschickte Auswahl und Konfiguration der Komiteeteilnehmer, auch die Hinzunahme anderer Modellarchitekturen wie beispielsweise Gaußprozessmodelle zum Komitee, liefern spannende Themen für zukünftige Forschung.

6 Anwendungen

Das Kapitel befasst sich mit der praktischen Anwendung der in dieser Arbeit vorgestellten Verfahren HILOMOT und HILOMOTDOE. In Abschnitt 6.1 wird die aktive Versuchsplanung eines Prozesses zur Schadenslokalisation im Rahmen eines Structural Health Monitoring-Systems gezeigt. Abschnitt 6.2 befasst sich mit der globalen Modellbildung eines Dieselmotoren-Datensatzes. In Abschnitt 6.3 wird die aktive Versuchsplanung für einen Dieselmotor am Motorenprüfstand präsentiert. Das Kapitel schließt mit einer Zusammenfassung ab.

6.1 Structural Health Monitoring

Beim Structural Health Monitoring (SHM) möchte man technische Strukturen oder Bauteile zerstörungsfrei während des laufenden Betriebs kontinuierlich überwachen. Aus wirtschaftlichen Gründen werden Strukturen oder Bauteile immer öfter nicht auf die vollständige Lebensdauer einer Anlage ausgelegt, sondern müssen in bestimmten Zeitabständen gewartet oder ausgetauscht werden [108]. In diesem Zusammenhang spielen daher Überwachungssysteme eine wichtige Rolle, die in der Lage sind, sich anbahnende Schäden frühzeitig zu *detektieren* und darüber hinaus auch zu *lokalisieren*. Die Forschungsbemühungen in diesem Feld gehen sogar soweit, dass man nach der Schadensdetektion und -lokalisation auch das Schadensausmaß und eine Prognose zur Restlebensdauer mit einem SHM-System anstrebt [98]. Die Anwendungsgebiete können sehr vielfältig sein, z.B. die Überwachung von Brücken, in der Luft- und Raumfahrt, bei Windkraftanlagen oder Rohrleitungen [157].

Die Hauptvorteile von SHM im Vergleich zu klassischen Verfahren der zerstörungsfreien Materialprüfung sind:

- Die Überwachung komplizierter Strukturen ist möglich, z.B. Flugzeugaußenwände.
- Zur Überwachung ist kein Ausbau der Bauteile nötig, was Kosten spart.
- Schäden an der Struktur können bereits in einem frühen Stadium erkannt werden.
- Mit SHM können die Strukturen durch permanent installierte Sensorik während des laufenden Betriebs überwacht werden.

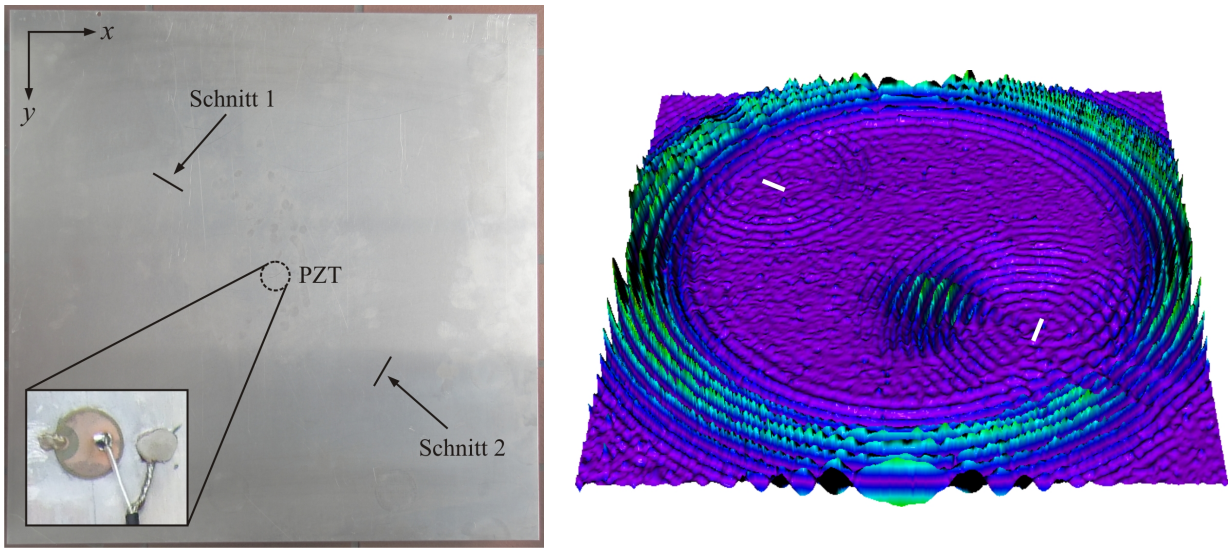

Bild 6.1: Links: Aluminiumplatte mit zwei eingebrachten Schnitten, die unterschiedlich zum piezoelektrischen Aktor ausgerichtet sind. Rechts: Schnappschuss der Wellenausbreitung, gemessen mit einem Laser-Doppler-Vibrometer (Quelle: [108]).

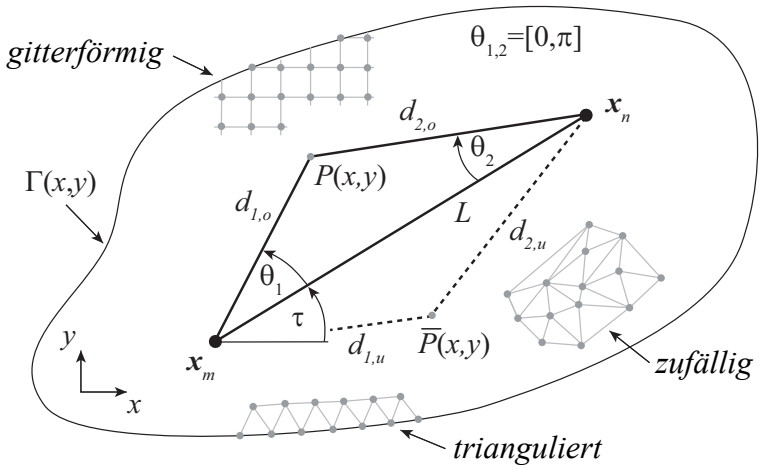

Bild 6.2: Die Berechnung der Schadenskarte kann nur räumlich diskretisiert innerhalb des Versuchsraums Γ(x,y) erfolgen. Standardmäßig werden die Stützstellen gitterförmig oder zufällig verteilt bzw. nach einem Triangulierungsverfahren (Quelle: [109]).

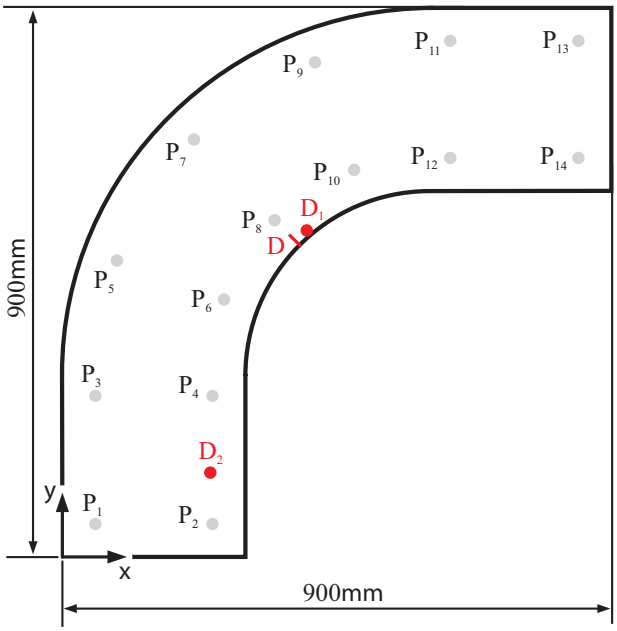

Bild 6.3: Geometrie der untersuchten Struktur mit den Positionen der 14 Aktoren P_1 bis P_{14}. Für die Simulation wurden die Schadenspositionen D_1 und D_2 gewählt. Im Experiment war der Schadensort bei Position D (Quelle: [109]).

Prinzipiell gibt es viele Ansätze, SHM durchzuführen [9],[40]. Eine Möglichkeit davon basiert auf dem Einsatz von Ultraschallwellen. Das hier untersuchte Anwendungsgebiet von SHM bezieht sich im Speziellen auf dünnwandige Strukturen, wie sie beispielsweise im Flugzeugbau Anwendung finden. Für die Experimente werden auf eine dünne Aluminiumplatte piezoelektrische Elemente aufgeklebt. Diese können sowohl als Sensor als auch als Aktor dienen. Die Struktur wird abwechselnd mit verschiedenen piezoelektrischen Aktoren aktiv angeregt. Die entstehenden Ultraschallwellen laufen dann durch die Struktur und können von den übrigen Piezoelementen gemessen werden. Die Idee ist nun, dass man einen ungeschädigten Referenzzustand der Struktur bestimmt, um im zukünftigen Betrieb aktuelle Messungen mit dem Referenzzustand vergleichen zu können. Ändert sich die Charakteristik der Messsignale im Vergleich zum ungeschädigten Zustand, kann diese Information zur Detektion und Lokalisation von Strukturschäden genutzt werden. Hierzu nutzt man zudem aus, dass die verwendeten Ultraschallwellen sensitiv bezüglich verschiedener Schadenstypen sind, wie z.B. Risse, Delaminationen in Verbundwerkstoffen oder auch Korrosionsschäden [108].

In Bild 6.1 sieht man den Schnappschuss einer Wellenausbreitung durch eine Aluminiumplatte [108]. Man erkennt deutlich, dass die vorhandenen Schäden je nach Ausrichtung zum Aktor Reflexionen der Wellen verursachen. Diese Reflexionen erzeugen ein zum Referenzzustand unterschiedliches Wellenbild und können durch geschickte Messung und Auswertung Aufschluss über den Schadensort liefern.

Mit Algorithmen, die z.B. in [108] vorgestellt werden, lässt sich eine sogenannte Schadenskarte zur Visualisierung der Schadenswahrscheinlichkeiten zu der überwachten Strukturgeometrie berechnen. Hierzu rechnet man für diskrete Punkte $P(x, y)$ bzw. $\bar{P}(x, y)$ auf der Plattengeometrie Intensitätswerte aus, siehe Bild 6.2. Der höchste Intensitätswert gibt Auskunft über den Schadensort. Klassische Strategien platzieren die Punkte innerhalb des Ver-

suchsraums $\Gamma(x,y)$ nach einem geometrischen Muster. Möglich hierbei sind beispielsweise die Triangulierung der Struktur, eine raumfüllende Verteilung oder auch eine gitterförmige oder zufällige Platzierung der Stützstellen. Ein wesentlicher Nachteil der geometrischen Platzierung ist jedoch, dass die zugrundeliegende Prozesscharakteristik keinen Einfluss auf die Stützstellenpositionen hat. Dadurch werden u.U. unnötig viele Punkte erzeugt, was sich durch eine intelligentere Platzierung vermeiden ließe. Die Auswertung ist durch die eingesetzten Visualisierungsverfahren sehr rechenintensiv, was für diesen Anwendungsfall den Einsatz einer aktiven Versuchsplanung mit HILOMTDOE motiviert. Die auszuwertenden Stützstellenpositionen auf der Plattengeometrie sollen unter Berücksichtigung der zunächst unbekannten Schadenskarte möglichst effizient platziert werden. Interessant sind vor allem Regionen auf der Struktur, die einen Schaden vermuten lassen. Daher wäre eine dichtere Platzierung in der Nähe des Schadens wünschenswert. Genau dieses Ziel wird nun durch den Einsatz des aktiven Lernens mit HILOMOTDOE erreicht, indem die virtuellen Messpunkte $P(x,y)$ und $\bar{P}(x,y)$ gemäß der Schadensintensitätswerte verteilt werden. Gleichzeitig erhält man ein nichtlineares HILOMOT-Modell des Prozesses.

Die hier gezeigten Ergebnisse entstanden in der Zusammenarbeit mit der Arbeitsgruppe von Professor Dr.-Ing. Fritzen[1]. Alle durchgeführten Simulationen und Messungen bezüglich SHM wurden dort durchgeführt, siehe auch [110, 109, 51].

Zunächst findet die Stützstellenerstellung für eine Schadenskarte mit HILOMOTDOE auf Basis von Simulationsdaten statt. Daraufhin erfolgt die Untersuchung bezüglich experimentell ermittelter Ultraschallsignale.

6.1.1 Simulationsergebnisse

Basierend auf den Lokalisationsmethoden, die in [108] vorgestellt wurden, fand die Diskretisierung für die Geometrie in Bild 6.3 zunächst auf der Basis von Simulationsdaten statt. Zur Erzeugung der Kandidatenpunkte innerhalb des Versuchsraums wurde die Plattengeometrie mit einem Polygonzug nachgebildet, um diese damit klassifizieren zu können. Die nichtkonvexe Struktur hat die Abmessungen $900mm$ x $900mm$ und eine Dicke von $1.5mm$. Bild 6.3 zeigt die Geometrie inklusive der Aktorpositionen P_1 bis P_{14}. Für die Simulation wurden zwei Schadenspositionen D_1 und D_2 als Szenario untersucht. Das nachfolgende Experiment beschränkte sich auf die Schadensposition $D{=}D_1$.

Für den zugrunde liegenden Prozess wurde die räumliche Diskretisierung mittels HILOMOTDOE (Single-Query-Strategie) durchgeführt. Bild 6.4 zeigt die Simulationsergebnisse für das Schadensszenario D_1. Während der Schadenslokalisation adaptiert das Verfahren die Schadensintensität in Abhängigkeit der geometrischen Koordinatenachsen x und y. Mit zunehmender Anzahl an Auswertungen steigt die Qualität des HILOMOT-Modells und desto präziser werden die Stützstellen in der Nähe des Schadens platziert. Mit insgesamt 30 Auswertungen ist die Modellqualität hoch genug, um eine genaue Schadensposition zu ermitteln. Der Schadensfall D_2 ist in Bild 6.5 gezeigt. Ähnlich zum ersten Szenario findet auch hier HILOMOTDOE die richtige Schadensposition und legt entsprechend die Stützstellen in der Schadensregion dichter. Dieser Fall unterstreicht die hohe Anpassungsfähigkeit des HILOMOTDOE-Verfahrens.

[1] Universität Siegen, Fakultät IV, Department Maschinenbau, Institut für Mechanik und Reglungstechnik - Mechatronik, Arbeitsgruppe für Technische Mechanik.

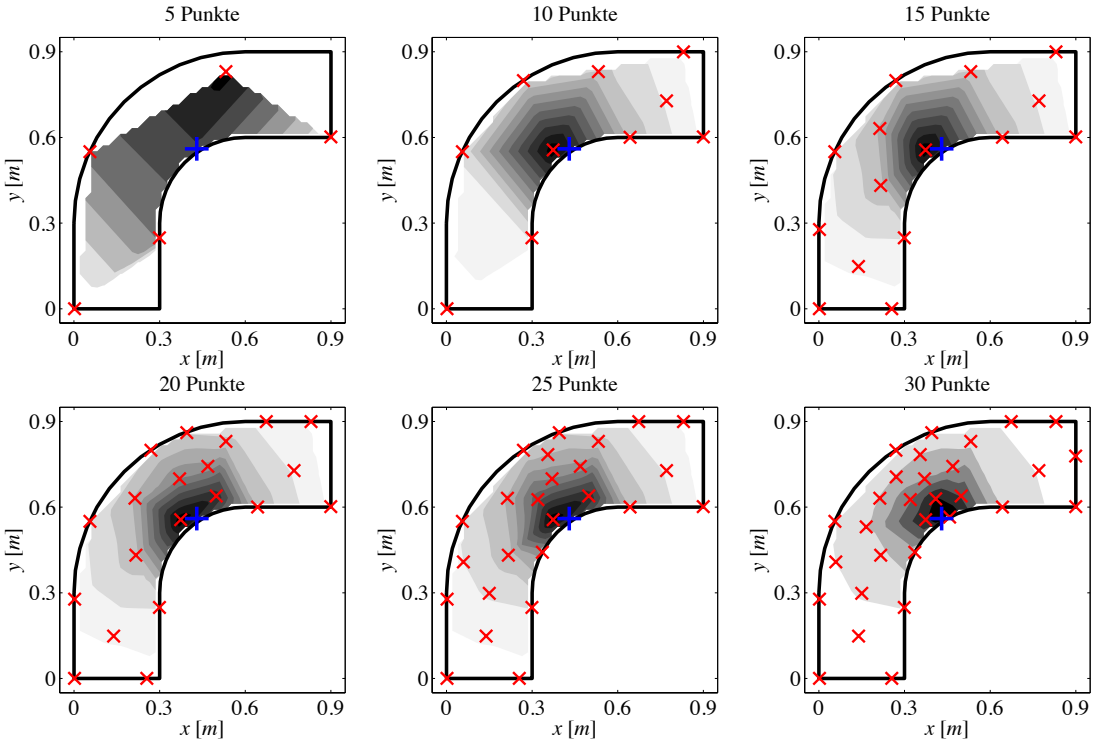

Bild 6.4: Simulationsergebnisse mit HILOMOTDOE für Schadensfall D_1. Der Schaden ist mit einem blauen + markiert, die Messpunkte mit rotem ×. Die Höhenlinien entsprechen dem gegenwärtigen Prozessmodell.

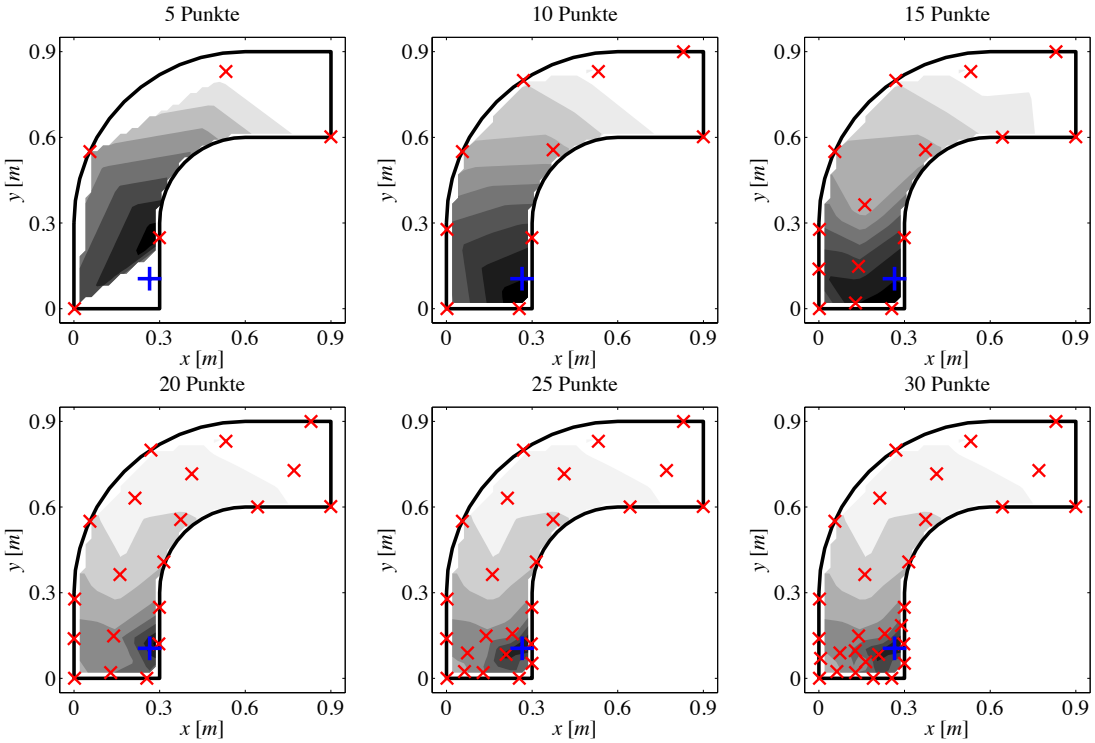

Bild 6.5: Simulationsergebnisse mit HILOMOTDOE für Schadensfall D_2.

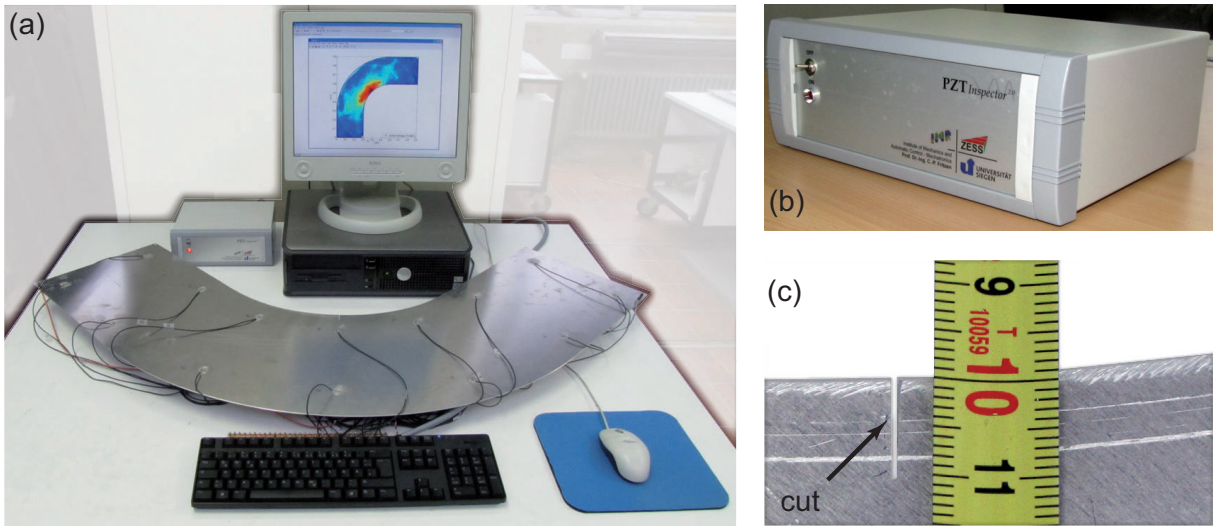

Bild 6.6: (a) Experimenteller Aufbau mit der untersuchten, nichkonvexen Struktur. (b) Umschalter für die Messdatenaufnahme. (c) Eingebrachter Schaden in Form eines $10mm$ langen und $1mm$ breiten Schnitts (Quelle: [109]).

6.1.2 Experimentelle Ergebnisse

Der experimentelle Aufbau ist in Bild 6.6 gezeigt. Er besteht im Wesentlichen aus drei Komponenten: Dem Rechner für die Messdatenverarbeitung, einem Umschalter für die Messdatenaufnahme bezüglich aller Aktor-Sensor-Kombinationen und der untersuchten Aluminiumplatte, die mit 14 aufgeklebten piezoelektrischen Aktoren und einem Temperatursensor bestückt wurde. Die Platte hat die gleichen Ausmaße wie die Platte in vorangegangener Simulation.

Die räumliche Diskretisierung wurde auch hier aktiv mit HILOMOTDOE durchgeführt. Im Unterschied zu den zuvor gezeigten Simulationen, wurde der Lernalgorithmus nun beim realen Prozess angewendet. Auch hier zeigt sich, dass HILOMOTDOE die Effizienz der Diskretisierung signifikant verbessern kann, siehe Bild 6.7. Mit 25 Auswertungen ergibt sich für die aktiv gelernten Stützstellen im Vergleich zu einem Gitteransatz ein deutlich besseres Modell zur Lokalisation des Schadens.

Wie bei den Simulationsergebnissen zeigt Bild 6.8 die Entwicklung der Diskretisierung für die experimentelle Konfiguration. Obwohl der Prozess hier im Gegensatz zur Simulation mit Rauschen behaftet ist, kann HILOMOTDOE die Stützstellen in die korrekte Region dicht platzieren, d.h. dort, wo tatsächlich auch der Schaden vorliegt. Bild 6.9 zeigt zudem die hohe Güte des HILOMOT-Modells. Das Maximum des Modellausgangs liegt präzise an der Position des wahren Schadens. Darüber hinaus zeigt sich die gute Generalisierungsleistung und Unempfindlichkeit gegenüber dem Rauschen.

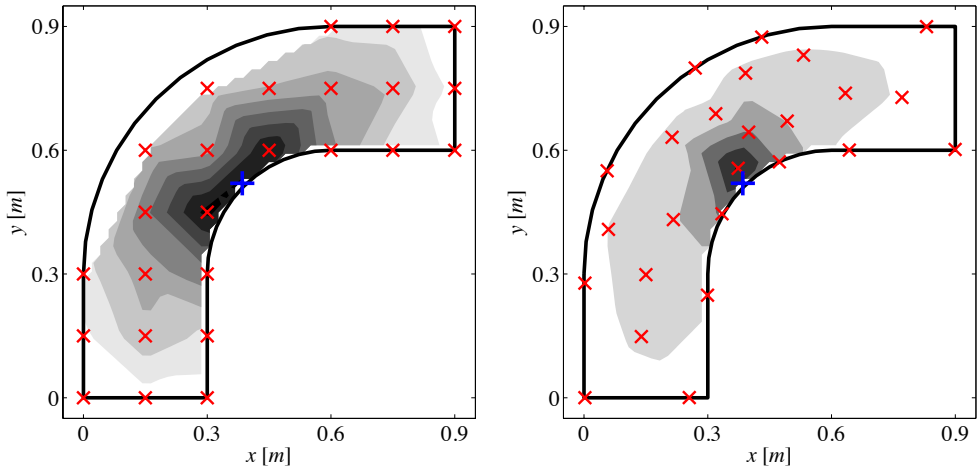

Bild 6.7: Experimentelle Ergebnisse mit 25 Auswertungen mit einer gitterförmigen Diskretisierung (links) und einer aktiv gelernten Auswertung (rechts). Das Bild zeigt, dass mit HILOMOTDOE bei gleicher Punktanzahl eine deutlich genauere Schadenslokalisation im Vergleich zu einer gitterförmigen Auswertung erzielt wird.

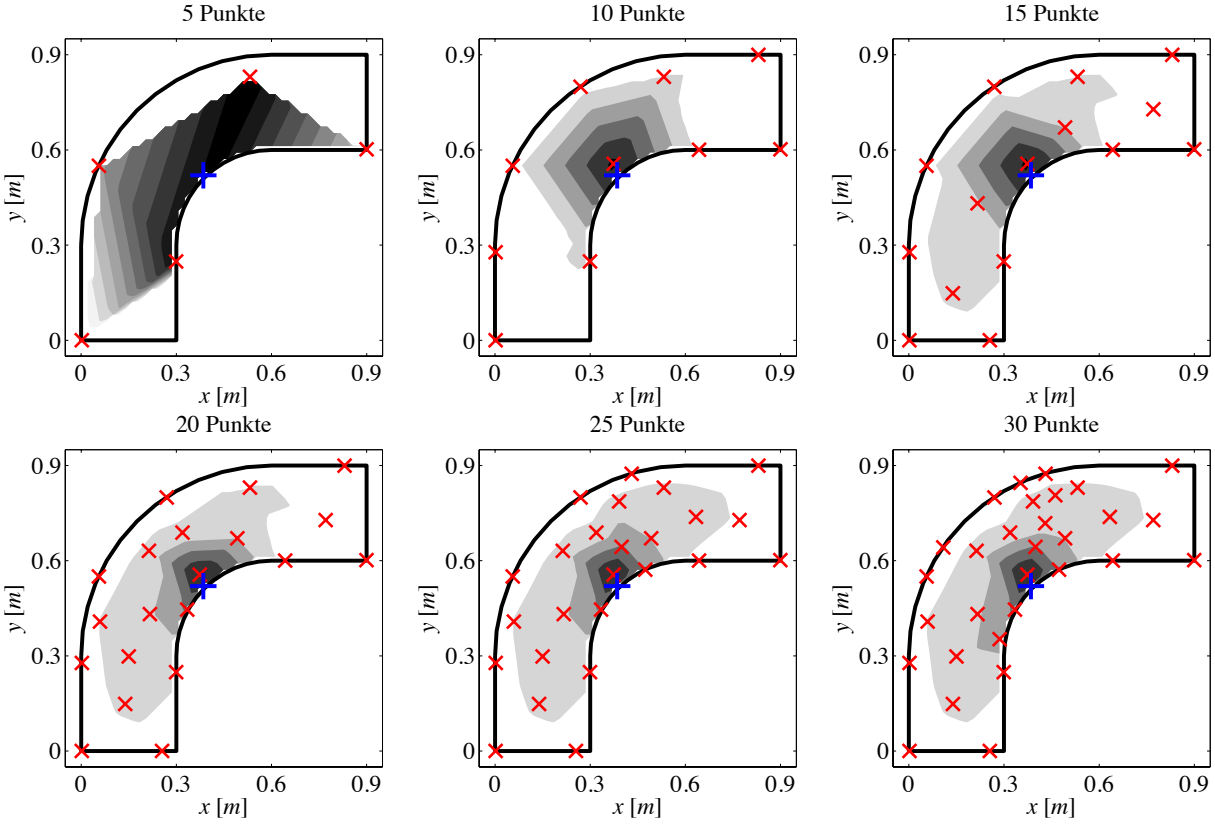

Bild 6.8: Experimentelle Ergebnisse mit Schadensszenario D aus Bild 6.3. Die mit HILOMOTDOE geplanten Auswertungen sind mit roten × markiert. Der Schaden ist mit einem blauen + gekennzeichnet. Das zugrunde liegende Prozessmodell ist mit den Höhenlinien illustriert. Mit zunehmender Anzahl an Auswertungen verbessert sich die modellbasierte Schadenslokalisation.

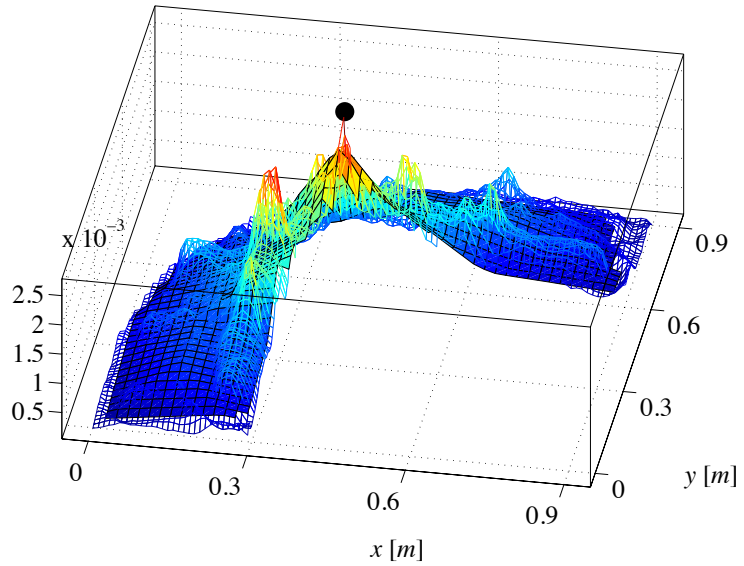

Bild 6.9: HILOMOT-Modell (undurchsichtig) im Vergleich zum Intensitätsprozess (durchsichtig). Der exakte Schaden ist mit dem schwarzen Punkt markiert.

6.2 Verbrauchs- und Emissionsmodellierung eines Dieselmotors

Der HILOMOT-Algorithmus wurde in Kooperation mit der IAV GmbH anhand realer Prüfstandsdaten eines modernen Dieselmotors für die globale Modellbildung getestet, siehe [48]. Dazu konnten Mitarbeiter der IAV GmbH die Modellierung mit Hilfe der in MATLAB implementierten Toolbox „LMNTOOL“ durchführen, welche in [49] vorgestellt wurde. Die objektorientiert programmierte Toolbox entstand im Rahmen dieser Arbeit in der Arbeitsgruppe von Professor Nelles an der Universität Siegen. Sie beinhaltet die Trainingsalgorithmen LOLIMOT und HILOMOT. Es werden automatisch lokale Modellnetze mit unterschiedlicher Konfiguration trainiert und anhand des korrigierten AIC-Kriteriums das beste Modell ausgewählt. Neben den Trainingsdaten können dem Verfahren auch Validierungs- und/oder Testdaten übergeben werden. Bei Vorgabe von Validierungsdaten, bestimmen diese über die Komplexität des Modells anstelle des AIC_c-Kriteriums. Testdaten dienen ausschließlich der Fehlerberechnung der fertig trainierten Modelle. Die vorgestellten Ergebnisse haben auch deshalb hohe praktische Relevanz, da sie von unabhängigen Anwendern erzeugt und daher kein Expertenwissen über die Toolbox bei der Modellbildung eingebracht wurde.

Nachfolgend werden zunächst die durchgeführten Messungen und die zum Vergleich herangezogenen Modellierungsverfahren, die im Rahmen der modellbasierten Kalibrierung von Verbrennungsmotoren üblich sind, kurz beschrieben. Daraufhin werden die Ergebnisse beschrieben. Der Fokus liegt hierbei sowohl auf dem Vergleich der Modellierungsverfahren als auch auf der Entwicklung der Modellgüte mit zunehmender Datenmenge.

6.2.1 Durchgeführte Messungen am Dieselmotor

Für die Kalibrierung eines modernen Dieselmotors muss eine große Anzahl an Stellgrößen berücksichtigt werden, was den Einsatz modellbasierter Methoden wie statistische Versuchsplanung (DoE) notwendig macht. Bei der zugrunde liegenden Motorvermessung fand die Verstellung folgender 11 Eingangsgrößen statt:

- Drehzahl,
- Last,
- Start der Haupteinspritzung,
- Ladedruck,
- Luftmenge,
- Raildruck,
- Drallklappenstellung,
- Ansteuerbeginn der ersten Voreinspritzung,
- Menge der ersten Voreinspritzung,
- Ansteuerbeginn der zweiten Voreinspritzung und
- Menge der zweiten Voreinspritzung.

Die Verteilung der gesamten Messpunkte erfolgte als eine Mischung aus statistisch optimaler und raumfüllender Versuchsplanung. Nach der Messung wurden die Daten nach einem raumfüllenden Kriterium reduziert, um Trainingsdatensätze mit verschiedener Punktzahl zu erzeugen. Die Datensätze hatten dann 349, 662, 944, 1323 und 1889 Messpunkte. Außerdem wurden die Daten so gefiltert, dass nur physikalisch sinnvolle Werte in Frage kommen. Beispielsweise wurde der effektive Kraftstoffverbrauch auf Werte unter 800 g/kWh beschränkt. Zur Beurteilung der Modellgüte stand ein Testdatensatz mit insgesamt 781 Punkten zur Verfügung. Diese Punkte wurden so platziert, dass sie innerhalb der konvexen Hülle des kleinsten Datensatzes lagen, um Extrapolation des Modells zu vermeiden. Die Berechnung der konvexen Hülle erfolgte lediglich für die ersten sechs Stellgrößen in der oberen Liste. Für die übrigen Eingangsgrößen wurden die oberen und unteren Grenzen in Abhängigkeit von Drehzahl und Last individuell vorgegeben.

6.2.2 Referenzmodelle

Zur Einordnung der Resultate mit HILOMOT fand die Modellbildung zusätzlich mit einem Polynomansatz sowie Gaußprozessmodellen (GPM) statt.

Polynommodell

Der für Kalibrieraufgaben meist verwendete Modellansatz ist das Polynommodell, weil es einfach zu verstehen ist und sowohl Training als auch Auswertung wenig rechenintensiv sind. Die Hauptnachteile sind jedoch die fehlende Flexibilität und ein oft unbrauchbares Extrapolationsverhalten.

Zur Modellbildung mit Polynomen wurde die Implementierung in der *IAV EasyDoE ToolSuite* [1] genutzt. Die Toolbox beinhaltet verschiedene Algorithmen zur Variablenselektion und zur robusten Regression, die auf die Polynommodelle angewendet wurden, um anschließend das beste Modell automatisch anhand statistischer Tests auszuwählen. Die Modelle

können zudem innerhalb der Toolbox zur Stellgrößenoptimierung genutzt oder zur weiteren Nachbearbeitung zu MATLAB exportiert werden. Darüber hinaus können verschiedene Zonen für bestimmte Drehzahl-Last-Bereiche definiert werden, um auf diese Weise als Anwender Vorwissen für die Modellbildung einzubinden. In diesem Vergleich findet jedoch keine Unterteilung in verschiedene Zonen statt.

Gaußprozessmodell

Gaußprozessmodelle [128, 17] sind mittlerweile im Bereich der modellbasierten Kalibrierung ein beliebtes Verfahren geworden. Die Modelle sind sehr flexibel und erfordern wie der HILOMOT-Algorithmus nur sehr wenig Interaktion mit dem Benutzer. Das macht die Methode zu einem robusten Hilfsmittel für die Generierung von Motormodellen.

Für den Vergleich wurde eine Implementierung der GPM mit der Software *IAV EasyDoE* genutzt, welche eine aktuelle Nachfolgeversion der EasyDoE Toolsuite darstellt und speziell für die Anwendung der modellbasierten Motorkalibrierung entwickelt wurde, siehe [14].

6.2.3 Ergebnisse

Die Ergebnisse wurden anhand der Wurzel des mittleren quadratischen Modellfehlers (RMSE) ausgewertet:

$$J_{\mathrm{RMSE}} = \sqrt{\frac{1}{N} \cdot \sum_{i=1}^{N} (y(i) - \hat{y}(i))^2}\,, \tag{6.1}$$

mit der Anzahl N der Trainings- bzw. Testdatenpunkte. Tabelle 6.1 fasst die Ergebnisse zusammen.

Exemplarisch sind die RMSE-Werte für die Modellbildung von NOx und HC in den Bildern 6.10 und 6.11 in Abhängigkeit von der Datenpunktanzahl dargestellt. Die Trainingsdatenfehler sind durch gestrichelte Linien, die Testdatenfehler mit durchgezogenen Linien verbunden.

Bild 6.10 zeigt, dass die Modellqualität auf Testdaten für die Ausgangsgröße NOx generell sehr ähnlich ist. Wie zu erwarten, verringert sich für alle drei Verfahren der Testdatenfehler mit zunehmender Anzahl an Trainingsdaten. Ferner zeigt sich, dass die Trainingsdatenfehler für das Polynommodell und das GPM mit zunehmender Punktanzahl ansteigt. Dies kann ein Indiz dafür sein, dass entweder unzuverlässige Messungen oder Messpunkte, die weit vom bisherigen Modell weg liegen, hinzugenommen wurden. Die Analyse eines normierten Histogramms könnte hier Aufschluss geben, ob es sich um Ausreißer handelt.

Tabelle 6.1: RMSE-Werte für die verschiedenen Modellansätze und Datenmengen.

		Polynom		**HILOMOT**		**GPM**	
	Anz. Punkte	Train	Test	Train	Test	Train	Test
NOx [ppm]	349	45	77	46	90	10	71
	662	52	68	44	70	13	62
	944	52	70	39	64	18	60
	1323	57	66	40	62	24	56
	1889	56	65	37	59	26	54
CO [ppm]	349	153	225	143	210	118	201
	662	199	196	190	190	24	217
	944	206	177	103	174	116	162
	1323	203	175	107	155	71	158
	1889	203	169	100	153	88	145
THC [ppm]	349	11	19	15	18	8	28
	662	13	14	13	15	10	18
	944	13	14	11	13	6	14
	1323	13	14	11	13	8	13
	1889	13	13	8	12	6	12
Ruß [FSN]	349	0.207	0.320	0.324	0.372	0.154	0.287
	662	0.231	0.256	0.252	0.275	0.150	0.229
	944	0.225	0.241	0.207	0.261	0.144	0.209
	1323	0.230	0.230	0.183	0.227	0.147	0.196
	1889	0.231	0.219	0.183	0.205	0.141	0.176
Spez. Verbrauch [g/kWh]	349	20.01	18.86	21.74	20.91	11.71	15.47
	662	22.57	16.60	17.78	15.62	11.63	14.76
	944	22.41	16.55	13.48	15.93	13.33	13.93
	1323	23.18	15.75	14.11	12.71	13.78	12.33
	1889	23.41	17.07	14.36	12.84	13.67	12.07

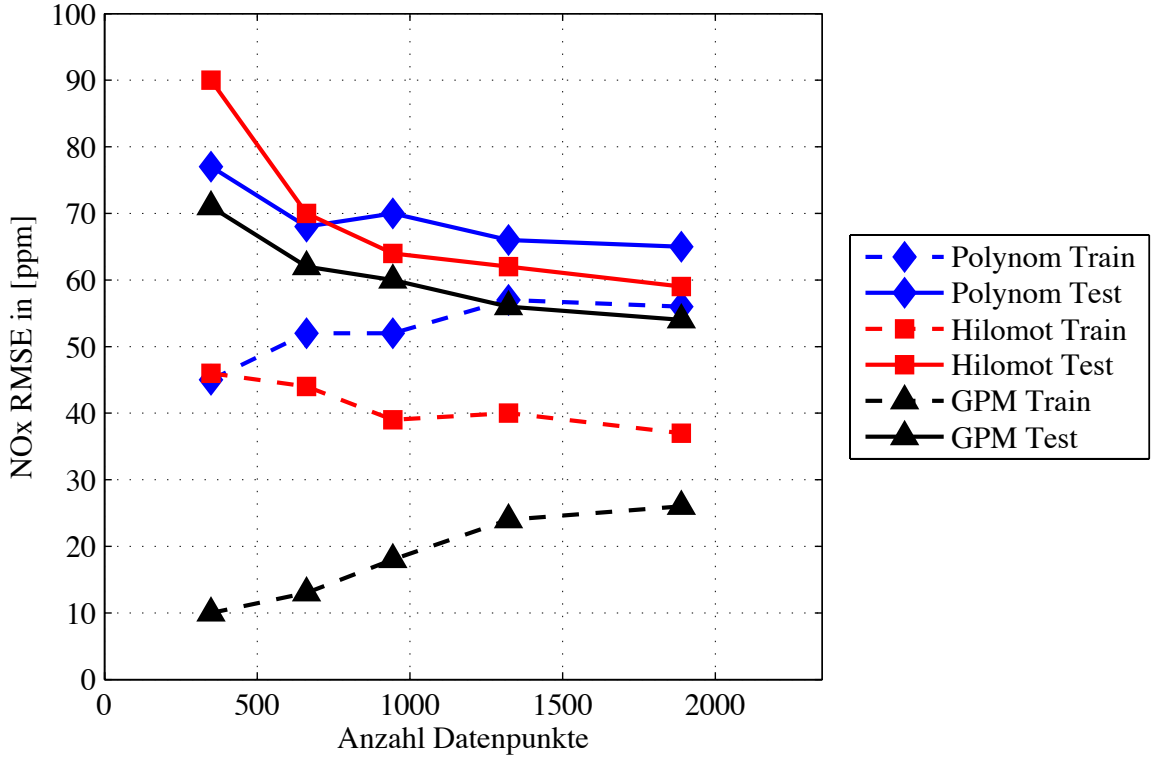

Bild 6.10: RMSE-Fehlerwerte für NOx

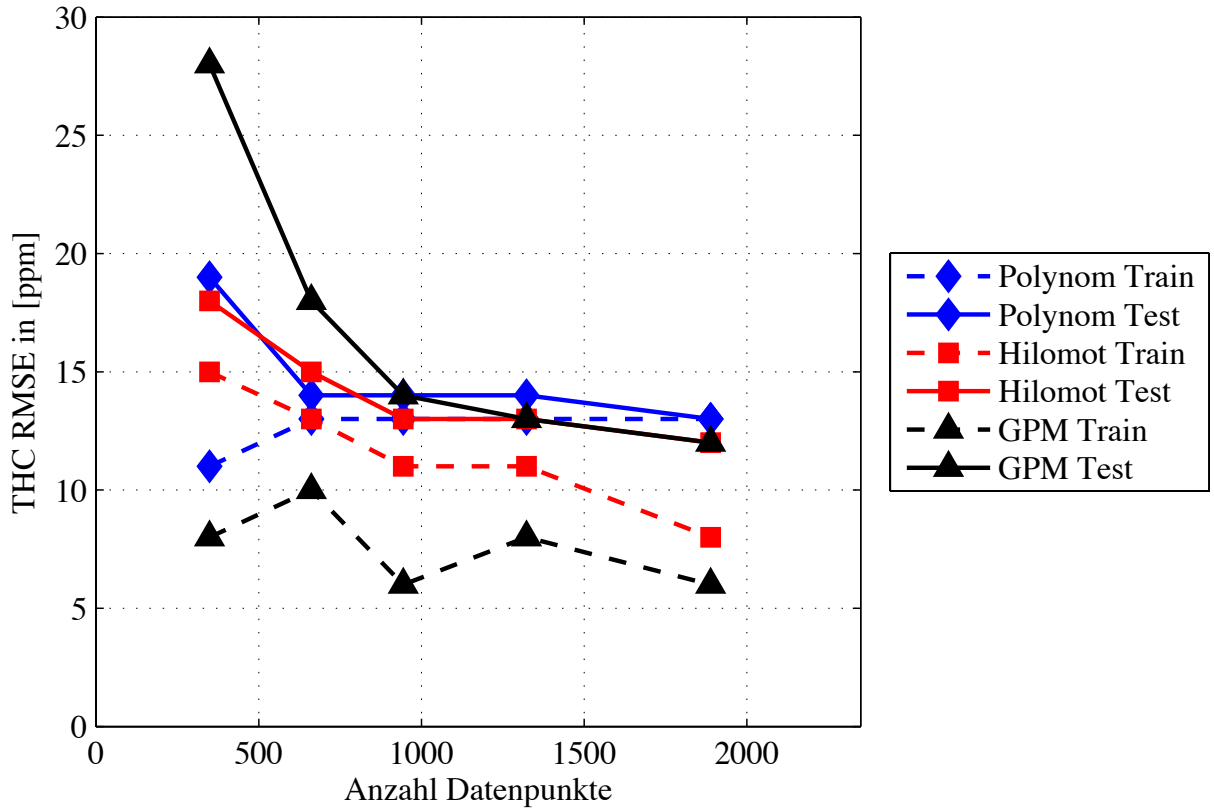

Bild 6.11: RMSE-Fehlerwerte für die HC-Emissionen

Bild 6.10 zeigt zudem, dass das HILOMOT-Modell dank der höheren Flexibilität bessere Ergebnisse liefert als das Polynommodell, sofern ausreichend Daten für das Training vorhanden sind. Polynome sind globale Approximatoren und liefern daher normalerweise auch dann brauchbare Ergebnisse, wenn nur sehr wenige Daten vorhanden sind, solange sich der Prozess durch polynomiale Nichtlinearitäten abbilden lässt und darüber hinaus keine Ausreißer in den Daten vorhanden sind, welche das Polynommodell maßgeblich verschlechtern können.

Beim Vergleich ist auch zu beachten, dass beim HILOMOT-Modell lokale Polynome zweiten Grades verwendet wurden, wohingegen das reine Polynommodell dritten Grades war. Wie sich in späteren Untersuchungen herausstellte, sind ein Großteil der Ergebnisunterschiede für geringe Datenmengen auf diese unterschiedliche Einstellung zurückzuführen. Das bedeutet, dass die Ergebnisse des reinen Polynommodells mit HILOMOT reproduziert werden konnten, wenn die lokalen Modelle dritten Grades gewählt wurden.

Der Verlauf des Trainingsdatenfehlers ist im Fall von HILOMOT-Modellen und GPM schwieriger zu bewerten als für Polynommodelle, da sich hierbei die Modellkomplexität mit unterschiedlicher Anzahl an Trainingsdatenpunkten verändert. Generell muss beim Vergleich von GPM mit HILOMOT-Modellen zudem beachtet werden, dass die Wahl der Komplexität bei HILOMOT nur in diskreten Stufen einstellbar ist, da sich die Modellkomplexität durch die Anzahl der lokalen Modelle ergibt. Daher kann sich u.U. die Parameteranzahl mit einem einzigen LM signifikant ändern. Dies hängt von der Dimensionalität und dem gewählten Grad des lokalen Polynoms ab.

Die Modellierungsergebnisse für die Ausgangsgröße der HC-Emissionen (englisch: Total Hydrocarbons, THC) in Bild 6.11 sind wieder für alle drei Modellansätze ähnlich, weil kein signifikanter Unterschied bei den Testdatenfehlern für mehr als 1000 Trainingsdatenpunkte erkennbar ist. Auch hier sinkt der Testdatenfehler mit zunehmender Datenmenge. Für weniger Datenpunkte zeigt sich im Fall von GPM eine schlechtere Güte als für das Polynommodell und das HILOMOT-Modell. Außerdem ist beim GPM ein größerer Unterschied zwischen Trainings- und Testdatenfehler zu sehen als bei den anderen beiden Modelltypen. Das ist ein Anzeichen für Overfitting; daher sollten die Ergebnisse mit GPM nie alleine auf Basis der Trainingsdaten beurteilt werden. Typischerweise ist die Leave-One-Out-Kreuzvalidierung ein geeignetes Mittel zur Bewertung der Modellgüte.

Zusammenfassend ergab der Modellvergleich, dass HILOMOT-Modelle eine sehr gute Alternative zu Polynommodellen für die Praxis darstellen. Darüber hinaus liefern sie vergleichbar gute Ergebnisse wie die Gaußprozessmodelle. Jedoch gibt es zudem den Vorteil, dass sich HILOMOT-Modelle deutlich schneller trainieren lassen, was die Möglichkeit bietet, diese Modellklasse ohne Probleme auch bei großen Datensätzen anzuwenden. Darüber hinaus sind HILOMOT-Modelle standardmäßig konservativ eingestellt und daher robust gegenüber Overfitting. Außerdem sind die Ergebnisse durch die Partitionierung interpretierbar.

Die durchgeführten Untersuchungen zeigen, dass die HILOMOT-Modellklasse bei geeigneter Datenbasis mit den Standardeinstellungen bessere Ergebnisse liefert als der Polynomansatz. Auch deshalb ist für zukünftige Versionen der IAV EasyDoE Toolsuite geplant, den HILOMOT-Ansatz als Alternative zu GPM und Polynommodellen verfügbar zu machen.

6.3 Aktive Online-Versuchsplanung am Motorenprüfstand

Dieser Abschnitt zeigt die Ergebnisse, welche in der Zusammenarbeit mit der Daimler AG erzielt wurden. Die in dieser Arbeit entwickelten Methoden HILOMOT und HILOMOTDOE konnten auf diese Weise auf die praktischen Anforderungen bei der modellbasierten Kalibrierung von Verbrennungsmotoren zugeschnitten werden. Alle durchgeführten Messungen fanden auf Prüfständen der Daimler AG statt und sind z.B. in [57] und [93] veröffentlicht worden. Die aktive Versuchsplanung erfolgte mit der Single-Query-Strategie, die in Kapitel 5.3.3 nachzulesen ist.

6.3.1 Motorenprüfstand und Automatisierungssystem

Eine nötige Voraussetzung für die Online-Versuchsplanung mit HILOMOTDOE ist, dass die Testläufe automatisiert durchführbar sind. Bild 6.12 zeigt die typische Anordnung eines Motorenprüfstands bei Mercedes-Benz Cars [94]. Die Komponenten sind: Das Prüfstandsautomatisierungssystem, das Indiziersystem, das Prüflaufautomatisierungssystem CAMEO[2] und die Software zur Kalibrierung und Messung [93].

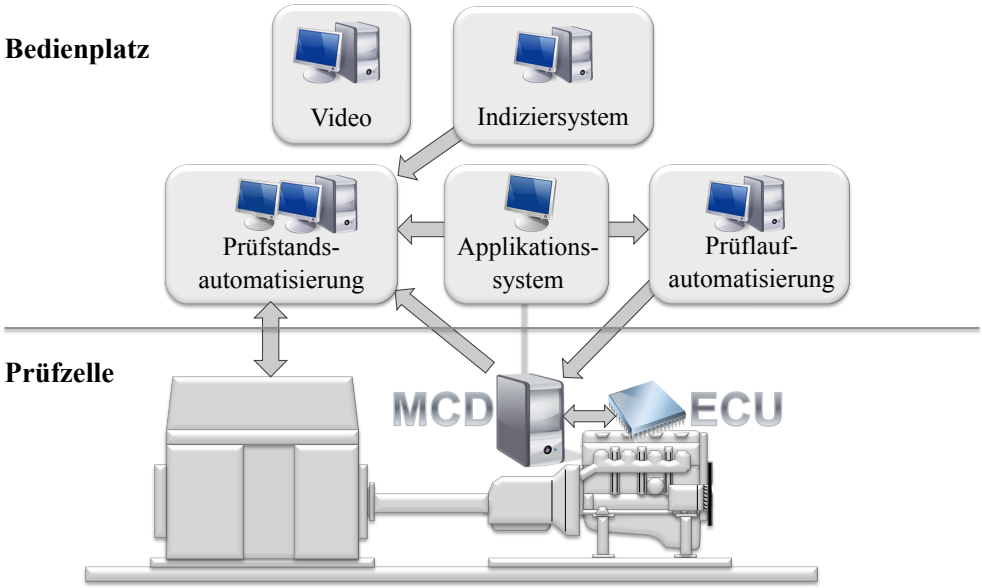

Bild 6.12: Typische Prüfstandsausstattung bei der Mercedes-Benz Cars Gruppe (Quelle: [94]).

Diese Standardkonfiguration des Prüfstandssystems muss erweitert werden, um eine adaptive Versuchsplanung zu ermöglichen. Um neue Messpunkte während des laufenden Motorversuchs einzuplanen, benötigt man eine zusätzliche externe Schnittstelle zum automatisierten Kalibrierungssystem. Dies kann beispielsweise für CAMEO über die sog. *iDoE*-Schnittstelle erfolgen. Prinzipiell kann dies aber auch für gleichwertige Systeme wie z.B. ORION[3] geschehen. Die iDoE-Schnittstelle ist eine Erweiterung der in [121] vorgestellten

[2]Prüfstandsautomatisierungssystem der Firma AVL GmbH.
[3]Prüfstandsautomatisierungssystem der Firma IAV GmbH.

Daimler Tool Suite und ermöglicht die folgenden Basisfunktionalitäten für ein externes MATLAB-System:

1. Senden eines initialen Versuchsplans an das automatisierte Kalibrierungssystem,
2. Hinzufügen neuer Messpunkte, die während des Versuchslaufs zu vermessen sind,
3. Empfangen der Messwerte und
4. Beendigung des Versuchslaufs.

Mit dieser Schnittstelle werden abwechselnd über ein MATLAB-System die neu zu messenden Queries an das automatisierte Kalibrierungssystem übermittelt und die Messwerte eingelesen. Der HILOMOTDOE-Algorithmus kann dadurch ebenfalls in MATLAB arbeiten. Bild 6.13 zeigt, welche Schritte des HILOMOTDOE-Algorithmus' während der Messung innerhalb der MATLAB-Umgebung ablaufen.

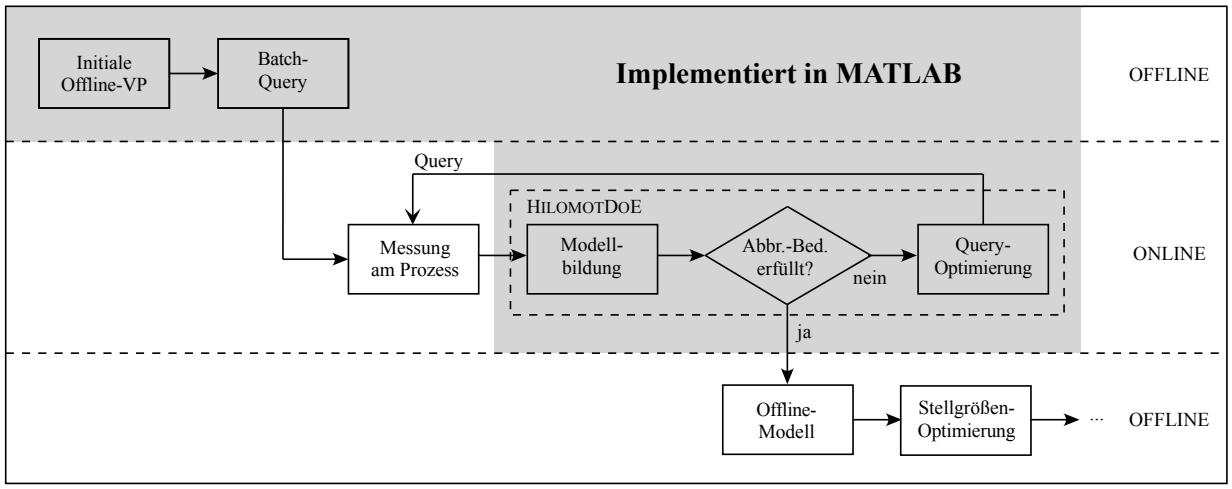

Bild 6.13: Die grau markierten Schritte laufen während der Prüfstandsmessung in einer MATLAB-Umgebung ab.

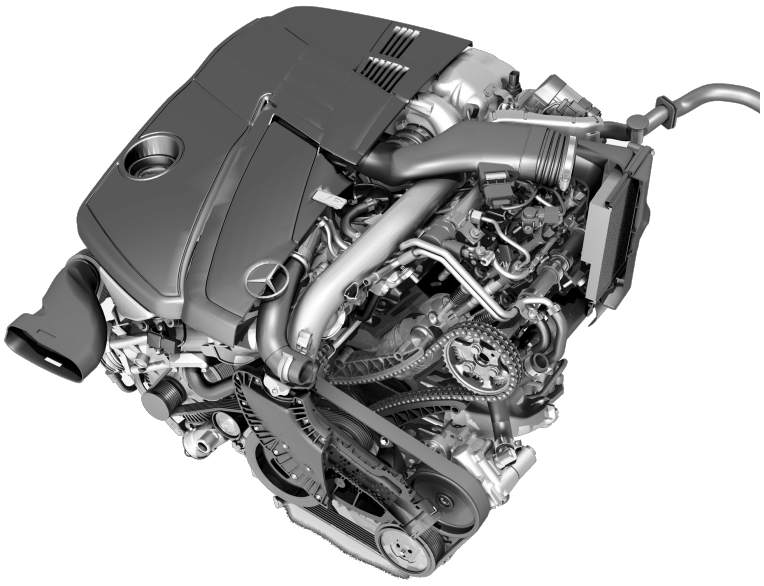

Bild 6.14: Mercedes-Benz V6 Dieselmotor OM 642 LS (Quelle: [93]).

Die Messungen erfolgten an einem Mercedes-Benz V6 Dieselmotor (interne Bezeichnung OM 642 LS), siehe Bild 6.14. Der Sechszylinder Dieselmotor hat einen Hubraum von 2987cm^3 und bringt eine maximale Leistung von 195kW bei 3800min^{-1} auf. Das Drehmoment beträgt 620Nm im Drehzahlbereich 1600-2400min^{-1}. Für detailliertere Informationen siehe [30].

Zur Validierung des HILOMOTDOE-Algorithmus' wurden mit diesem Motor die zwei folgenden Experimente durchgeführt: Zunächst fand eine Messung lokal für mehrere Drehzahl-Last-Kombinationen statt. Danach erfolgte die globale Versuchsplanung und Modellbildung für die gesamte Drehzahl-Last-Ebene.

6.3.2 Online-Versuchsplanung für einzelne Betriebpunkte

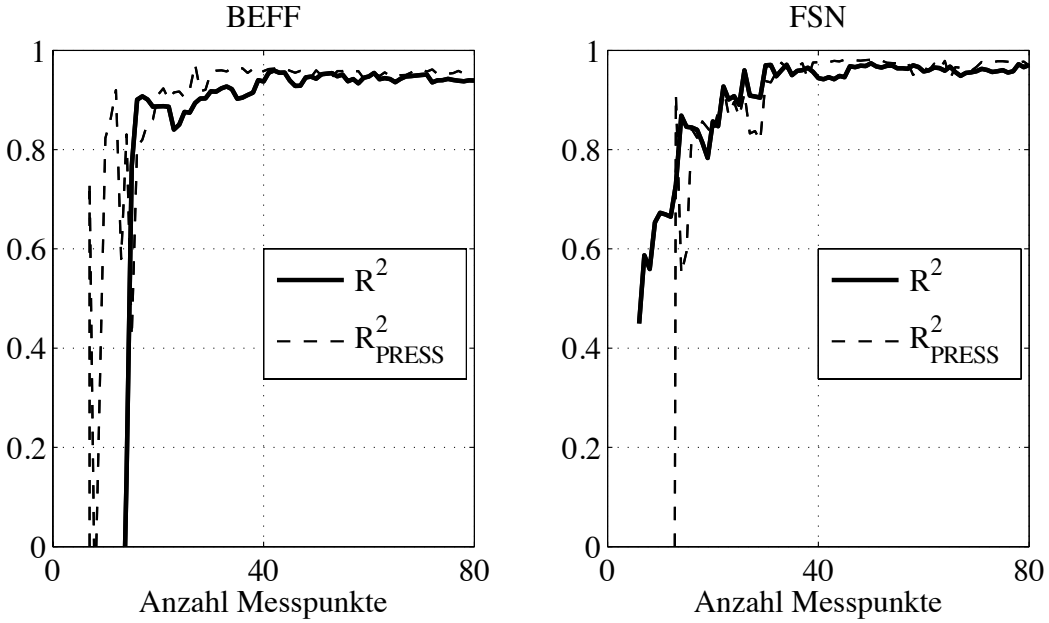

Bild 6.15: Versuchsplanung mit HILOMOTDOE bezüglich des spezifischen Verbrauchs BEFF. Gezeigt sind die Werte der Bestimmtheitsmaße R^2 und R^2_{PRESS} für die HILOMOT-Modelle bezüglich BEFF (links) und FSN (rechts). Man erkennt die gute Übereinstimmung der Fehlermaße.

Im ersten Schritt wurde HILOMOTDOE für die Versuchsplanung bezüglich lokaler Betriebspunkte[4] eingesetzt. Die Eingangsgrößen bei gegebener Drehzahl-Last-Kombination waren:

- Abgasrückführrate (AGR),
- Raildruck,
- Drallklappenstellung,
- 50%-Umsatzpunkt des Heizverlaufs (H_{50}-Umsatzpunkt) und
- Menge der Voreinspritzung.

Der H_{50}-Umsatzpunkt kann mit Hilfe einer Regelung am Motor vorgegeben werden. Eine genauere Beschreibung des Heizverlaufen bzw. des 50% Umsatzpunktes findet sich z.B. in

[4]Betriebspunkt = Eine Drehzahl-Last-Kombination.

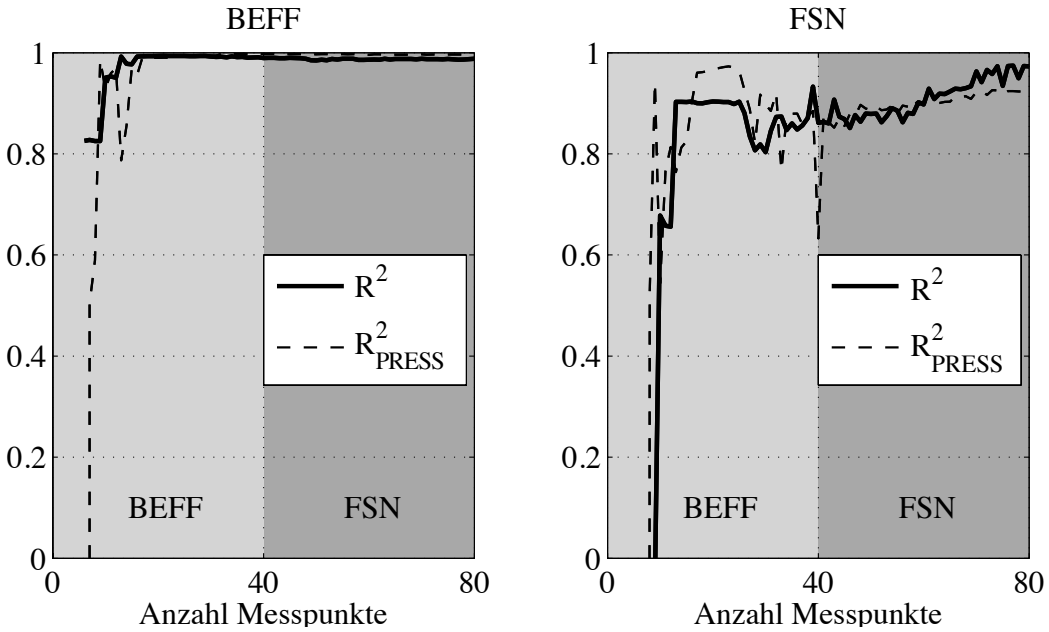

Bild 6.16: Versuchsplanung mit HILOMOTDOE. Die ersten 40 Messungen waren bezüglich der Zielgröße BEFF. Die Messungen 41 bis 80 waren bezüglich FSN. Interessant ist die Verbesserung des FSN-Modells nach Umschalten des Optimierungskriteriums der Versuchsplanung (rechts ab Messung 41).

[92]. Die betrachteten Ausgangsgrößen sind der effektive spezifische Kraftstoffverbrauch (BEFF in [g/kWh]) und die Schwärzungszahl FSN (*Filter Smoke Number*).

Zur Verifikation wurden für jeden Betriebspunkt 160 raumfüllend geplante Messungen durchgeführt und in einem Testdatensatz zusammengefasst. Alle in diesem Abschnitt gezeigten R^2-Fehlerverläufe wurden auf diesen separaten Testdaten berechnet.

Zum Vergleich mit dem HILOMOTDOE-Verfahren fand eine klassische D-optimale Versuchsplanung auf Basis eines Polynoms dritten Grades statt. Für ein Polynom dritten Grades ergeben sich bei den fünf Eingangsgrößen insgesamt 56 Parameter. Üblicherweise werden zur statistischen Absicherung ca. 30% bis 50% mehr Messungen durchgeführt als Parameter vorhanden sind, sogenannte *Overhead-Messungen*. Daher wurden insgesamt 80 Messpunkte je Betriebspunkt geplant. Die Versuchsplanung und die Modellbildung erfolgten nach der gegenwärtigen Standardvorgehensweise in der Motorenapplikation. In der nachträglichen Offline-Phase wurde die Modellbildung mit Hilfe des Softwarewerkzeugs CAMEO durchgeführt. Zur Schätzung der Polynommodelle wurden die Ausgangsgrößen mit einer Box-Cox-Transformation nichtlinear transformiert und außerdem eine schrittweise Regression zur Auswahl der wichtigsten Regressoren durchgeführt. Die resultierenden Modelle dienten zur Bewertung der Ergebnisse mit HILOMOT.

Im Normalfall stehen keine Validierungsdaten zur Verfügung und daher kann auch das Bestimmtheitsmaß R^2 nicht bezüglich separater Daten berechnet werden. Während der Prüfstandsmessungen ist es jedoch mit HILOMOTDOE möglich, die Modellqualität mit dem prädizierten Bestimmtheitsmaß R^2_{PRESS} auf Basis einer LOO-Kreuzvalidierung online zu beurteilen und den Versuch zu beenden, wenn die Modellgüte für die Kalibrierung ausreichend ist. Eine ausführliche Beschreibung des Fehlermaßes befindet sich in den

Kapiteln 3.5.2 und 5.4. Die Bilder 6.15 und 6.16 zeigen die gute Übereinstimmung der Bestimmtheitsmaße R^2 auf Testdaten und R^2_{PRESS}, welches lediglich mit den online verfügbaren Trainingsdaten berechnet wurde. Diese Online-Beurteilung des Modells ist beim D-optimalen Versuchsplan *nicht* möglich, da hier die Messpunkte *vorab* zu platzieren sind.

Bild 6.15 zeigt die Fehlerverläufe bezüglich des spezifischen Verbrauchs BEFF und der Schwärzungszahl FSN. Die insgesamt 80 Messpunkte wurden mit HILOMOTDOE geplant, wobei bezüglich der Ausgangsgröße BEFF aktiv gelernt wurde. Die FSN-Fehlerwerte ergeben sich durch die Modellierung dieser Größe mit denselben Daten. Man erkennt, dass sich die Modellgüte mit zunehmender Messpunktanzahl verbessert und sich ab ca. 40 Messungen keine signifikante Verbesserung des Fehlers ergibt. Das bedeutet, rund die Hälfte der Messungen sind redundant.

Die Versuchsplanung mit HILOMOTDOE kann immer nur bezüglich einer Zielgröße erfolgen. Da jedoch in der Praxis generell die Messung mehrerer Ausgangsgrößen gleichzeitig stattfindet, wurde im Rahmen dieser Untersuchung ein sequentieller Ansatz eingesetzt, d.h. die Vermessung bezüglich BEFF und FSN erfolgte nacheinander. Die ersten 40 Messungen waren zur Verbesserung der Zielgröße BEFF geplant, die Messungen 41 bis 80 in Bezug auf den Ausgang FSN. Bild 6.16 zeigt dieses Szenario. Man erkennt an diesem Beispiel sehr gut, wie sich das FSN-Modell nach Umschalten der Versuchsplanung auf FSN verbessert. Dies ist ein Beleg dafür, dass sich die aktive Versuchsplanung an den jeweiligen Prozess anpasst.

Zur Handhabung mehrerer Ausgangsgrößen bei der Versuchsplanung sind verschiedene Ansätze denkbar. Zum einen kann, wie beispielsweise in [96] vorgeschlagen, sequentiell vorgegangen werden. Nach jeder Einzelmessung wird einfach die Versuchsplanung auf eine andere Zielgröße umgeschaltet. Ein zweiter Ansatz wäre, nach jedem Query für alle Ausgänge ein Modell zu trainieren und eine gewichtete Präferenz der einzelnen Ausgangsgrößen bei der Versuchsplanung vorzunehmen. Hierbei muss ein Kompromiss bezüglich der verschiedenen Zielgrößen gefunden werden. Letztlich führen bei vielen Zielgrößen alle Strategien dazu, dass die aktive Versuchsplanung mit zunehmender Anzahl an Zielgrößen immer ineffektiver wird und in einer raumfüllenden Versuchsplanung resultiert.

Die durchgeführten Messung wurden dazu genutzt, um den in Kapitel 5.4 vorgestellten Ansatz der automatisierten Versuchsbeendigung in der Praxis umzusetzen. Dazu wurde die Methode allerdings zugunsten einer einfacheren Handhabung für das Prüfstandspersonal leicht angepasst. Das Vorgehen ist nun folgendermaßen:

1. Filterung des R^2_{PRESS}-Verlaufs mit einem Butterworth-Filter 1. Ordnung. Grenzfrequenz $f_g = 1/20 f_0$.
2. Rechteckfensterung der letzten R^2_{PRESS}-Werte mit vorgegebener Fensterbreite L.
3. Berechnung der Standardabweichung σ_{Fenster} der gefilterten R^2_{PRESS}-Werte im Fenster.
4. Wenn σ_{Fenster} kleiner ist als eine vorgegebene Schwelle ε, dann Stoppen der Messung.

Die zwei einzustellenden Parameter sind die Fensterbreite L und die Abbruchschwelle ε.

Zur Bewertung der Einstellungsparameter wurden zwei gegensätzliche Kriterien definiert:

1. *Performance-Verlust*: Um wie viel verschlechtert sich das Modell, wenn die Messung vor Erreichen der 80 Messpunkte beendet wird?

$$\text{Performance-Verlust} = 1 - \frac{R^2_{\text{Stop}}}{R^2_{\text{Ende}}}\,, \tag{6.2}$$

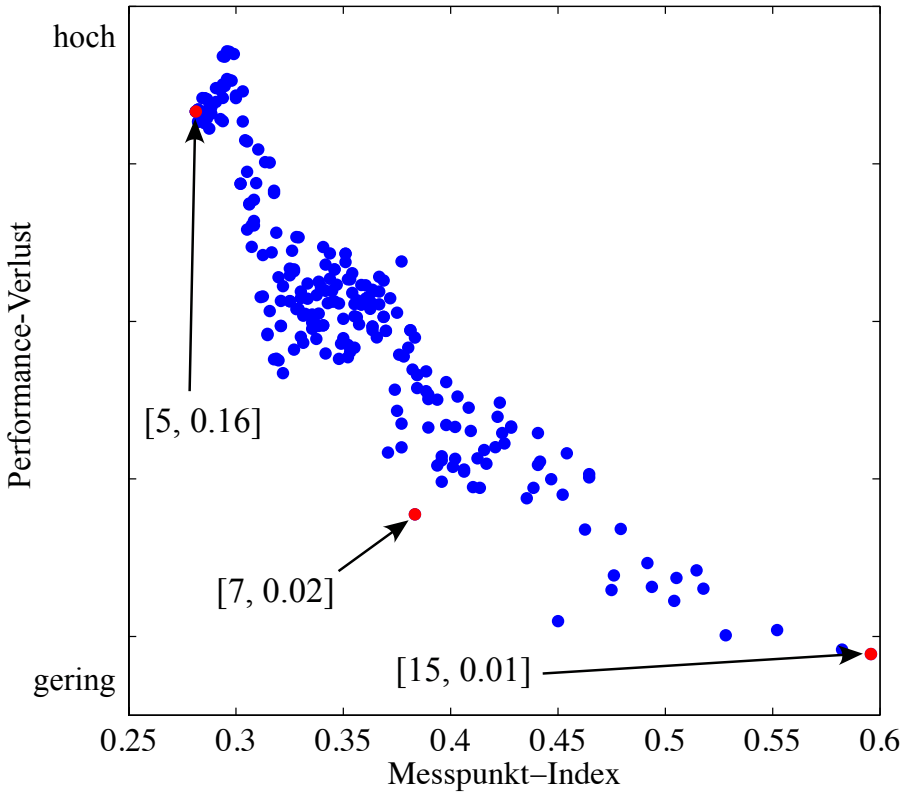

Bild 6.17: Performance-Verlust und Messpunkt-Index für verschiedene Filterrealisierungen, d.h. mit unterschiedlichen Fensterbreiten L und Abbruchschwellen ε, jeweils gemittelt über alle Versuchsläufe. Für drei Varianten sind beispielhaft die Parameter $[L, \varepsilon]$ gezeigt.

wobei R^2_{Stop} der Modellfehler bei vorzeitiger Versuchsbeendigung ist und R^2_{Ende} das Bestimmtheitsmaß mit allen Messungen.

2. *Messpunkt-Index*: Wie viele Messungen wurden anteilig bei vorzeitiger Versuchsbeendigung verwendet?

$$\text{Messpunkt-Index} = \frac{N_{\text{Stop}}}{N_{\text{Ende}}}\,, \tag{6.3}$$

wobei N_{Stop} die Anzahl der Messpunkte bei vorzeitiger Versuchsbeendigung ist und N_{Ende} der Anzahl aller Messungen.

Zur Optimierung der Fensterbreite L und des Schwellwerts ε wurde eine Gittersuche durchgeführt, bei der die Fensterbreite diskret zwischen 5 und 15 eingestellt und die Abbruchschwelle zwischen 0.01 in 0.01-Schritten bis zur Schwelle 0.25 variiert wurde. Für jede Realisierung wurden der Performance-Verlust und der Messpunkt-Index über alle Messläufe gemittelt. Da es sich hierbei um die Optimierung zweier Kriterien handelt, ergibt sich nicht ein einzelnes Optimum, sondern eine Lösungsmenge, aus der je nach Präferenz eine Lösung ausgesucht werden kann. In Bild 6.17 sind die Realisierungen dargestellt. Je breiter das Fenster und je niedriger die Schwelle desto kleiner wird der Performance-Verlust[5], je-

[5]Diese Ergebnisse sind in Zusammenarbeit mit der Daimler AG entstanden. Aus Geheimhaltungsgründen wird der Wertebereich für den Performance-Verlust nicht angegeben.

doch ergibt sich auch ein höherer Messpunkt-Index. Man kann sich z.B. drei Einstellungen nach verschiedenen Zielsetzungen aussuchen:

- Offensive Einstellung: Mit $L = 5$ und $\varepsilon = 0.16$ werden die Messungen im Schnitt bereits bei 28% gesetzten Gesamtpunktzahl beendet. Der Performance-Verlust ist dann allerdings im Vergleich zu anderen Einstellungen hoch.
- Konservative Einstellung: Mit $L = 15$ und $\varepsilon = 0.01$ ergibt sich der geringste durchschnittliche Performance-Verlust, jedoch auch die höchste mittlere Messpunktanzahl von 60%.
- Ausgewogene Einstellung: Ein guter Kompromiss mit einem mittleren Performance-Verlust und einem Messpunkt-Index von 38% wird mit der Einstellung $L = 7$ und $\varepsilon = 0.02$ erzielt.

Die resultierenden HILOMOT-Modelle lieferten in der ausgewogenen Einstellung zum Polynomansatz vergleichbare Modellgüten, der jedoch mit allen 80 Messpunkten ausgewertet wurde.

6.3.3 Globale Online-Versuchsplanung

Die zweite Anwendung für HILOMOTDOE ist eine globale Versuchsplanung. Die Eingangsgrößen sind in diesem Fall:

- Drehzahl,
- Motorlast,
- Abgasrückführrate (AGR),
- Raildruck,
- 50%-Umsatzpunkt des Heizverlaufs (H_{50}-Umsatzpunkt),
- Menge der ersten Voreinspritzung und
- Menge der zweiten Voreinspritzung.

Ziel war die Modellierung des effektiven spezifischen Kraftstoffverbrauchs BEFF.

Für den Vergleich wurden auch hier Referenzmodelle erstellt. Zum einen ein klassisches Polynommodell vierten Grades, zum anderen ein neuronales Netz innerhalb der Software CAMEO. Das sogenannte *Fast Neural Network* (FNN) entspricht der LOLIMOT-Methode, siehe z.B. [42]. Dazu fand die Versuchsplanung für insgesamt 40 Betriebspunkte raumfüllend mit einer S-optimalen Messpunkteplatzierung statt. Hierbei ist das Ziel, den Nearest-Neighbor-Abstand aller Punkte zu maximieren, siehe z.B. [90] oder [68]. Für das globale Motorverhalten war die Annahme, dass es sich mit einem Polynommodell vierten oder fünften Grades nachbilden lässt, was für den siebendimensionalen Eingangsraum die Planung einer Gesamtpunktemenge von 800 Messwerten motivierte. Insbesondere für globale Versuchspläne mit einem großen Drehzahl-Last-Bereich ist es schwierig, die Gesamtpunktzahl festzulegen, weil a priori vom Anwender keine genauen Annahmen über den nichtlinearen Prozess getroffen werden können [93]. Generell ist daher der Prüfstandsingenieur oft dazu geneigt, eher zu viele Messungen zu planen, als dies vielleicht mit genauem Prozesswissen nötig wäre. Für einen fairen Vergleich mit den Referenzmodellen wurde eine sehr große Menge von insgesamt 900 raumfüllend verteilten Testdatenpunkten vermessen.

Die Online-Versuchsplanung mit HILOMOTDOE fand mit der automatisierten Versuchsbeendigung statt, für die bei den vorangegangenen lokalen Betriebspunktmessungen die Para-

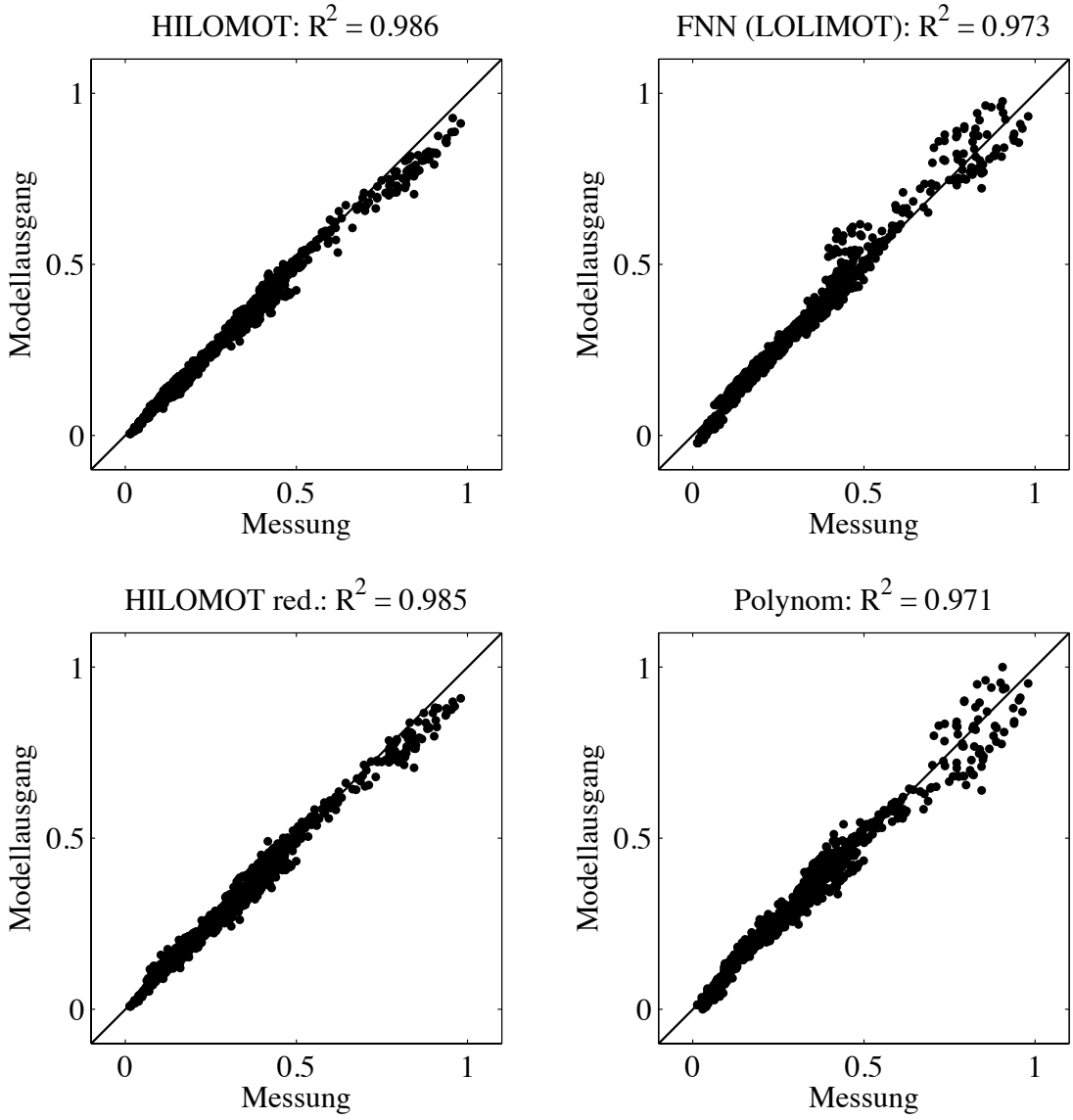

Bild 6.18: Korrelation zwischen Modellausgang und gemessenen Testdaten für den effektiven spezifischen Kraftstoffverbrauch BEFF für vier verschiedene Modelle.

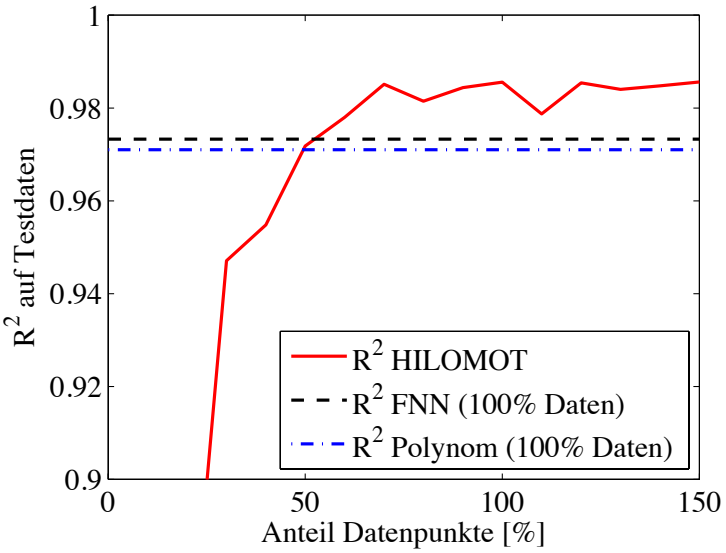

Bild 6.19: Bestimmtheitsmaß R^2 in Abhängigkeit der Datenmenge. 100% entspricht der Punktanzahl des Standardverfahrens von 800 Messwerten.

meter mit der ausgewogenen Einstellung resultierten. Für den Vergleich hat HILOMOTDOE die gleichen Betriebspunkte vermessen wie sie zuvor bei den Referenzmessungen geplant waren, d.h. die automatisierte Versuchsbeendigung galt jeweils für einen Betriebspunkt, sofern die prädizierten Modellfehler keine Modellverbesserung mit neuen Messungen versprachen. Dies führte für alle Betriebspunkte auf insgesamt 1200 Messpunkte mit HILOMOTDOE.

Bild 6.18 zeigt die Ergebnisse anhand der Korrelation zwischen Messwerten und Modellausgangswerten für die untersuchten Modelle. Die Referenzmodelle entsprechen dem konventionellen Vorgehen mit der Software CAMEO und wurden mit dem S-optimal vermessenen Datensatz trainiert. Das Training mit HILOMOT fand entsprechend mit den HILOMOTDOE-Messungen statt. Um einen fairen Vergleich zu gewährleisten, wurden die Messungen von HILOMOTDOE für alle Betriebspunkte prozentual reduziert, um auf die gleiche Punktmenge von 800 zu kommen wie beim S-optimalen Versuchsplan. Die Fehlerberechnung erfolgte für alle Modelle mit dem gleichen Testdatensatz (900 Messwerte).

Oben links in Bild 6.18 ist das Ergebnis von HILOMOT mit allen 1200 Messwerten fürs Training gezeigt, unten links das auf 800 Punkte reduzierte HILOMOT-Modell. Rechts sind die Ergebnisse mit FNN und Polynommodell zu sehen. Sowohl für das HILOMOT-Modell mit allen 1200 Datenpunkten als auch für das reduzierte HILOMOT-Modell ergeben sich bessere Fehlerwerte, als dies beim Polynommodell oder FNN der Fall ist.

Bild 6.19 zeigt zudem, dass für das HILOMOT-Modell sogar deutlich weniger als 800 Messwerte ausgereicht hätten, um im Vergleich eine bessere Güte zu erzielen als mit FNN oder dem Polynommodell. Die Modellgüte verbessert sich zwar mit zunehmender Punktanzahl zwischen 800 und 1200 Messwerten noch ein wenig, allerdings sollte man aufgrund hoher Prüfstandskosten den Messaufwand so gering wie möglich halten. Wie für die lokale Versuchsplanung gezeigt, kann das Verfahren zur automatisierten Versuchsbeendigung durchaus offensiver eingestellt werden, was in diesem Anwendungsfall zielführend wäre. Die gezeigten Ergebnisse legen nahe, dass im Vergleich zum konventionellen Vorgehen bei der Versuchsplanung bis zu 50% der Messungen mit HILOMOTDOE eingespart werden können.

Als Ausblick wäre interessant, am Prüfstandssystem auch Drehzahl und Last frei durch die Online-Versuchsplanung verstellen zu lassen und nicht vorab wie im Vergleich vorzugeben.

6.4 Zusammenfassung

Es wurden drei Anwendungsbeispiele für die Umsetzung sowohl der Modellbildungsmethode HILOMOT als auch der aktiven Versuchsplanungsmethode HILOMOTDOE in der Praxis vorgestellt. An dieser Stelle sei außerdem erwähnt, dass in [58] eine weitere, hier nicht aufgeführte Anwendung zur Optimierung der Stellgrößen einer Tablettenpresse vorgestellt wurde.

Die erste Anwendung betraf die aktive Versuchsplanung im Zusammenhang eines Structural Health Monitoring-Systems. Hierbei ging es um die Identifikation einer Schadenskarte zu einer Aluminiumstruktur, mit der eine Lokalisation des Schadens möglich ist. Ziel hierbei war es, mit möglichst wenigen Stützstellenauswertungen das Maximum der Schadenskarte zu lokalisieren. Mit dem HILOMOTDOE-Algorithmus wurden die Stützstellen aktiv in Abhängigkeit der Schadensintensitätswerte auf der untersuchten Struktur gelernt, um mit dem resultierenden Modell Aussagen zum Schadensort treffen zu können. Das Verfahren wurde zunächst anhand eines Simulationsmodells getestet. Hierbei hat sich gezeigt, dass mit HILOMOTDOE die Schadensorte durch eine dichtere Stützstellenplatzierung an Orten mit höheren Intensitätswerten zuverlässig identifiziert wurden. Außerdem zeigte sich die Adaptivität des HILOMOTDOE-Verfahrens. Ändert sich der Schadensort, werden dementsprechend auch die Stützstellen an dem veränderten Schadensort platziert. Neben der Simulation wurde die Methode mit Hilfe eines experimentellen Versuchsaufbaus auch anhand praktischer Messdaten untersucht. Die Ergebnisse zum durchgeführten Experiment decken sich mit den Ergebnissen aus der vorangegangenen Simulation.

Die zweite Anwendung bezog sich auf die Modellbildung. Das HILOMOT-Verfahren wurde zur Modellierung des Datensatzes eines zeitgemäßen, modernen Dieselmotors eingesetzt und mit den in der Automobilindustrie üblichen Verfahren der Polynommodelle und der Gaußprozessmodelle verglichen. Bei dem elfdimensionalen Prozess zeigte sich, dass HILOMOT-Modelle in der Standardeinstellung eine bessere Modellgüte als der Polynomansatz liefern. Jedoch zeigte sich auch der Vorteil in der Flexibilität der Gaußprozessmodelle. Diese lieferten im Vergleich leicht bessere oder ähnliche Ergebnisse wie die HILOMOT-Modelle. Allerdings hat sich auch gezeigt, dass im Fall der Gaußprozessmodelle eine Neigung zum Overfitting vorherrscht und ein größerer Rechenaufwand nötig ist. Insgesamt bietet der HILOMOT-Ansatz vielversprechende Vorteile für den industriellen Einsatz. Der Ansatz wird daher bei der Firma IAV weiter verfolgt und es ist geplant, das HILOMOT-Verfahren in zukünftigen Softwarewerkzeugen von IAV zu implementieren.

Die dritte Anwendung war die adaptive Online-Versuchsplanung zur automatisierten Kalibrierung von Verbrennungsmotoren. Der HILOMOTDOE-Algorithmus wurde für den industriellen Einsatz bei der Daimler AG in ein Prüfstandssystem integriert und zur aktiven Vermessung eines Dieselmotors eingesetzt. Es wurden zunächst Experimente zur lokalen Versuchsplanung einzelner Betriebspunkte durchgeführt, die auch der Parametrierung der automatisierten Versuchsbeendigung dienten. Danach erfolgte die globale Versuchsplanung für die gesamte Drehzahl-Last-Ebene mit HILOMOTDOE. Im Vergleich zur konventionellen

Vorgehensweise mit raumfüllender Versuchsplanung ergibt sich mit dem neuen HILOMOT-DoE-Verfahren deutliches Einsparungspotential, was zu Zeit- und Kosteneinsparungen bei der zukünftigen Applikation von Verbrennungsmotoren führt.

7 Zusammenfassung und Ausblick

Die vorliegende Arbeit beschäftigt sich mit der experimentellen Modellbildung nichtlinearer Prozesse durch lokal polynomiale Neuro-Fuzzy-Modelle. Das Ziel war die Entwicklung eines hierarchischen, achsenschrägen Partitionierungsverfahrens. Darüber hinaus war die Fragestellung, inwieweit sich lineare statistische Selektionsverfahren mit nichtlinearen Strukturoptimierungsverfahren für lokale Modellnetze kombinieren lassen. Außerdem sollte ein aktives Lernverfahren entwickelt werden, welches die Vorteile der lokalen Modellnetze für eine aktive Versuchsplanung nichtlinearer Prozesse voll ausschöpft.

Lineare Optimierung lokaler Modellparameter

Zunächst lag der Fokus auf der linearen Optimierung lokaler Modellparameter. Neben der impliziten Regularisierung durch eine lokale Schätzung werden in dieser Arbeit neue Ansätze zu einer expliziten Regularisierung für hierarchische lokale Modellnetze vorgestellt. Zum einen handelt es sich um eine Ridge Regression zur Regularisierung einer globalen Schätzung. Basierend auf einer Singulärwertzerlegung kann die Stärke der Regularisierung effizient bezüglich des Kreuzvalidierungsfehlers optimiert werden. Zum anderen ist die approximative Tikhonov-Regularisierung zweiter Ordnung ein Verfahren, bei dem die Krümmung des Modellausgangs bestraft wird. Ein aus der Literatur bekanntes approximatives Maß wurde in dieser Arbeit eigens für die Verwendung bei hierarchischen, achsenschrägen Partitionierungen entwickelt, bei denen die Schwierigkeit besteht, dass die Geometrie der Gültigkeitsfunktionen mathematisch nur äußerst schwierig zu beschreiben ist.

Wichtige Voraussetzung für ein gutes Modell ist das Auffinden eines guten Bias-Varianz-Kompromisses. Stehen keine separaten Validierungsdaten zur Verfügung, sind statistische Kriterien zur Einstellung der Modellkomplexität einzusetzen. Hierfür ist die Leave-One-Out-Kreuzvalidierung (LOOCV) ein wichtiger Spezialfall. Es wurde hergeleitet und gezeigt, wie man unter Ausnutzung einer QR-Zerlegung die LOOCV numerisch günstig auf die lokale Schätzung anwenden kann. Obwohl der nichtlineare Einfluss der Partitionierungsparameter dabei vernachlässigt wird, zeigt das Verfahren eine sehr gute Übereinstimmung mit einer vollständig durchgeführten Kreuzvalidierung, welche um Größenordnungen aufwändiger und damit langsamer ist. Neben der datengetriebenen Validierung bietet sich der Einsatz von Informationskriterien an. Das korrigierte AIC-Kriterium hat sich hier als sehr nützlich erwiesen und wird daher in den hier vorgestellten Algorithmen zum Training standardmäßig verwendet.

Im Falle einer lokal gewichteten Least Squares-Schätzung unterscheidet sich die Strukturselektion der wichtigsten Regressoren vom ungewichteten Standardvorgehen bei z.B. Polynommodellen. Zur Umsetzung bei lokalen Modellnetzen mussten einige Anpassungen vorgenommen werden. Die numerisch effiziente Implementierung eines sequentiellen t-Tests ist durch die Verwendung einer QR-Faktorisierung gewährleistet. Das Verfahren eignet sich sowohl für die Vorwärtsselektion als auch für die Rückwärtseliminierung.

Nichtlineare Strukturoptimierung lokaler Modellnetze

Um den Aufwand durch nichtlineare Optimierungsverfahren so gering wie möglich und das Training lokaler Modellnetze möglichst effizient zu gestalten, wurden in dieser Arbeit heuristische Verfahren vorgestellt und untersucht, die in der Lage sind, eine Optimierung der Modellstruktur automatisiert durchzuführen. Nach einer Übersicht zu Partitionierungsverfahren, die aus der Literatur bekannt sind, wurden ausgehend vom LOLIMOT-Algorithmus Methoden entwickelt, die eine deutliche Verbesserung der Modellflexibilität durch den Einsatz achsenschräger Partitionierungen erlauben.

Das neue Verfahren SUHICLUST („Supervised Hierarchical Clustering“) kombiniert dazu die Vorteile der Baumkonstruktion mit der Flexibilität von Produktraum-Clusteringverfahren. Diese Modelle verwenden eine parallele Modellstruktur. Ein weiteres Verfahren basiert auf einer hierarchischen Struktur, bei der sich viele Vorteile für ein achsenschräges Partitionierungsverfahren herauskristallisierten. Der resultierende HILOMOT-Algorithmus („Hierarchical Local Model Tree“) basiert auf der Idee von Hinging Hyperplane Tree-Modellen (HHT), die in [150] vorgestellt wurden und verallgemeinerte Hinge-Funktionen verwenden, die unterschiedliche Eingangsräume für die Partitionierung und die lokalen Modelle erlauben.

Durch den Einsatz linearer Strukturselektionsverfahren eröffnen sich neue Perspektiven für das Training lokaler Modellnetze. Die hier vorgestellte Methode der Strukturabwägung erlaubt es, während des Trainings mit HILOMOT nicht nur die wichtigsten Regressoren jedes Teilmodells zu selektieren, sondern darüber hinaus die Polynomkomplexität der lokalen Modelle individuell an verschiedene Regionen des Prozesses anzupassen. Dies erlaubt die Reduktion des Modells auf wenige Teilgebiete, die aber wiederum bezüglich der Regressoren so auf den Prozess abgestimmt sind, dass man sie zur lokalen Beschreibung des Prozessverhaltens sinnvoll nutzen kann.

Der vorgestellte HILOMOT-Algorithmus wurde zur Modellierung des Verbrauchs und von Emissionskenngrößen anhand der Messdaten eines realen Dieselmotors verifiziert.

Versuchsplanung mit lokalen Modellnetzen

Wesentlicher Gegenstand dieser Arbeit ist die passive und aktive Messdatengewinnung mit HILOMOT-Modellen. Diese Modellklasse bringt Vorteile mit sich, die für den Einsatz als aktives Lernverfahren zwingend erforderlich sind:

- Schnelles Training,
- deterministische Modelle,
- eine hohe Modellgüte,
- Robustheit für den Online-Einsatz und
- eine automatische Komplexitätsauswahl.

Der Fokus bei den vorgestellten Versuchsplanungsmethoden liegt auf der Minimierung des Modellfehlers. Hierfür lässt sich die Struktur der lokalen Modellnetze explizit ausnutzen: Der Prozess wird vom Modellbildungsalgorithmus in lokale Bereiche unterteilt, die unterschiedliches Prozessverhalten aufweisen. Es bietet sich daher an, das Wissen über die Gültigkeitsbereiche lokaler Modelle explizit für die Vergabe neuer Messpunkte zu nutzen. Diese Arbeit hat dazu drei neue Algorithmen vorgestellt, denen jeweils ein lokal raumfüllendes Versuchsplanungsverfahren zugrunde liegt. Die Methode ermöglicht durch die

Verwendung von Kandidatenpunkten eine einfache Einbindung vorab bekannter Versuchsraumgrenzen. Die aktiven Lernverfahren sind zudem in der Lage, automatisch die Messung zu beenden, wenn sich keine signifikante Modellverbesserung mehr durch neue Messungen ergibt. Darüber hinaus erwies sich der Einsatz von Modellkomitees, z.B. durch Erzeugung von Modellvarianten mittels Bootstrapping oder durch gewichtete Mittelung heterogener Modelle, als vielversprechend und wurde daher kurz vorgestellt.

Die neu entwickelte Methode HILOMOTDOE („Hierarchical Local Model Tree for Design of Experiments“) wurde zum aktiven Lernen zweier praktischer Anwendungen eingesetzt: Die Identifikation einer Schadenskarte im Rahmen eines Structural Health Monitoring Systems und die aktive Vermessung eines Dieselmotors am Motorenprüfstand.

Ausblick

Als Ausblick für zukünftige Forschung ergeben sich die folgenden Themenbereiche:

1. *Verbesserung von* HILOMOT: Die Verwendung einer analytischen Gradientenberechnung bezüglich der nichtlinearen Partitionierungsparameter verspricht im Gegensatz zur numerischen Berechnung das Potential, das Training mit HILOMOT insbesondere bei hoher Dimensionalität zu verbessern, weil dann die Auswertung eines numerisch approximierten Gradienten immer ineffizienter wird, siehe z.B. [36]. Außerdem kann das Einbringen der Nebenbedingung, dass z.B. immer genügend Messdaten innerhalb des lokalen Modellgebiets zur Schätzung vorliegen, die Effizienz steigern. Ferner ist zu überlegen, eine Ridge Regression für jedes lokale Modell separat durchzuführen. Dies ermöglicht die individuelle Anpassung der Regularisierung an die lokalen Trainingsdaten.
2. *Verbesserung der Versuchsplanung*: Für die praktische Umsetzung von HILOMOTDOE ist zu überlegen, die Versuchsplanung bezüglich mehrerer Ausgangsgrößen gleichzeitig durchzuführen. Darüber hinaus kann anstelle der Verwendung von diskreten Kandidatenpunkten eine kontinuierliche Query-Optimierung Verbesserungspotential liefern. Diese Methode funktioniert auch für sehr viele Eingangsgrößen und bietet eine höhere Genauigkeit der Query-Platzierung, ist jedoch auch aufwändiger zu berechnen. Bisher findet eine Klassifikation der lokalen Modelle nach dem lokalen Fehler statt. Anschließend erfolgt die Query-Platzierung raumfüllend im lokalen Gebiet. Anstelle der diskreten Vorgehensweise bietet eine kontinuierliche Überblendung der zwei Query-Kriterien eine höhere Flexibilität des Ansatzes.
3. *Sensitivitätsanalyse bezüglich wichtiger Eingangsgrößen*: Sowohl für die Modellbildung als auch für die Versuchsplanung bietet die Selektion der wichtigsten Eingänge bezüglich ihres Einflusses auf nichtlineare Zusammenhänge großes Potential. Sinnvollerweise kann der HILOMOT-Algorithmus zur Identifikation dieser Zusammenhänge für die Selektion genutzt werden.
4. *Dynamische Modellbildung*: Zur Verwendung von HILOMOT für dynamische Modelle ist die Optimierung des Ausgangsfehlers (NOE-Konfiguration) für alle lokalen Modelle gleichzeitig vielversprechend. Dazu kann eine Regularisierungstechnik verwendet werden, die trotz global durchgeführter Schätzung in einer lokalen Schätzung resultiert. Außerdem ist die Verwendung von lokalen Laguerre-Filtern sehr interessant, da diese inhärent stabil sind. Die Optimierung des Ausgangsfehlers kann hierfür mit linearen Methoden erfolgen, was als großer Vorteil zu bewerten ist.

A Support Vector Regression

Im Folgenden werden *Support Vector Machines* (SVM) im Rahmen der Funktionsapproximation hinsichtlich ihrer Eignung für Regressionsaufgaben untersucht. Eine Betrachtung der Eigenschaften von SVM für *Klassifikationsaufgaben* findet an dieser Stelle allerdings nicht statt. Studiert man die Literatur zum Thema Support Vector Regression (SVR), so sind folgende Punkte bemerkenswert:

- [134] beschreibt die Modifikation der ε-SVR zur ν-SVR, welche eine einfachere Parametrierung zur Folge haben soll. Jedoch wird auch bei ν-SVR-Anwendung vorgeschlagen, eine Vielzahl von Testläufen durchzuführen, um diesen Parameter letztendlich einzustellen.
- In [25] wird die Verwandtschaft von SVR zu Gauß'schen Prozessmodellen gezeigt. An keiner Stelle wird erwähnt, wie die SVR-Parameter in der Anwendung vom Benutzer auszuwählen sind.
- [61] liefert Aussagen zur SVR-Verlustfunktion im Vergleich zu Robust Regression. Weiterhin wird auf die Wahl der Parameter ε und λ, die zur Berechnung der Verlustfunktion nötig sind, eingegangen. Auch hier soll ε durch Ausprobieren ausgewählt und der Regularisierungsparameter λ durch Kreuzvalidierung ermittelt werden.
- [16] sieht keinen echten Fortschritt im Vergleich zu konventionellen, robusten Regressionsverfahren, Support Vector Machines auch zur Regression zu verwenden: *„The relative advantage in practice of support vector machine regression compared to any of several forms of robust regression is not clear."*

Neben der Literaturrecherche sollen die nachfolgenden numerischen Experimente zur empirischen Analyse der Support Vector Regression genutzt werden.

Bei Anwendung von Support Vector Regression wird die folgende ε-insensitive Verlustfunktion minimiert:

$$J\left(\underline{w}, \zeta, \zeta^*\right) = \frac{1}{2}\|\underline{w}\|^2 + C\sum_{i=1}^{N}\left(\zeta_i + \zeta_i^*\right) . \tag{A.1}$$

Dabei werden die sogenannten „Slack-Variables" ζ_i und ζ_i^* eingeführt, um nur diejenigen Datenpunkte zu bestrafen, welche außerhalb eines ε-insensitiven Gebiets liegen. Bild A.1 veranschaulicht die Vorgehensweise. C_{SVM} kann als Regularisierungsparameter aufgefasst werden. Für $C_{\mathrm{SVM}} \to 0$ tendiert das Modell zu global konstantem Verhalten, da in diesem Fall ausschließlich der Parametervektor $\underline{w}$ minimiert wird. Wenn C_{SVM} gegen ∞ läuft, wird ausschließlich der Fehler bezüglich der Datenpunkte außerhalb der ε-Röhre minimiert.

Für die zu Grunde liegenden Untersuchungen bot sich die LIBSVM-Toolbox [23], welche inklusive MATLAB-Interface frei im Internet verfügbar ist, zur Verwendung an. [76] empfiehlt u.a. diese Toolbox für Regressionsaufgaben. Der standardmäßig eingestellte Wert $\varepsilon = 0.1$ blieb beibehalten und RBF-Kernelfunktionen kamen zum Einsatz.

Zunächst ist die Frage interessant, wie die Parameter C_{SVM} und σ (Standardabweichung der RBF-Kernels) überhaupt einzustellen sind und herauszufinden, wie stark die Ergebnisse

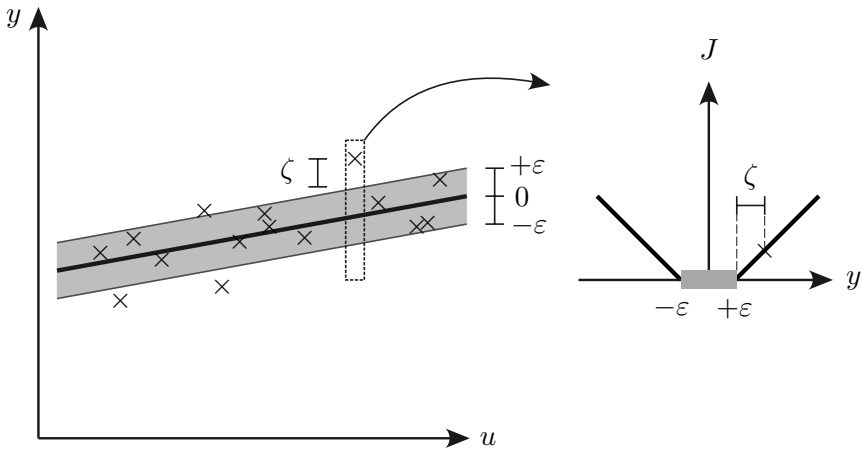

Bild A.1: Darstellung der ε-insensitiven Verlustfunktion [133].

für unterschiedliche Werte dabei variieren. Die Modellbildung erfolgte dazu für folgende eindimensionale Beispielfunktion:

$$y = 3\cos(5u)u^3 + d\,, \tag{A.2}$$

wobei y der Prozessausgang, u die Eingangsgröße und d normalverteiltes Rauschen mit der Rauschvarianz $\sigma_n^2 = 0.1$ ist. Für Training und Validierung wurden jeweils 30 im Eingangsraum gleichverteilte Datenpunkte generiert. Zusätzlich waren die Daten auf das Intervall $[0, 1]$ normiert. Das Training erfolgte mit verrauschten Daten, die Validierung mit unverrauschten Daten.

Für die Gauß-Kernels erschien zum einen die Standardabweichung von 1/2 zu groß, da in diesem Fall die Lokalität der Zugehörigkeitsfunktionen verloren geht. Zum anderen würde die Wahl $\sigma = 1/32$ zu einem schlechten Interpolationsverhalten des Modells führen. Daher gestaltete sich die Variation der Parameter C_{SVM} und σ folgendermaßen: $C_{\mathrm{SVM}} = \{10^{-1}, 10^{1}, 10^{3}\}$ und $\sigma = \{\frac{1}{16}, \frac{1}{8}, \frac{1}{4}\}$.

Als vergleichbare konventionelle Modellbildungsmethode wurde ein RBF-Netz so konstruiert, dass zunächst auf jedem Datenpunkt ein Neuron platziert und anschließend mit Hilfe von Strukturselektion die wichtigsten RBF-Neuronen ausgesucht wurden. Die Implementierung in MATLAB war dank der eingebauten Funktion `stepwisefit` einfach möglich. Um einen Vergleich mit SVM durchzuführen, wurden die Gauß'schen Zugehörigkeitsfunktionen beim RBF-Netz mit der gleichen Standardabweichung parametriert. Weiterhin waren bei der Strukturselektion immer genausoviele Neuronen auszuwählen, wie zuvor Support Vektoren beim SVM-Modell generiert wurden. Bild A.2 zeigt exemplarisch die verrauschten Trainingsdaten im Vergleich zu den Modellen. J_{G} ist dabei der NRMSE-Testdatenfehler[1]. Bei dieser Parameterstudie ist aufgefallen, dass mit *allen* untersuchten Parameter-Einstellungen das RBF-Netz mit Strukturselektion kleinere Testdatenfehler lieferte als das SVM-Modell. Weiterhin zeigte das SVM-Modell starke Schwankungen in der Modellgüte mit Variation der Modellparameter, wohingegen das RBF-Modell sehr robust in der Wahl der Standardabweichung war.

[1]NRMSE = Normalized Root Mean Squared Error: $J_{\mathrm{G}} = \sqrt{\frac{\sum_{i=1}^{N}(\hat{y}(\underline{u}(i)) - y(i))^2}{\sum_{j=1}^{N}(\hat{y}(\underline{u}(i)) - \overline{y})^2}}$.

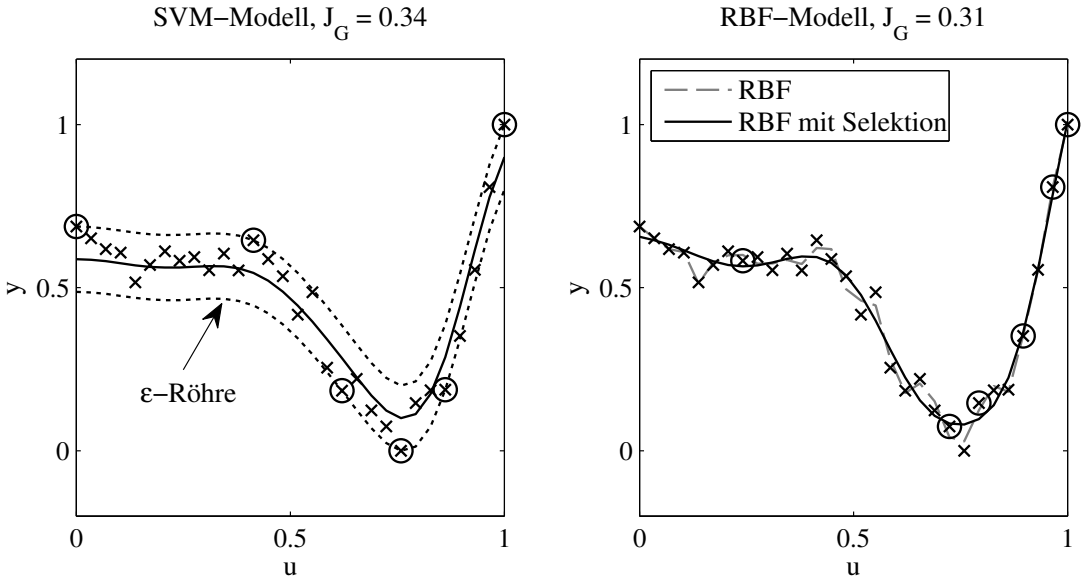

Bild A.2: Modellbildung mit SVM (links) und RBF mit Strukturselektion (rechts). Die Support Vektoren bzw. die ausgewählten RBF-Neuronen sind mit Kreisen markiert.

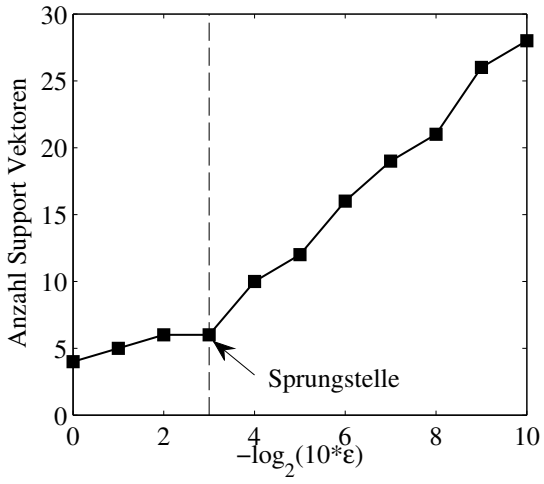

Bild A.3: Die Anzahl Support Vektoren steigt ab einem Schwellenwert für ε steil an. Die Wahl von ε gestaltet sich deshalb für den Benutzer schwierig.

Ein weiteres Experiment anhand der gleichen Beispielfunktion lieferte außerdem die Erkenntnis, dass die Wahl des ε-Parameters nur schwer getroffen werden kann. Betrachtet man den unverrauschten Prozess und variiert ε, so ergibt sich ab einem bestimmten Schwellenwert ein steiler Anstieg der Anzahl der Support Vektoren (siehe Bild A.3). Auch dies deutet auf die schlechte Parametrierbarkeit des SVM-Modells hin.

Aus der Literatur-Recherche zu SVM ging hervor, dass die Support Vector Regression insbesondere bei hochdimensionalen, dünn besetzten Eingangsräumen vorteilhaft sei. Daher ist auch der Vergleich von SVR und RBF in Abhängigkeit von der Anzahl der Eingänge

interessant. Der Vergleich wurde mit vier Beispielprozessen durchgeführt und wird exemplarisch mit dem folgenden Prozess veranschaulicht:

$$y = 1 - \prod_{i=1}^{p} \cos(u_i)\,. \tag{A.3}$$

wobei p die Anzahl der Eingangsgrößen ist. Der Vergleich geschah mit jeweils 3000 gleichverteilten Datenpunkten. Die Trainingsdaten waren mit normalverteiltem Rauschen mit den Varianzen $\sigma_n^2 = \{0, 0.05, 0.1\}$ beaufschlagt. Die Validierungsdaten waren unverrauscht.

Das SVM-Modell wurde wie folgt eingestellt: ε blieb auf dem Standardwert von 0.1. Eine zweistufige Gittersuche lieferte die Parameter C_{SVM} und σ, welche den kleinsten Testdaten-Fehler erzeugten. Für das RBF-Netz konnten anschließend die Standardabweichungen der Gauß-Funktionen und die Anzahl der Support Vektoren (entspricht der Anzahl zu selektierender Neuronen) übernommen werden. Die Ergebnisse dieser Untersuchung sind in Bild A.4 zusammengefasst.

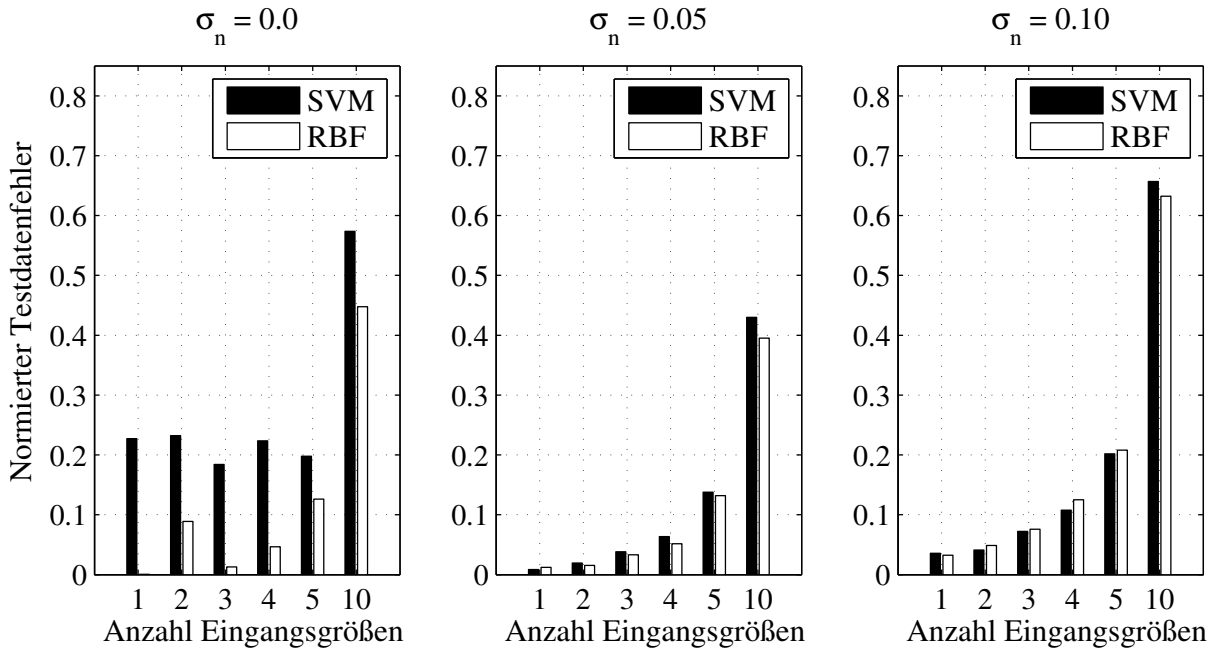

Bild A.4: Vergleichsstudie zu Support Vector Regression. Dargestellt sind jeweils die Testdaten-Fehler für 1D-, 2D-,...,5D- und 10D-Eingangsräume und für die drei verschiedenen Rauschvarianzen $\sigma_n^2 = \{0, 0.05, 0.1\}$.

Aufgrund der durchzuführenden Gittersuche gestaltete sich die Einstellung der Parameter der SVM kompliziert und zeitaufwändig. Die Rechenzeit der SVM blieb weitestgehend unbeeinflusst von der Eingangsraumdimensionalität. Dieser Vorteil relativierte sich jedoch durch die zeitintensive Gittersuche, welche beim RBF-Netz nicht nötig ist. Das RBF-Netz mit Strukturselektion lieferte bei allen Beispielen vergleichbare, oft sogar bessere Ergebnisse als das SVM-Modell, wobei die Rechenzeiten in den meisten Fällen kürzer ausfielen.

Als Fazit der durchgeführten Experimente ergibt sich, dass die Modellbildung mit Hilfe der SVM-Methode zwar durchaus ihre Berechtigung findet, allerdings keine herausragende Stellung im Bereich der Modellbildungswerkzeuge aufweisen kann.

B Anmerkungen zur Strukturselektion mit `stepwisefit` in MATLAB

Für die Anwendung von Strukturselektion bei klassischen Polynommodellen bietet MATLAB die Funktion `stepwisefit`. Hier übergibt man u.a. die Matrix $\underline{X}$ mit den potentiellen Regressoren und die Messwerte $\underline{y}$. Durch Anwendung eines sequentiellen t-Tests führt die Funktion dann eine schrittweise Regression durch. Ursprünglich war in dieser Arbeit die Überlegung, diese Funktion auch für die Strukturselektion der Regressoren von lokalen Modellen einzusetzen. Jedoch hat sich herausgestellt, dass `stepwisefit` für die Anwendung einer lokal gewichteten Schätzung ungeeignet ist.

Die wichtigsten Unterschiede von `stepwisefit` zu den in dieser Arbeit vorgestellten Methoden betreffen die folgenden Punkte:

1. Bei `stepwisefit` wird der Regressionsmatrix standardmäßig eine Spalte für den Gleichwert (engl.: Offset) des Modells hinzugefügt. Im Falle einer gewichteten LS Schätzung muss diese Gleichwert-Spalte, wie alle anderen Regressoren, mit der Wurzel der Gültigkeitswerte des lokalen Modells gewichtet werden. Dies ist in `stepwisefit` nicht vorgesehen.
2. Zur Schätzung der Rauschvarianz mit Hilfe des gegenwärtigen Modells benötigt man die Freiheitsgrade der Schätzung. Im ungewichteten Fall ergeben sich $N - n_x$ Freiheitsgrade, wenn N die Anzahl der Messwerte und n_x die Anzahl der Spalten von $\underline{X}$ sind. Die Freiheitsgrade bei der gewichteten Schätzung berechnen sich jedoch mit $\sum_{i=1}^{N} \underline{\Phi}_k(\underline{u}(i)) - n_{eff}$, wobei $\underline{\Phi}_k$ der Gültigkeit des betrachteten lokalen Modells mit Index k entspricht und n_{eff} der Anzahl der effektiven Parameter.
3. Die Kovarianzmatrix berechnet sich beim klassischen Polynommodell mit:

$$\mathrm{cov}\{\underline{\hat{\theta}}\} = \sigma^2 \left(\underline{X}^T \underline{X}\right)^{-1} . \tag{B.1}$$

   Bei der gewichteten Schätzung ergibt sich jedoch:

$$\mathrm{cov}\{\underline{\hat{\theta}}\} = \sigma^2 \left(\underline{X}^T \underline{Q}_k \underline{X}\right)^{-1} \underline{X}^T \underline{Q}_k^2 \underline{X} \left(\underline{X}^T \underline{Q}_k \underline{X}\right)^{-1} , \tag{B.2}$$

   mit $\underline{Q}_k = \mathrm{diag}(\underline{\Phi}_k)$. Dies hat einen entsprechend wichtigen Einfluss auf die Berechnung der t-Werte zur Regressorenauswahl.

C Einstellung des Interpolationsverhaltens

In [116] wurde gezeigt, dass im Rahmen einer lokalen Schätzung die Optimierung der Glattheit in den meisten Fällen nicht zielführend ist. Der Grund dafür ist, dass eine sehr kleine Glattheit, d.h. hartes Umschalten zwischen den LM, gleichzeitig den Fehler infolge der Vernachlässigung der Überlappungen zwischen den Gültigkeitsfunktionen minimiert. Auch wenn dies ein optimales Ergebnis liefert, sollten solche kleinen Überlappungen in der Praxis nicht realisiert werden, da normalerweise eher ein glattes Modellverhalten erwünscht ist. Im Gegensatz dazu tendiert die Optimierung der Glattheit bei Anwendung der globalen Schätzung dazu, sehr große Glattheitswerte ($\rightarrow \infty$) einzustellen. Die Modellflexibilität wäre ansonsten aufgrund der Lokalität der Gültigkeitsfunktionen eingeschränkt.

Betrachtet man die geringe Sensitivität des Modells bezüglich der Glattheit und alle Schwierigkeiten bei der nichtlinearen Optimierung der Glattheit, so kann generell empfohlen werden, die Glattheit *a-priori* festzulegen. Allerdings kommt es bei hierarchischen Modellstrukturen zu einem weiteren unangenehmen Effekt: Die Überlappungen der Gültigkeitsfunktionen hängen nämlich von der Hierarchiestufe der LM ab. Da die Teilungsfunktionen Ψ_i miteinander multipliziert werden, um die Gültigkeitsfunktionen Φ_i zu generieren, wird die Aktivität nachfolgender Neuronen mit jeder Hierarchiestufe reduziert. Aus diesem Grund ist eine *a-priori* Fixierung der Glattheit nur dann sinnvoll möglich, wenn gleichzeitig die resultierende Baumtiefe (oder Hierarchiestufe) der hierarchischen Modellstruktur vorab bekannt ist. Um diesen Nachteil zu umgehen, wurde im Rahmen dieser Arbeit ein neuer Algorithmus entwickelt und implementiert, der nach jedem Hinzufügen eines LM zur hierarchischen Baumstruktur die Glattheit automatisch anpasst, siehe auch [52, 53, 54].

Der Algorithmus zur automatischen Glattheitsanpassung sieht vor, dass nach jeder Teilung eines lokalen Modells die Glattheit σ des gesamten Modells angepasst wird. Dazu kann folgendes Optimierungsproblem definiert werden:

$$J = \left|\Phi_{\text{Schwelle}} - \min\left(\Phi_i(\underline{c}_i), \Phi_j(\underline{c}_j)\right)\right| \rightarrow \min_{\sigma} . \tag{C.1}$$

Der Wert Φ_{Schwelle} ist vom Benutzer wählbar und stellt einen Schwellenwert für die minimale Aktivität aller Gültigkeitsfunktionen dar. Durch die Teilung eines existierenden Modells aus einer Hierarchiestufe höher entstehen die Gültigkeitsfunktionen Φ_i und Φ_j. Die Übergänge zwischen allen lokalen Modellen werden so lange mit Hilfe des Glattheitsfaktors σ schärfer gestellt, bis die Verlustfunktion J kleiner als eine bestimmte Genauigkeit ε ist.

Der Schwellenwert Φ_{Schwelle} muss vom Benutzer dem vorliegenden Prozess entsprechend gewählt werden. Weist der Prozess z.B. hohe Gradienten und die Trainingsdaten kein signifikantes Rauschen auf, sollte man den Schwellenwert hoch wählen, d.h. nahe am Wert 1. Andererseits ist es sinnvoll den Schwellenwert klein zu wählen, falls der Prozess stark verrauscht oder nur wenig Wissen über den Prozess bekannt ist. Experimente zeigten, dass die Wahl $\Phi_{\text{Schwelle}} = 0.9$ in den meisten Anwendungen zu guten Ergebnissen führt.

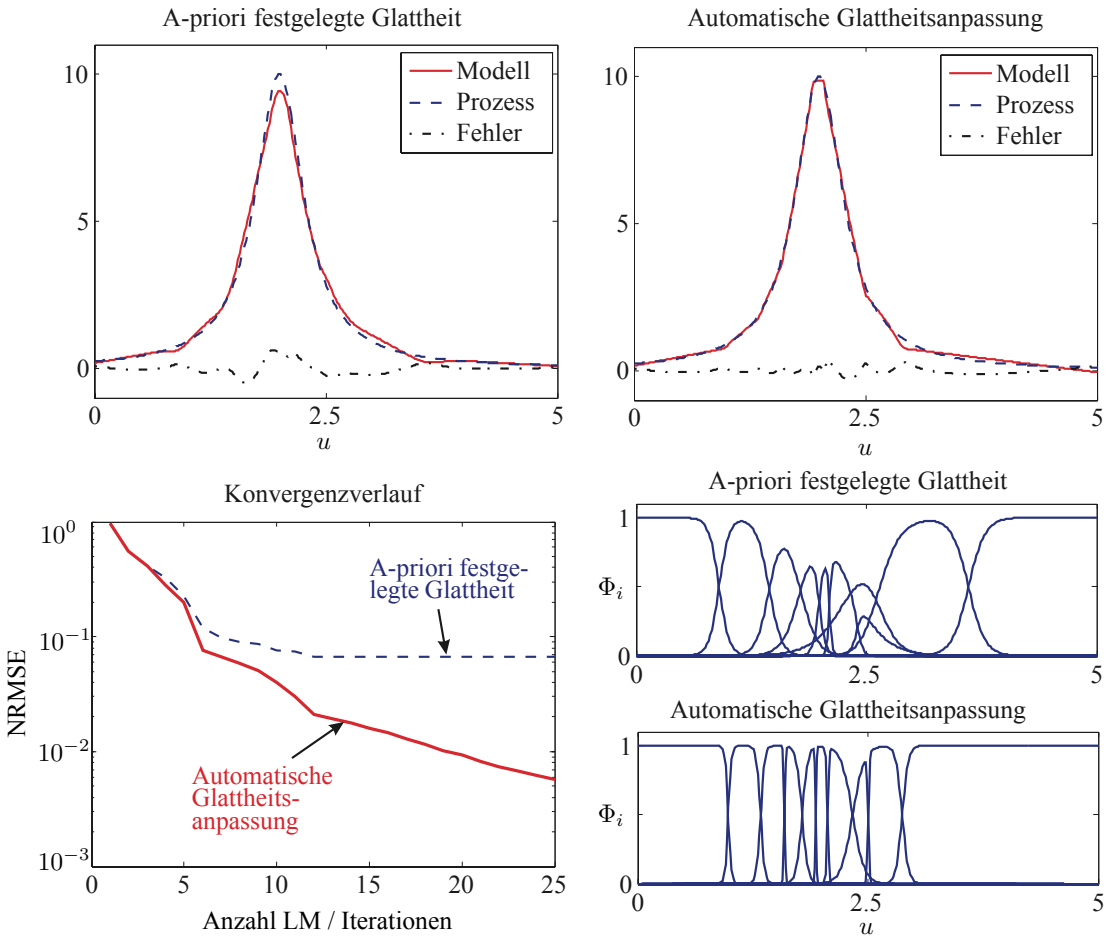

Bild C.1: Modellierung eines eindimensionalen Testprozesses, sowohl mit (oben links) als auch ohne die automatische Glattheitsanpassung (oben rechts). Die Konvergenzverläufe (unten links) und die resultierenden Partitionierungen (unten rechts) zeigen das verbesserte Auflösungsvermögen des Modells für Details im Prozess.

Nachdem die Glattheit des Modells manipuliert wurde, müssten die Parameter aller lokaler Modelle genau genommen nachfolgend neu geschätzt werden. Da aber die Parameter nur insignifikant von der Modell-Glattheit beeinflusst werden, kann auf eine nachträgliche Schätzung in den meisten Fällen verzichtet werden. Experimentelle Ergebnisse ergaben eine relative Parameter-Differenz von unter 0.1%.

Die Vorteile des Ansatzes zur automatischen Glattheitseinstellung kommen im nachfolgenden Beispiel zur Geltung. Folgender SISO-Prozess soll modelliert werden:

$$y = \frac{1}{(u-2)^2 + 0.1}\,, \tag{C.2}$$

mit $0 \leq u \leq 5$. Um diese Funktion gut nachbilden zu können, ist eine hohe Auflösung lokal in der Region $u \approx 2$ nötig. Aus diesem Grund stellt dieses Beispiel für den hierarchischen Strukturidentifikationsalgorithmus mit a-priori festgelegter Glattheit eine Art „Worst Case“-Szenario dar. In Bild C.1 sind sowohl das Modell mit a-priori fixierter Glattheit als auch mit automatisch angepasster Glattheit illustriert. Die Konvergenzverläufe sind bis zur Anzahl von 5 lokalen Modellen nahezu identisch. Allerdings fällt deutlich auf, dass mit

Tabelle C.1: Ergebnisse der Kennfeld-Modellierung.

Glattheits-Anpassung	#LM	NRMSE	Rechenzeit [s]
aus	15	0.0517	6.13
an	15	**0.0470**	**5.36**

zunehmender Anzahl an lokalen Modellen die Übergänge der Gültigkeitsfunktionen beim Modell mit a-priori festgelegter Glattheit zu weich sind. Die Verfeinerung des Modells um $u \approx 2$ bringt keine Verbesserung des Modellfehlers. In anderen Worten verliert das lokale Modellnetz die Eigenschaft eines universellen Approximators. Solch ein nicht zufrieden stellendes Modellverhalten ist die Konsequenz von zu kleinen Werten der Gültigkeitsfunktionen. Diese resultieren direkt aus der Multiplikation der Unterteilungsfunktionen aufgrund der hierarchischen Modellstruktur, welche vorab nicht bekannt ist. Der hier vorgestellte Algorithmus zur Glattheitseinstellung ist in der Lage, die Glattheit des gesamten Modells automatisch nach jeder Iteration des Partitionierungsalgorithmus' zu adaptieren. Wie die Konvergenzverläufe zeigen, kann die Modellgüte mit Hilfe des neuen Verfahrens mit jeder Iteration, im Gegensatz zum Modell mit a-priori fixierter Glattheit, signifikant verbessert werden.

Als Anwendungsbeispiel dient die Modellierung eines Motorkennfelds, siehe Bild C.2. Eingangsgrößen sind die Motordrehzahl in $[1/min]$ und die Einspritzmenge in $[mg]$. Als Ausgangsgröße dient das Drehmoment in $[Nm]$. Es stehen insgesamt 256 Messwerte für das Training mit HILOMOT zur Verfügung. Das Modell wurde einmal mit und einmal ohne den Algorithmus zur automatischen Glattheitsanpassung trainiert. Als Abbruchbedingung diente eine Anzahl von maximal 15 LM.

Man erkennt an den Ergebnissen in Tabelle C.1, dass sowohl die Modellgüte als auch die Rechenzeit[1] durch das Verfahren verbessert wurden. Zudem passt die gefundene Partitionierung in Bild C.2 besser zum Prozessverhalten.

[1]Verwendete Hardware: Intel Core 2 processor, 1.83 GHz, 2 GB RAM

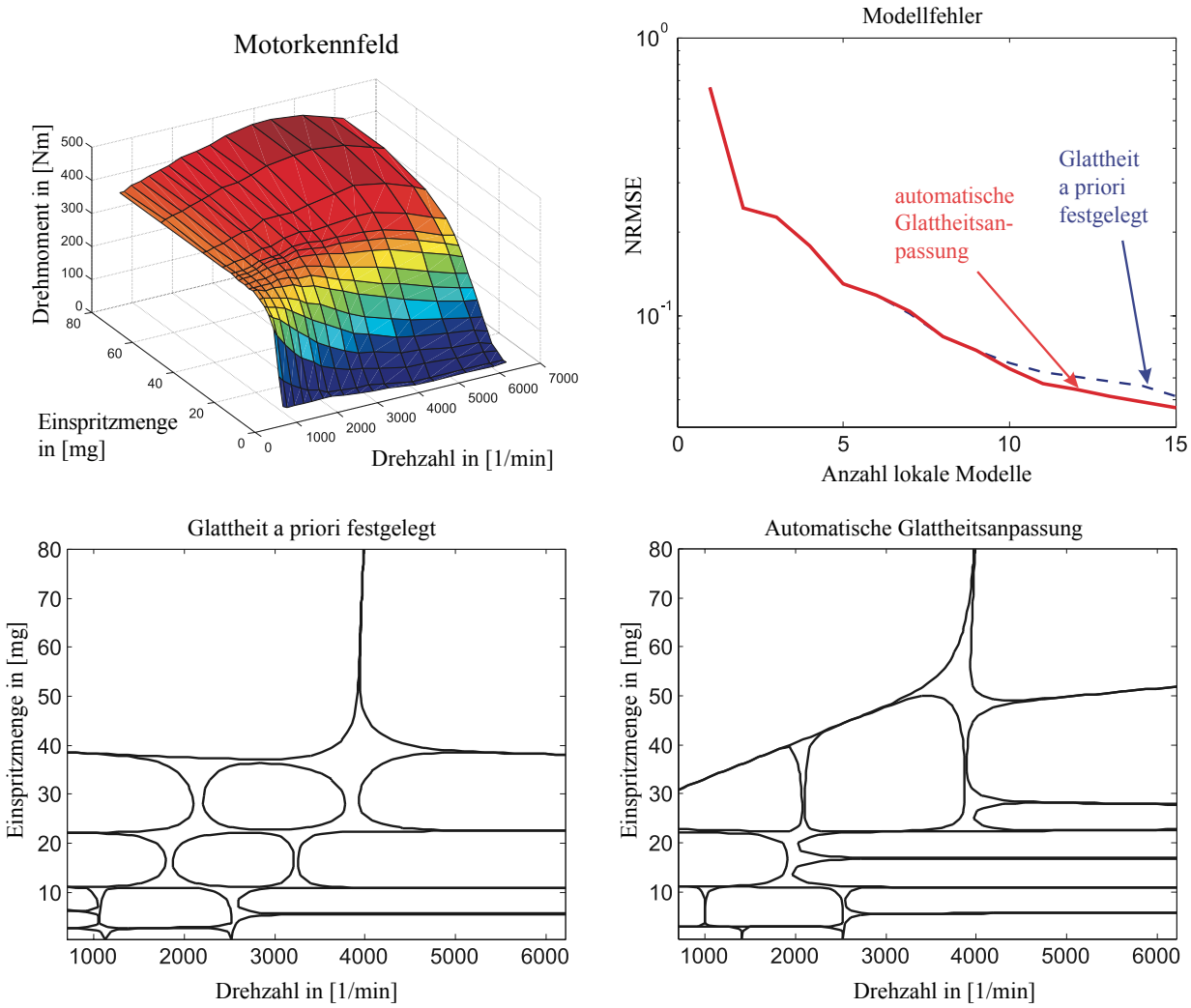

Bild C.2: Modellierung eines Motorkennfeldes mit und ohne automatische Glattheitsanpassung.

D Zur Robustheit des HILOMOT-Algorithmus

Anhand der folgenden vier Beispiele wird die Robustheit der Optimierung empirisch analysiert:

$$\text{Beispiel 1: } y = \frac{1}{0.1+\sum_{i=1}^{p}\frac{1-u_i}{p}}\,, \quad \text{Beispiel 2: } y = \frac{0.1^p}{\prod_{i=1}^{p}(1.1-u_i)}\,,$$
$$\text{Beispiel 3: } y = 1-\prod_{i=1}^{p}\cos(u_i)\,, \quad \text{Beispiel 4: } y = \frac{1}{p}\cdot\sum_{i=1}^{p}\cos(u_i)\,,$$

wobei p die Anzahl der Eingangsgrößen ist. Die Untersuchungen beschränkten sich auf bis zu fünf Eingänge. Weiterhin wurden die Trainingsdaten mit den drei Rauschvarianzen $\sigma_n^2 = \{0.0, 0.05, 0.1\}$ beaufschlagt.

D.1 Empirische Sensitivitätsanalyse bezüglich der Initialwerte

Standardmäßig wird bei HILOMOT so vorgegangen, dass vor der Optimierung zunächst die orthogonalen Schnitte in alle Eingangsrichtungen und die Schnittrichtung des Elternknotens getestet werden. Der Schnitt, welcher den geringsten globalen Modellfehler erzeugt, wird dann als Initialwert für die nichtlineare Schnittoptimierung gewählt. Um zu zeigen, dass die Optimierung robust bezüglich der Wahl der Initialwerte ist, werden zwei weitere Strategien getestet und mit der in HILOMOT implementierten Methode verglichen.

Die erste Alternativstrategie besteht darin, statt des besten den *schlechtesten* orthogonalen Schnitt als Anfangswert zu übernehmen. Wäre eine Sensibilität des Optimierungsverfahrens bezüglich der Anfangswerte vorhanden, müsste dies zu schlechteren Ergebnissen führen. Die zweite Strategie sieht vor, den Initialschnitt zwar durch das Zentrum des zu teilenden Modells zu legen, allerdings die Schnittrichtung zufällig zu wählen. Auch hier müsste eine große Schwankungsbreite der Ergebnisse erkennbar sein, falls eine hohe Empfindlichkeit bestünde.

Als Erstes findet ein Vergleich zwischen dem besten und dem schlechtesten achsenorthogonalen Schnitt als Startwert der Optimierung statt. Die Ergebnisse dieser Berechnungen zeigen keine qualitativen Unterschiede. Sie führen bei allen betrachteten Systemen zu gleich niedrigen, globalen Fehlern bei gleicher Anzahl von lokal linearen Modellen. Exemplarisch werden hier die 4 Beispiele für 5 Eingänge ohne Rauschen (Bild D.1) gezeigt, da hierbei die größte Differenz in den Konvergenzverläufen vorliegt.

Als nächster Schritt wird untersucht, ob sich zufällige Schnitte als Startwerte der Optimierung negativ auf das Ergebnis auswirken. Hierzu wurde jeder Prozess 30 mal mit verschiedenen zufälligen Schnitten initialisiert. Das Rauschen blieb dabei unverändert. Auch

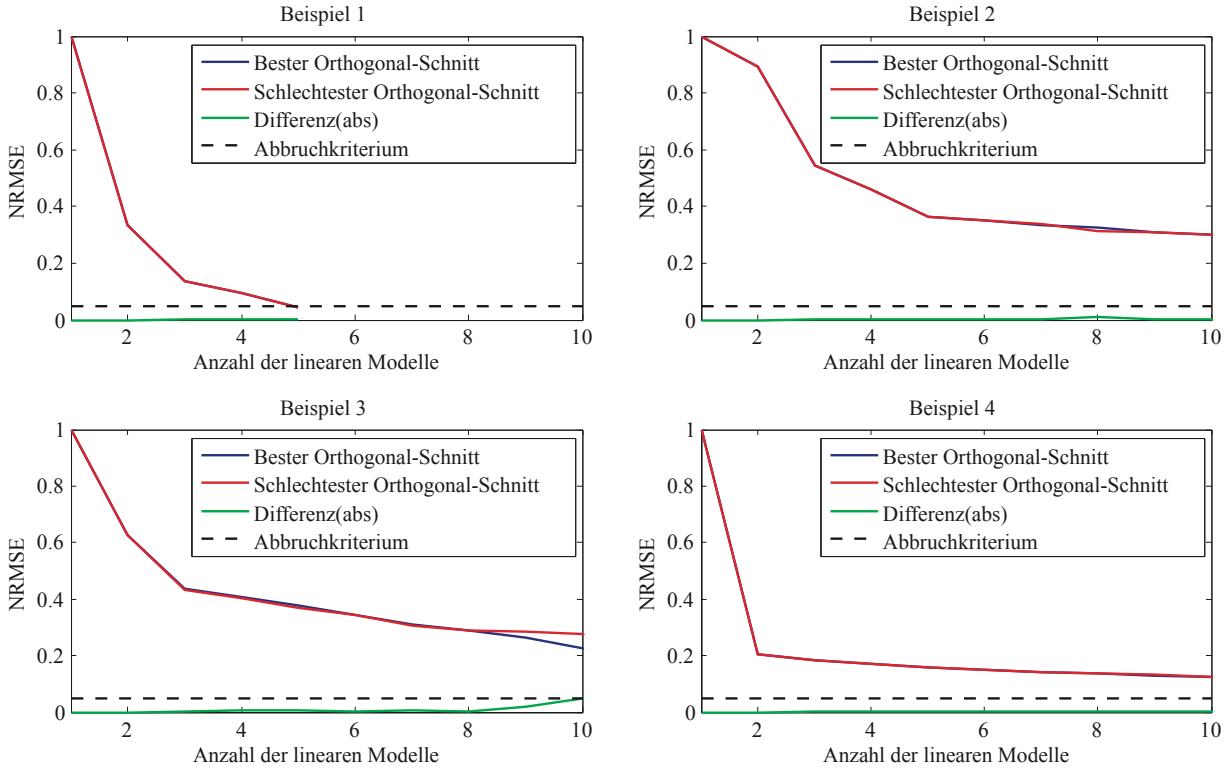

Bild D.1: Alle 4 Beispielprozesse zeigten mit 5 Eingängen und unverrauschten Trainingsdaten die größte Differenz innerhalb dieser Untersuchung.

hierbei stellt sich heraus, dass verschiedene Initialisierungsparameter keine qualitativ anderen Ergebnisse erzeugen. Es war zu erkennen, dass die Werte nur um einen schmalen Bereich variieren, sodass der Einfluss der Startwerte als klein einzustufen ist. Weiterhin ließen sich keine Ausreißer in den Ergebnissen feststellen. Alle betrachteten Systeme konvergierten mit verschiedenen Initialisierungen entweder zu einem ähnlichen Fehlerwert oder sind innerhalb von den vorgegebenen 10 Modellen unter einen Fehler von 5% gefallen.

Fasst man die geschilderten Untersuchungsergebnisse zusammen, ist die nichtlineare Optimierung innerhalb des HILOMOT-Algorithmus' unempfindlich gegenüber verschiedensten Startwerten und es besteht dank des überwachten Baumkonstruktionsverfahrens keine Gefahr zur Divergenz.

D.2 Vergleich zur simultanen Optimierung aller Schnitte

In HILOMOT wird pro Iteration des Algorithmus jeweils nur ein einzelner Schnitt optimiert. In diesem Vergleich wird untersucht, wie stark limitierend sich diese Beschränkung der Optimierung auf das Ergebnis auswirkt. Dazu wurde ein alternatives Verfahren entwickelt und implementiert, bei welchem in jeder Iteration der Baumkonstruktion alle, d.h. nicht nur die neu durchzuführende, Unterteilungen optimiert werden. Das führt zu einem größeren Optimierungsproblem (Anzahl der Parameter steigt linear an).

Zur Motivation wird anfangs ein einfaches 1-D Beispiel untersucht (siehe Bild D.2). Wie hierbei zu erkennen ist, kommt die gleichzeitige Optimierung aller Schnitte in diesem Beispiel mit einem lokalen Modell weniger aus. Bereits in der zweiten Iteration liefert die gleichzeitige Optimierung aller Schnitte ein sehr gutes Modell, wohingegen beim Standard-Verfahren drei Iterationen notwendig sind. Das bedeutet, beim Standardverfahren hätte man prinzipiell eine Iteration und damit zwei Modellparameter einsparen können.

Es bleibt also zu überprüfen, ob sich durch Verwendung des erweiterten Algorithmus' eine signifikante Verbesserung der Leistungsfähigkeit erzielen lässt. Die bei dieser Untersuchung verwendeten Prozesse sind identisch mit denen zur Sensibilitätsanalyse bezüglich der Initialwerte. Beide Verfahren lieferten annähernd gleiche Konvergenzverläufe. Der im Bild D.3 gezeigte Prozess 3 mit 5 Eingängen ohne Rauschen wies innerhalb der Untersuchung die größten Differenzen zwischen den beiden Konvergenzverläufen auf und soll somit exemplarisch die Zusammenhänge darstellen.

Die Methode, alle Schnitte gleichzeitig zu optimieren, führte zwar zu geringfügig besseren Ergebnissen, benötigte aber auch mehr Berechnungszeit. In Tabelle D.1 sind die Endwerte des globalen Fehlers (NRMSE) nach 10 lokalen Modellen aufgeführt. Bei einem globalen Fehler unter 5% wurde der Algorithmus abgebrochen. Die Untersuchung ergab, dass die Optimierung aller Schnitte deutlich mehr Zeit in Anspruch nimmt. Im Falle von Prozess 4 mit fünf Eingangsgrößen lag zwischen den Rechenzeiten ungefähr der Faktor 18, wobei die erzielte Genauigkeit bezüglich des NRMSE-Fehlers lediglich um 0,3% verbessert wurde. Somit übersteigt der Nachteil des großen Rechenaufwands den kleinen Vorteil der genaueren Ergebnisse in jedem Fall.

Tabelle D.1: Vergleich des globalen Fehlers (NRMSE) in % bei verschiedenen Optimierungsverfahren für unverrauschte Trainingsdaten.

Globaler Fehler in %	Prozess	1D	2D	3D	4D	5D
Ein Schnitt optimiert	1	<5	<5	<5	<5	<5
Alle Schnitte optimiert		<5	<5	<5	<5	<5
Ein Schnitt optimiert	2	<5	<5	13,27	24,39	28,83
Alle Schnitte optimiert		<5	<5	10,47	22,77	32,12
Ein Schnitt optimiert	3	<5	<5	8,51	15,62	22,05
Alle Schnitte optimiert		<5	<5	7,51	12,53	16,54
Ein Schnitt optimiert	4	<5	<5	<5	9,10	11,98
Alle Schnitte optimiert		<5	<5	<5	8,68	11,68

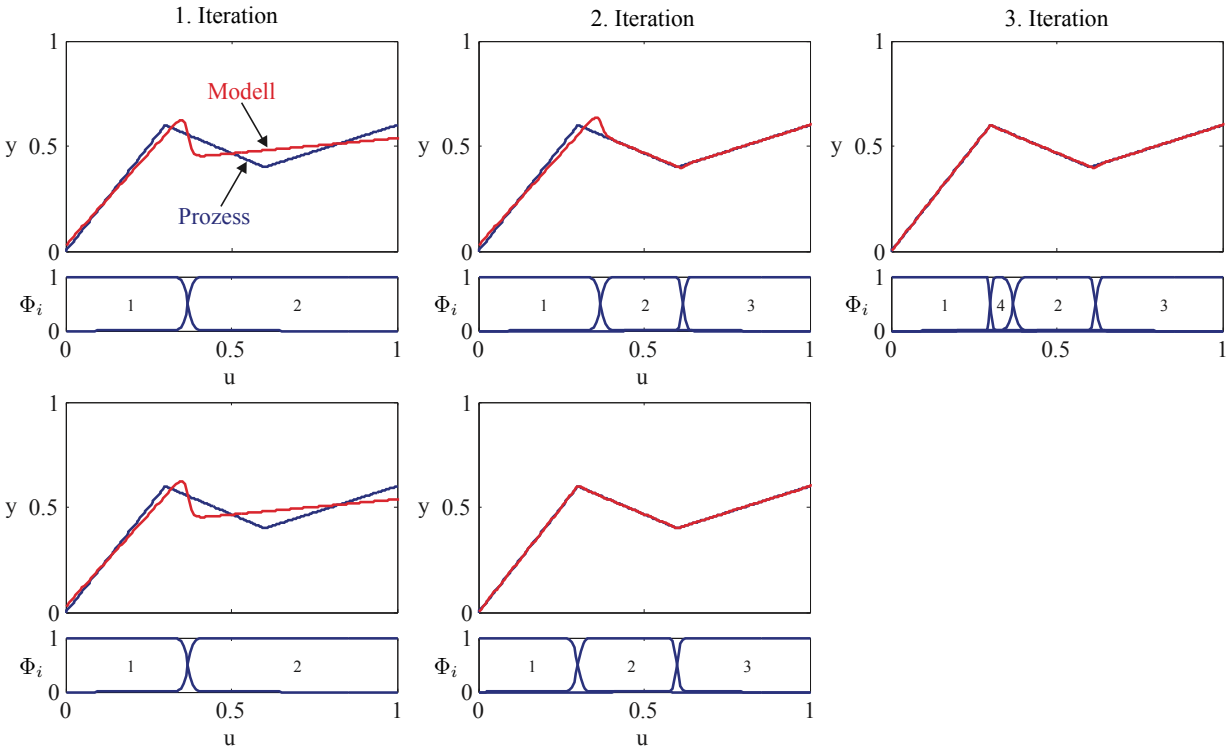

Bild D.2: Vorteil der Optimierung aller Schnitte (obere Zeile) gegenüber dem Standardverfahren (untere Zeile) anhand eines einfachen eindimensionalen Beispiels.

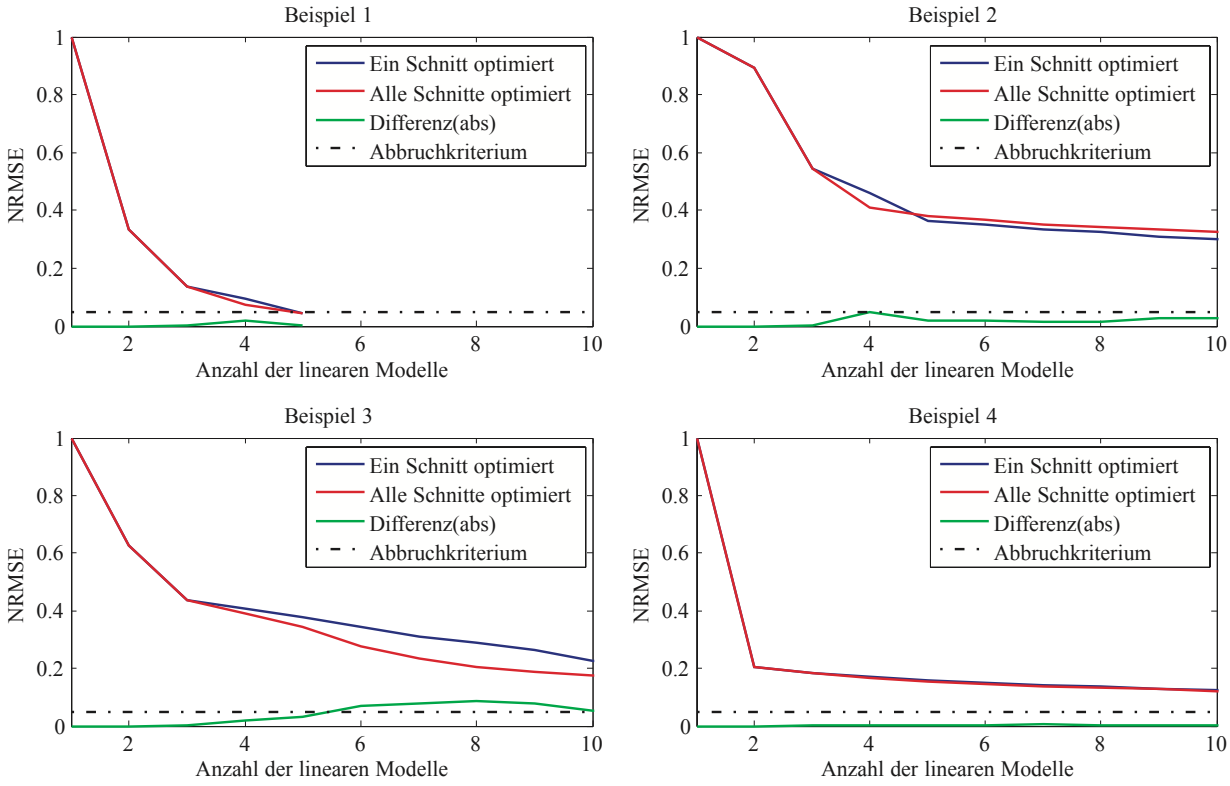

Bild D.3: Alle 4 Beispielprozesse mit 5 Eingängen und unverrauschten Trainingsdaten. Abbruchbedingung war entweder das Erreichen eines Fehlers unter 0.05 oder die maximale Anzahl von 10 lokalen Modellen.

Literaturverzeichnis

[1] *IAV EasyDoE-Toolsuite.* `http://www.iav.com/engineering/methods-tools/produkte/easydoe-toolsuite`

[2] Abonyi, J. ; Babuška, R. ; Szeifert, F.: Modified Gath-Geva fuzzy clustering for identification of Takagi-Sugeno fuzzy models. In: *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics* 32 (2002), Nr. 5, S. 612–621

[3] Akaike, H.: Information theory and an extension of the maximum likelihood principle. In: *Second international symposium on information theory* Bd. 1 Springer Verlag, 1973, S. 267–281

[4] Akaike, H.: A new look at the statistical model identification. In: *IEEE Transactions on Automatic Control* 19 (1974), Nr. 6, S. 716–723

[5] Angelov, P.P. ; Filev, D.P.: An approach to online identification of Takagi-Sugeno fuzzy models. In: *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics* 34 (2004), Nr. 1, S. 484–498. – ISSN 1083–4419

[6] Babuška, R.: *Fuzzy Modeling and Identification.* Delft, Niederlande, Dept. of Control Engineering, Delft University of Technology, Diss., 1996

[7] Babuška, R. ; Setnes, M. ; Kaymak, U. ; van Naute Lemke, H.R.: Simplification of Fuzzy Rule Bases. In: *European Congress on Intelligent Techniques and Soft Computing (EUFIT).* Aachen, 1996, S. 1115–1119

[8] Bach, Francis R.: Bolasso: model consistent Lasso estimation through the bootstrap. In: *Proceedings of the 25th international conference on Machine learning (ICML).* New York, NY, USA : ACM, 2008, S. 33–40

[9] Balageas, D. ; Fritzen, C.-P. ; Güemes, A.: *Structural health monitoring.* Wiley Online Library, 2010

[10] Bänfer, O. ; Hartmann, B. ; Nelles, O.: POLYMOT versus HILOMOT - a Comparison of Two Different Training Algorithms for Local Model Networks. In: *IFAC Symposium on System Identification (SYSID)* Bd. 16. Brüssel, Belgien, 2012, S. 1569–1574

[11] Bänfer, O. ; Nelles, O.: Polynomial Model Tree (POLYMOT) – A New Training Algorithm for Local Model Networks with Higher Degree Polynomials. In: *IEEE International Conference on Control & Automation (ICCA).* Christchurch, Neuseeland, Dezember 2009

[12] Bänfer, O. ; Nelles, O. ; Kainz, J. ; J.Beer: Lokale Modellnetze – Die zukünftige Modellierungsmethode für Steuergeräte? In: *ATZelektronik* (2008), Nr. 6

[13] Barron, A.R.: Approximation and Estimation Bounds for Artificial Neural Networks. In: *Machine Learning* 14 (1994), S. 115–133

[14] Baumann, W. ; Dreher, T. ; Stelzer, S.: DoE for Production Engine Calibration. In: *Proceedings of 7th conference Design of experiments (DoE) in engine development.*

[15] Bellman, R.E.: *Dynamic programming.* Princeton University Press, 1957 (Rand Corporation research study)

[16] Berk, R. A.: *Statistical Learning from a Regression Perspective.* New York, NY, USA : Springer Science and Business Media, 2008

[17] Bishop, C.M.: *Pattern Recognition and Machine Learning (Information Science and Statistics).* Secaucus, NJ, USA : Springer-Verlag New York, 2006

[18] Boyd, S. ; Vandenberghe, L.: *Convex Optimization.* Cambridge University Press, 2004

[19] Breiman, L.: Hinging Hyperplanes for Regression, Classification, and Function Approximation. In: *IEEE Transactions on Information Theory* 39 (1993), Mai, Nr. 3, S. 999–1013

[20] Breiman, L. ; J.H. Friedman R. Olshen R., C.J. S.: *Classification and Regression Trees.* New York : Chapman & Hall, 1984

[21] Burnham, K.P. ; Anderson, D.R.: Kullback-Leibler information as a basis for strong inference in ecological studies. In: *Wildlife Research* 28 (2001), Nr. 2, S. 111–119

[22] Burnham, K.P. ; Anderson, D.R.: *Model selection and multimodel inference: a practical information-theoretic approach.* Springer Verlag, 2002

[23] Chang, C.C. ; Lin, C.J.: LIBSVM: a library for support vector machines. In: *ACM Transactions on Intelligent Systems and Technology (TIST)* 2 (2011), Nr. 3, S. 27

[24] Chiu, S.L.: Fuzzy model identification based on cluster estimation. In: *Journal of intelligent and Fuzzy systems* 2 (1994), Nr. 3, S. 267–278

[25] Christiani, N. ; Shawe-Taylor, J.: *An Introduction to Support Vector Machines and other kernel-based learning methods.* Cambridge, UK : Cambridge University Press, 2000

[26] Cleveland, W.S. ; Devlin, S.J. ; Grosse, E.: Regression by Local Fitting: Methods, Properties, and Computational Algorithms. In: *Journal of Econometrics* 37 (1996), S. 87–114

[27] Cohn, D.A.: Neural network exploration using optimal experiment design. In: *Advances in neural information processing systems* 6 (1994), S. 679–686

[28] Cohn, D.A. ; Ghahramani, Z. ; Jordan, M.I.: Active learning with statistical models. In: *Arxiv preprint cs/9603104* (1996)

[29] Denna, M. ; Mauri, G. ; Zanaboni, A.M.: Learning fuzzy rules with tabu search-an application to control. In: *IEEE Transactions on Fuzzy Systems* 7 (1999), Nr. 3, S. 295–318

[30] Doll, G. ; Fausten, H. ; Noell, R. ; Spengel, C. ; Werner, P.: Der neue V6-Dieselmotor von Mercedes-Benz. In: *MTZ-Motortechnische Zeitschrift* 66 (2005), Nr. 9, S. 624–634

[31] Draper, N. ; Smith, H.: *Applied regression analysis*. Wiley, 1981

[32] Efron, B. ; Hastie, T. ; Johnstone, I. ; Tibshirani, R.: Least Angle Regression. In: *The Annals of Statistics* 32 (2004), Nr. 2, S. 407–451

[33] Emami, M.R. ; Turksen, IB ; Goldenberg, A.A.: Development of a systematic methodology of fuzzy logic modeling. In: *IEEE Transactions on Fuzzy Systems* 6 (1998), Nr. 3, S. 346–361

[34] Ernst, S.: Hinging Hyperplane Trees for Approximation and Identification. In: *IEEE Conference on Decision and Control (CDC)*. Tampa, USA, 1998, S. 1261–1277

[35] Fedorov, V.V. ; Hackl, P.: *Model-oriented design of experiments*. Springer Verlag, 1997

[36] Fischer, T. ; Hartmann, B. ; Nelles, O.: Increasing the Performance of a Training Algorithm for Local Model Networks. In: *World Congress of Engineering and Computer Science (WCECS)*. San Francisco, USA, Oktober 2012

[37] Friedman, J.H.: Multivariate Adaptive Regression Splines. In: *The Annals of Statistics* 19 (1991), März, Nr. 1, S. 1–141

[38] Gan, Q. ; Harris, C.J.: Fuzzy local linearization and local basis function expansion in nonlinear system modeling. In: *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics* 29 (1999), Nr. 4, S. 559–565

[39] Gath, I. ; Geva, A.B..: Unsupervised Optimal Fuzzy Clustering. In: *IEEE Trans. Pattern Analysis and Machine Intelligence* 11 (1989), Nr. 7, S. 773–781

[40] Güemes, J. ; Ostachowicz, W.: *New trends in structural health monitoring*. Springer, 2013

[41] Gustafson, D.E. ; Kessel, W.C.: Fuzzy Clustering with a Fuzzy Covariance Matrix. In: *IEEE Conference and Decision and Control*. San Diego, USA, 1979, S. 761–766

[42] Hafner, M. ; Schuler, M. ; Nelles, O.: Dynamical identification and control of combustion engine exhaust. In: *American Control Conference* Bd. 1 IEEE, 1999, S. 222–226

[43] Halton, J.H.: On the efficiency of certain quasi-random sequences of points in evaluating multi-dimensional integrals. In: *Numerische Mathematik* 2 (1960), Nr. 1, S. 84–90

[44] Hametner, C. ; Jakubek, S.: Nonlinear System Identification through Local Model Approaches: Partitioning Strategies and Parameter Estimation. In: *Modelling, Simulation and Identification, A. Mohamed (ed.)*, 2010, S. 179–194

[45] Hametner, C. ; Jakubek, S.: Engine model identification using local model networks in comparison with a multilayer perceptron network. In: *Proceedings of the 2nd International Multi-Conference on Complexity, Informatics and Cybernetics* Bd. 1202, 2011

[46] Hänsler, E.: *Statistische Signale: Grundlagen und Anwendungen*. Springer, 2001

[47] Hartmann, B. ; Bänfer, O. ; Nelles, O. ; Sodja, A. ; Teslic, L. ; Škrjanc, I.: Supervised Hierarchical Clustering in Fuzzy Model Identification. In: *IEEE Transactions on Fuzzy Systems* 19 (2011), Nr. 6, S. 1163

[48] Hartmann, B. ; Baumann, W. ; Nelles, O.: Axes-Oblique Partitioning of Local Model Networks for Engine Calibration. In: *Design of Experiments (DoE) in Engine Development*, Expert Verlag, Juni 2013

[49] Hartmann, B. ; Ebert, T. ; Fischer, T. ; Belz, J. ; Kampmann, G. ; Nelles, O.: LMNtool – Toolbox zum automatischen Trainieren lokaler Modellnetze. In: *Workshop Computational Intelligence*. Dortmund, Dezember 2012

[50] Hartmann, B. ; Ebert, T. ; Nelles, O.: Model-based design of experiments based on local model networks for nonlinear processes with low noise levels. In: *American Control Conference (ACC)* IEEE, 2011, S. 5306–5311

[51] Hartmann, B. ; Moll, J. ; Nelles, O. ; Fritzen, C.: Modeling of nonlinear wave velocity characteristics in a structural health monitoring system. In: *International Conference on Control Applications (CCA)* IEEE, 2010, S. 1011–1016

[52] Hartmann, B. ; Nelles, O.: Advantages of Hierarchical versus Flat Model Structures for High–Dimensional Mappings. In: *Workshop Computational Intelligence*. Bommerholz, Dezember 2009

[53] Hartmann, B. ; Nelles, O.: Automatic Adjustment of the Transition between Local Models in a Hierarchical Structure Identification Algorithm. In: *European Control Conference (ECC)*. Budapest, Ungarn, August 2009

[54] Hartmann, B. ; Nelles, O.: On the Smoothness in Local Model Networks. In: *American Control Conference (ACC)*. St. Louis, USA, Juni 2009

[55] Hartmann, B. ; Nelles, O.: Approximative Tikhonov Regularization for Smoothing Hierarchical Local Model Networks. In: *International Conference on Fuzzy Systems and Neural Computing (ICFSNC 2010)*. Bali, Indonesien, Juli 2010

[56] Hartmann, B. ; Nelles, O.: Structure Trade-off Strategy for Local Model Networks. In: *IEEE International Conference on Control Applications (CCA), 2012*. Dubrovnik, Kroatien, 2012

[57] Hartmann, B. ; Nelles, O.: Adaptive Test Planning for the Calibration of Combustion Engines – Methodology. In: *Design of Experiments (DoE) in Engine Development*, Expert Verlag, Juni 2013

[58] Hartmann, B. ; Nelles, O. ; Belič, A. ; Zupančič-Božič, D.: Local Model Networks for the Optimization of a Tablet Production Process. In: *Artificial Intelligence and Computational Intelligence* 5855 (2009), S. 241–250

[59] Hartmann, B. ; Nelles, O. ; Skrjanc, I. ; Sodja, A.: Global Supervised and Local Unsupervised Learning in Local Model Networks. In: *IFAC Symposium on System Identification (SYSID)*. Saint-Malo, Frankreich, Juli 2009

[60] Hartmann, B. ; Nelles, O. ; Skrjanc, I. ; Sodja, A.: Supervised Hierarchical Clustering (SUHICLUST) for Nonlinear System Identification. In: *IEEE Symposium on Computational Intelligence in Control and Automation (CICA)*. Nashville, USA, März 2009

[61] Hastie, T. ; Tibshirani, R. ; Friedman, J.: *The Elements of Statistical Learning – Data Mining, Inference, and Prediction. Second Edition.* New York, NY, USA : Springer Science+Business Media, 2009

[62] Haykin, S. ; Hwang, JN: Neural networks: a comprehensive foundation. In: *IEEE Transactions on Neural Networks* 6 (1995), Nr. 4, S. 1019–1021

[63] Haykin, S.S.: *Neural networks and learning machines*. Bd. 3. Prentice Hall, 2009

[64] Hoerl, A.E. ; Kennard, R.W.: Ridge Regression: Biased Estimation for Nonorthogonal Problems. In: *Technometrics* 12 (1970), Nr. 1, S. 55–67

[65] Hoffmann, F. ; Nelles, O.: Structure identification of TSK-fuzzy systems using genetic programming. In: *Proceedings Information Processing and Management of Uncertainty* (2000)

[66] Hoffmann, F. ; Nelles, O.: Genetic Programming for Model Selection of TSK Fuzzy Systems. In: *International Journal of Information Sciences* 136 (2001), S. 7–28

[67] Hoffmann, F. ; Schauten, D. ; Holemann, S.: Incremental evolutionary design of TSK fuzzy controllers. In: *IEEE Transactions on Fuzzy Systems* 15 (2007), Nr. 4, S. 563–577

[68] Hofwing, M.: Design of Experiments – A- D- I- S-optimality. In: *Proceedings of the 2nd International Conference on Engineering Optimization*, 2010

[69] Hohensohn, J. ; Mendel, J.M.: Two-Pass Orthogonal Least-Squares Algorithm to Train and Reduce Fuzzy Logic Systems. In: *IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*. Orlando, USA, 1994, S. 696–700

[70] Huang, Y.P. ; Wang, S.F.: Designing a fuzzy model by adaptive macroevolution genetic algorithms. In: *Fuzzy Sets and Systems* 113 (2000), Nr. 3, S. 367–379

[71] Huwendiek, O. ; Brockmann, W.: Function approximation with decomposed fuzzy systems. In: *Fuzzy sets and systems* 101 (1999), Nr. 2, S. 273–286

[72] Ilzarbe, L. ; Álvarez, M.J. ; Viles, E. ; Tanco, M.: Practical applications of design of experiments in the field of engineering: a bibliographical review. In: *Quality and Reliability Engineering International* 24 (2008), Nr. 4, S. 417–428

[73] Institute, SAS ; Publishing, Publishing S.: *Jmp 8 Design of Experiments User Guide*. SAS Institute, 2009

[74] Isermann, R.: *Modellgestützte Steuerung, Regelung und Diagnose von Verbrennungsmotoren*. Berlin, Heidelberg, New York : Springer Verlag, 2003

[75] Ishibuchi, H. ; Nozaki, K. ; Yamamoto, N. ; Tanaka, H.: Construction of Fuzzy Classification Systems with Rectangular Fuzzy Rules Using Genetic Algorithms. In: *Fuzzy Sets & Systems* 65 (1994), S. 237–253

[76] Izenman, A. J.: *Modern Multivariate Statistical Techniques – Regression, Classification, and Manifold Learning*. New York, NY, USA : Springer Science+Business Media, 2008

[77] Jakubek, S. ; Keuth, N.: A local neuro-fuzzy network for high-dimensional models and optimization. In: *Engineering Applications of Artificial Intelligence* 19 (2006), Nr. 6, S. 705–717

[78] Jang, J.-S.R.: ANFIS: Adaptive-Network-Based Fuzzy Inference Systems. In: *IEEE Transactions on Systems, Man, and Cybernetics* 23 (1993), Nr. 3, S. 665–685

[79] Jang, J.-S.R. ; Sun, C.T. ; Mizutani, E.: *Neuro-Fuzzy and Soft Computing: A Computational Approach to Learning and Machine Intelligence*. Englewood Cliffs : Prentice Hall, 1997

[80] Jesús Rubio, J. de: SOFMLS: online self-organizing fuzzy modified least-squares network. In: *IEEE Transactions on Fuzzy Systems* 17 (2009), Nr. 6, S. 1296–1309

[81] Jin, Y.: Fuzzy modeling of high-dimensional systems: complexity reduction and interpretability improvement. In: *IEEE Transactions on Fuzzy Systems* 8 (2000), Nr. 2, S. 212–221

[82] Johansen, T.A.: Identification of Non-linear System Structure and Parameters Using Regime Decomposition. In: *Automatica* 31 (1995), Nr. 2, S. 321–326

[83] Johansen, T.A.: Robust Identification of Takagi-Sugeno-Kang Fuzzy Models Using Regularization. In: *IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)* Bd. 1. New Orleans, USA, 1996, S. 180–193

[84] Johansen, T.A.: On Tikhonov Regularization, Bias and Variance in Nonlinear System Identification. In: *Automatica* 33 (1997), Nr. 3, S. 441–446

[85] Jones, D.R. ; Schonlau, M. ; Welch, W.J.: Efficient global optimization of expensive black-box functions. In: *Journal of Global optimization* 13 (1998), Nr. 4, S. 455–492

[86] Juang, C.F. ; Lin, C.T.: An online self-constructing neural fuzzy inference network and its applications. In: *IEEE Transactions on Fuzzy Systems* 6 (1998), Nr. 1, S. 12–32

[87] Kang, S.J. ; Woo, C.H. ; Hwang, H.S. ; Woo, K.B.: Evolutionary design of fuzzy rule base for nonlinear system modeling and control. In: *IEEE Transactions on Fuzzy Systems* 8 (2000), Nr. 1, S. 37–45

[88] Kasabov, N.K. ; Song, Q.: DENFIS: dynamic evolving neural-fuzzy inference system and its application for time-series prediction. In: *Fuzzy Systems, IEEE Transactions on* 10 (2002), Nr. 2, S. 144–154

[89] Kecman, V.: *Learning and soft computing: support vector machines, neural networks, and fuzzy logic models*. The MIT press, 2001

[90] Keuth, N. ; Ebner, T. ; Pflügl, H.: *Method for measuring a non-linear dynamic real system using design of experiment*. 2009. – EP Patent 2088486

[91] Kim, E. ; Park, M. ; Kim, S. ; Park, M.: A transformed input-domain approach to fuzzy modeling. In: *IEEE Transactions on Fuzzy Systems* 6 (1998), Nr. 4, S. 596–604

[92] Klein, P.: *Zylinderdruckbasierte Füllungserfassung für Verbrennungsmotoren*, Universitätsbibliothek Siegen, Diss., 2009

[93] Klein, P. ; Kirschbaum, F. ; Hartmann, B. ; Bogachik, J. ; Nelles, O.: Adaptive Test Planning for the Calibration of Combustion Engines – Application. In: *Design of Experiments (DoE) in Engine Development*, Expert Verlag, Juni 2013

[94] Klein, P. ; Linssen, R.: Moderne Methoden zur Motorgrundapplikation auf dem Prüfstand. In: *Elektronisches Management von Fahrzeugantrieben.* Frankfurt, Oktober 2012

[95] Kleppmann, W.: *Taschenbuch Versuchsplanung – Produkte und Prozesse optimieren*. München, Wien : Carl Hanser Verlag, 2003

[96] Klöpper, F.: *Entwicklung und Einsatz modellgestützter Online-Methoden zur Parameteroptimierung von Verbrennungsmotoren am Prüfstand*, Diss., 2009

[97] Kortmann, M.: *Die Identifikation nichtlinearer Ein- und Mehrgrössensysteme auf der Basis nichtlinearer Modellansätze*. Düsseldorf, Diss., 1989

[98] Kraemer, P.: *Schadensdiagnoseverfahren für die Zustandsüberwachung von Offshore-Windenergieanlagen*, Universität Siegen, Diss., 2011

[99] Lee, S.J. ; Ouyang, C.S.: A neuro-fuzzy system modeling with self-constructing rule generationand hybrid SVD-based learning. In: *IEEE Transactions on Fuzzy Systems* 11 (2003), Nr. 3, S. 341–353

[100] Lin, C.K. ; Wang, S.D.: Fuzzy system identification using an adaptive learning rule with terminal attractors. In: *Fuzzy sets and systems* 101 (1999), Nr. 3, S. 343–352

[101] Ljung, L.: *System identification*. Wiley Online Library, 1999

[102] Lughofer, E.D.: Flexfis: A robust incremental learning approach for evolving Takagi–Sugeno fuzzy models. In: *IEEE Transactions on Fuzzy Systems* 16 (2008), Nr. 6, S. 1393–1410

[103] Madár, J. ; Abonyi, J. ; Szeifert, F.: Genetic programming for the identification of nonlinear input-output models. In: *Industrial & engineering chemistry research* 44 (2005), Nr. 9, S. 3178–3186

[104] Mallows, C.L.: Some comments on Cp. In: *Technometrics* (1973), S. 661–675

[105] Mastorocostas, P.A. ; Theocharis, J.B. ; Kiartzis, S.J. ; Bakirtzis, A.G.: A hybrid fuzzy modeling method for short-term load forecasting. In: *Mathematics and Computers in Simulation* 51 (2000), Nr. 3, S. 221–232

[106] Miller, A.J.: *Subset Selection in Regression*. New York : Chapman & Hall, 1990 (Statistics and Applied Probability)

[107] Mitterer, A.: *Optimierung vielparametriger Systeme in der KFZ-Antriebsstrangentwicklung*. Deutschland, Diss., 2000

[108] Moll, J.: *Strukturdiagnose mit Ultraschallwellen durch Verwendung von piezoelektrischen Sensoren und Aktoren*, Universität Siegen, Diss., 2011

[109] Moll, J. ; Hartmann, B. ; Chaaban, R. ; Fritzen, C.-P. ; Nelles, O.: A Novel Online Learning Approach for Ultrasonic Imaging Applied to a Non-Convex Structure. In: *(Interner Bericht)* (2011)

[110] Moll, J. ; Schulte, R.T. ; Hartmann, B. ; Fritzen, C.-P. ; Nelles, O.: Multi-Site Damage Localization in Generally Anisotropic plate-like Structures Using an Active Guided Wave Structural Health Monitoring System. In: *Smart Materials and Structures* 19 (2010), Nr. 4

[111] Montgomery, D.C. ; Peck, E.A. ; Vining, G.G. ; Vining, J.: *Introduction to linear regression analysis*. Bd. 3. Wiley New York, 2001

[112] Moody, J. ; Darken, C.J.: Fast Learning in Networks of Locally-Tuned Processing Units. In: *Neural Computation* 1 (1989), Nr. 2, S. 281–294

[113] Murray-Smith, R.: *A Local Model Network Approach to Nonlinear Modeling*. Strathclyde, UK, University of Strathclyde, Diss., 1994

[114] Murray-Smith, R. ; Gollee, H.: A constructive learning algorithm for local model networks. In: *Proceedings of the IEEE Workshop on Computer-Intensive Methods in Control and Signal Processing, Prag, Tschechien* Citeseer, 1994, S. 21–29

[115] Murray-Smith, R. ; Johansen, T.A.: Local learning in local model networks. In: *IEE International Conference on Artificial Neural Networks*, 1995, S. 40–46

[116] Nelles, O.: *Nonlinear System Identification.* Berlin : Springer, 2001

[117] Nelles, O.: Axes-Oblique Partitioning Strategies for Local Model Networks. In: *IEEE International Symposium on Intelligent Control (ISIC).* München, Oktober 2006

[118] Nelles, O.: Neuronale Netze: Eine Übersicht. In: *Informationsfusion in der Mess- und Sensortechnik*, Universitätsverlag Karlsruhe, http://digbib.ubka.uni-karlsruhe.de/volltexte/1000004875, 2006

[119] Nelles, O. ; Fink, A. ; Babuška, R. ; Setnes, M.: Comparison of Two Construction Algorithms for Takagi-Sugeno Fuzzy Models. In: *International Journal of Applied Mathematics and Computer Science* 10 (2000), Nr. 4, S. 835–855

[120] Nelles, O. ; Sinsel, S. ; Isermann, R.: Local Basis Function Networks for Identification of a Turbocharger. In: *IEEE UKACC International Conference on Control.* Exeter, UK, Sept. 1996, S. 7–12

[121] Nitzschke, E. ; Kirschbaum, F. ; Linssen, R.: Calibration Methods in Engine Development at Mercedes-Benz Cars – The Daimler Tool Suite. In: *4th International Symposium on Development Methodology.* Wiesbaden, November 2011

[122] N.N.: *MATLAB Fuzzy Logic Toolbox: User's Guide, Version 2.2.7.* Natick, MA : The MATHWORKS Inc., 2008

[123] N.N.: *MATLAB Model-Based Calibration Toolbox: User's Guide, Version 3.4.* Natick, MA : The MATHWORKS Inc., 2008

[124] N.N.: *MATLAB Statistics Toolbox: User's Guide, Version 6.* Natick, MA : The MATHWORKS Inc., 2008

[125] Nurmela, K.J.: *Stochastic Optimization Methods in Sphere Packing and Covering Problems in Discrete Geometry and Coding Theory.* Citeseer, 1997

[126] Poland, J.: *Modellgestützte und evolutionäre Optimierungsverfahren für die Motorentwicklung*, Diss., 2002

[127] Pucar, P. ; Millnert, M.: Smooth Hinging Hyperplanes: A Alternative to Neural Nets. In: *European Control Conference (ECC).* Rom, Italien, 1995, S. 1173–1178

[128] Rasmussen, C.E.: *Gaussian processes for machine learning.* MIT Press, 2006

[129] Rifkin, R.M. ; Lippert, R.A.: Notes on regularized least squares. (2007)

[130] Ronco, E. ; Gawthrop, P.J.: Incremental model reference adaptive polynomial controller network. In: *IEEE Conference on Decision and Control.* New York, USA, 1997, S. 4171–4172

[131] Roubos, H. ; Setnes, M.: Compact fuzzy models through complexity reduction and evolutionary optimization. In: *IEEE International Conference on Fuzzy Systems* Bd. 2, 2000, S. 762–767

[132] Schohn, G. ; Cohn, D.: Less is More: Active Learning with Support Vector Machines. In: *Proceedings of the Seventeenth International Conference on Machine Learning* Morgan Kaufmann Publishers Inc., 2000, S. 839–846

[133] Schölkopf, B.: *Support Vector Learning*. Berlin, Technische Universität Berlin, Diss., 1997

[134] Schölkopf, B. ; Smola, A.: *Learning with Kernels – Support Vector Machines, Regularization, Optimization, and Beyond*. Cambridge, Massachusetts, London, England : MIT Press, 2002

[135] Schwarz, G.: Estimating the dimension of a model. In: *The annals of statistics* 6 (1978), Nr. 2, S. 461–464

[136] Sequenz, H. ; Schreiber, A. ; Isermann, R.: Identification of nonlinear static processes with local polynomial regression and subset selection. In: *Proceedings of the 15th IFAC Symposium on System Identification*, 2009

[137] Shi, Y. ; Eberhart, R. ; Chen, Y.: Implementation of evolutionary fuzzy systems. In: *IEEE Transactions on Fuzzy Systems* 7 (1999), Nr. 2, S. 109–119

[138] Shorten, Robert ; Murray-Smith, Roderick: Side-Effects of Normalising Basis Functions in Local Model Networks. In: *Multiple Model Approaches to Modelling and Control* 8 (1997), S. 211–228

[139] Sjöberg, J. ; Zhang, Q. ; Ljung, L. ; Benveniste, A. ; Delyon, B. ; Glorennec, P.-Y. ; Hjalmarsson, H. ; Juditsky, A.: Nonlinear Black-Box Modeling in System Identification: A Unified Overview. In: *Automatica* 31 (1995), Nr. 12, S. 1691–1724

[140] Sobol, I.M.: On quasi-Monte Carlo integrations. In: *Mathematics and Computers in Simulation* 47 (1998), Nr. 2-5, S. 103–112

[141] Stoica, P. ; Selen, Y.: Model-order selection: a review of information criterion rules. In: *IEEE Signal Processing Magazine* 21 (2004), Nr. 4, S. 36–47

[142] Strokbro, K. ; Umberger, D.K. ; Hertz, J.A.: Exploiting Neurons with Localized Receptive Fields to Learn Chaos. In: *Journal of Complex Systems* 4 (1990), Nr. 3, S. 603–622

[143] Sugeno, M. ; Kang, G.T.: Structure Identification of Fuzzy Model. In: *Fuzzy Sets & Systems* 28 (1988), Nr. 1, S. 15–33

[144] Sugeno, M. ; Yasukawa, T.: A fuzzy-logic-based approach to qualitative modeling. In: *IEEE Transactions on fuzzy systems* 1 (1993), Nr. 1, S. 7–31

[145] Sung, A. ; Klöpper, F. ; Mitterer, A. ; Wachtmeister, G. ; Zell, A.: Modellbasierte Online-Optimierung in der Simulation und am Motorenprüfstand. In: *Motortechnische Zeitschrift MTZ* 68 (2007), Nr. 1, S. 42–48

[146] Takagi, T. ; Sugeno, M.: Fuzzy Identification of Systems and its Application to Modeling and Control. In: *IEEE Transactions on Systems, Man, and Cybernetics* 15 (1985), Nr. 1, S. 116–132

[147] Teslic, L. ; Hartmann, B. ; Nelles, O. ; Skrjanc, I.: Nonlinear System Identification by Gustafson–Kessel Fuzzy Clustering and Supervised Local Model Network Learning for the Drug Absorption Spectra Process. In: *IEEE Transactions on Neural Networks* 22 (2011), Nr. 12, S. 1941–1951

[148] Tibshirani, R.: Regression Shrinkage and Selection via the Lasso. In: *Journal of the Royal Statistical Society. Series B (Methodological)* 58 (1996), Nr. 1, S. 267–288

[149] Tikhonov, A.N. ; Arsenin, V.Y.: *Solutions of Ill-Posed Problems*. New York : Wiley, 1977

[150] Töpfer, S.: *Hierarchische neuronale Modelle für die Identifikation nichtlinearer Systeme*. Düsseldorf, Diss., 2002

[151] Töpfer, S. ; Martini, E. ; Kunde, C.: Stationäre Motorvermessung mit neuronalen Netzen. In: Isermann, R. (Hrsg.): *Modellgestützte Steuerung, Regelung und Diagnose von Verbrennungsmotoren*. Springer, 2003, Kapitel 8, S. 131–152

[152] Tsekouras, G. ; Sarimveis, H. ; Kavakli, E. ; Bafas, G.: A hierarchical fuzzy-clustering approach to fuzzy modeling. In: *Fuzzy Sets and Systems* 150 (2005), Nr. 2, S. 245–266

[153] Tutmez, B. ; Hatipoglu, Z.: Comparing two data driven interpolation methods for modeling nitrate distribution in aquifer. In: *Ecological Informatics* 5 (2010), Nr. 4, S. 311–315

[154] Wallet, B.C. ; Marchette, D.J. ; Solka, J.L. ; Wegman, E.J.: A genetic algorithm for best subset selection in linear regression. In: *Computing Science and Statistics* (1997), S. 545–550

[155] Wang, L.-X. ; Mendel, J.M.: Fuzzy Basis Functions, Universal Approximation, and Orthogonal Least-Squares Learning. In: *IEEE Transactions on Neural Networks* 3 (1992), September, Nr. 5, S. 807–814

[156] Wang, L.X.: Analysis and design of hierarchical fuzzy systems. In: *IEEE Transactions on Fuzzy Systems* 7 (1999), Nr. 5, S. 617–624

[157] Wikipedia: *Structural Health Monitoring*. `http://de.wikipedia.org/wiki/Structural_Health_Monitoring`. – (abgerufen am 30.05.2013)

[158] Wu, B. ; Yu, X.: Fuzzy modelling and identification with genetic algorithm based learning. In: *Fuzzy Sets and Systems* 113 (2000), Nr. 3, S. 351–365

[159] Zimmerschied, R.: *Identifikation nichtlinearer Prozesse mit dynamischen lokal-affinen Modellen*, Technische Universität Darmstadt, Diss., 2008

[160] Zou, H. ; Hastie, T.: Regularization and variable selection via the elastic net. In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 67 (2005), Nr. 2, S. 301–320