UNIVERSITÄT DER BUNDESWEHR MÜNCHEN
Fakultät für Elektrotechnik und Informationstechnik

Dynamische Kompensation von nichtlinearen Verzerrungen mit genetischer Programmierung

Xuhui Li

Vorsitzender des Promotionsausschusses:	Prof. Dr.-Ing. K. Landes
1. Berichterstatter:	Prof. Dr.-Ing. K. Tröndle
2. Berichterstatter:	Prof. Dr.-Ing. U. Appel

Tag der Prüfung: 22.07.2003

Mit der Promotion erlangter akademischer Grad:
Doktor-Ingenieur (Dr.-Ing.)

München, den 12. August 2003

Vorwort

Die vorliegende Arbeit entstand während meiner Tätigkeit als wissenschaftlicher Mitarbeiter am Institut für Informationstechnik der Universität der Bundeswehr München.

Diese Zeit war für mich ein großer Gewinn. Ich möchte allen Mitarbeitern und Kollegen für das hervorragende Arbeitsklima am Institut danken.

Mein besonderer Dank gilt meinem Doktorvater Herrn Professor Dr.-Ing. Karlheinz Tröndle für die Betreuung dieser Arbeit.

München, im April 2003

Xuhui Li

Zusammenfassung

In der vorliegenden Arbeit werden zwei Identifikations- bzw. Kompensationsverfahren von nichtlinearen Systemen unter der Anwendung der GP vorgestellt, nämlich das rekursive GP-Verfahren und der Kombinationsalgorithmus.

Diese Arbeit konzentriert auf die Entwicklung und die Simulation des Kombinationsalgorithmus, der auf dem Wiener-Modell basiert.

Dieser Kombinationsalgorithmus ist sehr leistungsfähig in der Identifikation bzw. Kompensation von nichtlinearen Systemen, die mit einem Wiener-Modell hinreichend gut beschreibbar sind.

Abb. 1 veranschaulicht den Algorithmus. Mit Hilfe der GP wird das Ziel-System durch ein Wiener-Modell nachgebildet. Die größte Besonderheit dieses Algorithmus liegt darin, dass die GP die Struktur des NL-Anteils des Wiener-Modells automatisch suchen kann.

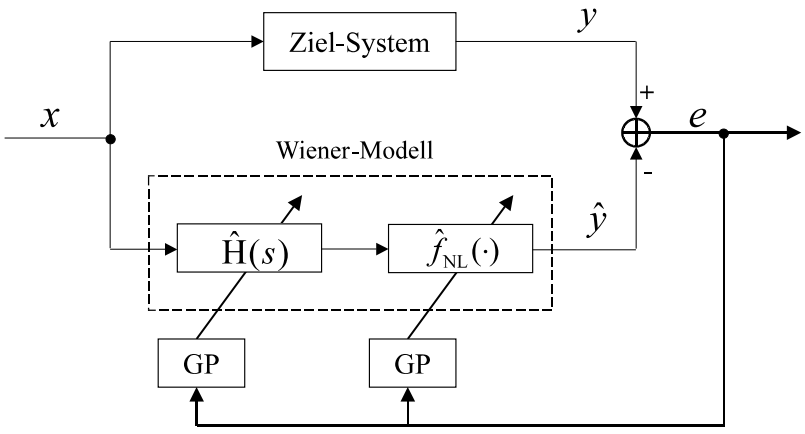


Abbildung 1: Veranschaulichung des Kombinationsalgorithmus

Im Wesentlichen wird der Kombinationsalgorithmus durch die folgenden Leistungen gekennzeichnet:

1. Hohe Identifikationsgenauigkeit des Wiener-Modells.
2. Niedriger Rechenaufwand des Wiener-Modells.
3. Hohe Kompensationsgenauigkeit.
4. Niedriger Kompensationsrechenaufwand.

Die erste Aufgabe (Identifikation des Ziel-Systems) wird durch einen Iterationsprozess mit Hilfe der GP und der Konstanten-Optimierung ausgeführt. Wenn der NL-Anteil, der von der GP in der ersten Aufgabe gefunden wird, kompliziert ist, wird die zweite Aufgabe (Vereinfachung des Modells) durch einen Quotienten aus zwei Polynomen mit Hilfe der GP gelöst. Die dritte Aufgabe (Kompensation des NL-Anteils) wird durch die GP mit Hilfe der abschnittsweisen Optimierung (AO) sowie der abschnittsweisen Identifikation (AI) bzw. der abschnittsweisen Kompensation (AK) implementiert. Dabei werden mehrere Kompensatoren entstehen, die zu den verschiedenen Ausgangswertebereichen des Ziel-Systems passen. Wenn die Kompensatoren kompliziert sind, wird die vierte Aufgabe (Vereinfachung der Kompensatoren) durch die GP mit Hilfe des Quotienten aus zwei Polynomen fertiggestellt.

Die Abb. 2 veranschaulicht den dynamischen Kompensationsverlauf durch mehrere Kompensatoren. Dabei verteilt der Entscheider das Ausgangssignal des Ziel-Systems auf einen geeigneten Kompensator gemäß der Größe des Ausgangssignals des Ziel-Systems.

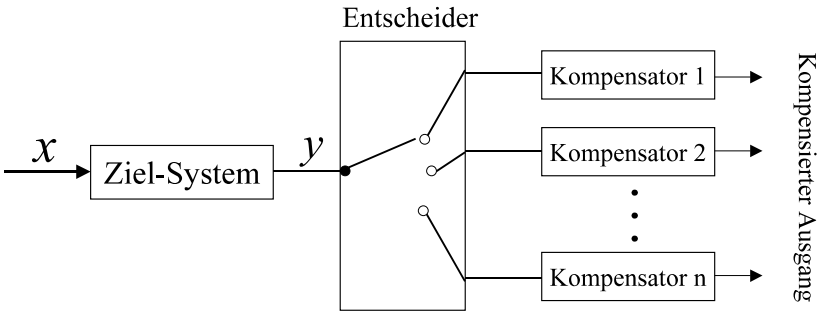


Abbildung 2: Beschreibung der dynamischen Kompensation

Der Kombinationsalgorithmus ist auch für Hammerstein-Modelle geeignet. Bei der Identifikation wird ein Hammerstein-Modell an Stelle des Wiener-Modells eingesetzt. Es wird dann die Pre-Kompensation verwendet.

Um den Kombinationsalgorithmus zu testen, werden einige Beispiele demonstriert, z.B. Ziel-Systeme mit unterschiedlichen Nichtlinearitäten und Ziel-Systeme mit verschiedenen Sättigungskennlinien.

Durch die Ergebnisse der Simulation des Kombinationsalgorithmus werden die folgenden Schlussfolgerungen gezogen:

- Der Kombinationsalgorithmus ist geeignet für die Systemidentifikation, wenn das zu identifizierende System mit einem Wiener-Modell exakt oder hinreichend genau beschrieben werden kann.

- Die Genauigkeit dieses Kombinationsalgorithmus ist nicht von der Nichtlinearität des nichtlinearen Ziel-Systems sondern nur von der Beschreibungsgenauigkeit des Wiener-Modells für das Ziel-System abhängig.
- Die Genauigkeit dieses Kombinationsalgorithmus ist stark von der Beschreibungsgenauigkeit des Wiener-Modells abhängig.
- Die Genauigkeit dieses Kombinationsalgorithmus ist um einen wesentlichen Faktor (5–300) besser als die Genauigkeit des rein linearen Modells mit vergleichbarem Rechenaufwand.
- Der Kombinationsalgorithmus kann sehr hohe Modellierungsgenauigkeiten (10^{-3}–10^{-4}) mit einem relativ kleinen Rechenaufwand erreichen, wenn der NL-Anteil des Ziel-Systems ein Sättigungssystem ist. Er bildet daher eine wichtige Ergänzung für konventionelle Volterra-Modelle.
- Wenn der Kombinationsalgorithmus einen komplizierten NL-Anteil des Wiener-Modells findet, kann die GP durch eine Post-Verarbeitung die NL-Kennlinie auch durch einen Quotienten aus zwei Polynomen darstellen, ohne dass ein wesentlicher Verlust für die gesamte Modellierungsgenauigkeit entsteht.
- Der Kombinationsalgorithmus ist ungeeignet für nichtlineare Ziel-Systeme, die mit einem Wiener-Modell nicht hinreichend gut beschrieben werden können.

Inhaltsverzeichnis

Kapitel 1

Einleitung

Die lineare Systemtheorie ist gut erforscht und bildet eine wichtige Grundlage jedes Elektroingenieurstudiums. Die Linearität bedeutet, dass ein Eingangssignal $x(t) = ax_1(t) + bx_2(t)$ ein Ausgangssignal $y(t) = ay_1(t) + by_2(t)$ erzeugt, also

$$y(t) = \varphi\{ax_1(t) + bx_2(t)\} = ay_1(t) + by_2(t) \tag{1.1}$$

für alle zulässigen Werte x_1, x_2 gilt. Hierbei sind a und b beliebige Konstanten und

$$\varphi\{x_1(t)\} = y_1(t), \varphi\{x_2(t)\} = y_2(t). \tag{1.2}$$

Wenn der Operator φ diese Bedingung erfüllt, heißt das System lineares System.

Für die Identifikation des LZI-Systems stehen viele Verfahren wie z.B. AR, ARX, MA, MAX, ARMA, OE, BJ, FIR, IIR Modelle [Ljung87], [Ljung98], [Nelles00] zur Verfügung. Diese Verfahren sind in der Matlab Identifikations-Toolbox implementiert.

In der Tat ist aber die Linearität eine mathematische Approximation, d.h. es gibt kaum *echt* lineare Systeme in der Natur. Mehr oder weniger weisen fast alle praktischen Systeme eine Nichtlinearität auf.

Die Anforderungen der praktischen Anwendung bestimmen die Trennlinie zwischen der linearen und der nichtlinearen System-Modellierung.

Ein System muss nichtlinear modelliert werden, wenn es durch lineare Methoden nicht mehr hinreichend genau beschrieben werden kann.

Die Nichtlinearität besitzt zwei ganz unterschiedliche Wirkungen in der Anwendungen:

1. die Nichtlinearität ist eine *Nebenwirkung*, d.h. sie ist unerwünscht. z.B. bei Sensoren, Aktoren und Übertragungskanälen. In diesen Fällen sollte man beim Design oder Aufbau die Nichtlinearität so gut wie möglich *vermeiden*.

2. die Nichtlinearität spielt eine *wichtige* Rolle im System, d.h. die Nichtlinearität wird unbedingt benötigt, sie stellt das wichtige oder wesentliche Verhalten des Systems dar, z.B. in modernen nichtlinearen Steuerungen (um höhere Steuerungsgenauigkeit zu erreichen), bei Solitonen in der Lichtwellenleitungskommunikation (um eine höhere Kommunikationsgeschwindigkeit zu erreichen) oder in Frequenzmultiplikatoren . In diesen Fällen möchte man die Nichtlinearität des Systems ausnutzen.

Der erste Fall der Nebenwirkung, ist derjenige, bei dem eine Kompensation der nichtlinearen Verzerrungen erforderlich ist, wobei aber weder beim Design noch beim Aufbau die Nichtlinearität betrachtet wird, sondern bei der Kompensation mittels digitaler Signalverarbeitung. In [Frank97, Kafka02] wird dazu ein Entwurf entwickelt, der auf dem Volterra-Reihen Modell [Schetzen80] basiert.

Der zweite Fall entspricht der Identifikation eines nichtlinearen Systems. Der zweite Fall bildet in gewisser Weise die Basis zur Implementierung des ersten Falls, weil der Kompensator ein nichtlineares System sein muss.

Ein nichtlineares Modell besteht aus zwei Teilen: der Modell-Struktur und den entsprechenden Modell-Parameter. Zur Identifikation eines nichtlinearen Systems mit Gedächtnis stehen viele klassische aber auch moderne Verfahren zur Verfügung.

Die typischen klassischen Verfahren sind z.B. NARX, NARMAX, NOE, NJB, NFIR, NIIR, Volterra-Reihen, Kolmogorov-Polynome usw. [Sjoberg95a, Ljung92, Giancarlo01, Basso02]. Die typischen modernen Verfahren sind z.B. Neuronale Netze(NN), Fussy-Modelle, Neuron-Fussy Modelle, Wavelet Transformator usw. [Sjoberg95b, Pham99, Nelles01, Mallat98, Sureshbabu95].

Fast alle klassischen und modernen Identifikations-Verfahren gehören zur sogenannten Black-Box Methode und setzen eine angenommene allgemeine Modell-Struktur voraus. Die Schätzung der Parameter des Modells bildet die Kernaufgabe solcher Methoden.

Wegen der zeitlichen Änderung der inneren Zustände und/oder der äußeren Bedingungen verändern sich die inneren Regeln eines *dynamisches* Systems (Prozesses). Wenn der Prozess über die Zeit läuft, wie z.B. bei der Aktienbörse, Flugprozessen eines modernen Jagdflugzeugs oder bei Sensoren mit Umgebungseinflüssen usw.

Damit ergeben sich die folgenden Frage- bzw. Aufgabenstellungen:

- Wie kann man dynamische Prozesse besser identifizieren?
- Wie kann man dynamische Prozesse besser kompensieren, falls die Kompensation von nichtlinearen Verzerrungen möglich und nötig ist.
- Unter welchen Bedingungen kann man die Identifikation und Kompensation solcher dynamischer Prozesse besser implementieren?

Mit adaptiven Methoden kann man den dynamischen Prozess simulieren. Dabei passt der adaptive Algorithmus normalerweise nur die Parameter des Modells an, aber die Modell-Struktur bleibt fest.

Um einen dynamischen Prozess besser zu simulieren, soll die Modell-Struktur auch dynamisch sein. In dieser Arbeit wird ein anderer Weg versucht, nämlich das Problem mit der Genetischen Programmierung (GP) zu lösen, weil die GP die Modell-Struktur gemäß den Trainingsdaten automatisch ändern kann, um das Modell den Trainingsdaten besser anzupassen.

Die GP ist ein Verfahren zur automatischen Erzeugung von Computerprogrammen [Koza92, Banzhaf98, Koza99]. Die GP ahmt den evolutionären Prozess in der Natur nach. Grundsätzlich könnte die GP bei komplexen Prozess sehr lange laufen. Während eines GP-Laufes werden mehrere Individuen(Programme) erzeugt, deren Güte oder Qualität durch eine vordefinierte Fitnessfunktion und den vorliegenden Trainingsdaten beurteilt wird.

Beim Identifikationsprozess sucht die GP automatisch Schritt für Schritt eine geeignete *Modellstruktur* im vorgegebenen Suchraum, wobei die Modell-Struktur sehr große Freiheitsgrade (beliebige Nichtlinearität) besitzen kann. Grundsätzlich braucht die GP keine streng mathematischen Kenntnisse über das zu identifizierende System. Theoretisch kann die GP die Lösung einer beliebigen Aufgabe finden, solange der Suchraum der GP die Lösung der Aufgabe umfasst.

Wichtige Punkte sind dabei

- Für einen Identifikationsprozess hängt die Güte der Individuen der GP von der Vollständigkeit der Trainingsdaten ab.
- Die Veränderung der Prozessregeln kann durch die Trainingsdaten ausgedrückt werden.
- Die sich dynamisch verändernden neuen Trainingsdaten führen zur Veränderung der Individuen der GP, die sich den neuen Trainingsdaten besser anpassen. D.h. die Evolution der GP entwickelt sich in eine neue Richtung . Die Veränderungen der Trainingsdaten sind oft verursacht von den Systemstrukturänderungen. Die Modellstruktur-Suche ist die Stärke der GP, deswegen passt die GP zur Behandlung solcher dynamischer Systeme.

Die Ziele dieser Arbeit sind:

- Aufbau der Algorithmen zur Identifikation dynamischer nichtlinearer Systeme mit der GP.
- Aufbau von Algorithmen zur Kompensation der dynamischen nichtlinearen Verzerrungen mit der GP.

- Entwicklung und Simulation eines Algorithmus, der auf dem Wiener-Modell basiert. In dieser Arbeit wird er Kombinationsalgorithmus genannt.
- Verbesserung dieses Kombinationsalgorithmus.

Aufbau dieser Arbeit

Im Kapitel 2 wird die Grundlage sowie eine Klassifikation der Systeme als Vorbereitung dargestellt. Dabei wird auch der Aufbau der dynamischen Identifikation und Kompensation vorgestellt.

Im Kapitel 3 werden die GP-Kenntnisse und GP-Methoden als Vorbereitung dargestellt. Durch umfangreiche Analysen und Vergleiche werden einige allgemeine Prinzipien zur Verbesserung der GP-Suchfähigkeit dargestellt.

Kapitel 4 beschäftigt sich mit der Systemidentifikation sowie der Kompensation bei gedächtnislosen nichtlinearen Systemen mit starken Störungen.

Kapitel 5 beschäftigt sich mit der Identifikation von Systemen mit Gedächtnis. Dabei werden einige konkrete Entwürfe zur Verbesserung der GP-Suchfähigkeit dargestellt.

Im Kapitel 6 wird ein auf dem Wiener-Modell basierender Kombinationsalgorithmus entwickelt, der die GP und konventionelle iterative Algorithmen kombiniert. Für den Kombinationsalgorithmus werden drei verschiedene Implementierungsentwürfe vorgestellt.

Kapitel 7 beinhaltet einige Anwendungsbeispiele.

Kapitel 2

Grundlagen der Systeme

2.1 System und Modell

Definition 2.1: *Ein konkretes oder konkret vorstellbares Gebilde, welches ein beobachtbares Verhalten zeigt, wobei dieses Verhalten als Ergebnis des Zusammenwirkens seiner Teile angesehen werden kann, heißt System.*

Ein System kann als eine Vorschrift, durch die einer Anregung x eine Antwort y zugeordnet wird, definiert werden. Die Vorschrift kann durch ein Symbol $\mathbf{T}$ bezeichnet werden, wie dies in Gl. (2.1) gezeigt wird.

$$y = \mathrm{T}(x) \tag{2.1}$$

In dieser Beschreibung wird $\mathbf{T}$ ein Operator genannt. Die Darstellung, dass y die Antwort des Systems auf x ist, bedeutet, dass eine Vorschrift $\mathbf{T}$ *existieren muss*, durch die einem beliebigen zulässigen x ein *einziges* y zugeordnet werden kann. Abb. 2.1 zeigt diesen Zusammenhang der Zuordnung.

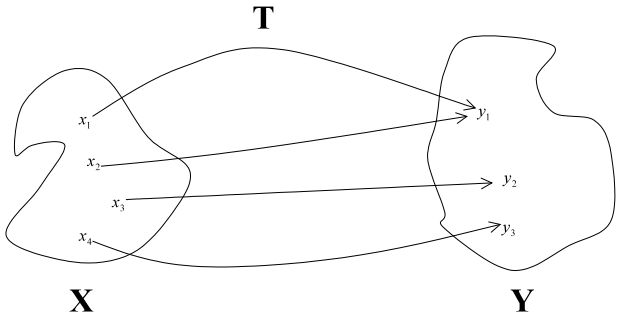


Abbildung 2.1: Zuordnung der Menge der Antworten zur Menge der Anregungen durch den Operator $\mathbf{T}$.

Hierbei wird $\mathbf{X}$ als die Menge der Anregungen und $\mathbf{Y}$ als die Menge der Antworten bezeichnet. Der Operator $\mathbf{T}$ muss folgende Merkmale besitzen:

Die Zuordnung muss Eins-zu-Eins oder Mehrere-zu-Eins sein. Im Gegensatz dazu ist Eins-zu-Mehreren ausgeschlossen.

In der Praxis nennt man den interessierenden Teil eines Gebildes das System und im Gegensatz dazu die anderen Teile die Umgebung des Systems. Ein System könnte ein technisches System, ein Biosystem, ein Ökosystem usw. sein. Die Umgebung könnte z.B. die Störung sein. Im Allgemeinen besitzt ein System mindestens einen Ein- und einen Ausgang, wie dies die Abb. 2.2 zeigt.

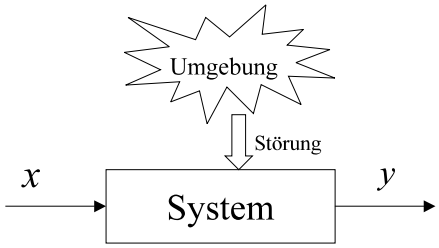


Abbildung 2.2: Zusammenhang zwischen dem System und seiner Umgebung. Die Umgebung beeinflusst das System normalerweise als Störung.

Definition 2.2 *Ein Modell ist ein Gebilde, welches anstelle des eigentlich interessierenden Originals gestellt wird und mit diesem in wesentlichen Eigenschaften übereinstimmt aber auch andere (unwesentliche) Eigenschaften aufweist.*

Wie in Abb. 2.1 angedeutet, kann das System als ein Zuordnungszusammenhang **T** zwischen dem Ein- und Ausgang *symbolisch* definiert werden. Der Zuordnungszusammenhang **T** kann sehr einfach oder sehr kompliziert sein. Durch den Zuordungszusammenhang **T** können die Verhaltensweisen des Systems vollständig beschrieben werden.

Normalerweise ist der echte Zusammenhang **T** nicht ergründbar, wenn das System relativ kompliziert ist. In der Tat interessiert man sich auch kaum für den echten Zusammenhang **T**. In der Praxis benötigt man ein Näherungsmodell, das die wesentlichen Verhaltensweisen des Systems beschreibt.

Je nach den unterschiedlichen gestellten Forderungen[1] kann das Modell linear oder nichtlinear sein. Durch das mathematische Modell kann man das System untersuchen, um z.B. ein System besser zu verstehen oder besser zu steuern. Das Modell zu gewinnen ist die Aufgabe der Modellierung.

[1] Man kann mit einem linearen Modell ein nichtlineares System modellieren, wenn die Forderung der Genauigkeit nicht hoch ist oder die Nichtlinearität des Systems nicht stark ist.

Die in dieser Arbeit behandelten Systeme werden der Einfachheit halber nur als kausale, passive Systeme mit einem Eingang und einem Ausgang angenommen. Ein passives System ist ein System, das kein Ausgangssignal abgibt, wenn kein Eingangssignal vorliegt. Es erzeugt also z.B. keinen konstanten Offset. Die Kausalität bedeutet, dass das Ausgangssignal nicht von zukünftigen Werten des Ein- und Ausgangssignals abhängt. Die Kausalität ist besonders wichtig bei der Kompensation von nichtlinearen Verzerrungen .

Die Reduktion auf einen Ein- und einen Ausgang (SISO[2]) ist keine echte Beschränkung, weil ein System mit mehreren Ein- und Ausgängen(MIMO[3]) aus mehreren MISO[4] Systemen zusammengesetzt werden kann, wie dies in Abb. 2.3 gezeigt wird.

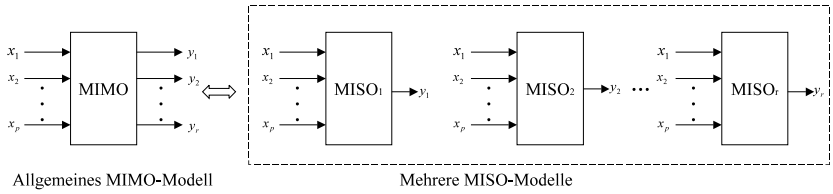


Abbildung 2.3: Ein MIMO-Modell kann aus mehreren MISO-Modellen zusammengesetzt werden.

Obwohl in der Mathematik die Erweiterung von SISO zu MISO nicht einfach ist, gibt es dennoch keinen wesentlichen Unterschied zwischen MISO und SISO bei der GP. Tatsächlich löst die GP immer die *Quasi*-MISO Aufgabe bei der Identifikation eines Systems mit Gedächtnis, wie dies in Abb. 2.4 gezeigt wird.

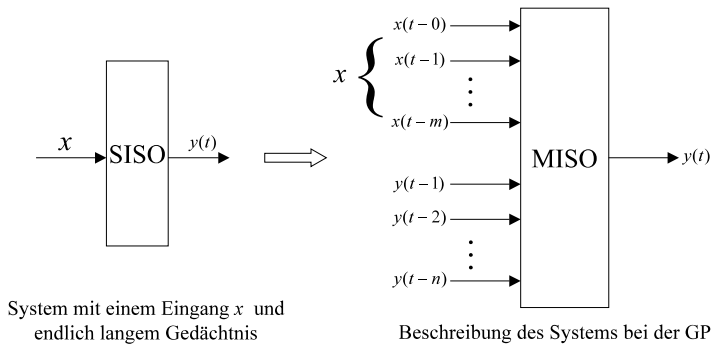


Abbildung 2.4: Identifikation eines Systems mit Gedächtnis.

[2]Single Input Single Output
[3]Multiple Input Multiple Output
[4]Multiple Input Single Output

Ein System mit *einem* Eingang x und *Gedächtnis* ist hinsichtlich der Beschreibung ähnlich wie MISO in der GP, weil der aktuelle Eingangswert, die vergangenen Eingangswerte und die vergangenen Ausgangswerte als unabhängige Variablen in der Terminalmenge der GP betrachtet werden können.

2.2 Systemidentifikation und Modellierung

Definition 2.3: *Identifikation ist das Bilden eines mathematischen Modells von einem vorliegenden Ziel-System, indem man*

- *die Eingangs- und Ausgangssignale des vorliegenden Ziel-Systems beobachtet,*
- *und anschließend ein Modell sucht, das das beobachtete Systemverhalten am besten beschreibt.*

Systemidentifikation ist ein Verfahren, ein Modell zu gewinnen, das ein gegebenes System hinreichend gut beschreiben kann. Systemidentifikation bedeutet, für ein gegebenes unbekanntes System ein Modell zu suchen, mit dem das gegebene System hinreichend *Äquivalent* nachgeahmt werden kann. Dabei bedeutet *Äquivalent* normalerweise, dass der mittlere quadratische Fehler zwischen dem Ausgang des Modells und dem Ausgang des gegebenen Systems kleiner als ein vorgegebener Wert ε ist, nämlich $\int e^2(t)dt \leq \varepsilon$, dabei ist $e(t) = y(t) - \widehat{y}(t)$, wie dies die Abb. 2.5 zeigt.

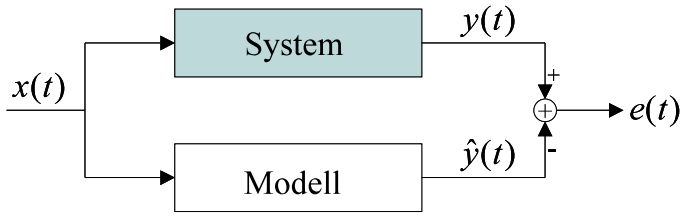


Abbildung 2.5: System und sein äquivalentes Modell

Systemidentifikation ist die *experimentelle* Ermittlung des zeitlichen Verhaltens eines Systems. (Im Gegensatz dazu ist die Modellierung die *theoretische* Ermittlung.) Sie gehört zu den *datengetriebenen* Verfahren, d.h. sie beruht auf den Ein- und Ausgangsdaten des Systems. Bei der Systemidentifikation gibt es im wesentlichen zwei Aufgaben:

1. Bestimmung der Struktur des Modells.
2. Schätzung der Parameter des Modells.

Die konventionellen Identifikationsverfahren basieren normalerweise auf einer "bekannten" Struktur, d.h. zunächst muss eine Modell-Struktur vorgegeben werden (z.B. durch theoretische Analyse) oder *angenommen* werden (z.B. für neuronale Netze und Fussy-Modell), dann schätzt man durch Betrachtung der Ein- und Ausgangsdaten des Systems die Parameter des Modells. Wenn die Ergebnisse nicht gut genug übereinstimmen, kann man die Struktur adaptieren oder neue Strukturen suchen/annehmen (z.B. die Regeln des Fussy-Modells, die verborgenen Schichten der neuronalen Netze oder die Ordnung der Polynom-Modelle verändern).

Systemidentifikation ist ein Versuchsprozess.

Systemidentifikation ist im allgemeinen ein iterativer Vorgang mit folgenden Schritten: Strukturbestimmung →Parameterschätzung →Modellauswertung→ neue Strukturbestimmung→ neue Parameterschätzung→ ···. Dieser Vorgang läuft bis die Forderung an die Genauigkeit erfüllt ist, wie dies die Abb. 2.6 (b) zeigt.

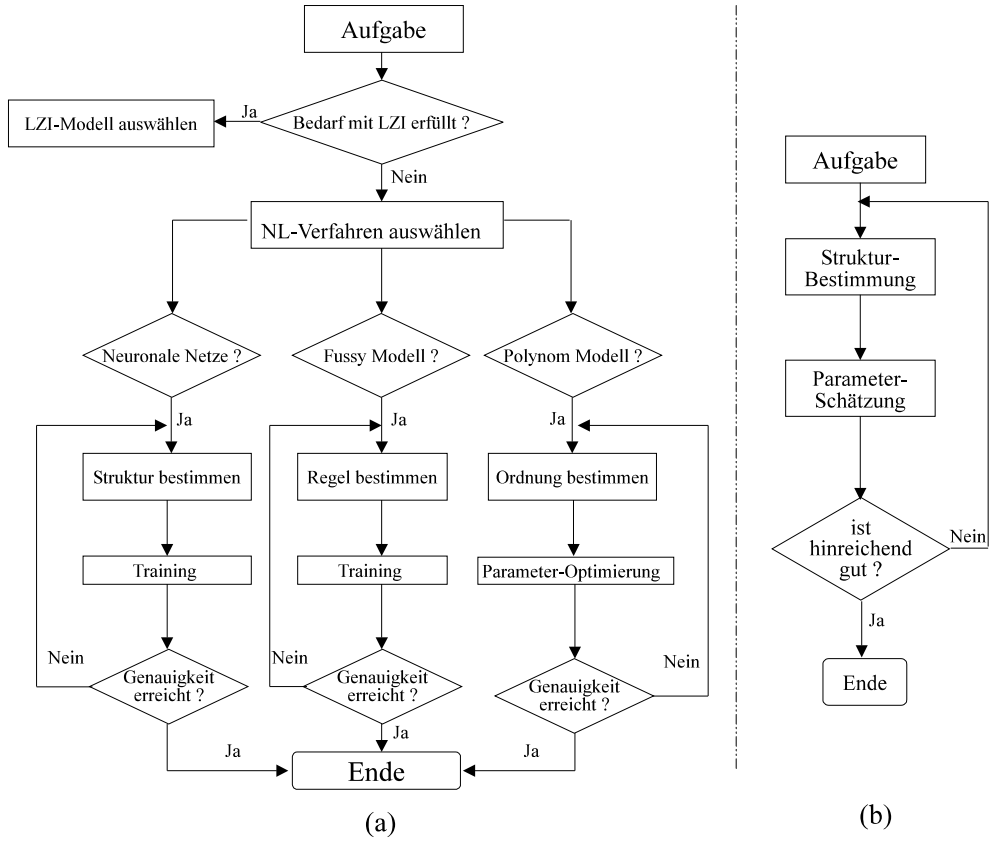


Abbildung 2.6: (a)–Allgemeiner Ablauf der Systemidentifikation mit neuronalen Netzen, Fussy-Modellen und Polynom-Modellen. (b)–Vereinfachter Ablauf der Systemidentifikation.

Jedenfalls muss man zunächst die erste Aufgabe, die Struktur zu bestimmen, bearbeiten, erst dann kann man die zweite Aufgabe anfangen. Abb. 2.6 (a) zeigt den allgemeinen Ablauf bei der Systemidentifikation mit neuronalen Netzen, Fussy-Modellen und Polynom-Modellen. Je nach den Anforderung der Aufgabe (z.B. Genauigkeit) oder dem Vorwissen über das System entscheidet man zunächst, ob ein LZI-Modell verwendet werden kann. Wenn die Genauigkeit mit LZI-Modellen nicht erfüllbar ist, *muss* man die Lösung mit NL-Modellen suchen.

Bei den NL-Modellen entscheidet man zunächst, ob neuronale Netze, Fussy-Modelle oder Polynom-Modelle verwendbar sind. Wenn eines der NL-Verfahren ausgewählt ist, muss man zunächst die Modell-Struktur für das Verfahren bestimmen, anschließend werden die Parameter des Modells durch bestimmte Algorithmen ermittelt oder optimiert.

Dabei sind die Struktur der neuronalen Netze, die Regeln des Fussy-Modells und die Ordnung des Polynom-Modells so zu verstehen, dass sie die gleiche Bedeutung besitzen, nämlich die *"Modell-Struktur"* des entsprechenden NL-Verfahrens zu bestimmen. Auch die Trainingsphase und die Parameter-Optimierung besitzen die gleiche grundsätzliche Bedeutung, nämlich die möglichst optimale Bestimmung der Parameter des Modells.

Wegen der Besonderheit der GP ist dieser zweistufige Ablauf bei der GP gemischt, d.h. die GP sucht gleichzeitig die Struktur des Modells und die entsprechenden Parameter. Dies ist einerseits der Vorteil der GP, weil die GP in diesem Ablauf neue Strukturen[5] des Modells finden kann, anderseits ist dies aber auch ein Nachteil, weil die GP in diesem Ablauf mehr Rechenaufwand benötigt. Außerdem erniedrigt die Struktursuche auch die Suchfähigkeit für die numerischen Parameter. Durch konventionelle Optimierungsverfahren (z.B. kleinste Quadrate) kann man diesen Nachteil jedoch vermeiden.

Definition 2.4: *Die Modellierung ist das Bilden eines mathematischen Modells eines Systems durch die Anwendung der physikalischen Gesetze, gemäß derer das System sich verhält.*

Die Modellierung ist ein *klassisches* Verfahren, ein Modell zu gewinnen. Sie basiert auf der Analyse der physikalischen (mechanischen oder elektronischen) Strukturen des Systems und der physikalischen Gesetze, die diese beschreiben. Die Unterschiede zwischen der Systemidentifikation und der Modellierung sind:

1. Die Systemidentifikation beruht auf der Analyse der Ein- und Ausgangswerte

[5]Dadurch ergeben sich noch andere Probleme, z.B. Overfitting, weil die von der GP erzeugten Strukturen sehr kompliziert gegenüber den idealen Strukturen sein können. Durch die Methoden, die wir im Abschnitt 2.2.2 vorstellen werden, kann dies größtenteils bewältigt werden. Außerdem zeigt die GP noch, dass die GP für sehr komplizierte Aufgaben kaum gute Ergebnisse liefert. Also ist die Aufspaltung der komplizierten Aufgabe in kleinere Aufgaben sehr wichtig, um damit die Basissuchfähigkeit der GP anzupassen. Kapitel 6 beschreibt diese Idee.

des Systems.

2. Die Modellierung muss nicht auf Ein- und Ausgangswerte des Systems zurückgreifen, obwohl die Bestimmung mancher Parameter des Modells möglicherweise von diesen Werten abhängig ist.

3. In der Systemidentifikation verwendet man eine allgemeine Modellstruktur. D.h. die Modellstruktur und die Struktur des Systems müssen nicht identisch sein.

4. Bei der Modellierung wird die Modellstruktur aus der physikalischen Analyse des Systems abgeleitet, also sind die Modellstruktur und die Systemstruktur normalerweise identisch.

5. Das Modell, das aus der Modellierung hervorgeht, besitzt mehr Treffsicherheit und erfasst die Systemeigenschaften insgesamt besser als das Modell aus der Systemidentifikation.

2.2.1 Simulation und Prädiktion

Definition 2.5: *Simulation ist das systematische Erproben des Verhaltens von geplanten, sich in der Entwicklung befindlichen oder bereits existierenden Systemen. Dabei wird ein Simulationsmodell zugrundegelegt, welches die für die Simulation relevanten Aspekte des Systems nachbildet.*

Definition 2.6: *Prädiktion ist eine Verquickung der vorherigen verfügbaren Daten bis zum Zeitpunkt t zur Vorhersage von zukünftigen Werten zu verschiedenen Zeitpunkten t+1,t+2,t+3,... .*

Für die Systemidentifikation gibt es zwei unterschiedliche Fehlertypen zwischen dem System und dem Modell, einen Simulationsfehler und einen einstufigen Prädiktionsfehler. In Abb. 2.5 definierten wir nur $\varepsilon = \int e^2(t)dt$, und $e(t) = y(t) - \widehat{y}(t)$ für die Identifikation. Dabei besitzt $\widehat{y}(t)$ zwei verschiedene Formen je nach der Forderung, welche durch die Aufgabe vorgegen ist, nämlich die System-Simulation oder die System-Prädiktion, wie die Gleichungen (2.2) und (2.3) es beschreiben.

$$\widehat{y}_s(t) = \widehat{f}_s(x(t), x(t-1), x(t-2)..., \widehat{y}_s(t-1), \widehat{y}_s(t-2), ...) \tag{2.2}$$

$$\widehat{y}_p(t) = \widehat{f}_p(x(t), x(t-1), x(t-2)..., y(t-1), y(t-2), ...) \tag{2.3}$$

$$\widehat{y}_{pl}(t) = \widehat{f}_{pl}(x(t), x(t-1), x(t-2)..., y(t-l), y(t-l-1), ...) \tag{2.4}$$

Die Gl. (2.2) zeigt die Situation der Simulation. Dabei ist der Ausgang des Modells *nur* von den Eingangswerten des Systems aber nicht von den Ausgangswerten[6] des

[6]Aber es ist möglich, dass der Ausgang des Modells von den vergangenen Ausgangswerten des Modells abhängig ist.

Systems abhängig. In diesem Fall wird der Restfehler ε als sogenannter Simulationsfehler ε_s bezeichnet.

Die Gl. (2.3) zeigt die Situation der Prädiktion. Dabei ist der Ausgang des Modells nicht nur von den Eingangswerten sondern auch von den Ausgangswerten des Systems abhängig. In diesem Fall wird der Restfehler ε als ε_p bezeichnet und Prädikationsfehler genannt. Genauer gesagt, heißt er einstufiger Prädikationsfehler. Außerdem zeigt Gl.(2.4) die mehrstufige Prädikation, und der Fehler heißt entsprechend mehrstufiger Prädiktionsfehler.

Dabei ist ersichtlich, dass die Prädiktion als ein Sonderfall der Simulation betrachtet werden kann. Wenn die Prädikationsstufe $l \longrightarrow \infty$ geht, geht die Prädiktion in die Simulation über. Im allgemeinen ist der Restfehler der Simulation viel größer als der Restfehler der einstufigen Prädiktion. Außerdem gibt es bei der Simulation noch ein großes Problem, nämlich das Stabilitätsproblem[7] des Modells wegen der Akkumulierung des Fehlers. Mit übereinstimmendem Modell (für Simulation und Prädiktion) zeigt die Abb. 2.7 den Unterschied zwischen der einstufigen Prädiktion und der Simulation in normalen Fällen.

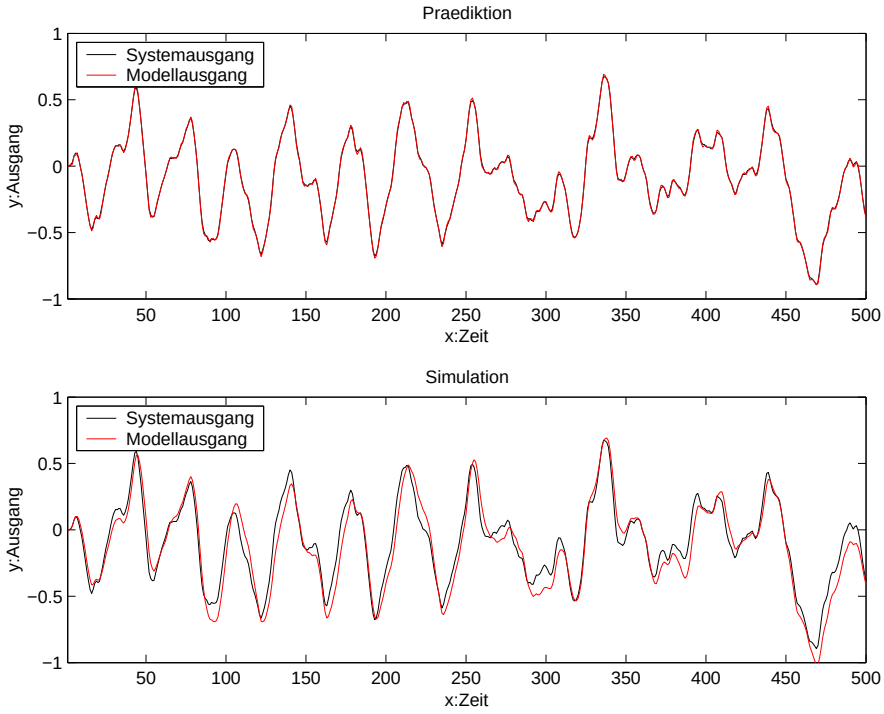


Abbildung 2.7: Einstufige Prädiktion und Simulation. Dabei ist das Modell stabil, aber der Simulationsfehler ist viel größer als der Prädiktionsfehler.

[7] Zu diesem Problem gibt es keine allgemeine theoretische Lösung für allgemeine nichtlineare Systeme.

Bei übereinstimmendem Modell für Simulation und Prädiktion zeigt Abb. 2.8 einen besonderen Signalverlauf. In diesem Fall (Abb. 2.8) ist das Ergebnis der einstufigen Prädiktion sehr gut, aber das Ergebnis der Simulation sehr schlecht, weil das Modell für die Simulation nicht stabil ist.

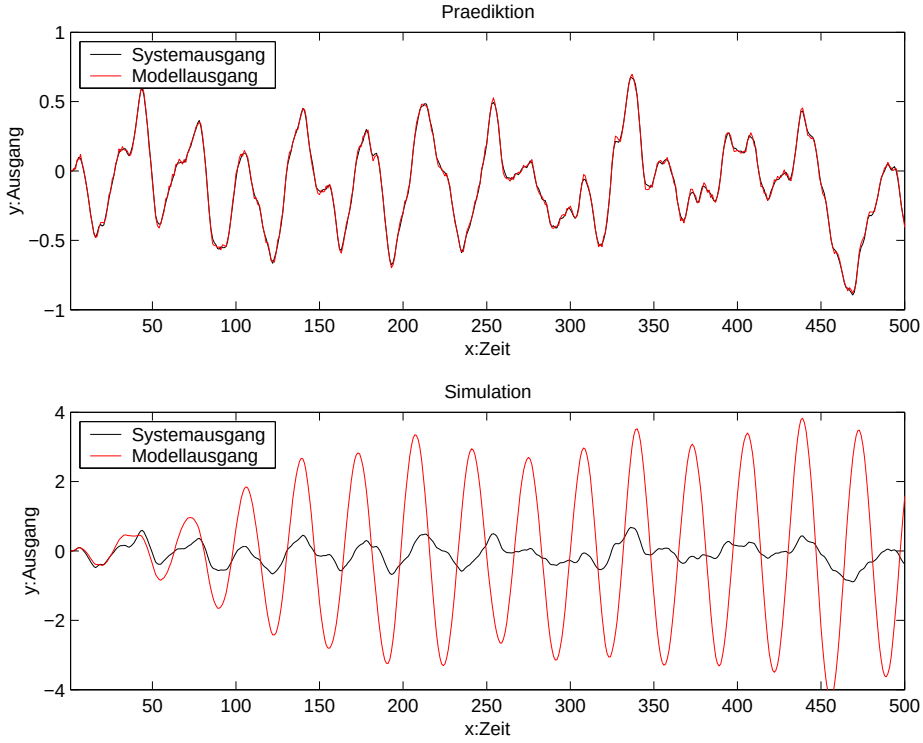


Abbildung 2.8: Einstufige Prädiktion und Simulation–wobei das Modell nicht stabil ist.

In dieser Arbeit ist der Zweck der Identifikation des Systems die Simulation des Systems. Also werden alle Restfehler sowie normierte Restfehler mit dem Simulationsfehler $e_s(t)$ berechnet.

Um ein gutes Ergebnis zu bekommen, verwendet man bei der Fitnessberechnung den einstufigen Prädiktionsfehler.

Bei der GP wird der Start der Evolution sehr erschwert, wenn man von Anfang an, d.h. bereits bei den ersten Generationen den Simulationsfehler verwendet. Demgegenüber ist es viel günstiger für die GP, wenn für die Berechnung der Fitness der Prädiktionsfehler gemäß Gl. (2.3) verwendet wird. Wenn dann gegen Ende des GP-Prozesses vom Prädiktionsfehler wieder auf den Simulationsfehler für die Fitnessberechnung über gegangen wird, erzielt man wesentlich bessere Ergebnisse. Bei den hier durchgeführten Untersuchungen wurde daher von folgender Fehlerberechnung ausgegangen:

1. Bei der Fitnessberechnung wird der Prädiktionsfehler verwendet, um die Evolution der GP zu verstärken.

2. Bei der Konstanten-Optimierung wird der Simulationsfehler verwendet, um den kleinsten Simulationsfehler zu finden.

2.2.2 Training, Validierung und Test

Obergrenze der Genauigkeit des Modells

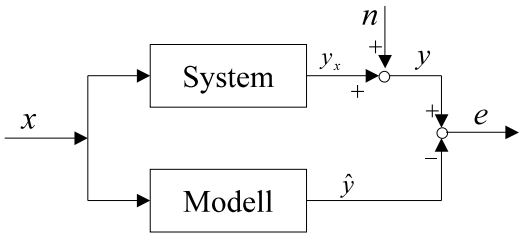


Abbildung 2.9: Systemidentifikation mit Rauschen, wobei y_x der nicht zugängliche Ausgang des Systems ist. y ist der Ausgang des Systems mit Rauschen.

Das rauschfreie System stellt eine ideale Situation dar. In der Praxis sind die Systeme jedoch mehr oder weniger verrauscht. Abb. 2.9 zeigt ein System mit Rauschen, wobei y_x den unmessbaren Ausgang des Systems ohne Rauschen, $\widehat{y}$ den Ausgang des Modells und y den messbaren Ausgang des Systems mit Rauschen n darstellt. Dabei soll gelten $\mathrm{E}\{n\}$=0. Also gilt die folgende Gleichung:

$$\mathrm{E}\{e^2\} = \mathrm{E}\{(y-\widehat{y})^2\} = \mathrm{E}\{(y_x+n-\widehat{y})^2\} = \mathrm{E}\{(y_x-\widehat{y})^2\} + \mathrm{E}\{n^2\} \tag{2.5}$$

Dabei steht E für den Erwartungswert. Der erste Term auf der rechten Seite in der Gl. (2.5) stellt den Fehler zwischen dem echten (unmessbaren) Ausgang des Systems und dem Ausgang des Modells dar, er heißt der Modell-Fehler. Der zweite Term stellt die Varianz des Rauschens dar. Wenn das Modell das System exakt beschreiben kann, d.h. $\widehat{y} = y_x$ ist, also wenn der Modell-Fehler Null ist, erreicht $\mathrm{E}\{e^2\}$ sein Minimum, d.h. $\mathrm{E}\{e^2\}$=$\mathrm{E}\{n^2\}$. Weil der Term $\mathrm{E}(n^2)$ für vorgegebenes Rauschen ein Konstante ist, wird nur der erste Term $\mathrm{E}\{(y_x-\widehat{y})^2\}$ im folgenden weiter verarbeitet.

$$\begin{aligned}
\mathrm{E}\{(y_x-\widehat{y})^2\} &= \mathrm{E}\{[y_x-\mathrm{E}\{\widehat{y}\}-(\widehat{y}-\mathrm{E}\{\widehat{y}\})]^2\} \\
&= \mathrm{E}\{[y_x-\mathrm{E}\{\widehat{y}\}]^2\} + \mathrm{E}\{[\widehat{y}-\mathrm{E}\{\widehat{y}\}]^2\} \\
&= [y_x-\mathrm{E}\{\widehat{y}\}]^2 + \mathrm{E}\{[\widehat{y}-\mathrm{E}\{\widehat{y}\}]^2\}
\end{aligned} \tag{2.6}$$

Gl.(2.6) zeigt, dass der Modell-Fehler aus zwei Teilen besteht. Davon wird $[y_x - \mathrm{E}\{\widehat{y}\}]^2$ als Bias-Fehler bezeichnet, und $\mathrm{E}\{[\widehat{y}-\mathrm{E}\{\widehat{y}\}]^2\}$ Varianz-Fehler genannt. Also,

$$\text{Modell-Fehler} = \text{Bias-Fehler}+\text{Varianz-Fehler}. \tag{2.7}$$

Hier ist zu beachten, dass der Modell-Fehler und der Bias-Fehler in der Gl.(2.7) quadratische Fehler sind. Bias-Fehler bedeutet die *systematische* Ablenkung zwischen dem identifizierten Ziel-System und dem Modell. Er wird von nicht ausreichender Flexibilität oder Komplexität[8] des Modells verursacht, deshalb heißt er auch Struktur-Fehler. Wenn z.B. ein Polynom-System 3. Ordnung durch ein Polynom-System 2. Ordnung angenähert wird, ergibt sich dennoch ein Fehler zwischen den Beiden, *auch wenn* die Parameter im Polynom-System 2. Ordnung perfekt geschätzt werden können. In diesem Fall ist der Fehler prinzipieller Natur und kann nicht durch die Parameter-Optimierung beseitigt werden, weil die zu geringe Polynom-Ordnung den Fehler verursacht. Ein solcher Fehler heißt Bias-Fehler. Abb. 2.10(a) zeigt den typischen Zusammenhang zwischen dem Bias-Fehler und der Komplexität des Modells.

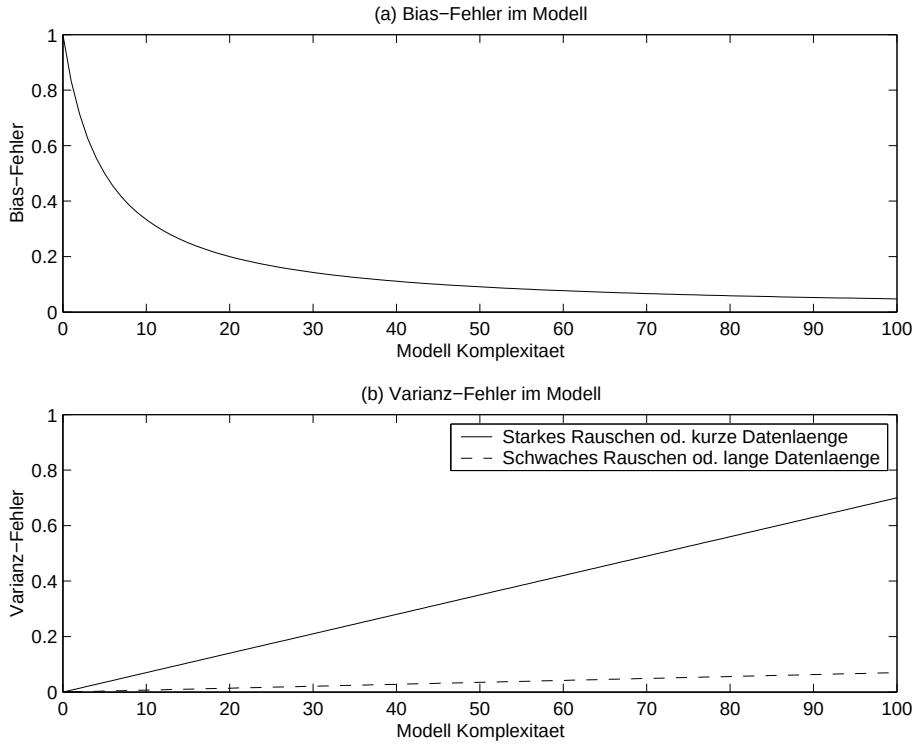


Abbildung 2.10: Der typische Zusammenhang zwischen dem Bias-Fehler,dem Varianz-Fehler und der Komplexität des Modells

Aus Abb. 2.10(a) wird ersichtlich:

1. Je größer die Komplexität des Modells ist, desto kleiner ist der Bias-Fehler[9].

[8]Komplexität des Modells kann theoretisch als ein Wert c definiert werden. Je größer der Wert c ist, desto komplizierter ist das Modell.

[9]Voraussetzung: das Verhalten des Algorithmus der Parameterschätzung ist identisch für alle Modelle.

2. Je größer die Komplexität des Modells wird, desto langsamer ist das Abnehmen des Bias-Fehlers.

3. Wenn eine bestimmte Komplexität des Modells erreicht ist, nähert sich der Bias-Fehler etwa einer Konstanten (theoretisch gegen Null[10]).

Das bedeutet, dass zu komplizierte Modelle nicht sehr sinnvoll sind. Außerdem werden neue Probleme, wie z.B. Overfiting, erzeugt, wenn das Modell zu kompliziert wird.

Der Varianz-Fehler beschreibt die Abweichung zwischen dem von einer *endlich langen* und *verrauschten* Datenmenge geschätzten Parameter und seinem optimalen Wert. Der Varianz-Fehler beschreibt die *Unsicherheit* der geschätzten Modell-Parameter. Im Wesentlichen verursacht das Rauschen im System diesen Fehler, z.B. wenn ein Polynom-System 3. Ordnung $P_a = a_0 + a_1 x + a_2 x^2 + a_3 x^3$ durch ein Polynom-System 3. Ordnung $P_b = b_0 + b_1 x + b_2 x^2 + b_3 x^3$ approximiert wird, ist normalerweise $b_i \neq a_i, i = 0, 1, 2, 3$, weil $b_i, i = 0, 1, 2, 3$ aus verrauschten Daten geschätzt wurde und diese Werte abhängig vom Rauschen sind. Dieser Fehler heißt Varianz-Fehler. In [Ljung87] wird präsentiert, dass der Varianz-Fehler bei großer Trainingsdatenmenge dem Zusammenhang nach Gl.(2.8) folgt.

$$\text{Varianz-Fehler} \sim \sigma^2 \frac{c}{\mathrm{N}} \tag{2.8}$$

Wobei σ^2 die Varianz des Rauschens ist. N die Länge der Trainingsdaten und c ein Wert, der der Komplexität des Modells entspricht. Hier ist zu beachten, dass die Gl.(2.8) auf einer hinreichend langen Trainingsdatensequenz basiert. Beim LZI-Modell ist c die Anzahl der Parameter des Modells. Beim nichtlinearen Modell ist c nicht so einfach zu bestimmen. Wenn ein nichtlineares System durch ein nichtlineares Modell mit *linearen Parameter* angenähert werden kann, ist c die äquivalente Anzahl der Parameter. z.B. $f_1(x) = a_0 + a_1 x$ besitzt zwei Parameter, also $c = 2$, und $f_2(x) = sin(ax) = b_0 + b_1 x + b_2 x^2 + b_3 x^3 + \cdots$ besitzt einen Parameter a, aber es ist $c \neq 1$. Gemäß der Tailor-Entwicklung kann $f_2(x)$ viele lineare Parameter $b_0, b_1, b_2, \cdots$ besitzen. Wegen der Abhängigkeit dieser Parameter ist auch $c \neq$ Anzahl der linearen Parameter. Theoretisch können wir c wie folgt einschränken:[11]

Anzahl der nichtlinearen Parameter$< c <$Anzahl der linearen Parameter

Je geringer die Anzahl der Parameter in einem Modell ist, desto genauer können diese Parameter geschätzt werden

Abb. 2.10(b) zeigt den typischen Zusammenhang zwischen dem Varianz-Fehler und der Komplexität eines Modells. Aus dieser Abbildung wird deutlich:

[10] Voraussetzung: der Algorithmus der Parameterschätzung ist perfekt.
[11] Die Komplexitätsbewertung eines nichtlinearen Modells ist eine schwierige Aufgabe.

1. Der Varianz-Fehler nimmt etwa linear mit dem Anstieg der Komplexität des Modells zu.

2. Je stärker das Rauschen ist, d.h. je größer σ^2 ist, desto größer ist der Varianz-Fehler.

3. Je länger die Trainingsdatensequenz ist, d.h. je größer N ist, desto kleiner ist der Varianz-Fehler.

Gemäß Gl.(2.7) werden die Fehler nach Abb. 2.10 (a) und Abb. 2.10 (b) aufsummiert, so dass wir den zusammengesetzten gesamten Modellfehler nach Abb. 2.11 erhalten.

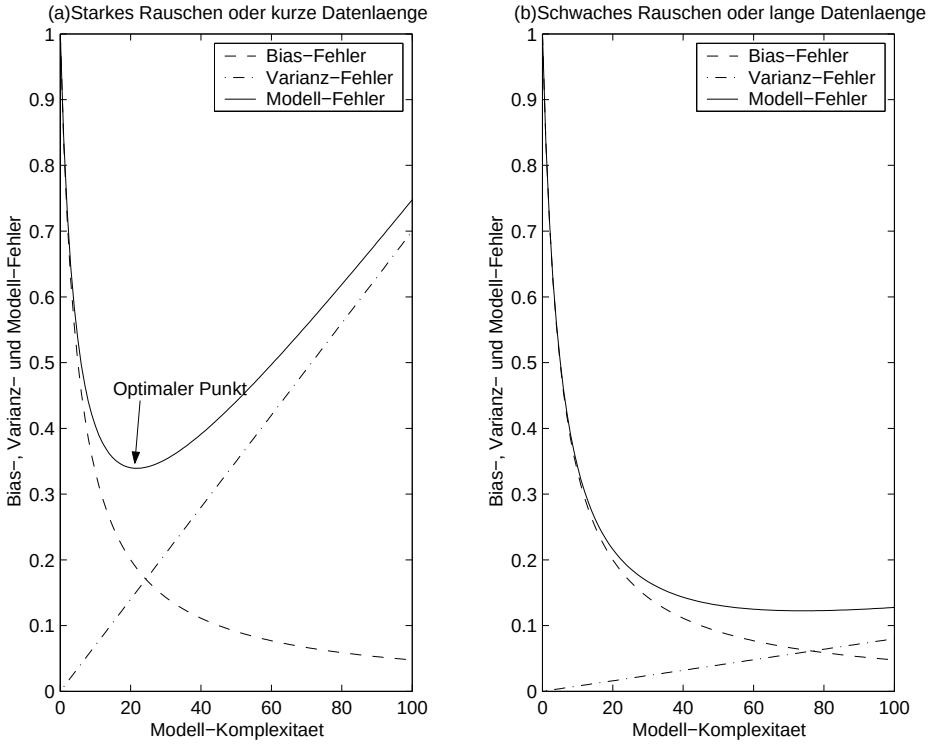


Abbildung 2.11: Zusammenhang zwischen Bias-Fehler, Varianz-Fehler und gesamtem Modell-Fehler

Hier gibt es zwei Sonderfälle:

1. Das Rauschen ist sehr stark, d.h. σ^2 (Varianz des Rauschens) ist sehr groß oder die Datenlänge ist sehr kurz, d.h. N ist sehr klein, also ist $\frac{\sigma^2}{N}$ sehr groß. Abb. 2.11(a) zeigt diesen Fall. Aus dieser Abbildung ist ersichtlich, dass es einen deutlich ausgeprägten optimalen Punkt[12] in Abhängigkeit von der Komplexität des Modells gibt. Wenn die Komplexität des Modells sich der Nähe dieses Punktes nähert, wird der Modell-Fehler zum Minimum.

[12]Dieser Punkt ist nicht der beste Punkt der Komplexität für den Bias-Fehler

2. Das Rauschen ist schwach, d.h. σ^2 ist relativ klein, und die Datenlänge ist relativ lang, dann ist $\frac{\sigma^2}{\mathrm{N}}$ sehr klein. Abb. 2.11(b) zeigt diesen Fall. Aus dieser Abbildung wird ersichtlich, dass wenn eine ausreichende Modell-Komplexität erreicht ist, der Modell-Fehler nicht mehr wesentlich zu verbessern ist. [13]

Falls der Bias-Fehler nicht berücksichtigt wird, d.h. das Modell besitzt eine ausreichende Komplexität, also der Bias-Fehler ist ungefähr Null und die Daten sind ausreichend lang[14], ist der Modell-Fehler≈Varianz-Fehler≈ $\sigma^2 \frac{c}{\mathrm{N}}$. Dieser Fehler bildet die Obergrenze der Genauigkeit des Modells bei der Situation mit Rauschen.

Training, Validierung und Test

Wir interessieren uns hier nur für die von Abb. 2.11(a) gezeigte Situation, d.h. starkes Rauschen oder kurze Trainingsdaten. Abb. 2.11(a) zeigt, theoretisch gibt es einen optimalen Punkt, bei dem das Modell den kleinsten Modell-Fehler(Bias-Fehler plus Varianz-Fehler) hat. In der Praxis ist dieser Punkt nicht bestimmbar, weil Bias-Fehler und Varianz-Fehler unbekannt sind. Es is zu beachten, dass Gl.(2.8) und Abb. 2.11 (a) auf einer *statistischen* Sicht basieren, d.h. das Ergebnis beruht auf ausreichender Statistik. In der Theorie gilt für die Systemidentifikation:

Zu komplizierte oder zu einfache Modelle sind nicht günstig.

Mit *einer* Datenmenge (z.B. Trainingsdaten) kann man tatsächlich den Varianz-Fehler des Modells *nicht* beschreiben, wenn die Datenmenge relativ klein oder das Rauschen sehr stark ist, weil die zufälligen Eigenschaft des Rauschens in diesem Fall größtenteils verloren gehen. Das trainierte Modell versucht sich folglich an das System und das Rauschen[15] anzupassen. Wenn das Modell sich an das Rauschen in den Trainingsdaten anzupassen versucht, ergibt sich eine Erscheinung, die Overfiting genannt wird. Dabei gibt es für Overfiting zwei Typen[16]:

1. Das Rauschen ist nicht sehr stark, aber die Trainingsdaten sind zu kurz. In diesem Fall versucht das Modell sich dem ganzen Ausgangssignal (System plus Rauschen) anzupassen, wie es in Abb. 2.12 (b) gezeigt wird.

2. Die Trainingsdaten sind nicht sehr kurz, aber das Rauschen ist sehr stark, in diesem Fall versucht das Modell sich dem Ausgangssignal des Systems und den

[13] Es gibt ein allgemeines Prinzip: das einfachste ist das beste, wenn mehrere Modelle die Anforderung gleichzeitig erfüllen. Gemäß diesem Prinzip ist der optimaler Punkt, der Punkt mit der kleinsten Komplexität.

[14] Die Voraussetzung für die Gültigkeit der Gl.(2.8)

[15] Eigentlich ist es wegen der Zufälligkeit des Rauschens unmöglich, dass das Rauschen im Zeitbereich beschrieben wird, wenn die Daten sehr lang sind

[16] Tatsächlich gibt es noch einen dritten Typ: ohne Rauschen, sehr kurze Daten, aber sehr kompliziertes Modell, wie in Abb. 2.12 (c) gezeigt. Tatsächlich ist das der reine Bias-Fehler des Modells

durchschnittlichen Werten[17] des Rauschens in kurzer Zeit anzupassen, wie es in Abb. 2.12 (a) gezeigt wird.

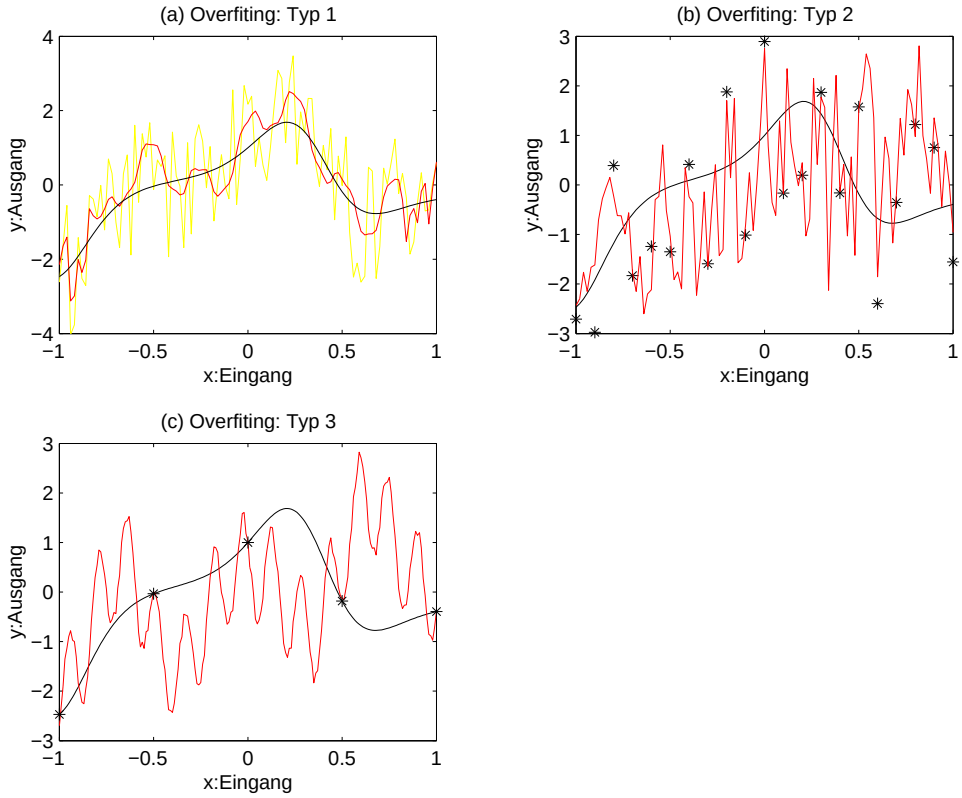


Abbildung 2.12: Unterschiedliche Arten von Overfiting: (a) die GP versucht sich an die kurzen durchschnittlichen Werte des Rauschens anzupassen. (b) die GP versucht sich an das Rauschen anzupassen. (c) die GP passt sich den Abtastungspunkten genau an, aber sie passt sich nicht den unabgetasteten Punkten (Bereiche) an.

Dadurch werden die Ursachen des Overfittings deutlich:

1. Rauschen.
2. Verlieren der zufälligen Eigenschaft des Rauschens.

Die Maßnahme zur Überwindung des Overfiting sind also:

1. Verkleinern des Rauschens.
2. Verstärken der zufälligen Eigenschaften des Rauschens.

Um das Overfitting zu überwinden, kann man unabhängige Trainingsdaten und Validierungsdaten verwenden.

[17] Für eine Rauschenquelle mit $E(n)$=0, ist die Voraussetzung die, dass die Länge N$\rightarrow \infty$ sein muss.

Die wichtigste Maßnahme zur Verkleinerung des Rauschens ist die Filterung der Signale, aber sie ist normalerweise beschränkt[18]. Die wichtigste Maßnahme für die Verstärkung der Zufälligkeit des Rauschens ist die Vergrößerung der Datenlänge, aber sie ist nicht für jeden Fall[19] geeignet. Mit einer zweiten unabhängigen Datenmenge(z.B. Validierungsdaten) kann man den Varianz-Fehler beschreiben. Weil das Modell in der ersten und zweiten Datenmenge den gleichen Bias-Fehler besitzt und wegen der Unabhängigkeit der beiden Datenmengen, das Modell bei der zweiten Datenmenge den Varianz-Fehler relativ besser darstellen kann, wie es in Abb. 2.13 gezeigt ist.

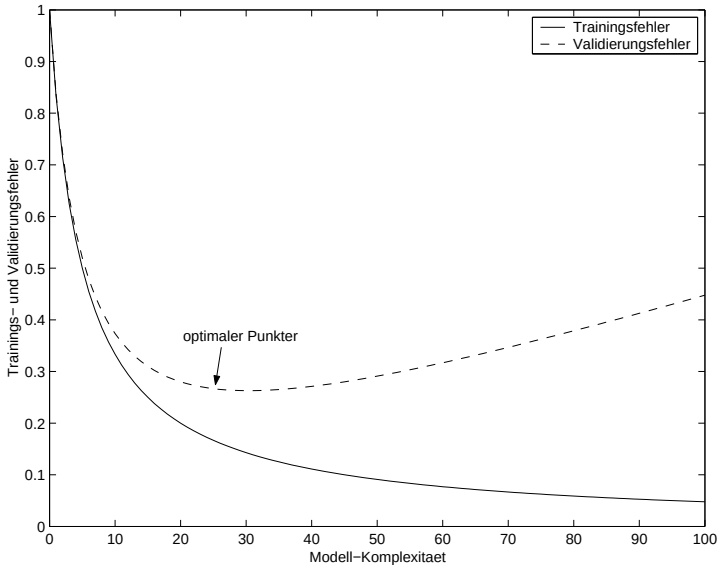


Abbildung 2.13: Zusammenhang zwischen Trainingsfehler und Validierungsfehler

Um das Verhalten des Modells zu testen, wird noch eine unabhängige Testdatenmenge benötigt. Mit der Testmenge wird also nur eine höhere Vertrauenswürdigkeit des Verhaltens des Modells erreicht, es findet keine weitere Verbesserung des Modells statt, wie in Abb. 2.14 gezeigt wird.

1. Mit der Trainierungsdatenmenge werden die (verschiedenen) Modelle trainiert.
2. Mit der Validierungsdatenmenge wird das Modell mit dem kleinsten Modell-Fehler (Bias-Fehler plus Varianz-Fehler) untersucht.
3. Mit der Testdatenmenge wird das Verhalten des ausgewählten Modells hinsichtlich seiner Vertrauenswürdigkeit getestet.

[18]Einerseits ist die Wirkung der Filterung vom Filter und von der Datenläge abhängig, z.B. wenn die Datenlänge sehr kurz ist, ist es schwierig, die Daten zu filtern, anderseits ist es manchmal verboten, die Daten zu filtern, wenn z.B. Chaosdaten vorliegen.

[19]z.B. die vorhandenen Daten sind kurz.

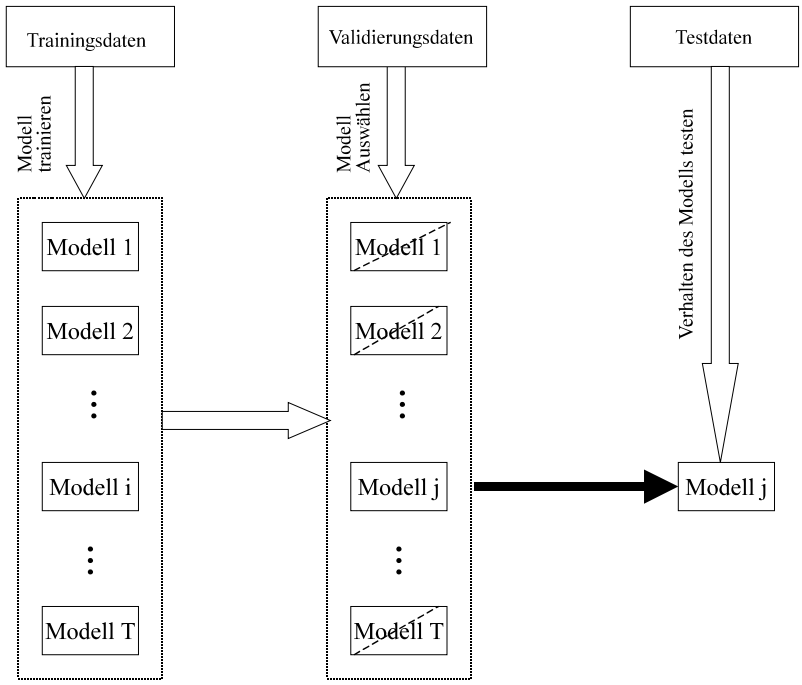


Abbildung 2.14: Zusammenhang und Ablauf von Training, Validierung und Test.

2.3 Strategien zur Verbesserung eines Modells

Für die nichtlineare Systemidentifikation gibt es viele unterschiedliche Verfahren, wie in Abb. 2.6 gezeigt wurde. Das Verhalten des Modells ist abhängig von dem verwendeten Verfahren und der zu lösenden Aufgabe. Bei jedem Verfahren ist die Systemidentifikation ein iterativer Vorgang. Wenn alle Verfahren versucht wurden, und die Forderungen z.B. hinsichtlich der Genauigkeit noch nicht erfüllt sind, kann man die folgenden zwei Strategien verwenden, nämlich die serielle und die parallele Modellstruktur.

Die Grundidee für diese beiden Strategieen ist die Haupt- und Nebenmodellstruktur. Normalerweise wird dabei das Nebenmodell als eine Ergänzung des Hauptmodells betrachtet.

Wegen der unterschiedlichen Struktur ergeben sich für diese beiden Strategien eigene Merkmale.

1. Die parallele Struktur ist allgemeiner als die serielle Struktur.

2. Zur ergänzenden Korrektur und Verfeinerung des Modells ist die parallele Struktur gut geeignet.

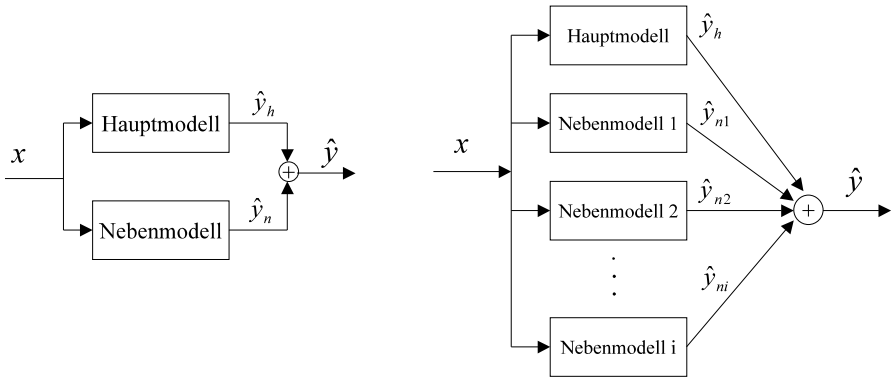
3. Die Komplexität des ganzen Modells ist in der parallelen Struktur die Summe der Komplexitäten aller Teil-Modelle. Im Gegensatz dazu ist die Komplexität des kompletten Modells in der seriellen Struktur viel größer als die Summe der

Komplexität aller Teil-Modelle. Das bedeutet, dass man mit einem einfachen Hauptmodell und einem einfachen Nebenmodell sehr komplizierte Gesamt-Modelle darstellen kann.

4. Es ist möglich mit der seriellen Struktur hierache Systeme zu beschreiben.

2.3.1 Parallele Struktur

Abb. 2.15 a) zeigt die parallele Struktur.



a) Haupt- Nebenmodell Struktur b) Erweiterung der Haupt- Nebenmodell Struktur

Abbildung 2.15: Parallele Strukturen zur Verbesserung der Genauigkeit eines Modells.

In dieser parallelen Struktur sind das Hauptmodell und das Nebenmodell parallel geschaltet. Die beiden Modelle können verschiedene Sub-Strukturen besitzen und mit verschiedenen Verfahren identifiziert werden. Dabei wird zunächst das Hauptmodell identifiziert, dafür ist das typische Verfahren die theoretische Analyse (Modellierung) oder die experimentelle Ermittelung. Wenn das Hauptmodell die Aufforderungen der Aufgabe nicht erfüllt, dient das Nebenmodell zur Verbesserung (Ergänzung) des Hauptmodells. Das typische Identifikationsverfahren für das Nebenmodell ist ein datengetriebenes Verfahren.

Der Ausgang zur Identifikation des Nebenmodells ist $\widehat{y}_n = y - \widehat{y}_h$. Dabei ist $\widehat{y}_h$ der aktuelle Ausgang des Hauptmodells, wenn x am Eingang anliegt. y ist der Ausgang des Ziel-Systems. Außerdem können beide Modelle durch konventionelle Verfahren gemeinsam optimiert werden. Die parallele Struktur kann erweitert werden, wie dies in Abb. 2.15 b) gezeigt wird. Hierbei ist die Reihenfolge der Ermittelung der Modelle: Hauptmodell, Nebenmodell 1, Nebenmodell 2,usw..

In diesem Fall ist zu beachten, dass unter bestimmten Voraussetzungen[20] das

[20] z.B. Ohne Rückkopplung vom Ausgang zum Modelleingang.

Hauptmodell und die Nebenmodelle kombinierbar sind. Diese Strategie bildet die Basis des rekursiven GP-Verfahrens.

2.3.2 Serielle Struktur

Abb. 2.16 a) zeigt die serielle Struktur.

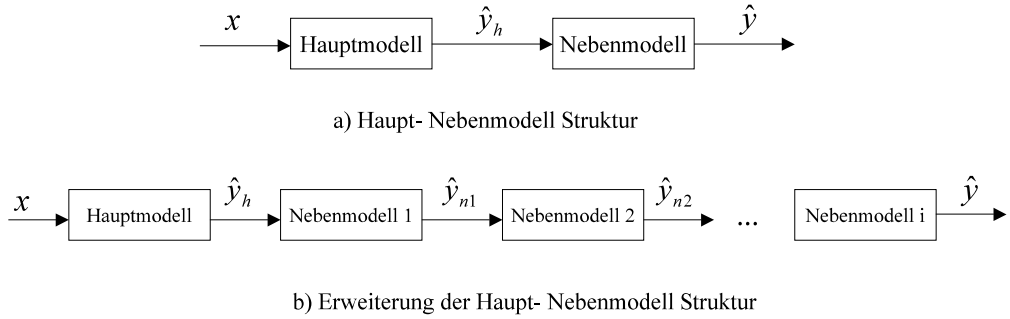


Abbildung 2.16: Serielle Struktur zur Verbesserung der Genauigkeit eines Modells

In dieser Struktur sind das Hauptmodell und das Nebenmodell in Kette geschaltet. Die beiden Modelle können verschiedene Strukturen besitzen und mit verschiedenen Verfahren identifiziert werden. Dabei wird zunächst das Hauptmodell identifiziert, dafür ist das typische Verfahren wieder die theoretische Analyse (Modellierung) oder die experimentelle Ermittelung. Wenn die Aufforderungen der Aufgabe noch nicht erfüllt sind, wird das Nebenmodell zur Verbesserung des Hauptmodells nachgeschaltet und identifiziert. Das typische Identifikationverfahren für das Nebenmodell ist ein datengetriebenes Verfahren, nämlich die experimentelle Ermittelung des Modells.

Der Eingang zur Identifikation des Nebenmodells ist der aktuelle Ausgang des Hauptmodells und der Ausgang ist der Ausgang des Systems. Außerdem können beide Modelle durch konventionelle Verfahren gemeinsam optimiert werden. Die Struktur kann weiter erweitert werden, wie in Abb. 2.16 b) gezeigt. Hierbei ist die Reihenfolge der Ermittelung der Modelle: Hauptmodell, Nebenmodell 1, Nebenmodell 2, usw..

In diesem Fall ist zu beachten, dass beide Strategien allgemein sind und die Wirkung abhängig von der Aufgabe ist. Normalerweise ist die parallele Struktur noch allgemeiner[21] als die serielle Struktur. Außerdem können natürlich die beiden Strategien gemischt verwendet werden.

[21]Es ist ersichtlich, dass die Struktur eine Obermenge der Volterra-Reihenentwicklung ist.

2.4 Systemklassifikation

Das Ziel dieses Abschnitt ist es: eine für die GP geeignete formale Systembeschreibung zu entwickeln.

Die automatische Erzeugung neuer Strukturen ist der wichtigste Vorteil der GP bei der Systemidentifikation. Im Wesentlichen besitzt die GP bei der Identifikation zwei Arten von Freiheitsgraden, d.h. Freiheitsgrade hinsichtlich der Struktur und der Parameter. Grundsätzlich kann die GP gleichzeitig in beiden Freiheitsgraden suchen. Hier ist zu beachten, dass nicht garantiert ist, dass die GP eine bessere Struktur als die traditionellen Modell-Strukturen erzeugen oder finden kann.

Die GP bietet nur eine neue *Möglichkeit* zur Erzeugung von besseren Strukturen. Die GP benötigt nur eine *formale* Beschreibung, die einerseits für die Aufgabe ausreichende Freiheitsgrade haben muss, anderseits die Freiheitsgrade aber auf ein notwendiges Minimum beschränkt, um bessere Ergebnisse zu bekommen. Dies bildet das grundlegende Kriterium, mit dem die Grammatik der GP bzw. ihre Terminal- und Nichtteminalmenge ausgewählt werden muss. Die GP kontrolliert die strukturellen Freiheitsgrade und die Freiheitsgrade hinsichtlich der Parameter mittels Terminal- und Nichtteminalmenge sowie der Grammatik der GP. Theoretisch gilt, je größer die Freiheitsgrade sind, desto wahrscheinlicher ist es, dass im Suchraum ein gute Lösung enthalten wird. Aber normalerweise gilt auch:

Je größer die Freiheitsgrade sind, desto schwieriger (langwieriger) ist es für die GP eine Lösung im Suchraum zu finden.

Um die formale Beschreibung der Systeme der GP anzupassen, werden die Systeme wie folgt klassifiziert:

- Lineare zeitinvariante Systeme.
- Nichtlineare Systeme ohne Gedächtnis.
- Nichtlineare Systeme mit endlich langem Gedächtnis.
- Nichtlineare Systeme mit unendlich langem Gedächtnis.

2.4.1 Lineare Zeitinvariante Systeme (LZI-Systeme)

Auf oberster Ebene sind die Systeme in lineare und nichtlineare Systeme eingeteilt. Das ist sinnvoll zum Verstehen des Begriffs der Kompensation von nichtlinearen Verzerrungen. Außerdem kann ein LZI-System, das mit der Gl. (2.9) genauer beschrieben werden kann, mit der GP ebenfalls identifiziert werden.

$$y(t) = \sum_{i=0}^{M} a_i x(t-i) + \sum_{j=1}^{N} b_j y(t-j) \tag{2.9}$$

2.4.2 Nichtlineare (NL) Systeme ohne Gedächtnis

Gedächtnislose bzw. statische NL-Systeme zeichnen sich anschaulich dadurch aus, dass der Systemausgang schlagartig auf Null zurückkehrt, sobald am Eingang kein Signal mehr anliegt, d.h. der Ausgangswert ist nur vom aktuellen Eingangswert abhängig. Vergangene Eingangswerte (und Ausgangswerte) zu Zeitpunkten $t < t_0$ haben keinen Einfluss auf den Ausgangswert zum Zeitpunkt $t = t_0$ wie dies in der Abb. 2.18 a) gezeigt wird. Ein solches System kann durch die Gl. (2.10) formal beschrieben werden:

$$y(t) = f(x(t)) \tag{2.10}$$

Dabei ist f eine beliebige Funktion, die *Kennlinie* genannt wird, so wie dies in der Abb. 2.17 gezeigt ist.

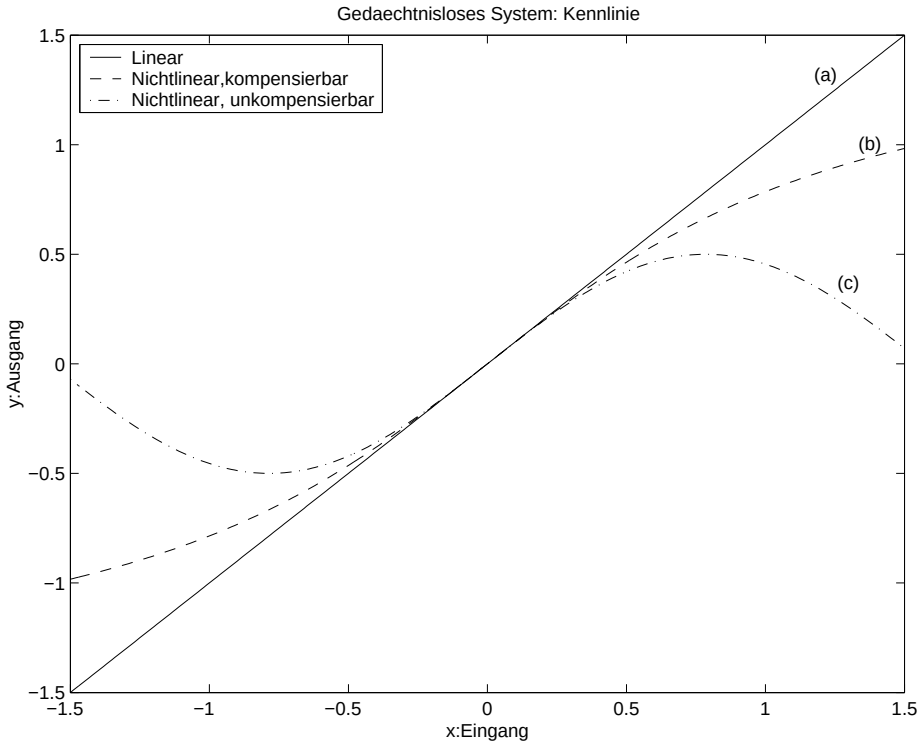


Abbildung 2.17: Gedächtnislose Systeme können durch eine System-Kennlinie beschrieben werden. Kennlinie (a) ist eine Gerade; Kennlinie (b) ist eine monotone Kennlinie; Kennlinie (c) ist eine nicht monotone Kennlinie.

Die formale Beschreibung der Gl. (2.10) bedeutet für die GP, dass ihre Terminalmenge nur auf den aktuellen Wert beschränkt ist. In diesem Fall bedeutet passives

System, dass die Kennlinie immer durch den Ursprung geht. Wenn diese Kennlinie eine gerade Linie ist, spricht man davon, dass das System ein passives lineares gedächtnisloses System ist. Im Gegensatz dazu spricht man bei Kennlinien, die von der Geraden abweichen von passiven,nichtlinearen,gedächtnislosen Systemen. Insbesondere wenn diese Kennlinie monoton ist, ist es schon bewiesen, dass das System theoretisch kompensierbar ist. Bei nicht monotoner NL-Kennlinie geht man davon aus, dass das NL-System im Allgemeinen theoretisch nicht kompensierbar ist.

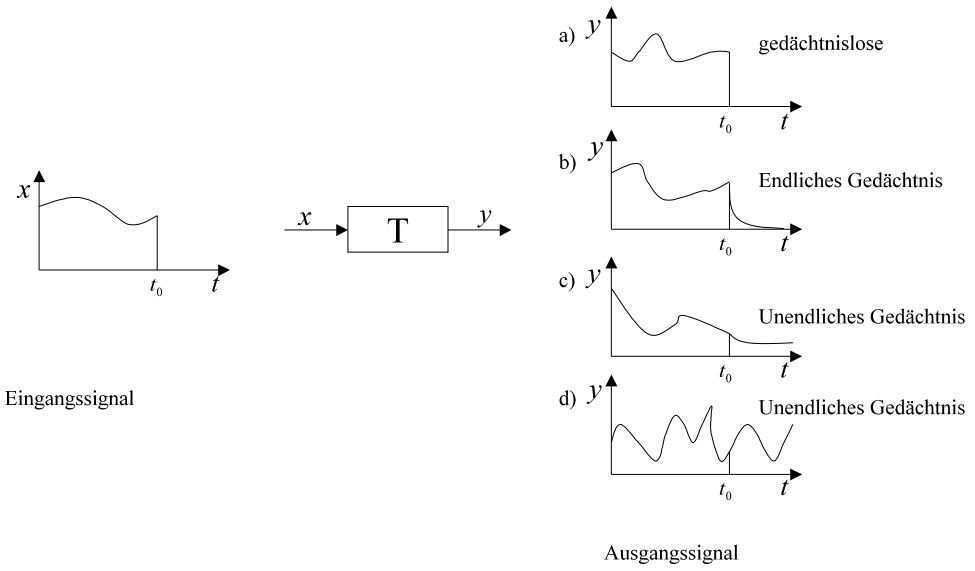


Abbildung 2.18: Anschauliche Darstellung der Systeme ohne Gedächtnis, mit endlich langem Gedächtnis sowie mit unendlich langem Gedächtnis.

2.4.3 NL-Systeme mit endlich langem Gedächtnis

Systeme mit endlich langem Gedächtnis zeichnen sich anschaulich dadurch aus, dass der Systemausgang nach einer bestimmten Zeit gegen Null geht, wenn am Eingang kein Signal mehr anliegt, so wie dies in Abb. 2.18 b) gezeigt ist. Der Ausgang des Systems hängt von aktuellen und vergangenen Eingangswerten ab. Die Beschränkung auf vergangene Ausgangswerte ist für die Erfüllung der Kausalität notwendig. Dies lässt sich formal wie folgt schreiben als

$$y(t_0) = f(x(t \mid t \leq t_0), y(t \mid t < t_0)) \tag{2.11}$$

wobei $f(\cdot)$ eine beliebige Funktion ist. Weil der Ausgang nicht nur von den aktuellen Eingangswerten abhängt, lassen sich derartige Systeme nicht mehr durch eine Kennlinie wie in Abb 2.17 darstellen.

Die formale Beschreibung der Gl. (2.11) bedeutet für die GP, dass ihre Terminalmenge sich auf die aktuellen Eingangswerte sowie die Werte der Vergangenheit des

Eingangs- bzw. Ausgangssignals beschränken kann. Ein Sonderfall liegt vor, wenn $f(\cdot)$ eine lineare Funktion ist. In diesem Fall ist das System ein LZI-System. Ansonsten spricht man davon, dass das System ein nichlineares System mit Gedächtnis ist.

Die nichtlinearen Systeme, die in diese Klasse fallen, sind z.B. Systeme mit Sättigung, unstetige Systeme oder Polynom-Systeme.

In der Theorie ist zwar noch nicht bekannt, unter welchen notwendigen und hinreichenden Voraussetzungen ein nichtlineares System kompensierbar ist, die Praxis zeigt jedoch, dass im Allgemeinen ein NL-System mit schwacher Nichtlinearität kompensierbar oder zum Größtenteil kompensierbar ist. Wenn ein NL-System eine starke Nichtlinearität besitzt, denkt man normalerweise, dass das System nicht kompensierbar ist. Dabei gibt es eine Ausnahme. Es ist möglich, ein stark nichtlineares System zu kompensieren, wenn es mit einem Wiener-Modell oder Hammerstein-Modell beschrieben werden kann. Zwischen starker und schwacher Nichtlinearität gibt es aber keine strenge Trennlinie. Das wichtigste Maß, mit dem die Nichtlinearität ausgedrückt werden kann, ist der Klirrfaktor K. K wird durch die Gl.(2.12) definiert.

$$\mathrm{K} = \sqrt{\frac{\mathrm{A}_2^2 + \mathrm{A}_3^2 + \cdots}{\mathrm{A}_1^2 + \mathrm{A}_2^2 + \mathrm{A}_3^2 + \cdots}} \cdot 100[\%] \tag{2.12}$$

A_i ist die Amplitude der Signalanteile bei der Frequenz if_0 am Ausgang[22] und f_0 ist die Frequenz des Eingangssignals.

2.4.4 NL-Systeme mit unendlich langem Gedächtnis

Systeme mit unendlich langem Gedächtnis zeichnen sich anschaulich dadurch aus, dass der Systemausgang für eine unendlich lange Zeit auf Werten ungleich Null bleiben kann, auch wenn das Eingangssignal abgeschaltet ist. Abb 2.18 c) und d) zeigen solche Fälle. Derartige Systeme besitzen nicht-flüchtige Speicher, die durch interne Zustände S_i definiert werden. Der Ausgang des Systems hängt dann nicht nur von den Ein- und Ausgangswerten, sondern auch von den internen Zuständen ab. Dies lässt sich formal wie folgt beschreiben:

$$y(t_0) = f(x(t \mid t \leq t_0), y(t \mid t < t_0), S_i) \tag{2.13}$$

Alle derartigen Systeme sind ohne Ausnahme als nichtlinear zu bezeichnen und die nichtlinearen Verzerrungen von derartigen Systemen sind prinzipiell nicht kompensierbar.

[22] Eine wichtige Eigenschaft eines nichtlinearen Systems ist es, dass neue Frequenzen(Oberwellen) am Ausgang erzeugt werden können.

2.5 Beschreibung der Nichtlinearität

Das Ziel dieses Abschnitts ist es, die wichtigsten Beschreibungsformen der Nichtlinearität zu diskutieren.

Es gibt viele unterschiedliche mathematische Beschreibungen der Nichtlinearitäten und entsprechend gibt es auch unterschiedliche Identifizierungsverfahren. Die Wichtigsten sollen im Folgenden beschrieben werden.

2.5.1 Allgemeine Beschreibung

Die Gl. (2.14) zeigt die allgemeinste Form der Beschreibung eines NL-Systems.

$$y(t) = f(x(t), x(t-1), x(t-2), ..., y(t-1), y(t-2), y(t-3), ...) \tag{2.14}$$

Wobei $f(\cdot)$ eine beliebige Funktion ist. Die Beschreibungsform ist für die Systemidentifikation durch GP geeignet, solange die Funktionsmenge gegeben ist.

2.5.2 Volterra-Systeme

Die Volterra-Reihenentwicklung zeigt, dass ein System unter bestimmten Voraussetzungen mit Gl. (2.15) beschrieben werden kann.[Schetzen80]

$$\begin{aligned} y(t) \;=\; & \sum_{n_1=0}^{\infty} h_1(n_1)x(t-n_1) + \\ & \sum_{n_1=0}^{\infty}\sum_{n_2=0}^{\infty} h_2(n_1,n_2)x(t-n_1)x(t-n_2) + \\ & \sum_{n_1=0}^{\infty}\sum_{n_2=0}^{\infty}\sum_{n_3=0}^{\infty} h_3(n_1,n_2,n_3)x(t-n_1)x(t-n_2)x(t-n_3) + \cdots + \\ & \sum_{n_1=0}^{\infty}\cdots\sum_{n_p=0}^{\infty} h_p(n_1,\cdots,n_p)x(t-n_1)\cdots x(t-n_p) \end{aligned} \tag{2.15}$$

Dabei heißt $h_1(n_1)$ zeitdiskreter Volterra-Kern der 1. Ordnung, $h_2(n_1, n_2)$ heißt zeitdiskreter Volterra-Kern der 2. Ordnung und $h_3(n_1, n_2, n_3)$ der zeitdiskrete Volterra-Kern der 3. Ordnung u.s.w.. Tatsächlich stellt Gl.(2.15) eine unendlich lange Reihenentwicklung dar. Das Wichtigste ist, dass durch diese Reihenentwicklung ein linearer Systemanteil (h_1) und ein nichtlinearer Systemanteil abgespalten wird. Den linearen Anteil des Systems abzuspalten ist wesentlich und die erste wichtige Aufgabe bei der Kompensation von nichtlinearen Verzerrungen.

Abb. 2.19 zeigt die Struktur der Volterra-Reihenentwicklung. Dabei kann man klar sehen, dass der lineare Anteil des Systems vollständig abgespalten wird, weil die Volterra-Reihenentwicklung keine Rückkopplung vom Ausgang zum Eingang besitzt. Damit ist der Ausgang des linearen Anteils des Systems nur vom Eingang des Systems abhängig. Dies bildet die Basis für die Kompensation der nichtlinearen Verzerrungen mit der GP.

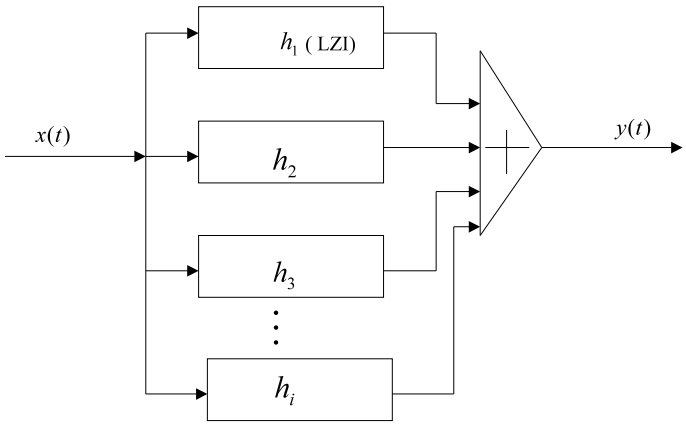


Abbildung 2.19: Beschreibung der Volterra-Reihenentwicklung eines NL-Systems.

2.5.3 Polynom-Systeme

Die Gl. (2.16) zeigt die allgemeine Form der Beschreibung der Polynom-Systeme.

$$y(t) = \mathrm{P}_r(x(t), x(t-1), x(t-2), ..., y(t-1), y(t-2), ...) \tag{2.16}$$

Wobei $\mathrm{P}_r(\cdot)$ ein Polynom r-ten Grades mit den entsprechenden Argumenten bezeichnet. Hierbei ist der Begriff "Polynom" so zu verstehen, dass der Ausgang eines Polynom-Systems die Verknüpfung beliebiger Potenzen und Produkte aktueller und vergangener Ein- und Ausgangswerte ist.

Das Kolmogorov-Polynom ist einer Sonderfall der allgemeinen Polynom-Systeme. Im Kolmogorov-Polynom gibt es keine Bestandteile von $y(t-1), y(t-2), y(t-3), ...$ Also ist es garantiert, dass das Kolmogorov-Polynom stabil ist.

2.5.4 Neuronale Netze

Der Grundbaustein eines neuronalen Netzwerkes ist das Neuron, wie es in Abb. 2.20 gezeigt wird. Die Verbindungsstärke von Neuron s_i zu Neuron s_j wird als Gewicht w_{ij} bezeichnet. Zur Berechnung der Aufgabe werden die Einflüsse der anderen Neuronen gewichtet und aufsummiert, auf diese Summe wird dann eine einfache nichtlineare Ausgabefunktion angewandt. Durch unterschiedliche Vernetzungsstruktur der Neuronen, wie z.B. MLP[23]-Struktur, sowie die Komplexität der Vernetzung, wie z.B.

[23]MultiLayer Proganation

die Anzahl der verborgenen Schichten im MLP, kann eine beliebige Nichtlinearität nachgebildet werden.

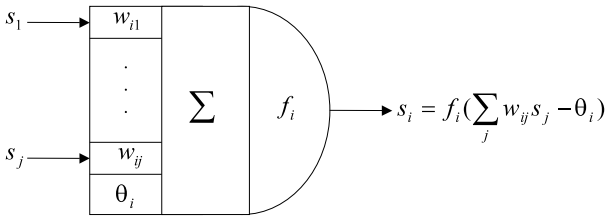


Abbildung 2.20: Der Grundbaustein: das Neuron

Neuronale Netze erhalten ihre speziellen Gewichte w_{ij}, indem sie für ihre Anwendung trainiert werden. Das Training wird mit für die Anwendung typischen Daten über einen Adaptions-Algorithmus, wie z.B. das Backpropagation-Verfahren durchgeführt. Neuronale Netze haben typischerweise ein endliches Gedächtnis. Hier ist zu beachten, dass die neuronalen Netze keine Basis für die Kompensation bilden können, weil in den neuronalen Netzen kein linearer Teil abgespaltet werden kann.

2.5.5 Halbphysikalische Modelle

Halbphysikalische Modelle besitzen, wie ihr Name zeigt, eine "halbphysikalische" Struktur. Der Sinn der halbphysikalischen Modelle ist:

1. Das halbphysikalische Modell kann manche künstlichen Systeme *exakt* darstellen, weil die Struktur des halbphysikalischen Modells mit der Struktur des künstlichen Systems identisch sein könnte.

2. Das halbphysikalische Modell kann manche nichtlinearen Systeme besser *nachbilden.*

3. Die Kompensation der halbphysikalischen Modelle ist viel einfacher und deutlicher.

Die wichtigsten halbphysikalischen Modelle sind: Wiener-Modell, Hammerstein-Modell und Wiener-Hammerstein-Modell. Ein LZI-System mit nach schaltetem statischen NL-System bildet ein Wiener-Modell, wie es in Abb. 2.21 a) gezeigt wird. Ein statisches NL-System mit nachgeschaltetem LZI-System bildet ein Hammerstein-Modell, wie es in Abb. 2.21 b) gezeigt wird. Ein LZI-System mit nachgeschaltetem statischem NL-System und nachgeschaltetem LZI-System bildet ein Wiener-Hammerstein-Modell, wie es in Abb. 2.21 c) gezeigt wird. Im Kapitel 6 wird ein Kompensationsalgorithmus, der auf dem Wiener-Modell basiert, entwickelt.

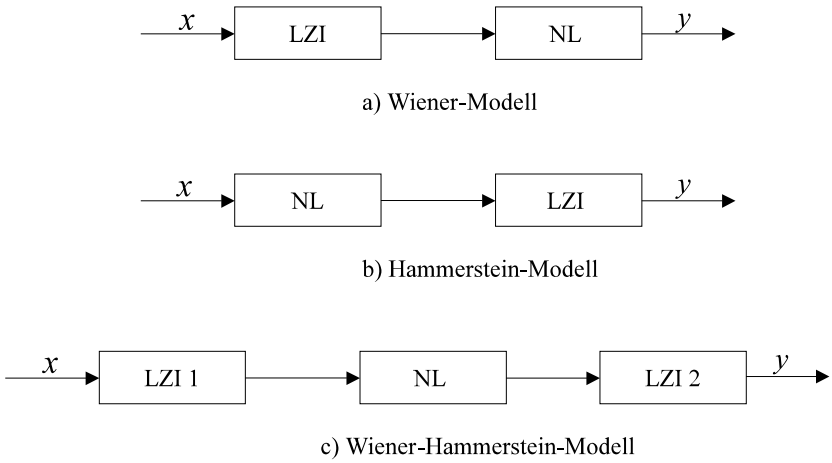


Abbildung 2.21: Drei wichtige halbphysikalische Modelle: a) Wiener-Modell. b) Hammerstein-Modell. c) Wiener-Hammerstein-Modell. Dabei bedeutet NL das gedächtnislose nichtlineare System.

2.6 Kompensation von nichtlinearen Verzerrungen

Im Allgemeinen bedeutet die Kompensation von nichtlinearen Verzerrungen, dass die nichtlinearen Verzerrungen des Systems beseitigt werden. Wie in Abb. 2.22 gezeigt wird, geschieht dies durch ein Kompensationssystem so dass das Gesamtsystem ein lineares System wird.

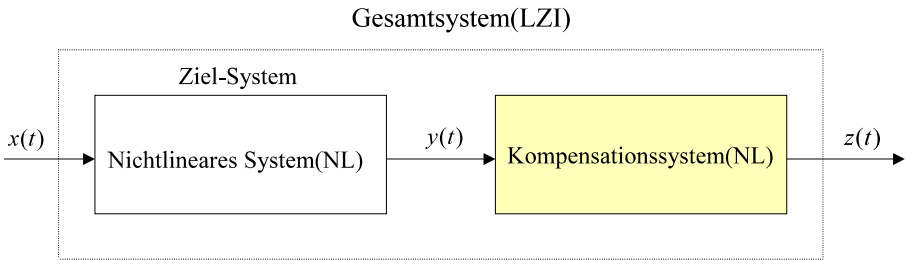


Abbildung 2.22: Allgemeine Beschreibung der Kompensation von nichtlinearen Verzerrungen.

Um den Begriff der Kompensation genauer zu definieren, wird im Folgenden die Kompensation in Entzerrer und Linearisierer aufgeteilt.

Das System, dessen nichtlineare Verzerrungen kompensiert werden sollen, wird als Ziel-System bezeichnet. Das System, das die Kompensation durchführt, wird als Kompensationssystem bezeichnet.

Definition 2.7 [Kafka02]: *Hat eine Kaskade aus einem nichtlinearen Ziel-System und einem nichtlinearen Kompensationssystem eine resultierende Betragsübertragungsfunktion, die für alle betrachteten Frequenzen konstant Eins ist, d.h. werden alle linearen und nichtlinearen Übertragungseigenschaften vollständig kompensiert, so bezeichnet man dieses Kompensationssystem als nichtlinearen Entzerrer. Ist das Kompensationssystem in der Kaskade vor dem nichtlinearen Ziel-System angeordnet, so spricht man von einem Pre-Entzerrer, ist es dahinter angeordnet, so spricht man von einem Post-Entzerrer.*

Definition 2.8 [Kafka02]: *Hat eine Kaskade aus einem nichtlinearen Ziel-System und einem nichtlinearen Kompensationssystem eine resultierende Betragsübertragungsfunktion, die der linearen Betragsübertragungsfunktion des Systems entspricht, d.h. werden alle nichtlinearen Verzerrungen vollständig kompensiert, so bezeichnet man dieses Kompensationssystem als nichtlinearen Linearisierer. Ist das Kompensationssystem in der Kaskade vor dem nichtlinearen Ziel-System angeordnet, so spricht man von einem Pre-Linearisierer, ist es dahinter angeordnet, so spricht man von einem Post-Linearisierer.*

2.6.1 Überblick über die Kompensationsprozesse

Die Ermittelung des Kompensationssystems ist sehr ähnlich wie die Ermittelung des inversen Systems. Nicht jedes nichtlineare System kann kompensiert werden. Dafür braucht man einige grundlegende Untersuchungen. Abb. 2.23 zeigt den Prozess.

Für ein gegebenes nichtlineares Ziel-System entscheidet man zunächst, ob das Ziel-System kompensierbar ist. Wenn die Entscheidung "Nein" lautet, ist die exakte Kompensation dieses nichtlinearen Ziel-Systems nicht möglich. Wenn die Entscheidung "Ja" ist, kommt die 2. Entscheidung, d.h. ob das Ziel-System mit einem Wiener-Modell beschrieben werden kann. Wenn die Entscheidung "Ja" lautet, kommt der Kombinationsalgorithmus, der im Kapitel 6 ausführlich dargestellt wird, zur Lösung grundsätzlich in Frage. Wenn die Entscheidung "Nein" ist, kommt die 3. Entscheidung, d.h. ob man einen LZI-Anteil von diesem nichtlinearen Ziel-System abspalten kann. Wenn die Entscheidung "Ja" ist, verwenden wir ein rekursives GP Verfahren , das im Kapitel 5 ausführlich dargestellt wird. Wenn die Entscheidung "Nein" ist, ist es schwierig, mit der GP die Aufgabe zu lösen, weil die GP den Ausgang des LZI-Anteils des nichtlinearen Ziel-Systems benötigt.

Hier ist zu beachten, obwohl wir gemäß Abb. 2.23 den Prozess beschreiben können, ergibt sich trotzdem eine Schwierigkeit, nämlich die, der durchführbaren Entscheidungskriterien. Zur Zeit gibt es noch keine allgemeine Methode, gemäß derer aus den Ein- und Ausgangssignalen des nichtlinearen Ziel-Systems die 3 Entscheidungen allgemein zu treffen sind. Als Alternative macht man den Prozess umgekehrt. D.h. zunächst identifizieren wir das nichtlineare Ziel-System mit einem Wiener-Modell. Wenn der Identifizierungsalgorithmus konvergiert, d.h. der Restfehler der Modellie-

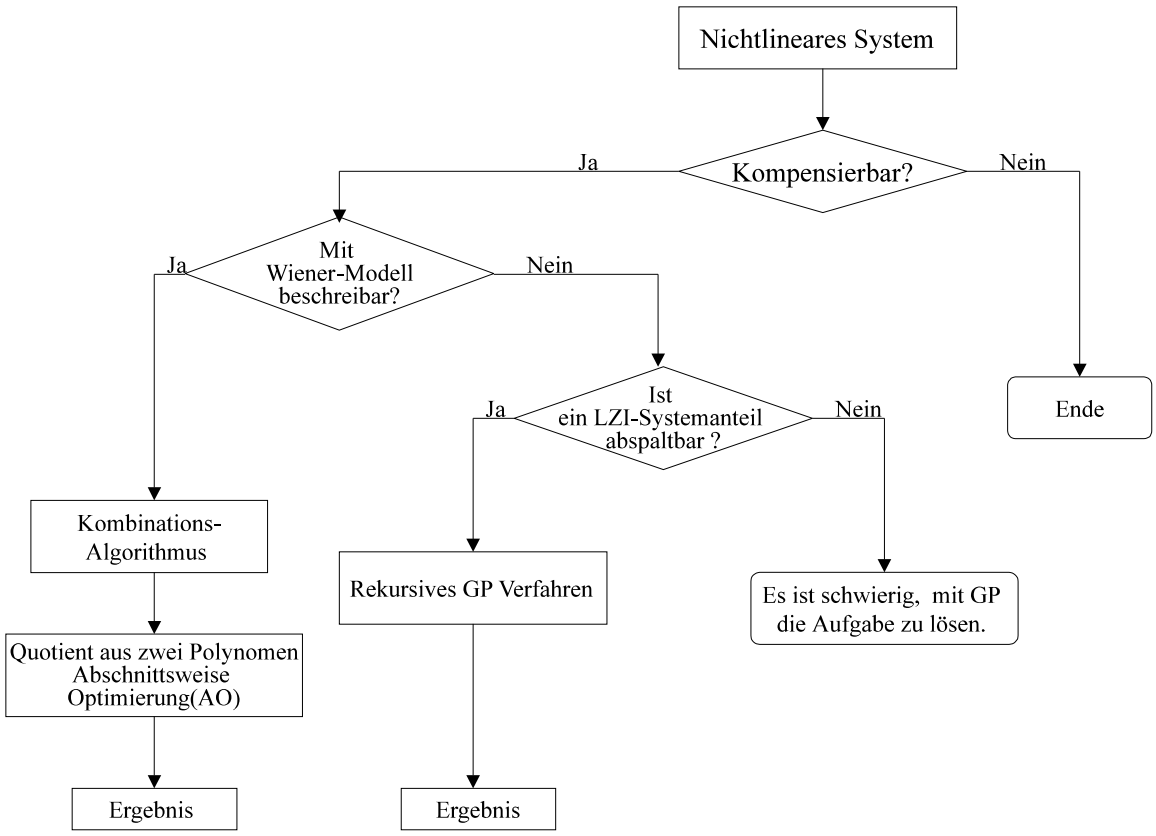


Abbildung 2.23: Ablaufdiagramm des Kompensationsprozesses für ein nichtlineares Ziel-System.

rung kleiner als ein vorgegebener Wert ε ist, dann kann man annehmen, dass das nichtlineare Ziel-System mit einem Wiener-Modell beschrieben werden kann, ansonsten nimmt man an, dass das nichtlineare Ziel-System mit einem Wiener-Modell nicht beschreibbar ist.

Ähnlich wie oben beschrieben können wir mit NFIR-Filtern (Nichtlineares FIR[24]), NIIR-Filtern (Nichtlineares IIR[25]) oder Volterra-Reihen versuchen, das nichtlineare Ziel-System zu identifizieren. Wenn der Restfehler kleiner als der vorgegebene Wert ε ist, nimmt man an, dass der LZI-Anteil im NFIR, NIIR oder der Volterra-Reihe der LZI-Anteil des Ziel-Systems ist. Hier ist zu beachten, dass in diesem Fall neuronale Netze ungeeignet sind, weil in solchen Netzen gar keine LZI-Anteile entstehen können.

Die Kompensation mit der GP soll auf gemessenen Ein- und Ausgangssignalen des Ziel-Systems basieren. Also gibt es insgesamt 4 Entwürfe für die Kompensation.

[24]Finite Impulse Response
[25]Infinite Impulse Response

2.6.2 Vier Entwürfe der Kompensation

Die vier Entwürfe [Frank97, Kafka02] der Kompensation sind: Pre- und Post-Entzerrung, Pre- und Post-Linearisierung. Im folgenden wird der Aufbau zur Identifikation des Entzerrers und Linearisierers dargestellt. Dieser Aufbau ist geeignet, wenn das Ziel-System mit Volterra-Reihen, NFIR- oder NIIR-Systemen beschrieben werden kann.

Pre- und Post-Entzerrung

Ziel der Entzerrung ist es, die linearen und nichtlinearen Verzerrungen des Ziel-Systems gleichzeitig zu kompensieren. In [Frank97, Kafka02] wird gezeigt, dass Pre- und Post-Entzerrer grundsätzlich die gleiche Struktur besitzen. D.h. das Ziel-System und der Entzerrer sind austauschbar. Abb. 2.24 zeigt den Aufbau zur Auffindung des Pre- und Post-Entzerrers eines nichtlinearen Ziel-Systems. Hierbei ist der Eingang zur Auffindung des Entzerrers der Ausgang des Ziel-Systems, und der Ausgang zur Auffindung des Entzerrers ist der Eingang des Ziel-Systems mit $3T_0$ Verzögerung, um die Kausalität des Entzerrers zu gewährleisten. Dabei ist T_0 das Abtast-Intervale.

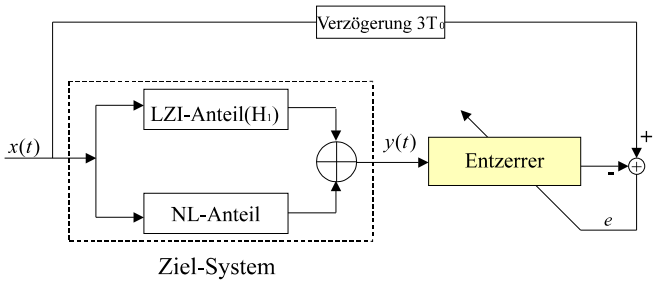


Abbildung 2.24: Aufbau zur Auffindung des Pre- und Post-Entzerrers

Post-Linearisierung

Ziel der Linearisierung ist es, die lineare Übertragungsfunktion des Ziel-Systems beizubehalten. Es werden nur die nichtlinearen Verzerrungen kompensiert. In [Kafka02] wird gezeigt, dass Pre- und Post-Linearisierer unterschiedliche Strukturen besitzen, d.h. Ziel-System und Linearisierer sind nicht austauschbar. Abb. 2.25 zeigt den Aufbau zur Auffindung des Post-Linearisierers eines nichtlinearen Ziel-Systems. Hierbei ist der Eingang zur Auffindung des Post-Linearisierers der Ausgang des Ziel-Systems und der Ausgang zur Auffindung des Linearisierers ist der Eingang des Ziel-Systems durch H_1 (linearer Anteil des Ziel-Systems) mit $2T_0$ Verzögerung, um die Kausalität des Post-Linerisierers zu gewährleisten, wobei T_0 das Abtast-Intervale ist.

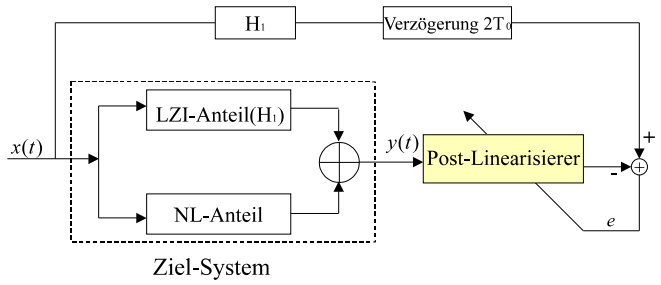


Abbildung 2.25: Aufbau zur Auffindung des Post-Linearisierers

Pre-Linearisierung

Abb. 2.26 zeigt den Aufbau zur Auffindung des Pre-Linearisierers eines nichtlinearen Ziel-Systems. Hierbei ist der Eingang zur Auffindung des Pre-Linearisierers der Ausgang des Ziel-Systems durch H_1^{-1} und der Ausgang zur Auffindung des Linearisierers ist der Eingang des Ziel-Systems mit $3T_0$ Verzögerung, um die Kausalität des Pre-Linearisierers zu gewährleisten.

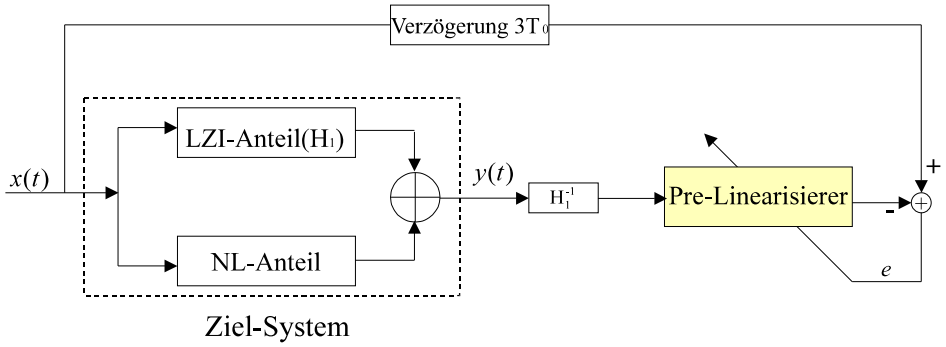


Abbildung 2.26: Aufbau zur Auffindung des Pre-Linearisierers

Durch Abb. 2.24, 2.25 und 2.26 können wir sehen, dass die Kompensationsaufgabe (Pre- und Post-Entzerrung, Pre- und Post-Linearisierung) in eine Identifikationsaufgabe umgewandelt wird.

2.6.3 Theoretische Überlegungen

Existenzproblem des Kompensationssystems

Nicht jedes NL-System kann kompensiert werden.

In Abb. 2.26 wird deutlich gezeigt, dass die Mindest-Voraussetzung für die Existenz eines Pre-Linearisierers darin besteht, dass der LZI-Anteil des Ziel-Systems und seine kausale inverse Funktion existieren müssen.

In Abb. 2.25 wird deutlich gezeigt, die Mindest-Voraussetzung für die Existenz eines Post-Linearisierers besteht darin, dass der LZI-Anteil des Ziel-Systems existiert.

Obwohl in Abb. 2.24 nicht deutlich gezeigt wird, was die notwendigen Voraussetzungen für die Existenz eines Post- und Pre-Entzerrers sind, zeigt die theoretische Analyse doch, dass der LZI-Anteil des Ziel-Systems und seine kausale inverse Funktion existieren müssen[Frank97, Kafka02].

Hierbei bedeutet *Mindest-Voraussetzung*, wenn das nichtlineare Ziel-System nur 3. Ordnung ist und ein schwaches nichtlineares System vorliegt, ist die Voraussetzung für die Kompensation noch ausreichend. Wenn das Ziel-System höher als 3. Ordnung oder ein stark nichtlineares System ist, ist es aber nicht sicher, dass das System kompensierbar ist. In dieser Arbeit nehmen wir an, dass das Ziel-System theoretisch kompensierbar ist, ungeachtet der Größe der Ordnung.

Abtastfrequenzproblem

In [Frank97] werden die notwendigen Abtastfrequenzen für unterschiedliche Anwendungen beim nichtlinearen System bis zur 3. Ordnung hergeleitet. In dieser Arbeit haben wir angenommen, dass die Abtastfrequenz genügend hoch ist, weil die GP sehr oft ein nichtlineares System über der 3. Ordnung behandelt.

Um den Begriff der Kompensation besser zu verstehen, werden im Folgenden ein paar Beispiele gezeigt.

Beispiel 1: Kompensierbares statisches System, seine inverse Funktion ist exakt beschreibbar in der Mathematik. z.B. $y = f(x) = x^3$, seine inverse Funktion (Kompensationsfunktion) $x = f^{-1}(y) = \sqrt[3]{y}$.

Beispiel[26] 2: Kompensierbares statisches System, seine inverse Funktion ist vorhanden in der Theorie aber nicht exakt beschreibbar. z.B. zeigt Gl[27].(2.17) eine Original-NL-Kennlinie, Gl.(2.18) zeigt eine Näherung für ihre inverse Funktion $\widehat{f}^{-1}$ aus der GP. Abb. 2.27 zeigt anschaulich diese Kennlinie und Kompensationswirkung. Der normierte Kompensationsfehler beträgt $\approx 2 \cdot 10^{-5}$.

$$y = f(x) = \frac{cos(cos(sin(cos(\frac{(x+0.9331)}{cos(sin(cos(sin(sin(x+6))))})) + 3)) + 0.5403) + x}{cos(sin(cos(1.086x + 6.5157) - 1))} \qquad (2.17)$$

[26]Das Beispiel ist aus einem Zwischenergebnis bei der Kompensation von nichtlinearen Verzerrungen

[27]Dafür ist es schwierig, die inverse Funktion in der Mathematik direkt zu ermitteln.

$$\widehat{x} = \widehat{f}^{-1}(y) = a_{17}(a_{15}sin(a_{13}sin(sin(a_{11}sin(sin(sin(a_9sin(sin(sin(a_7sin(a_5 (exp(a_4(sin(sin(a_1 - a_2y)))(sin(exp(a_3y)))) - a_6) - a_8)))) - a_{10}y))) - a_{12}y)) - a_{14}) - a_{16}y) \tag{2.18}$$

Wobei: $a_1 = 9.7457, a_2 = 1.0840, a_3 = 0.3139, a_4 = 1.0937, a_5 = 1.2555$
$a_6 = 7.0258, a_7 = 1.6055, a_8 = 3.0161, a_9 = 2.2767, a_{10} = 0.5884, a_{11} = 0.8617$
$a_{12} = 1.2827, a_{13} = 0.4395, a_{14} = -0.2633, a_{15} = 1.5041, a_{16} = 0.9759, a_{17} = -0.6246$

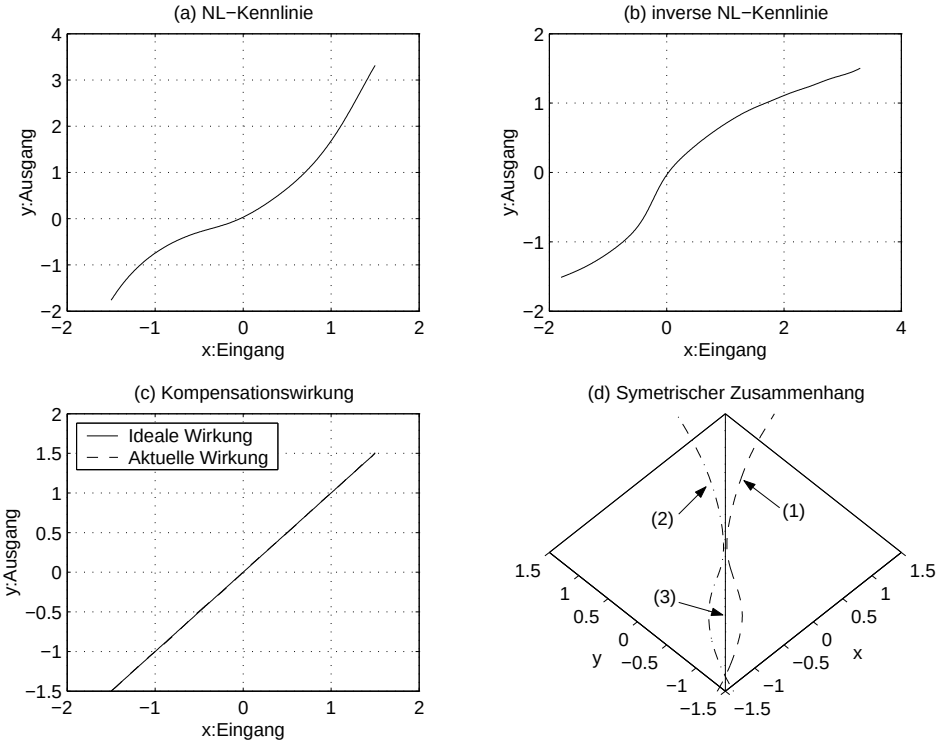


Abbildung 2.27: (a) Original-NL-Kennlinie, monoton im Bereich [-1.5,1.5]. (b) Kennlinie der inversen Funktion. (c) Kompensationswirkung. (d) der Zusammenhang dieser Kennlinie: (1)-Original-NL-Kennlinie, (2)-Inverse NL-Kennlinie, (3)-Grade $y = x$, Kennlinie (1) und (2) sind hinsichtlich $y = x$ symmetrisch.

Hierbei ist zu beachten:

1. Es ist sehr wahrscheinlich, dass die Gl.(2.18) nicht die beste Lösung für f^{-1} ist, wie sie z.B. hinsichtlich der Genauigkeit und Komplexität zum Ausdruck kommt.

2. Die Lösung ist nur wirksam im Bereich $x \in [-1.5, 1.5]$.

3. Das Beispiel zeigt nur die Fähigkeit der GP bei der Kompensation eines gedächtnislosen Systems.

Beispiel 3: Unkompensierbares statisches System, da seine inverse Funktion in der Theorie nicht vorhanden ist, wie z.B. $f = x^2$, theoretisch gibt es keine *Eins-zu-Eins*[28] Funktion f^{-1}.

Beispiel 4: Kompensierbares System mit endlich langem Gedächtnis, wie z.B. das in Gl.(2.19) gezeigte System. Tatsächlich ist dies ein kompensierbares Wiener-System mit der Kennlinie $(x + x^3)$.

$$\begin{aligned} y(i) &= [0.5x(i) + 0.3x(i-1) + 0.2x(i-2) + 0.1x(i-3)]^3 + \\ &\quad [0.5x(i) + 0.3x(i-1) + 0.2x(i-2) + 0.1x(i-3)]. \end{aligned} \tag{2.19}$$

Beispiel 5: Unkompensierbares System mit endlich langem Gedächtnis: wie z.B. das in Gl. (2.20) gezeigte System, dabei ist $x \in [-a, a]$, und $a \gg 1$, also ist das System ein stark nichtlineares System[29].

$$y(i) = 0.1x(i) + x^2(i-1). \tag{2.20}$$

Beispiel 6: Größtenteils kompensierbares System mit endlich langem Gedächtnis, wie z.B. das in Gl. (2.21) gezeigte System, dabei gilt $x \in (-1, 1)$, also ist das System ein schwach nichtlineares System.

$$y(i) = x(i) + 0.1x^2(i-1). \tag{2.21}$$

Durch die obige Beispiele können wir folgende Erkenntnisse gewinnen:

1. Die Nichtlinearität des Systems ist normalerweise stark vom zulässigen Eingangswertebereich des Systems abhängig. D.h. wenn der Eingangswertebereich vergrößert wird, ist es möglich, dass ein schwach nichtlineares System in ein stark nichtlineares System übergeht.

2. Normalerweise ist nur ein schwach nichtlineares System kompensierbar, aber manche stark nichtlinearen Systeme sind auch kompensierbar, wie das 1. und 4. Beispiel.

2.7 Dynamische Identifikation und Kompensation

Ein Zeitvariantes System ist ein System, das seine Übertragungseigenschaften über der Zeit verändert. Diese Veränderung über der Zeit besitzt dabei zwei Bedeutungen:

1. Die Parameter des zeitvarianten Systems verändern sich über der Zeit.

2. Die Struktur des zeitvarianten Systems verändert sich über der Zeit.

[28] Definition des Systems

[29] Natürlich ist es möglich, dass das System ein schwach nichtlineares System werden kann, wenn das Eingangssignal auf einem sehr kleinen Bereich beschränkt wird. Die Beurteilung des schwach oder stark nichtlinearen Systems ist also abhängig vom Wertebereich des Eingangssignals.

Im 1. Fall handelt es sich um ein adaptives System, während es sich im 2. Fall um ein *dynamisches* System handelt. Die Mehrheit der dynamischen Systeme ist in der Praxis nichtlinear.

Die Systemregeln entsprechen der Struktur des Modells. Um ein dynamisches System besser beschreiben zu können, sollte das Modell auch dynamisch sein. D.h. während die Systemregeln sich ändern, verändert sich auch die Modell-Struktur.

Bei der Identifikation eines nichtlinearen Systems spielt die Modell-Struktur eine sehr wichtige Rolle, weil die Modell-Struktur die Fähigkeit zur Beschreibung der Prozessdaten bestimmt, wenn der Algorithmus zur Schätzung der Parameter im Modell perfekt ist. Zurzeit gibt es noch keine allgemeine nichtlineare Modell-Struktur, die für alle Anwendungen geeignet ist.

Die wichtigste Methode zur nichtlinearen Systemidentifikation ist die sogenannte Black-Box Methode. Bei der Black-Box Methode liegt der schwierigste Teil nicht in der Schätzung der Parameter sondern in der Auswahl der geeigneten Modell-Struktur (Black-Box Modell-Struktur). Es gibt zahlreiche Black-Box Modell-Strukturen. Manche arbeiten für konkrete Aufgabe besser als andere. Beim Vorliegen eines konkreten Black-Box Modells entspricht die Anzahl der Parameter in diesem Modell der Ordnung oder Komplexität des Modells .

Eine typische Methode zur Wahl der Anzahl der Parameter eines Black-Box Modells ist es, zunächst eine größere Anzahl der Parameter auszuwählen und nach der ersten Schätzung der Parameter manche Parameter zu eliminieren.

Bei der Prozessidentifikation sucht die GP automatisch Schritt für Schritt eine geeignete Modellstruktur im Suchraum, die sehr große Freiheitsgrade (beliebige Nichtlinearität) besitzen kann. Grundsätzlich braucht die GP keine streng mathematischen Kenntnisse über das zu identifizierende System. Die Suche der Modell-Struktur ist die Stärke der GP. Daher ist es eine gute Idee, mit der GP ein dynamisches System zu identifizieren.

2.7.1 Aufbau zur dynamischen Identifikation mit der GP

Die Abb. 2.28 zeigt den System-Aufbau, mit dem die GP ein dynamisches System identifizieren kann.

Auf der Abb. 2.28 gibt es 2 Prozessoren. Der Prozessor 1 führt die Systemsimulation bzw. Prädiktion des dynamischen Systems gemäß dem eingeprägten Modell durch. Der Prozessor 2 analysiert bzw. überwacht den Fehler zwischen dem Systemausgang und dem Modellausgang. Die Aufgabe von Prozessor 2 ist es:

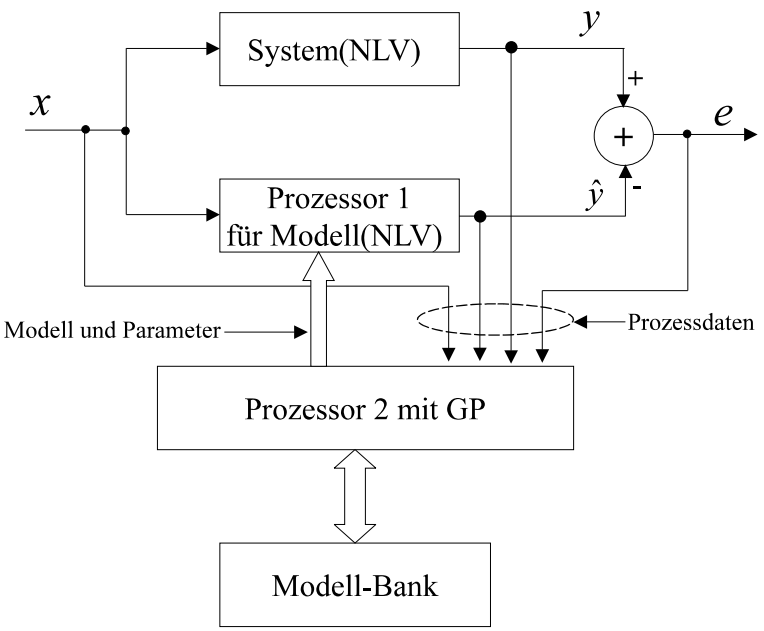


Abbildung 2.28: Aufbau zur dynamischen Identifikation mit der GP.

1. Aus den Ein- und Ausgangssignalen des dynamischen Systems ein Initial-Modell des dynamischen Systems zu bestimmen. In der Anlaufphase identifiziert er mindestens ein Modell des dynamischen Systems, das ihm in der Modell-Datenbank zur Verfügung steht.

2. Er verwaltet die Modell-Datenbank, in der die verschiedenen zur Situation passenden Modelle gespeichert sind.

3. Er generiert geeignete neue Modelle für den Prozessor 1 gemäß der Überwachung des ganzen Systems(dynamisches System und Modell), wenn sich das ganze System im Betrieb befindet und dynamisch verändert. Das könnte z.B. sein:

   (a) Ein vorhandenes Modell aus der Modell-Datenbank wird mit neuen Optimierungsparametern gemäß den aktuellen Ein- und Ausgangssignalen ausgestattet und auf Prozessor 1 geladen. Dieser Fall ist ähnlich wie bei den adaptiven Verfahren.

   (b) Alle in der Modell-Datenbank vorhanden Modelle werden in die GP-Population geladen. Danach läuft die GP mit diesen Initialisierungsmodellen und den aktuellen Prozessdaten wieder neu an, um das beste Modell zu finden und auf den Prozessor 1 zu laden. Dies ist der Vorgang der dynamischen Identifikation eines dynamischen nichtlinearen Systems mit Vorwissen-Initialisierung.

   (c) Ein neues Modell, gemäß der aktuellen Signale an den Ein- und Ausgängen wird mit Hilfe der GP bestimmt, und auf Prozessor 1 geladen. Dieser

Fall ist vergleichbar mit der Identifikation eines allgemeinen nichtlinearen Systems. Dies ist der Vorgang der dynamischen Identifikation eines dynamischen nichtlinearen Systems ohne Vorwissen-Initialisierung.

Wir können diesen Ablauf wie folgt zusammenfassen.

Am Anfang gibt es mindestens ein Modell für den Prozessor 1 in der Modell-Datenbank. Dieses Modell wird vom Prozessor 2 identifiziert. Mit diesem Modell kann das ganze System anlaufen. Während das ganze System läuft, überwacht der Prozessor 2 das Ergebnis des Systems.

Wenn der Fehler zwischen dem Systemausgang und dem Modellausgang größer als ein vorgegebener Wert ist, passt der Prozessor 2 zunächst durch die Überwachung der aktuellen Ein- und Ausgänge die Parameter in diesem Modell an, um die geforderte Genauigkeit der Simulation/Prädiktion zu erreichen. Wenn die geforderte Genauigkeit erfüllt ist, wird das Modell mit den neuen Optimierungsparametern auf Prozessor 1 geladen. Dieser Fall ist ähnlich wie bei den adaptiven Verfahren und zeitaufwandsarm. Er ist geeignet für die Zeitphasen des Systems, in denen das System zeitvariante Parameter und eine zeitinvariante Struktur besitzt.

Wenn diese Anpassung der Parameter die geforderte Genauigkeit nicht erfüllt, lädt die GP alle vorhandenen Modelle aus der Modell-Datenbank in die GP-Population. Danach läuft die GP mit diesen aktuellen Prozessdaten wieder an, und das beste Modell wird erneut auf den Prozessor 1 geladen, wenn die geforderte Genauigkeit erfüllt ist. Dabei ergibt sich ein Vorteil, d.h. die GP kann das von der GP automatisch erzeugte Vorwissen verwenden, weil die vorher funktionierenden Modelle die nützlichen Funktions-Blöcke für die neuen Zeitphasen möglicherweise umfassen. Dieser Fall ist geeignet für die Zeitphasen des Systems, in denen das System teilweise nach neuen Regeln arbeitet. In diesen Phasen bestimmt der Prozessor 2 das hauptsächliche Verhalten des Systems.

Wenn die Genauigkeit durch die obigen Maßnahmen nicht erfüllt sind, muss die GP ein ganz neues Modell mit den aktuellen Prozessdaten identifizieren. Dieser Fall ist wie eine Initialisierung ohne Vorwissen, d.h. es wird ein vollig neues Modell identifiziert. Dieser Fall passt für die Zeitphasen, in denen das System ein ganz neues Verhalten zeigt. Dieser Fall wird sehr selten eintreten, wenn die Anzahl der Modelle in der Modell-Datenbank sehr groß ist.

Durch diesen dynamischen Identifikationsprozess wird ein Datenfluss (wie z.B. von der Aktienbörse) automatisch segmentiert und für jedes Daten-Segment wird gleichzeitig ein geeignetes Modell erzeugt. Diese Segmentation ist sehr schwierig für den Menschen aber, wenn man vom Rechenaufwand absieht, sehr leicht für die GP.

Die Suche der Modell-Struktur gemäß den veränderlichen Prozessdaten ist die Stärke der GP. Die Voraussetzung für die Funktion des oben dargestellten Aufbaus

ist es, dass die GP eine hinreichend gute Identifikationsfähigkeit für jedes isolierte Segment hat. Diese Voraussetzung wird in Kapitel 4 und 5 genauer untersucht.

2.7.2 Aufbau zur dynamischen Kompensation mit der GP

In Abschnitt 2.6.2 wurden 4 Entwürfe der Kompensation dargestellt. Dadurch können wir sehen, dass das Kompensationssystem auf einem bestimmtem Ziel-Systemmodell basiert. Das bedeutet, dass das Ziel-System zeitinvariant sein muss. Aber in zahlreichen technischen Fällen kann es sein, dass das Ziel-System zeitvariant ist, weil sich die Bedingungen oder die Umgebung verändern. Zeitvariant besitzt dabei zwei Bedeutungen:

1. Die Parameter des Ziel-Systems sind zeitvariant.
2. Die Regeln des Ziel-Systems sind zeitvariant.

Folglich muss das Kompensationssystem auch zeitvariant sein, um eine bessere Kompensationswirkung zu erreichen. Im 1. Fall handelt es sich um ein adaptives System, während es sich im 2. Fall um eine *dynamische* Modellierung bzw. Kompensation handelt. Zur Vereinfachung beschäftigen wir uns nur mit der Situation, in der ein Ziel-System mit einem Wiener-Modell beschrieben werden kann.

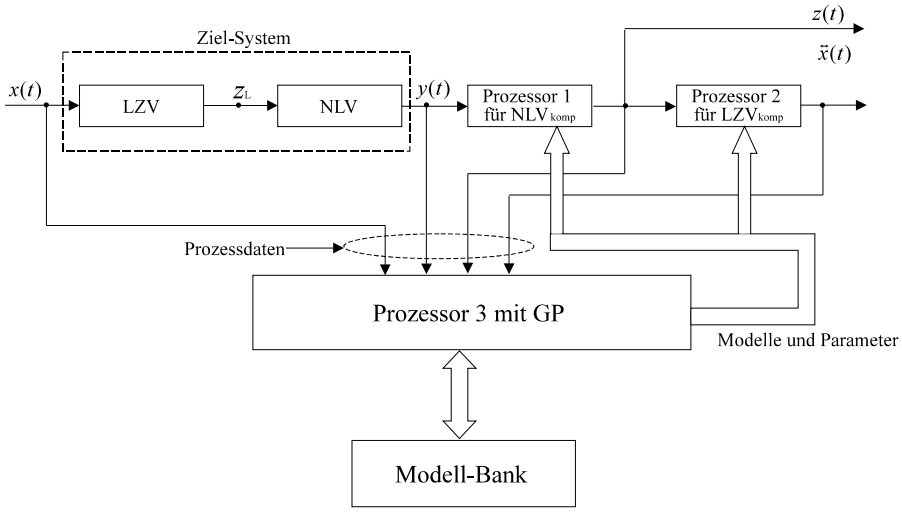


Abbildung 2.29: Aufbau zur dynamischen Kompensation, die auf einem Wiener-Modell basiert.

Abb. 2.29 zeigt einen Aufbau zur dynamischen Kompensation mit der GP. Dabei gibt es 3 Prozessoren. Der Prozessor 1 führt die Linearisierung des Ziel-Systems gemäß dem ihm entsprechenden Modell durch, d.h. er kompensiert die Nichtlinearität NL. Der Prozessor 2 führt die Entzerrung des Ziel-Systems gemäß dem vorgegebenen Modell durch. Der Prozessor 3 analysiert/überwacht das ganze System. Die Aufgaben von Prozessor 3 sind:

1. Aus den Ein- und Ausgangssignalen des Ziel-Systems bestimmt er ein Paar Initial-Modelle des Ziel-Systems und berechnet die entsprechenden Kompensationsmodelle. In der Anlaufphase identifiziert er mindestens ein Modellpaar des Ziel-Systems und ein dazugehöriges Kompensationsmodellpaar, das ihm in der Modell-Bank zur Verfügung steht.

2. Er verwaltet die Modell-Bank, in der die verschiedenen zur Situation passenden Modelle gespeichert sind.

3. Er generiert geeignete neue Modelle für den Prozessor 1 und 2 gemäß der Überwachung des ganzen Systems, wenn sich das ganze System im Betrieb befindet. Es könnte z.B. sein:

   (a) Ein vorhandenes Modell aus der Modell-Bank wird mit neuen Optimierungsparameter gemäß den aktuellen Ein- und Ausgangssignalen bestimmt und auf Prozessor 1 und 2 geladen. Dieser Fall ist ähnlich wie bei den adaptiven Verfahren.

   (b) Ein neues Modell, gemäß der aktuellen Signale an den Ein- und Ausgängen wird mit Hilfe der GP bestimmt.

4. Er ist verantwortlich für die Erstellung neuer Modelle, wenn es in der Modell-Bank keine geeigneten Modelle gibt.

Wir können diesen Ablauf wie folgt zusammenfassen.
Am Anfang gibt es mindestens ein Modellpaar für die Prozessoren 1 und 2 in der Modell-Bank. Mit diesen Modellen kann das ganze System anlaufen. Während das ganze System läuft, überwacht der Prozessor 3 das ganze System.

Einerseits passt der Prozessor 3 durch die Überwachung der aktuellen Ein- und Ausgänge die Parameter in diesem Modell an, um die geforderte Genauigkeit der Kompensation zu erreichen. Andererseits, wenn das Modell die Anforderung nicht erfüllen sollte, muss der Prozessor 3 ein neues Modell zur Kompensation aus den aktuellen Ein- und Ausgängen identifizieren. Die Prozessoren 1 und 2 führen anschließend die Kompensation mit den neuen Modellen durch. Gleichzeitig werden die neuen Modelle in der Modell-Bank gespeichert. Die Modell-Bank erweitert sich also, während das System läuft.

Die Identifikation eines neuen Modells ist zeitaufwendig, wenn das durchgeführte alte Modell die Anforderung nicht erfüllt, versucht der Prozessor 3 zuerst die Parameteroptimierung durchzuführen, dann versucht er das vorhandene Modell durch ein anderes aus der Modell-Bank zu ersetzen und erst dann versucht er die Identifikation eines ganz neuen Modells.

Daraus können wir klar ersehen, dass der Prozessor 3 den Kern des ganzen Systems bildet. Die Modellierung und die Parameteroptimierung sind die zwei wichtigsten Aufgaben dieses Prozessors. In Kapitel 6 werden diese beiden Aufgaben im Detail diskutiert, was den Schwerpunkt dieser Arbeit bildet.

Kapitel 3

Grundlagen der genetischen Programmierung

Seit den 50er Jahren beschäftigt man sich mit der Frage, wie Computer lernen können, ein Problem zu lösen, ohne explizit dafür programmiert zu werden.

Ab Mitte der 80er Jahre wurde dann erstmals versucht diese Idee mittels evolutionärer Prinzipien umzusetzen.

Einer der Vorreiter dieses Konzepts war bzw. ist John Koza[Koza92, Koza94, Koza99], dessen Arbeiten große Bedeutung erlangt haben.

Charles Darvin hat die Entwicklung der Gattungen und Arten der Lebewesen dieser Erde, die sogenannte Evolution, als Optimierungsprozess gedeutet: die tüchtigsten Individuen jeder Art haben die größte Chance, ihr Erbgut weiterzugeben. Dies wird heute Darwin'sches Prinzip der natürlichen Selektion genannt. Dadurch werden mit der Zeit die günstigsten Eigenschaften auf die ganze Art ausgebreitet.

Moderne genetische Theorien zeigen: das Erbgut ist in den DNA-Molekülsträngen codiert. Generation für Generation wird durch die Vererbung und allmähliche Veränderung die günstigere genetische Information in den DNA-Molekülsträngen an Nachkommen weitergegeben.

Die codierten DNA-Molekülsträngen sind sehr ähnlich wie die codierten binären Informations-Ketten im Computer bzw. in Computerprogrammen. Diese Ähnlichkeit bildet die Basis der evolutionären Algorithmen (EA).

3.1 Evolutionäre Algorithmen (EA)

Evolutionäre Algorithmen [Baeck97, Fogel00] modellieren die natürliche Vererbung und Evolution im Computer nach, also die Entwicklung hin zu komplexeren Organis-

men aus zeitlich früheren und damit meist einfacheren Lebensformen. Evolutionäre Algorithmen arbeiten mit einer Population von Individuen. Jedem Individuum der Population wird dabei eine bestimmte Fitness zugeordnet. In diesem Fitnessmaß ist codiert die Aufgabe enthalten, die der evolutionäre Algorithmus lösen soll. Je besser die Lösung ist, die das Individuum beschreibt, desto besser ist dessen Fitness.

Allgemeinen lassen sich in einem evolutionären Algorithmus drei Stufen unterscheiden:

1. Zu Beginn wird eine Population mit zufällig erzeugten Individuen initialisiert.
2. Probabilistisch werden in jeder Generation Individuen aus der Population ausgewählt und bezüglich ihrer Fitness miteinander verglichen. Nur auf die Individuen mit der jeweils besten Fitness werden nach dem Selektionsprinzip die folgenden genetischen Operationen angewendet:
   (a) Reproduktion durch identisches Kopieren in die neue Population.
   (b) Mutation einzelner zufällig ausgewählter Stringpositionen.
   (c) Austauschen von Teilstrukturen ausgewählter Individuen (Kreuzung).

   Durch Anwendung dieser an die Natur angelehnten Operationen wird die Population in die nächste Generation überführt. Dabei bleibt normalerweise die Populationsgröße konstant.
3. Ist das Abbruchkriterium hinsichtlich des Fitnessmaßes erfüllt, bildet das Individuum mit dem besten Fitnesswert in der Population das Ergebnis des Algorithmus, d.h. es wurde eine approximative Lösung der Aufgabe gefunden.

Ein Abbruch erfolgt in der Regel nach Erreichen einer maximalen Generationsanzahl oder eines bestimmten besten Fitnesswertes(Erfolgskriterium). Das allgemeine Schema verdeutlicht den probabilistischen Charakter evolutionärer Algorithmen, der mehrere unabhängige Versuche nötig macht, um ein zuverlässiges Ergebnis für ein Problem zu erzielen.

Über die Generationen nimmt die Fitness der Individuen in der Population zu. Dies begründet sich einmal darin, dass Individuen mit guter Fitness häufiger selektiert und reproduziert werden als andere, und zum anderen in der Tatsache, dass die Rekombination zweier Elternindividuen mit guter Fitness mit großer Wahrscheinlichkeit Nachkommen entstehen lässt, die gleich gute oder noch bessere Fitnesswerte aufweisen.

EA umfassen eine Menge der verschiedensten Algorithmen, die auf dem evolutionären Prinzip basieren. Dabei kann man EA in Genetische Algorithmen (GA), Genetische Programmierung (GP), Evolutionstrategien (ES) und Evolutionäre Programmierung (EP) unterteilen. Der größte Unterschied besteht darin, dass GA, GP,

ES und EP verschiedene Repräsentationen der Lösung verwenden und verschiedene Ziele erreichen. Die Repräsentationsform von GA ist z.B. eine codierte binäre Kette und die Repräsentationsform von GP sind durchführbare Computerprogramme.

Es gibt viele unterschiedliche Anwendungsmöglichkeit von EA:

1. Planung: z.B. Bei dem Problem eines Handelsreisenden (Travelling-Salesman-Problem), der die Aufgabe hat, nach der kürzesten Verbindung zwischen mehreren Städten zu suchen, liegt ein EA-fähiges Problem vor.

2. Handel und Finanzierung.

3. Entwurf von elektrischen oder digitalen Systemen. (Evolutionäre Hardware EHW).

4. Modellierung und Identifikation.

5. Regelung: z.B. Agenten und Robotersteuerung.

6. Klassifikation: z.B. Mustererkennung.

7. Bild- und Signalverarbeitung.

8. Kunst.

Obwohl es so viele Varianten und verschiedene Anwendungen von EA gibt, ist der Ablauf der evolutionären Algorithmen doch weitestgehend identisch. Abb. 3.1 zeigt den Prozess der evolutionären Algorithmen. Aus dieser Abbildung kann man klar ersehen, dass der Kern von EA erstens die Fitnessberechnung, zweitens die Selektion gemäß der Fitness und drittens die Fortpflanzung in die nächste Generation mit entsprechenden genetischen Operatoren ist.

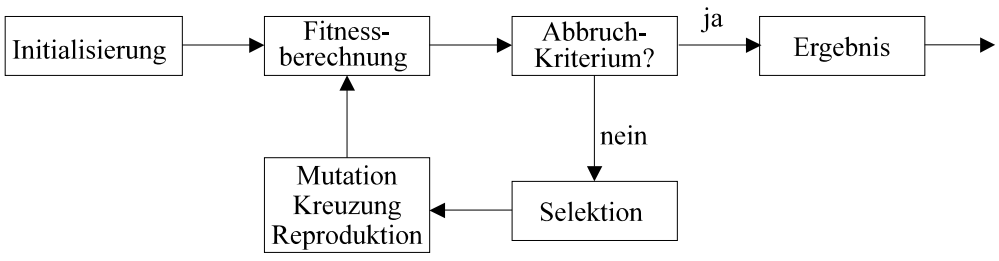


Abbildung 3.1: Ablauf der evolutionären Algorithmen.

Für diese drei Stufen der EAs gibt es viele unterschiedliche Schemata der Realisierung. Im folgenden Abschnitt werden die wichtigsten Schemata, die man bei GP verwenden kann, ausführlich dargestellt.

3.2 Grundkonzept der GP

Genetische Programmierung ist ein von Koza entwickelter Zweig der Evolutionären Algorithmen, speziell der genetischen Algorithmen, mit dem Ziel der automatischen Programmerzeugung. Computer sollen hier in die Lage versetzt werden, sich weitgehend selbst zu programmieren.

Grundsätzlich gehört die GP zum Gebiet der Suchalgorithmen. Wenn GP eine Aufgabe lösen möchte, sucht sie die Lösung in einem bestimmten Suchraum, der von der Terminalmenge, der Funktionenmenge und anderen Parametern gebildet wird.

Normalerweise ist der Suchraum eine riesige Zahl, z.B. von 10^{23} verschiedenen Programmen selbst bei der Verwendung eines einfachen Kerns[Horner96]. Dies bedeutet, dass die Wahrscheinlichkeit sehr klein ist, durch eine rein zufällige Suche eine Lösung zu finden. In der Tat ist GP viel schneller als eine zufällige Suche.

GP ist also nicht eine zufällige Suche, obwohl es viele Zufälligkeiten in der GP gibt. Der wesentliche Unterschied zwischen GP und konventionellen Such-Algorithmen (z.B. Zufalls-Suche, Gradienten-orientierte Suche) besteht darin, dass die GP das evolutionäre Prinzip verwendet und dieses kann zu ganz verschiedenen Ergebnissen führen. Dabei spielen die Fitnessfunktion, die Selektionsoperatoren und die genetischen Operatoren eine wichtige Rolle.

3.2.1 Funktionenmenge und Terminalmenge

Die Menge aller möglichen ausführbaren Programme, die sich aus einer bestimmten Menge von Funktionen und Terminalen zusammensetzen lassen, bildet den Suchraum der genetischen Programmierung. Die Terminalmenge umfasst die Menge aller erlaubten Eingaben für die Funktionen, und wird mit F bezeichnet. Die Funktionenmenge F setzt sich im Allgemeinen aus

1. arithmetischen Operationen (wie +, -, *, /),
2. mathematischen Funktionen (wie sin, cos, usw.),
3. boolschen Operationen (wie AND, NOT, XOR, usw.),
4. Verzweigungen (wie IF-THEN-ELSE usw.),
5. Iterationen (wie FOR, DO-UNTIL, usw.),

zusammen.

Elemente der Teminalmenge (T) sind:

1. Konstanten.

2. Variablen.

Wird als Repräsentation der Programme eine hierarchische Baumstruktur gewählt, stehen die Funktionen an den inneren Knoten des Baumes und die Terminale an den Blättern.

Die Funktionenmenge und die Terminalmenge müssen die folgenden Merkmale erfüllen.

1. Angemessenheit: Mit den vorhandenen Elementen der beiden Mengen muss es möglich sein ein Programm zu finden, das das Problem löst.

2. Abgeschlossenheit: Jede Funktion der Funktionenmenge sollte als Argument jeden Wert zulassen, den beliebige andere Funktionen der Funktionenmenge potentiell ergeben können, sowie jeden Wert den beliebige Konstanten oder Variablen der Terminalmenge annehmen können.

Außerdem sollen die Funktionenmenge und die Terminalmenge die folgenden Merkmale erfüllen, damit mit höherer Wahrscheinlichkeit eine erfolgreiche Lösung zu finden ist:

1. Ein gewisses Vorwissen des Anwenders über die Gestalt der Lösung sollte an die GP weitergegeben werden.

2. Die beiden Mengen sollten nicht unnötig gross gewählt werden.

Dies wird normalerweise durch die Wahl der sogenannten GP-Grammatik implementiert. Das Vorwissen und seine Implementierung in der GP bilden den Schwerpunkt für die Anwendung der GP in verschiedenen Fachgebieten.

3.2.2 Fitnessfunktion

Die GP sollte über die Fitnessfunktion ein möglichst *kontinuierliches und flaches* Feedback erhalten, um sich der gesuchten Lösung inkrementell annähern zu können. Eine Verbesserung der Fitness steht dann im Allgemeinen in direkter Beziehung zu einer Verbesserung im Verhalten des betreffenden Individuums. Die Fitnessfunktion ist also eine von den zu lösenden Problemen abhängige Funktion. Hierbei ist zu beachten:

1. Die Fitnessunktion ist ein Messmaß für die Fitness, also ist sie identisch für alle Individuen in der Population. Übrigens wirkt die Fitnessfunktion ähnlich wie die Umgebung in der Natur.

2. Ein Fitnesswert ist ein für ein Individuum berechneter konkreter Wert.

Je nach Anwendung kann die Fitness auf sehr unterschiedliche Weise gemessen werden. Der unmittelbar aus der jeweiligen Anwendung abgeleitete Fitnesswert wird als Rohfitness r bezeichnet. Häufig arbeitet man jedoch nicht mit der Rohfitness, sondern transformiert diesen Wert in andere Größen.

Rohfitness bzw. standardisierte Fitness

Der ursprünglich ermittelte Fitnesswert ist im Allgemeinen ohne prinzipielle Begrenzung. Er ist je nach Aufgabenstellung und Art der Berechnung zu minimieren oder zu maximieren.

Für die meisten Probleme lässt sich die Rohfitness eines Individuums definieren als die Summe der Fehler (Abstände) zwischen den GP-Ergebnissen und den tatsächlichen Ergebnissen. Je geringer die Summe ist, desto besser ist das Individuum. Die folgende Gleichung ist ein Beispiel für eine Rohfitnessfunktion:

$$r(i,k) = \sum_{j=1}^{\mathrm{N}_e} \parallel \mathrm{C}(i) - \mathrm{Z}(j) \parallel \tag{3.1}$$

Wobei:
$r(i,k)$: Die Rohfitness des Individuums i in der Generation k.
i: i-tes Individuum.
k: k-te Generation.
j: j-tes Trainingsbeispiel.
N_e: Anzahl der Trainingsbeispiele.
$\parallel \cdot \parallel$: Abstand.
$\mathrm{C}(i)$: Die Ausgabe des Individuums i.
$\mathrm{Z}(j)$: Die Ausgabe des Trainingsbeispiels j.

Für andere Probleme kann die Fitness mit Hilfe einer Punktzahl ausgedrückt werden, z.B. im Sinne von richtig erkannten Fällen aus einer Menge von Trainingsbeispielen. Hier würde eine hohe Punktzahl einer guten Fitness entsprechen.

Die Standardisierte Fitness ergibt sich durch eine Transformation aus der Rohfitness und wird mit $s(i,k)$ bezeichnet. Sie wird durch die Gl.(3.2) definiert.

$$s(i,k) = \begin{cases} r(i,k) & \text{falls min. } r(i,k) \text{ gut ist} \\ r_{max} - r(i,k) & \text{falls max. } r(i,k) \text{ gut ist} \end{cases} \tag{3.2}$$

Wobei:
$s(i,k)$: Die standardisierte Fitness des Individuums i in der Generation k.
r_{max}: Die mögliche maximale Fitness in der Generation k.

Für die standardisierte Fitness gilt:

1. Die Standardisierte Fitness $s(i,k) \in [0,\infty)$.

2. Je kleiner die standardisierte Fitness ist, desto besser ist das Individuum.

Ein Individuum erreicht dann im besten Fall den standardisierten Fitnesswert 0.

Adjustierte Fitness

Die adjustierte Fitness ist eine Größe, die durch eine weitere Transformation aus der standardisierten Fitness hervorgeht. Sie wir mit $a(i,k)$ bezeichnet. Sie ist in Gl. (3.3) definiert.

$$a(i,k) = \frac{1}{1+s(i,k)} \tag{3.3}$$

Wobei:
$a(i,k)$: Die adjustierte Fitness des Individuums i in der Generation k ist.

Es gilt:

1. Die adjustierte Fitness $a(i,k) \in [0,\ 1]$.

2. Je größer die adjustierte Fitness ist, desto besser ist das Individuum.

Normalisierte Fitness

Die normalisierte Fitness ist eine transformierte, adjustierte Fitness und wird mit $n(i,k)$ bezeichnet. Sie wird durch die Gl. (3.4) definiert.

$$n(i,k) = \frac{a(i,k)}{\sum_{j=1}^{\mathrm{M}} a(j,k)} \tag{3.4}$$

Wobei:
$n(i,k)$: Die normalisierte Fitness des Individuums i in der Generation k.
M: Die Populationsgröße.

Es gilt:

1. Normalisierte Fitness $n(i,k) \in [0,\ 1]$.

2. $\sum_{i=1}^{\mathrm{M}} n(i,k) = 1$

3. Je größer die normalisierte Fitness innerhalb einer Population ist, desto besser ist das Individuum.

Das wichtigste Ziel der Fitnessberechnung ist es, eine Wahrscheinlichkeit zur Auswahl der Eltern beim GP-Durchlauf anzubieten. Es ist ersichtlich, dass die normalisierte Fitness $n(i,k)$ direkt als diese Wahrscheinlichkeit betrachtet werden kann. Wenn man also von Fitness redet, so meint man normalerweise die normalisierte Fitness.

3.2.3 Auswahl der Fitnessfunktion

Die Fitnessfunktion ist eine vom zu lösenden Problem abhängige Funktion. Dafür ist ein kontinuierlicher und flacher Verlauf wichtig. Um die flache Charakteristik zu erfüllen, kann man, wenn die Ableitung der 1. Ordnung von C und Z bekannt ist, die Gl.(3.1) in die Gl. (3.5) umschreiben, weil die 1. Ableitung bzw. höhere Ableitungen die flache Eigenschaft des Systems darstellen.

$$r(i,k) = \sum_{j=1}^{N_e} (\parallel \mathrm{C}(i,j) - \mathrm{Z}(j) \parallel + \gamma \cdot \parallel \mathrm{C}'(i,j) - \mathrm{Z}'(j) \parallel) \tag{3.5}$$

Wobei $\gamma > 0$ ein Parameter ist, mit dem der sensitive Grad dargestellt wird. Und C', Z' sind die 1. Ableitung von C und Z.

Außerdem können zusätzliche Faktoren, wie z.B. die Programmgröße, in die Fitnessberechnung einfließen, um eine möglichst kompakte Lösung zu bekommen, obwohl die Programmgröße die Komplexität der Lösung nicht genau beschreiben kann.

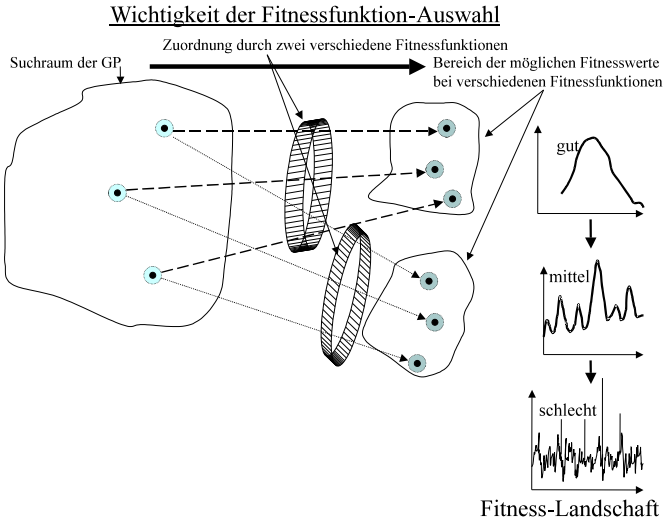


Abbildung 3.2: Fitnesslandschaft verschiedener Fitnessfunktionen. Die rechte Seite der Abbildung zeigt drei mögliche Fitnesslandschaft.

Für ein bestimmtes Problem gibt es unterschiedliche Fitnessfunktionen, die unterschiedliche Wirkungen in der GP erzeugen. Die allgemeine Folgerung ist: je flacher die Fitnesslandschaft ist, desto günstiger ist die Fitnessfunktion für die GP-Durchläufe. Abb. 3.2 zeigt diesen Fall. Durch eine Fitnessfunktion wird der Suchraum der GP in einem Bereich den Fitnesswerten zugeordnet. Die verschiedenen Zuordnungen ergeben unterschiedliche Verteilungseigenschaften der Fitnesswerte. Dies heißt die Fitness-Landschaft. Die rechte Seite der Abbildung zeigt drei Beispiele für die Fitness-Landschaften.

3.2.4 Dynamisierung der Fitnessfunktion

Die Fitnessfunktion hat eine ähnliche Wirkung wie die Umgebung in der Natur. Bei der Evolution in der Natur verändert sich die Umgebung immer, wenn lange Zeit vergeht. Die Dynamisierung der Fitnessfunktion simuliert diese Erscheinung in der Natur. Dynamisierung der Fitnessfunktion bedeutet, dass in der GP verschiedene Fitnessfunktionen verwendet werden, wenn die GP verschiedene Generationen durchläuft. In Kapitel 4 verwenden wir z.B. die Dynamisierung der Fitnessfunktion, um das Overfitting zu überwinden.

Abb. 3.3 zeigt diese Idee mit dem Vergleich zwischen der Natur und der GP. Die linke Seite dieser Abbildung zeigt den evolutionären Prozess in der Natur. Die rechte Seite der Abbildung zeigt die Dynamisierung der Fitnessfunktion in der GP.

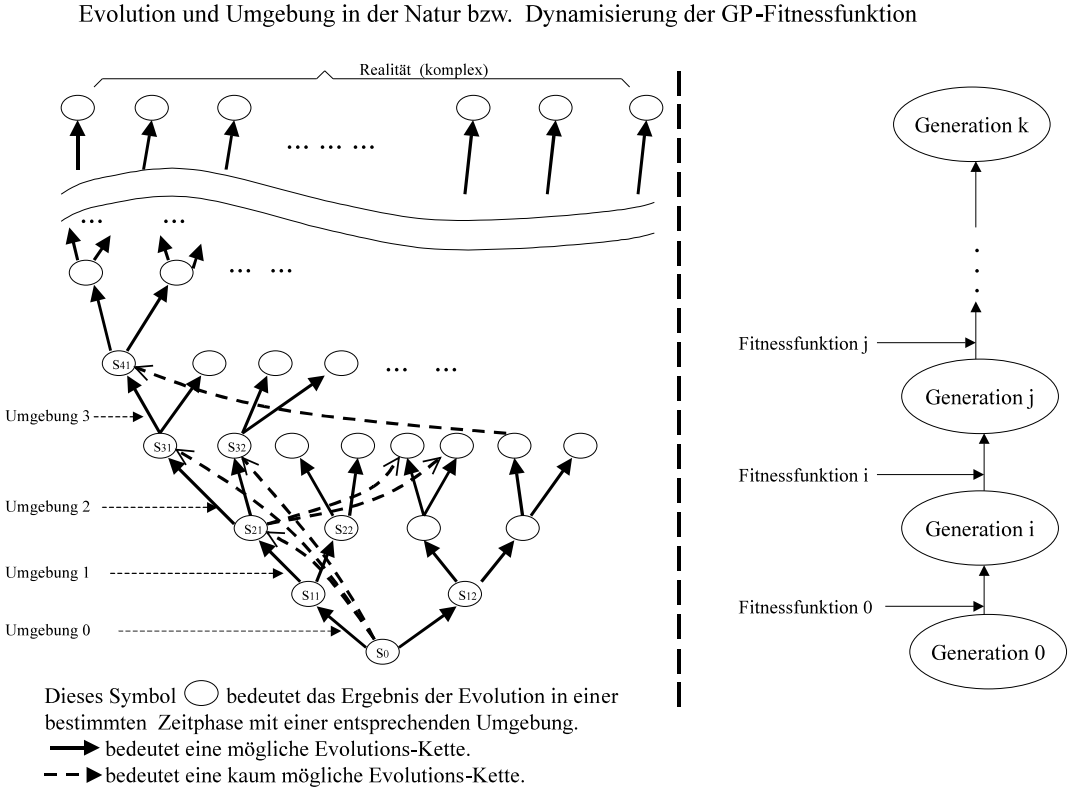


Abbildung 3.3: Dynamisierung der Fitnessfunktion

Der ganze evolutionäre Prozess wird in viele Sub-Prozesse eingeteilt. Jeder Sub-Prozess hat ein lokales Ziel, das durch eine geeignete lokale Umgebung durchgeführt wird. Wenn ein Sub-Prozess z.B. in der Stufe i fertig ist, d.h. das lokale Ziel für diesen Sub-Prozess erreicht ist, dann folgt der nächste Sub-Prozess z.B. $i+1$. Die Reihenfolge der Sub-Prozesse ist nicht umkehrbar, d.h. die Ergebnisse im Sub-Prozess i bilden die Basis für die weitere Evolution im nächsten Sub-Prozess $i+1$. Mehrere

Sub-Prozesse bilden eine evolutionäre Kette. Am Ende erreichen die mehrstufigen evolutionären Prozesse (Kette) das endgültige Ziel.

3.2.5 Selektionsoperator

Selektion ist eine Strategie, die auf der Wahrscheinlichkeit basiert. Dabei werden die Eltern in einer Population zur Fortpflanzung ausgewählt. Der Selektionsoperator ist ein Name für die gesamte Selektionsstrategie. Die Selektionsmethode muss sicherstellen, dass die Individuen in der Population eine möglichst gute Fitness erreichen. Dazu ist es notwendig, dass gute Individuen in der Population häufiger, aber nicht ausschließlich, reproduziert werden.

Fitness-Proportionale Selektion

Bei der Fitness-Proportionalen Selektion werden die Eltern mit der Wahrscheinlichkeit $p_i = n(i,k)$ ausgewählt. Wobei $n(i,k)$ die normalisierte Fitness gemäß Gl. (3.4) ist. Dann folgt ein entsprechender genetischer Operator. Die Fitness-Proportionale Selektion ist der wichtigste Selektionsoperator.

Über-Selektion

Bei der Über-Selektion werden die Eltern mit der gewichteten Wahrscheinlichkeit von $n(i,k)$ ausgewählt. Dann folgt ein entsprechender genetischer Operator. Die Gewichtungsfunktion soll folgende Voraussetzungen erfüllen:

Je besser das Individuum ist, desto größer ist sein Gewichtswert.

Durch die Wirkung der Über-Selektion haben die besseren Individuen überproportional bessere Chancen zur Fortpflanzung als bei der proportionalen Selektion. Die schlechteren Individuen besitzen dann überproportional geringere Chancen zur Fortpflanzung als bei der proportionalen Selektion. Eine typische Über-Selektion ist:

1. Die ganze Population wird in zwei Gruppen unterteilt. Die 1. Gruppe besitzt die besten Individuen, die z.B. 32% der Fitnesssumme der ganzen Population besitzen. Und die 2. Gruppe besitzt die anderen Individuen.
2. Für die 1. Gruppe wird die Wahrscheinlichkeit mit der Konstanten z.B. 0.8/0.32 gewichtet, und für die 2. Gruppe wird die Wahrscheinlichkeit mit der Konstanten 0.2/0.68 gewichtet, wie in Gl.(3.6) gezeigt.

$$p(i,k) = \begin{cases} \frac{0.8}{0.32} \cdot n(i,k) & \text{für 1. Gruppe.} \\ \frac{0.2}{0.68} \cdot n(i,k) & \text{für 2. Gruppe.} \end{cases} \tag{3.6}$$

Wobei:
$n(i,k)$: Normalisierte Fitness des Individuums i in der Generation k.
$p(i,k)$: Gewichtete Wahrscheinlichkeit des Individuums i in der Generation k.

Rang-Selektion

Bei der Rang-Selektion werden die Eltern mit der gewichteten Wahrscheinlichkeit von $n(i,k)$ ausgewählt. Dann folgt ein entsprechender genetischer Operator. Die Gewichtungsfunktionen basieren auf den Ordnungen der Fitness der Individuen. Mit den Ordnungen werden die Individuen in Populationen sortiert. Für die lineare Rang-Selektion ist die Wahrscheinlichkeit eine lineare Funktion vom Rang R_i, wie in Gl.(3.7) gezeigt wird.

$$p_i = \frac{1}{\mathrm{M}}[p^- + (p^+ - p^-)\frac{\mathrm{R}_i - 1}{\mathrm{M} - 1}] \tag{3.7}$$

Wobei:
R_i: Der Rang für das Individuum i, $i = 1, 2, \cdots \mathrm{M}$. M: Die Populationsgröße. $\frac{p^-}{\mathrm{M}}$: Die Wahrscheinlichkeit, mit der das schlechteste Individuum in der Population ausgewählt werden soll. $\frac{p^+}{\mathrm{M}}$: Die Wahrscheinlichkeit, mit der das beste Individuum der Population ausgewählt werden soll. p_i: Die erneut berechnete Wahrscheinlichkeit für das Individuum i.
Die Gl. (3.7) zeigt, dass p_i nicht direkt von $n(i,k)$ abhängig ist, aber der Rang R_i ist von $n(i,k)$ abhängig. Für p^- und p^+ soll Gl. (3.8) erfüllt werden.

$$p^- + p^+ = 2 \tag{3.8}$$

Die Wirkungen der Rang-Selektion sind:

1. Die besten Individuen in der Population werden relativ unterdrückt, im Gegensatz dazu werden die schlechtesten Individuen relativ erhobt.

2. Die Individuen lassen sich besser unterschieden.

Abb. 3.4 zeigt den Unterschied zwischen der Fitness-Proportionalen Selektion, der Über-Selektion und der Rang-Selektion. Aus dieser Abbildung kann man klar ersehen, dass der einzige Unterschied in der Gewichtenfunktion besteht.

Wettkampf-Selektion

Die Wettkampf-Selektion basiert nicht auf der Konkurenz der ganzen Population sondern auf der Konkurenz einer Sub-Menge der Population. Dieser Prozess ist im Folgendem dargestellt.

1. Bestimmte Individuen, d.h. die Wettkampf-Menge, werden mit gleicher Wahrscheinlichkeit *zufällig* aus der ganzen Population ausgewählt.

2. Für diese Wettkampf-Menge wird die Fitness der Individuen berechnet.

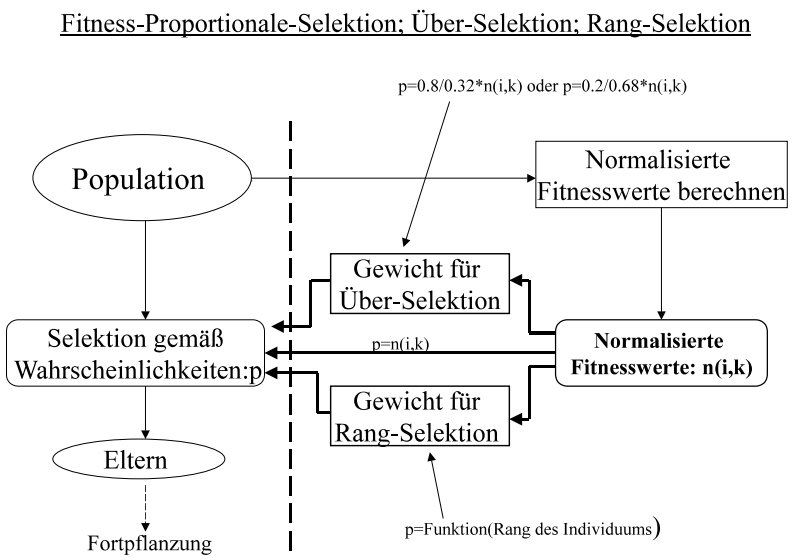


Abbildung 3.4: Fitness-Proportionale Selektion, Über-Selektion und Rang-Selektion

3. Die Selektion zur Fortpflanzung dieser Wettkampf-Menge ist *gleich* wie bei den oben erwähnten Selektionsstrategien.

4. Eine neue Submenge, die die gleiche Anzahl von Individuen wie die Wettkampf-Menge besitzt, entsteht.

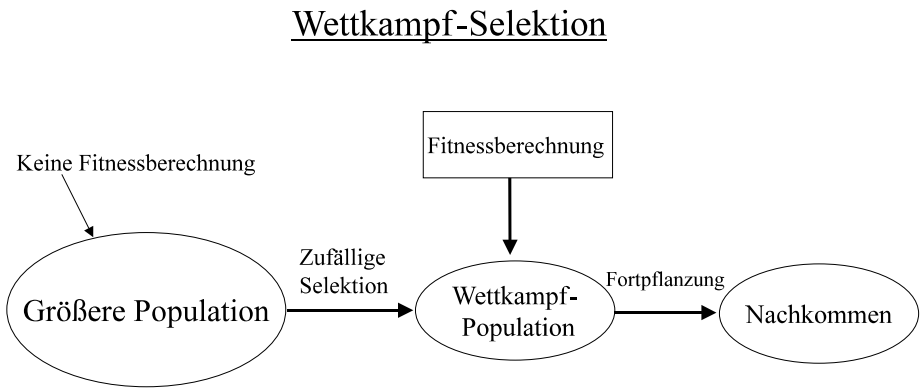


Abbildung 3.5: Wettkampf-Selektion

Abb. 3.5 zeigt den Selektions-Prozess bei der Wettkampf-Selektion.

Die Vorteile der Wettkampf-Selektion sind:

1. Sie passt zu parallelen Algorithmen.

2. Durch die Auswahl der Anzahl der Wettkampf-Individuen kann man den Selektionsdruck regulieren.

Elite-Selektion

In vielen Anwendungen wird das beste Individuen der aktuellen Generation auf jeden Fall in die Population der nächsten Generation übernommen. Man bezeichnet diese Vorgehensweise als Elite-Selektion oder Elitismus.

Durch Elite-Selektion ist gewährleistet, dass die besten bisher gefundenen Lösungen eines GP-Laufes nicht mehr aus der Population verlorengehen.

3.2.6 Genetische Operatoren

Ein Genetischer Operator ist eine Methode, mit der die Nachkommen von Eltern oder Elternteilen erzeugt werden. Genetischer Operator ist ein Name für alle Fortpflanzungsvorgänge von Eltern und Nachkommen. Davon sind die Fitnessproportionale-Reproduktion, die Kreuzung und die Mutation besonders wichtig.

Fitnessproportionale-Reproduktion

Die Fitnessproportionale-Reproduktion ist einfach ein Kopieren von ausgewählten Individuen in die nächste Generation. Für die Reproduktion gibt es keinen Unterschied zwischen Eltern und Nachkommen. Für die Reproduktion braucht die GP nur einen Elternteil. Die Fitnessproportionale-Reproduktion ist wichtig, um sicherzustellen, dass das beste Individuum in der nächsten Generation nicht schlechter ist als das beste Individuum in der letzten Generation. Der Prozess der Reproduktion ist in der folgenden Abb. 3.6 dargestellt.

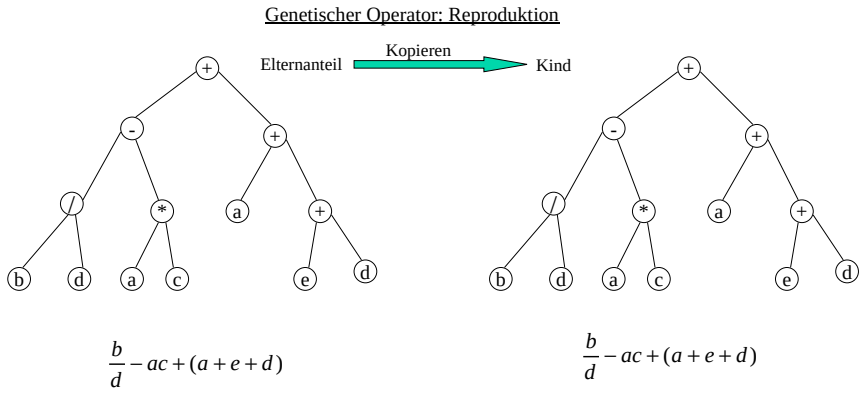


Abbildung 3.6: Genetischer Operator: Reproduktion

1. Der Elternteil wird durch die obige Selektionsmethode mit der Reproduktionswahrscheinlichkeit p_r z.B. 10% ausgewählt.

2. Der Elternteil wird in die nächste Generation kopiert, wie es in Abb.3.6 gezeigt wird.

Kreuzung

Für die Kreuzung braucht die GP zwei Eltern. Nach der Kreuzung entstehen zwei Nachkommen. Die Eltern und die Nachkommen sind normalerweise unterschiedlich. Der Prozess der Kreuzung ist im Folgenden dargestellt:

1. Zwei Eltern werden durch die obige Selektionsmethode mit der Kreuzungswahrscheinlichkeit p_k z.B. 90% ausgewählt.

2. Der Kreuzungspunkt für jeden Elternteil wird zufällig[1] ausgewählt.

3. Die Teilbäume der Eltern werden miteinander ausgetauscht.

4. Es entstehen zwei Nachkommen, wie dies Abb. 3.7 zeigt.

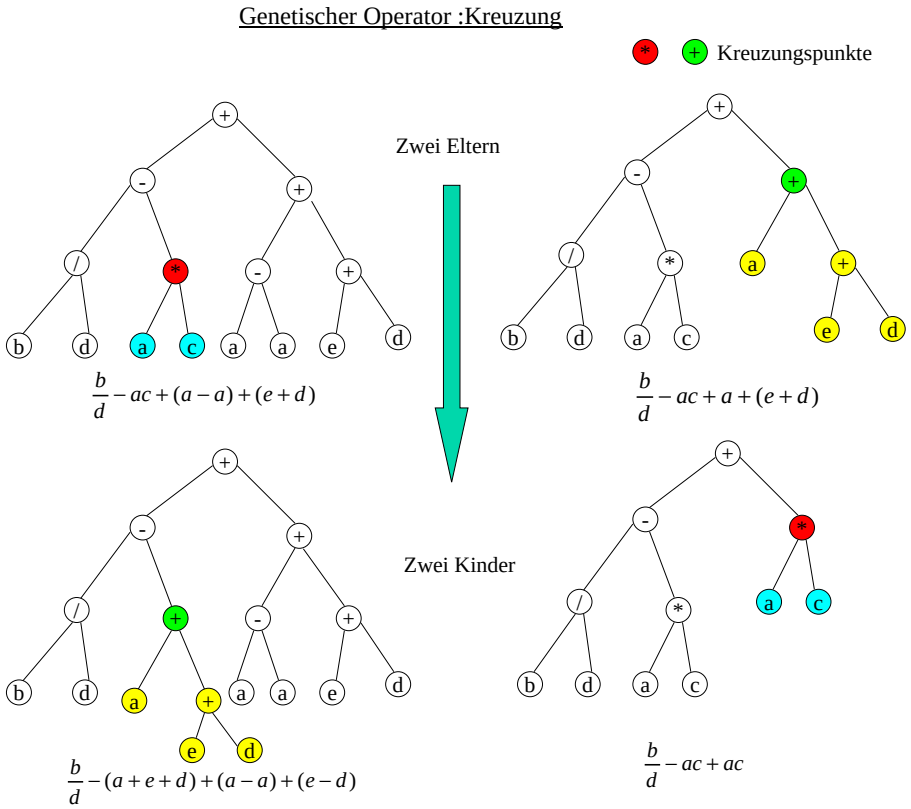


Abbildung 3.7: Genetischer Operator: Kreuzung

[1]Normalerweise werden innere Knoten mit der Wahrscheinlichkeit p_{ip}=90% , und außere Blätter mit der Wahrscheinlichkeit p_{ep}=10% ausgewählt.

Mutation

Für die Mutation braucht die GP nur ein Elternteil. Nach der Mutation entsteht nur ein Nachkomme. Der Elternteil und der Nachkomme sind normalerweise unterschiedlich. Der Prozess der Mutation ist im Folgenden dargestellt:

1. Der Elternteil wird durch die obige Selektionsmethode mit der Mutationswahrscheinlichkeit p_m z.B. 10% ausgewählt.
2. Der Mutationspunkt im Elternteil wird zufällig[2] ausgewählt.
3. Ein neues Individuum wird zufällig erzeugt[3].
4. Der Teilbaum im Elternteil wird durch dieses Individuum ersetzt.
5. Ein Nachkomme ist entstanden, wie es in Abb.3.8 gezeigt wird.

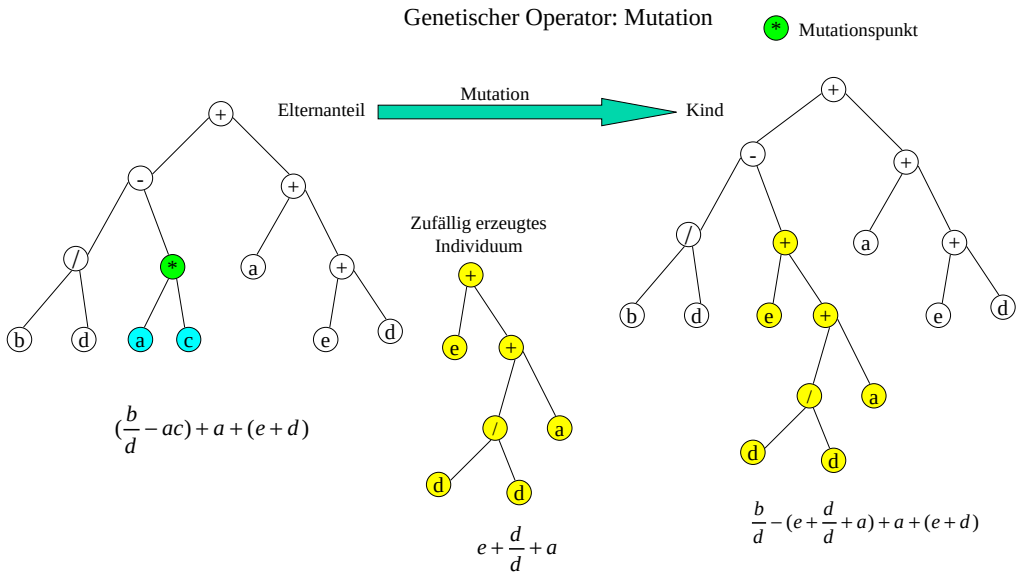


Abbildung 3.8: Genetischer Operator: Mutation

3.2.7 Automatisch Definierte Funktionen (ADF)

Wie wir wissen, ist es möglich und oft vorteilhaft komplexe Problemstellungen in Teilprobleme zu zerlegen und zunächst die einzelnen Teilprobleme zu lösen (modulare Programmierung). Dies führt oft zu übersichtlicheren und kürzeren Programmen,

[2]Normalerweise werden innere Knoten mit der Wahrscheinlichkeit p_{ip}=90% , und außere Blätter mit der Wahrscheinlichkeit p_{ep}= 10% ausgewählt.

[3]Die Methode ist entspricht der Initialisierung der GP

weil Teilfunktionen wiederholt aufgerufen werden können.

Deswegen wurden, um die Effizienz von GP bei komplexen Problemstellungen zu erhöhen, verschiedene Modularisierungskonzepte vorgeschlagen, von denen die "Automatisch Definierten Funktionen" (ADF) am gebräuchlichsten sind. ADF's sind inzwischen eine Standarderweiterung vieler GP-Systeme.

Dabei geht es darum modular aufgebaute Programme zu generieren, die aus einem Hauptprogramm und den ADF's bestehen, wobei das Hauptprogramm die generierten ADF's nicht zwingend aufrufen muss.

Es werden außerdem nur die konkreten Inhalte des Hauptprogramms und der ADF's mit evolutionären Mechanismen generiert, d.h. es muss zuvor schon ein grobes Gerüst festgelegt werden. Insbesondere, wieviele Parameter einer Funktion übergeben werden.

3.2.8 Kontexfreie Grammatik (CFG)

Zuerst müssen wir festhalten, dass CFG für die GP nicht unbedingt notwendig ist. Das bedeutet, dass man mittels der CFG oder ohne die CFG gleiche GP-Programme implementieren kann. Die GP-Programme mit der CFG bieten eine Möglichkeit, mit der man die Programmstruktur eines Individuums teilweise kontrollieren kann.

Eine formale Sprache ist eine Menge von Zeichenketten, die durch eine Grammatik erzeugt werden. Eine Grammatik ist sowohl das mathematische System zur Definition einer formalen Sprache, als auch ein Satz von Regeln zur Feststellung der syntaktischen Gültigkeit eines konkreten Satzes.

Eine kontextfreie Grammatik ist ein 4-Tupel $G = \{N, T, P, S\}$.
wobei,
N: die Menge der Nichtterminalsymbole,
T: die Menge der Terminalsymbole, und
P: die Ableitungsregel $N \times (N \cup T)$ ist.
S: ist ein besonderes Nichtterminalsymbol, auch Startsymbol genannt.

Ein legaler Satz fängt bei CFG immer mit dem Startsymbol S an, dann kann das Startsymbol irgendwelche Symbole aus der Terminalmenge und der Nichtterminalmenge ableiten.

1. Falls das abgeleitete Symbol ein Terminalsymbol ist, wird der Ableitungsprozess geschlossen.

2. Falls das abgeleitete Symbol ein Nichtterminalsymbol ist, wird der Ableitungsprozess gemäß der Ableitungsregel P weiter geführt. Der Prozess läuft bis zum Ende, so dass alle Symbole nicht mehr weiter ableitbar sind.

Dabei ist ersichtlich, dass der Erzeugungsprozess eines legalen Satzes bei CFG und der Prozess der GP-Initialisierung fast identisch sind. Das ist der Grund, warum GP-Programme mittels CFG leichter implementiert werden können. Dabei gilt:

1. Die CFG-Terminalmenge entspricht der GP-Terminalmenge.

2. Die CFG-Nichtterminalmenge entspricht der GP-Funktionenmenge.

3. Die CFG-Abteilungsregel P ist ähnlich wie die GP-Abgeschlossenheit.

4. Das CFG-Startsymbol S hat keine Bedeutung bei der GP. Tatsächlich ist es auch nur ein Anfangs-Zeichen für alle legalen Sätze bei der CFG. Außerdem gibt es keine andere Bedeutung mehr.

Wenn ein GP-Programm durch CFG implementiert wird, nennt man dies CFG-GP. Durch den geeigneten Entwurf der Ableitungsregel P in der CFG liefert die CFG-GP eine Möglichkeit, den Suchraum der GP zu beschränken.

3.3 Aufbau der GP

3.3.1 Aufbau der GP

Ein vollständiges GP-Programm sollte mindestens die folgenden Funktionenblöcke enthalten, d.h. die Funktionenmenge, die Terminalmenge, die Fitnessfunktion, die Genetischen Operatoren und die Selektionsoperatoren. Außerdem benötigt die GP noch ein paar wichtige Steuerungsparameter, wie Populationsgröße, maximale Anzahl der Generationen, maximale Tiefe der von der GP erzeugten Programme und die Wahrscheinlichkeiten für die verschiedenen genetischen Operatoren.

Abb. 3.9 zeigt die Bestandteile der GP bzw. das logische Flussdiagramm für die Problemermittelung. Im oberen Teil dieser Abbildung kann man sehen:

1. Die Aufgabenstellung und die Aufgabenanalyse, dann folgt eine Entscheidung, d.h. ist die GP für die Aufgabe geeignet? Wenn die Entscheidung "Ja" lautet, dann entstehen die GP-Durchläufe.

2. Nach den GP-Durchläufen liefert die GP eines oder mehrere Ergebnisse, dann folgt eine weitere Entscheidung, d.h. hat das Ergebnis aus der GP die Aufgabe erfüllt? Wenn die Entscheidung "Ja" lautet, kommt die Anwendung. Wenn die Entscheidung "Nein" lautet, gibt es zwei Möglichkeiten, 1. Neue GP-Durchläufe werden initiert oder 2. Die Aufgabenanalyse wird neu gemacht.

Der untere Teil der Abb. 3.9 zeigt die allgemeinen Bestandteile der GP-Durchläufe. Die Besonderheit dieses Teils ist von der Aufgabe abhängig. Die wichtigste Aufgabe nach der Aufgabenanalyse ist es, diese GP-Bestandteile zu bestimmen, weil die nutzlosen Elemente in der Terminalmenge oder der Funktionenmenge die Suchfähigkeit der GP vermindern.

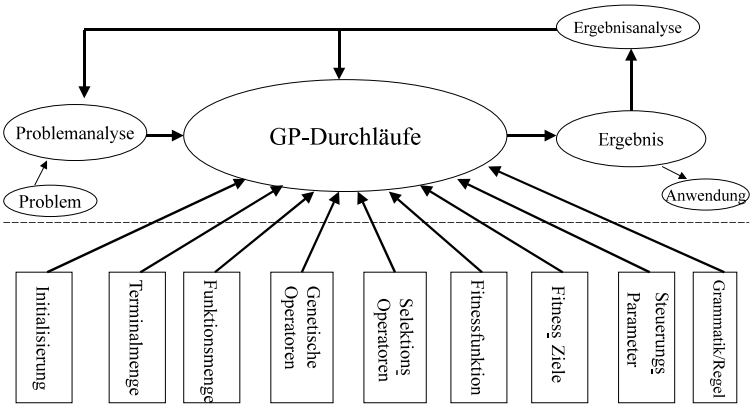


Abbildung 3.9: Bestandteile der GP

3.3.2 Die zwei bedeutendsten Entwurfsmethoden von GP-Durchläufen

Es gibt viele unterschiedliche Entwurfsmethoden von GP-Durchläufen. Die zwei bedeutendsten Entwurfsmethoden sind die GA-Entwurf basierten Methoden und die ES-Entwurf basierten Methoden. Abb. 3.10 a) zeigt die Methoden des GA-Entwurfs, b) die Methoden des ES-Entwurfs.

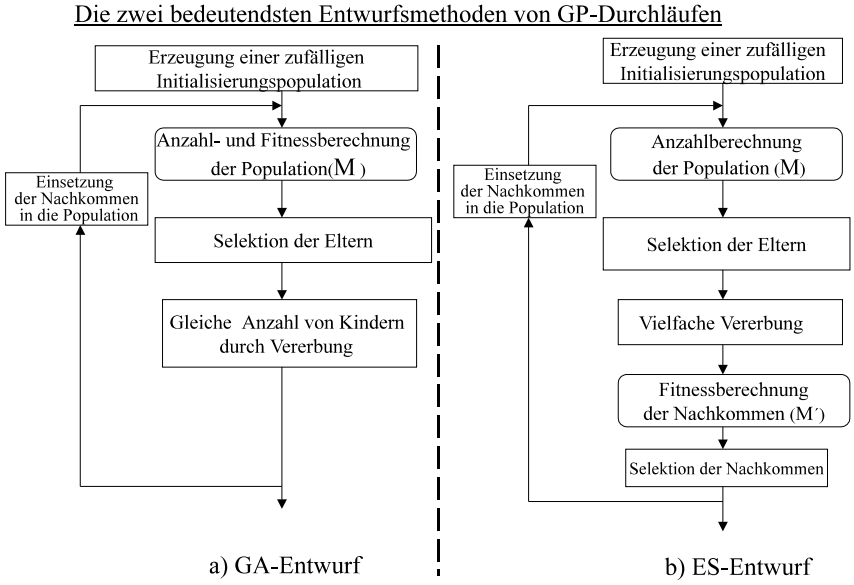


Abbildung 3.10: Die zwei bedeutendsten Entwurfsmethoden für GP-Durchläufe

Der wichtigste Unterschied besteht in der Fortpflanzungsmethode der Nachkom-

men. Beim GA-Entwurf erzeugen die Eltern immer die gleiche Anzahl an Nachkommen. Beim ES-Entwurf erzeugen die Eltern mehr Nachkommen als die Eltern. Nach dieser mehrfachen Fortpflanzung folgt eine einfache Selektion, d.h. nur die besten Individuen in der Nachkommenschafft werden in die nächste Generation eingesetzt, damit die Populationsgröße M sich nicht verändert. Abb. 3.11 zeigt anschaulich den Unterschied zwischen dem GA-Entwurf und dem ES-Entwurf mit Kreuzung.

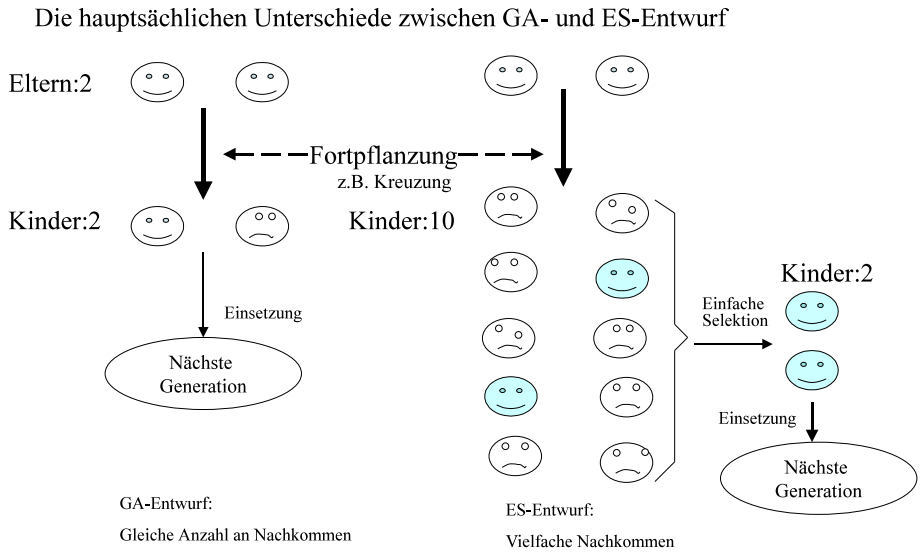


Abbildung 3.11: Unterschied zwischen dem GA-Entwurf und dem ES-Entwurf.

3.4 Durchführung des GP-Laufes

Um die GP lauffähig zu machen, braucht man einige Vorbereitungen und folgende Schritte:

3.4.1 Vorbereitung des Programmlaufs

1. Bestimmen der Terminalmenge T.
2. Bestimmen der Funktionenmenge F.
3. Bestimmen einer geeigneten Fitnessfunktion.
4. Bestimmen der Steuerparameter $p_r, p_k, p_m, p_{ip}, p_{ep}$, M, G_{max}, $D_{initial}$, $D_{created}$.
5. Bestimmen des Abbruchkriteriums.
6. Bestimmen der Fitness-Ziele und der Trainingsdaten.

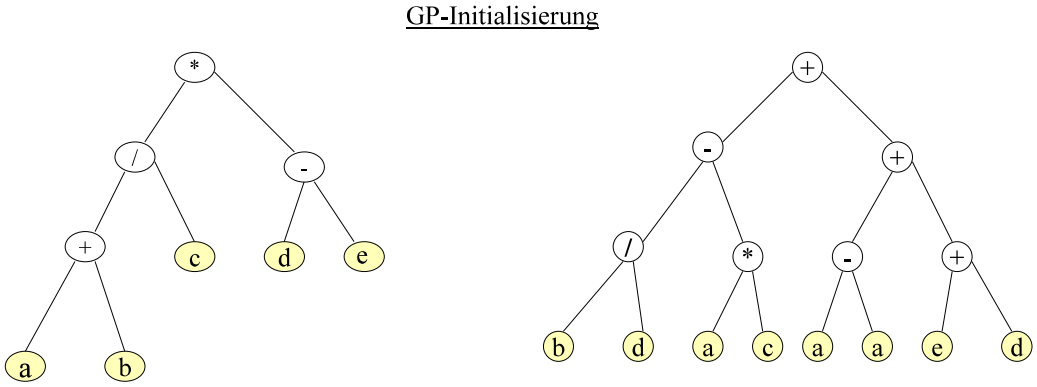
3.4.2 Initialisierung

GP-Initialisierung bedeutet, eine Anfangspopulation wird zufällig erzeugt. Dafür gibt es drei Methoden:

1. Volle Methode: Jeder Baumzweig aller baumbasierten erzeugten Programme muss bis zur maximalen initialisierten Tiefe $D_{initial}$ erweitert werden, wie in Abb. 3.12 (b) gezeigt.

2. Wachsende Methode: Jeder Baumzweig aller baumbasierten erzeugten Programme muss nicht bis zur maximalen initialisierten Tiefe $D_{initial}$ erweitert werden, sondern nur bis zu einem Blatt(ein Element der Terminalmenge), wie in Abb.3.12 (a) gezeigt.

3. Ramped halb-und-halb Methode: Zuerst wird die Anfangspopulation mit der Tiefe 2, 3, ..., $D_{initial}$ gruppiert, dann werden die Programme in jeder Gruppe halb mit voller Methode, halb mit wachsender Methode erzeugt. Die Tiefe ist die maximale Länge des Baumzweigs der baumbasierten erzeugten Progamme. Wobei Annahme: T = {a, b, c, d, e} ; F={+, -, *, / } ; $D_{initial} = 4$; T $\bigcup$ F ={a, b, c, d, e, +, -, *, /}.



(a) Wachsende Methode

(b) Volle Methode

Abbildung 3.12: GP-Initialisierung.

3.4.3 Fitnessberechnung

Die Fitnessberechnung wird gemäß Gl.(3.1), Gl.(3.3) und Gl.(3.4) gemacht. Sie bildet die Basis[4] der Selektion. Dies ist ein zeitaufwendiger Prozess in der GP.

[4]Die normalisierte Fitness kann direkt oder nach bestimmter Verarbeitung (siehe Über-Selektion und Rang-Selektion) in die Selektionswahrscheinlichkeit umgesetzt werden

3.4.4 Überprüfung des Abbruchkriteriums

Für das Abbruchkriterium gibt es normalerweise drei Situationen.

1. Eine vollständig korrekte Lösung wurde gefunden.
2. Eine approximierte Lösung mit vorgegebener Genauigkeit wurde gefunden.
3. Die maximale Anzahl an Generationen wurde erreicht.

Davon bedeuten die 1. und die 2. Situation, dass ein erfolgreicher GP-Durchlauf stattgefunden hat, und die 3. Situation, dass der GP-Durchlauf erfolglos war.

3.4.5 Bilden der neuen Population

Bilden einer neuen Population ist ein Prozess, in dem eine neue Generation durch Selektionsoperatoren und genetische Operatoren gebildet wird und der folgende Schritte umfaßt.

1. Reproduktion mit der Wahrscheinlichkeit p_r (siehe Reproduktion).
2. Kreuzung mit der Wahrscheinlichkeit p_k (siehe Kreuzung).
3. Mutation mit der Wahrscheinlichkeit p_m (siehe Mutation).
4. zurück zur Fitnessberechnung.

Dabei wird nur ein GP-Durchlauf dargestellt. Durch eine Schleife kann man mehrere GP-Durchläufe durchführen.

Dieser Prozess ist in Abb. 3.13 gezeigt.

3.5 Weitergehende Diskussion über die GP

3.5.1 Vergleich der Intelligenz von GP und der menschlichen Intelligenz

GP ist ein Prozess wie bei allen anderen EAs. GP ist auch eine relativ allgemeine Methode. Für eine konkrete komplizierte Aufgabe garantiert GP einerseits nicht, dass sie die Lösung (in einer bestimmten Zeit und in entsprechendem Umfang) findet, anderseits bietet sie eine Möglichkeit an, dass die Lösung der GP besser ist als die Lösung, die der Mensch findet. Das heißt *"human competitive Intelligenz"* von GP.

Abb. 3.14 verdeutlicht diesen Begriff. A zeigt grundsätzlich einen Bereich, in dem die Lösung einer Aufgabe möglicherweise liegen kann. B zeigt einen Bereich, in dem der Mensch die Lösung gut suchen oder finden kann. Wegen der Beschränkung der menschlichen Kenntnisse und Fähigkeiten ist der Bereich B kleiner als der Bereich A. C zeigt einen Bereich, in dem die GP die Lösung gut suchen oder finden kann. Es gilt E=B$\cap$C und D=B$\cup$C. Wenn die Lösung einer Aufgabe:

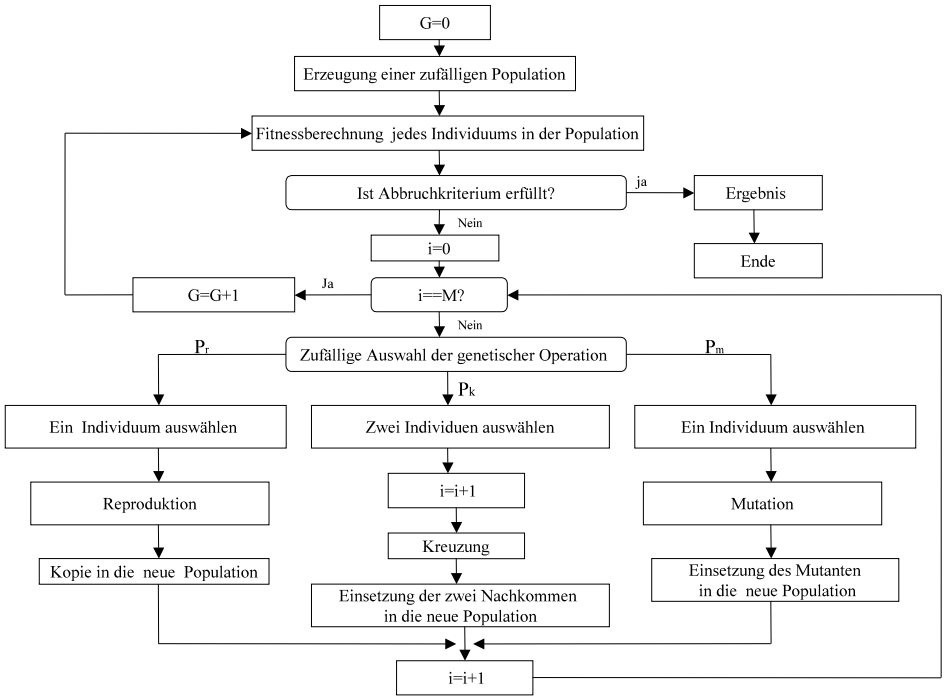


Abbildung 3.13: Ein Beispiel: Ablauf der GP. G ist der Generationszähler, i ist Populationszähler, M ist Populationsgröße.

1. nicht im Bereich D liegt, kann weder der Mensch noch die GP die Lösung finden.
2. im Bereich B liegt, kann der Mensch die Lösung finden.
3. im Bereich C liegt, kann die GP die Lösung finden.
4. im Bereich E liegt, können der Mensch und die GP die Lösung finden.
5. im Bereich F liegt, kann die GP die Lösung finden, aber der Mensch kann die Lösung nicht finden.

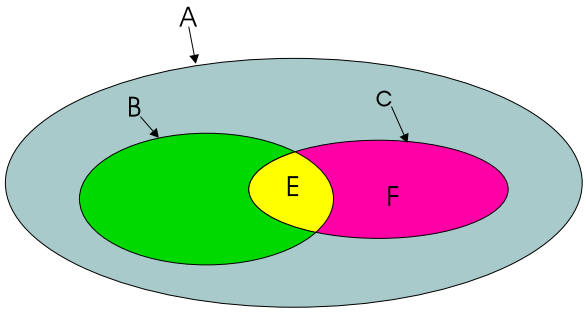


Abbildung 3.14: Human competitive Intelligenz

Aus dieser Abbildung 3.14 können wir ersehen:

1. Die GP kann manche Aufgabe automatisch lösen, die vorher der Mensch nur sehr aufwendig zu lösen vermochte.

2. Die GP kann den Raum des menschlichen Denkens erweitern. D.h. es ist möglich, dass die GP eine Lösung einer Aufgabe findet, die vorher der Mensch nicht lösen konnte.

3.5.2 Konvergenz-Problem der GP

GP ist ein Suchalgorithmus und sie hat einen theoretischen Suchraum. Bei endlicher Zeit (Anzahl der Generationen) sowie endlichem Umfang (Populationsgröße) und begrenzter Anzahl der Durchläufe ist es möglich, dass die GP die beste Lösung im ganzen Suchraum nicht findet. Das heißt Konvergenz-Problem der GP.

Die wichtigste Ursache des Konvergenz-Problems der GP ist Frühreife (*Premature*). Eigentlich ist die Premature ein biologischer Begriff. Premature bedeutet hier, dass ab einem bestimmten Durchlauf auch bei verlängerter Laufzeit es nahezu unmöglich ist, das Ergebnis zu verbessern. Premature ist ähnlich wie das Festhängen an einer lokalen Lösung bei einer globalen Optimierungsaufgabe.

Mit der Abb. 3.15 können wir diesen Begriff klarer darstellen. Dabei bezeichnet der Bereich A den theoretischen Suchraum der GP. Im Bereich A gibt es viele kleine Bereiche, die die verschiedenen Durchläufe bilden. In jedem kleinen Bereich gibt es einen Konvergenz-Punkt. Dieser Punkt ist wahrscheinlich nicht die beste Lösung für den ganzen Bereich A.

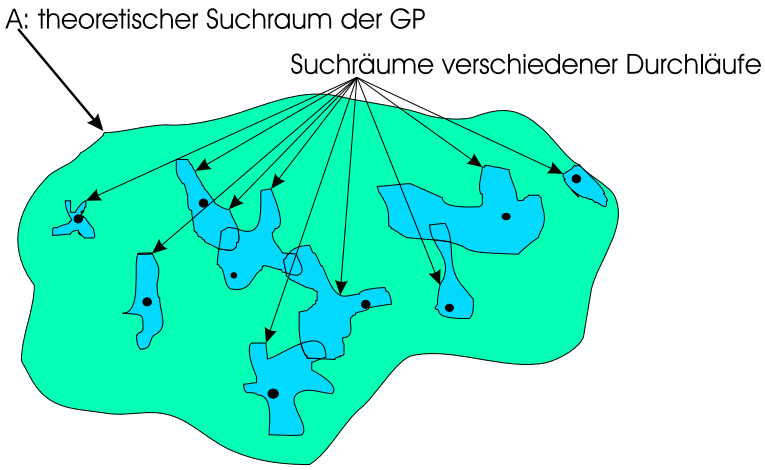


Abbildung 3.15: Konvergenz-Problem der GP: Premature

Aus dieser Abbildung können wir gleichzeitig auch sehen, welches die wichtigsten Gegenmaßnahmen gegen die Premature sind:

1. Die Populationsgröße vergrößern.

2. Die Wahrscheinlichkeit der Mutation vergrößern.

3. Den theoretischen Suchraum verkleinern.

4. Die Fitnessfunktion verbessern, um die Fitness-Landschaft "glatter" zu machen.

Aber die Vergrößerung der Populationsgröße verursacht einen Anstieg des Rechenaufwands. Dazu gibt es einige Erfahrungswerte für die Populationsgröße, z.B. Populationsgröße=500, 1000, 5000, 10000[Banzhaf98]. Gemäß der Komplexität einer konkreten Aufgabe sollte man die Populationsgröße geeignet wählen. Außerdem kann es besser sein, dass die GP mehrere unabhängige Durchläufe macht. Aber die Voraussetzung ist, dass die GP eine geeignete minimale Populationsgröße erreichen kann.

Die Vergrößerung der Wahrscheinlichkeit der Mutation verursacht einen Anstieg der Zufälligkeit. Die Geschwindigkeit der Konvergenz wird dabei langsamer. Zu große Wahrscheinlichkeit der Mutation zerstört das Prinzip der GP, d.h. das evolutionäre Prinzip. Die Erfahrungswerte der Wahrscheinlichkeit der Mutation liegen bei 0.05–0.1. In dieser Arbeit nutzen wir 0.3, wenn die Populationsgröße 10000 ist. Je größer die Populationsgröße ist, desto größere Wahrscheinlichkeiten für die Mutation kann man normalerweise verwenden .

Die Verkleinerung des theoretischen Suchraums benötigt Vorwissen zu einer Aufgabe. Theoretisch gilt: je größer das Vorwissen ist, desto kleiner kann man den theoretischen Suchraum werden lassen.

Die Verbesserung der Fitnessfunktion spielt auch eine wichtige Rolle. Aber es gibt keine allgemeine Regel für die Definition der Fitnessfunktion. D.h. die Fitnessfunktion ist abhängig von der zu lösenden Aufgabe oder von der besonderen Beschreibungsform der GP. In Kapitel 4 entwickeln wir eine Kombinationsfitnessfunktion, die die standardisierte Fitnessfunktion und AKF/KKF kombiniert, um das Overfitting zu überwinden. In [Nikolay01] wird eine Fitnessfunktion, die auf der Beschreibungsform der Basisfunktion basiert, entwickelt.

3.5.3 Der Mensch hilft der GP die Lösung zu finden

In den obigen Abschnitten haben wir dargelegt, dass die GP nicht allmächtig ist. Der riesige theoretische Suchraum beinhaltet zwar die Möglichkeit, dass die GP eine gute Lösung finden kann, aber Premature beschränkt die Suchfähigkeit der GP im ganzen theoretischen Suchraum, d.h. es gibt keine Sicherheit dafür, dass die GP die beste Lösung in diesem ganzen theoretischen Suchraum findet.

Alle beliebigen Kombinationen aus der Terminalmenge und der Funktionenmenge bilden den theoretischen Suchraum der GP. Für ganz unbekannte Probleme (Black-Box) ist die Suche in diesem Raum notwendig, für andere Probleme (Grey-Box) ist das nicht notwendig, weil man in diesem Fall schon weiß, dass manche Kombinationen unsinnig sind. Ein kleinerer aber die Lösung enthaltender Suchraum ist immer günstig für die GP. Das Vorwissen über eine Aufgabe, das dem Menschen zur Verfügung steht, bildet eine Möglichkeit, den Suchraum zu verkleinern.

Abb. 3.16 zeigt die Idee, dass der Mensch der GP hilft. Dabei bezeichnet der Bereich A den theoretischen Suchraum der GP. Der Bereich B ist eine Sub-Menge von Bereich A. Die Größe des Bereichs B wird durch das Vorwissen über die gestellte Aufgabe bestimmt. Je mehr Vorwissen da ist, desto kleiner wird dieser Bereich B. In diesem Fall wenn die GP abläuft, sucht sie die Lösung nur im Bereich B. Normalerweise realisiert man die Beschränkung des Suchraums der GP durch eine bestimmte vorgegebene Grammatik. Z.B. für die Identifikation der LZI-Systeme verwendet man eine Funktionenmenge wie {+, -, *, /, sin, cos, ...}, aber es ist ersichtlich, dass nicht jede Funktion in der Funktionenmenge die Elemente in der Terminalmenge ableiten kann, weil die Individuen, wie z.B. $sin(x(t-i))$, $x(t-i)\cdot x(t-j)$, $x(t-i)/x(t-j), i \neq j$ usw., ungültig sind, weil sie nicht linear sind. Nur die Individuen, die die Form von Gl. (3.9) besitzen, sind gültig.

$$\sum_{i=1}^{M} a_i x(t-i) + \sum_{j=1}^{N} b_j y(t-j) \tag{3.9}$$

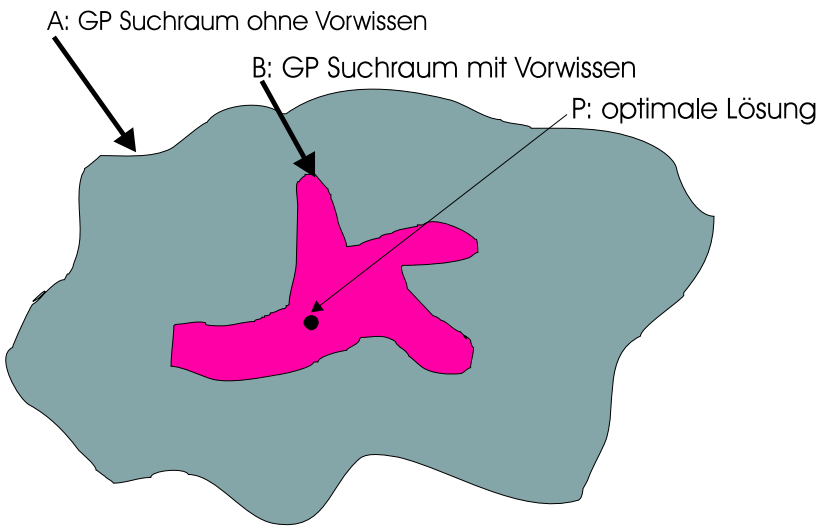


Abbildung 3.16: Verkleinerung des Suchraums der GP mit Hilfe des Menschen.

3.5.4 GP unterstützt die menschliche Entscheidung

Wenn Die Lösung einer Aufgabe sehr versteckt ist, d.h. der Suchraum sehr groß ist, können der Mensch und auch die GP mit gewissem Aufwand die Lösung nicht finden. Die GP hat aber ein wichtiges Merkmal, d.h. obwohl sie bei sehr großem Suchraum die perfekte Lösung für eine Aufgabe nicht finden kann, findet sie doch eine oder mehrere "Ersatz-Lösungen". Jede Ersatz-Lösung der GP bildet einen möglichen Lösungsraum, worin die optimale Lösung der Aufgabe liegen kann. In diesen verkleinerten Lösungsräumen kann man dann weiter erfolgreich suchen.

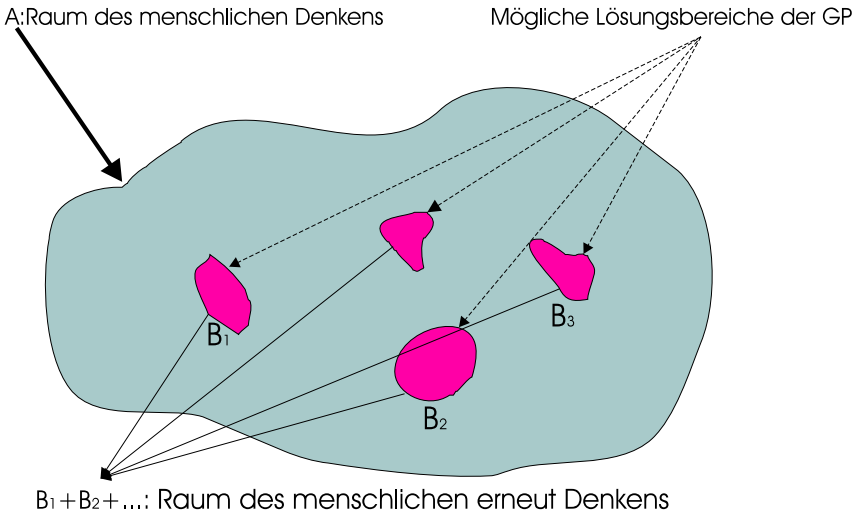


Abbildung 3.17: GP-unterstützte menschliche Entscheidung: Die GP findet die Ersatz-Lösungen. Der Mensch verarbeitet diese Ersatz-Lösungen weiter.

Abb. 3.17 zeigt die Idee, dass die GP die menschliche Entscheidung unterstützt. Dabei bedeutet Bereich A den Raum des menschlichen Denkens. Die Bereiche B_1, B_2,... bedeuten die Lösungsräume, in denen die GP eine Ersatz-Lösung gefunden hat. Also ist es möglich, dass diese von der GP verkleinerten Lösungsräume die perfekte Lösung der Aufgabe enthalten. In diesen verkleinerten Räumen kann man nun die Lösung der gestellten Aufgabe besser lokalisieren.

Ein Beispiel dazu ist die verwendete Konstanten-Optimierung. Zuerst sucht die GP eine Struktur, die die wesentliche Eigenschaft des Systems möglicherweise darstellt, dann optimiert man die Konstanten in dieser Struktur mit bestimmten bekannten Algorithmen, um ein besseres Ergebnis zu bekommen. Dabei bilden eine oder mehre Strukturen aus der GP die wichtigsten Ersatz-Lösungen, nämlich die verkleinerten Lösungsräume. Daraufhin kann die Ersatz-Lösung weiter verarbeitet werden. Für die weiteren Bearbeitungen der GP-Lösungen ist es möglich:

1. Konstanten-Optimierung durchzuführen.
2. Die Struktur teilweise zu modifizieren.

Ein anderes Beispiel dieser Idee sind die GP-Spiele. GP-Spiele bieten dem Menschen manche gute Möglichkeit gegen den Gegner an, dann muss der Mensch selbst entscheiden, welche er nehmen mag. Der Prozess besteht also aus:

1. Die GP bietet eine oder mehre Ersatz-Lösungen.
2. Der Mensch verarbeitet diese Ersatz-Lösungen weiter.

Kapitel 4

Identifikation und Kompensation von gedächtnislosen nichtlinearen Systemen

Gedächtnislose nichtlineare Systeme sind eine System-Art, deren Ausgang nicht von vergangenen Eingangswerten und Ausgangswerten abhängig ist. Mit einer Übertragungskennlinie kann die Eigenschaft eines solchen Systems vollkommen beschrieben werden. Zur Identifikation solcher Systeme stehen viele Verfahren zur Verfügung. In diesem Kapitel wird die GP-Systemidentifikation diskutiert.

4.1 Identifikation der gedächtnislosen NL-Systeme ohne Rauschen

Im vorherigen Kapitel wurde das Prinzip der GP erläutert. Nun werden einige Experimente beschrieben, um die Fähigkeit und das Verhalten der GP bei Aufgaben der Systemidentifikation zu testen.

4.1.1 Identifikation eines NL-Systems mit Sättigungskennlinie

Ein NL-System mit einer Sättigungskennlinie wird sehr oft in Nachrichtensender verwendet. Abb. 4.1 a) zeigt ein solches System im Eingangsbereich [-2, 2] mit folgender Kennlinie:

$$y = \begin{cases} 1 & 2 \geq x > 1 \\ x & -1 \leq x \leq 1 \\ -1 & -2 \leq x < -1 \end{cases} \tag{4.1}$$

Die Gl.(4.2) zeigt die von der GP identifizierte Annäherungskennlinienfunktion.

$$\widehat{y} = sin(\frac{0.9894 \cdot sin(sin(x))}{cos(1.0844 \cdot sin(\frac{\frac{sin(x)}{6.2018}}{cos(sin(\frac{}{cos(sin(0.9023 \cdot sin(cos(1.6348 \cdot sin(cos(x))))))}))}))}) \tag{4.2}$$

Die Abb. 4.1 b) zeigt diese von der GP gefundene Kennlinie. Der Fehler zwischen dem Ausgang des Ziel-Systems und dem Ausgang des von der GP identifizierten Modells ist $6.3 \cdot 10^{-6}$.

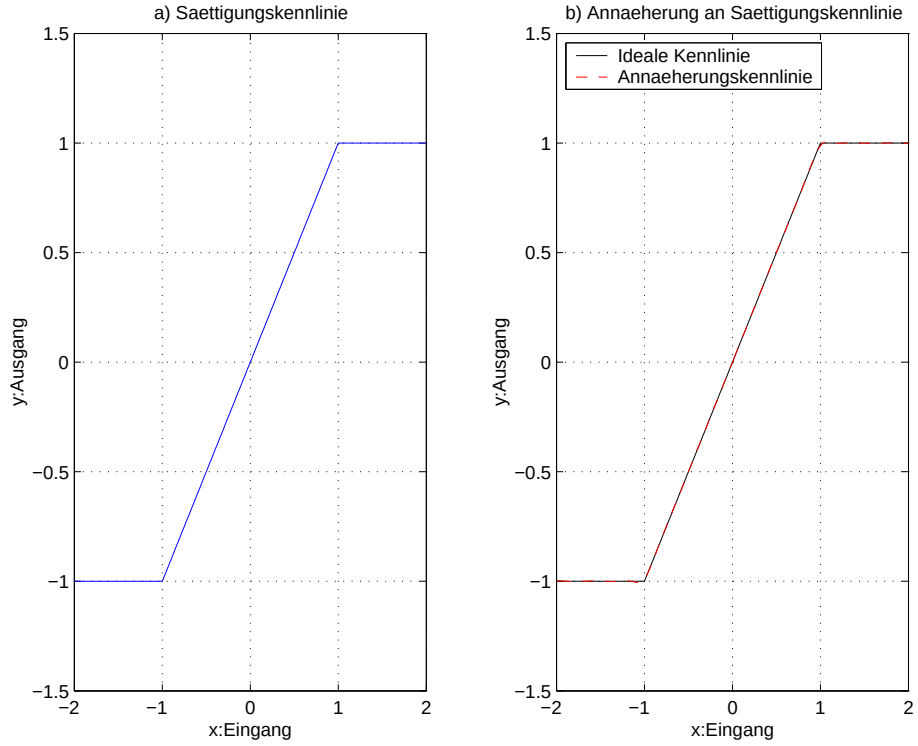


Abbildung 4.1: Sättigungskennlinie und von der GP identifizierte Kennlinie

4.1.2 Identifikation eines NL-Systems mit Sign-Kennlinie

Die Sign-Funktion ist eine sehr häufig angewendete Funktion in der Mathematik und der Signalverarbeitung, wie sie in Abb. 4.2 a) gezeigt wird. Gl. (4.3) zeigt die von der GP gefundene exakte Lösung, und Gl. (4.4) zeigt die von der GP gefundenen Annäherungsfunktion im Eingangsbereich [-1, 1].

$$\widehat{y} = \frac{x}{|x|} \tag{4.3}$$

$$\begin{aligned} \widehat{y} \;=\; & sin(1.9442 \cdot sin(sin(sin(1.9382 \cdot sin(2.1415 sin(sin(2.0348 \cdot sin(2.3625 \\ & \cdot sin(sin(2.6547 \cdot sin(sin(2.0184 \cdot sin(3.0808 \cdot sin(x)))))))))))))) \end{aligned} \tag{4.4}$$

Abb. 4.2 b) zeigt die Kennlinie der Gl. (4.4). Der Fehler zwischen dem Ausgang des Ziel-Systems und dem Ausgang des von der GP identifizierten Modells ist $3.0 \cdot 10^{-11}$. Dieses Beispiel zeigt, dass es möglich ist, eine nahezu exakte Lösung durch die GP zu finden, wenn bestimmte Voraussetzungen erfüllt sind.

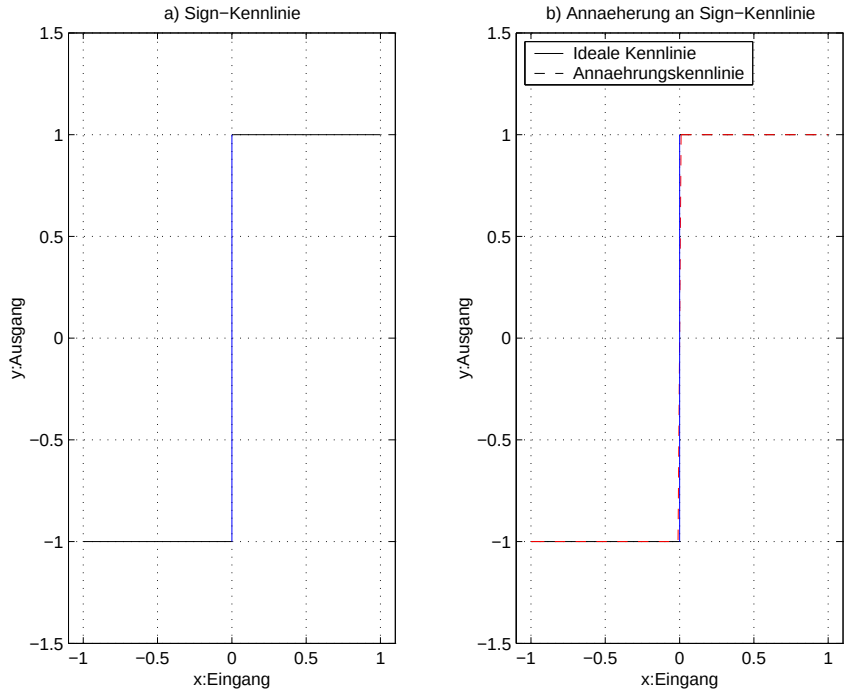


Abbildung 4.2: Sign-Kennlinie und durch die GP identifizierte Kennlinie

4.1.3 Identifikation und Kompensation eines NL-Systems mit einer Polynom-Kennlinie 3. Ordnung

Gl.(4.5) zeigt die Kennlinienfunktion eines NL-Systems, und Abb. 4.3 a) zeigt die Kennlinie im Eingangsbereich [-1, +1].

$$y = f(x) = x + x^2 + x^3 \tag{4.5}$$

Gl. (4.6) zeigt die von der GP identifizierte Annäherungskennlinienfunktion, die in Abb. 4.3 b) veranschaulicht wird.

$$\widehat{y} = \widehat{f}(x) = \frac{1.1464 \cdot cos(0.6704 \cdot cos(sin(e^z)) + 3.1634) + 1.0450 \cdot e^{(0.9750 \cdot x)}}{sin(cos(sin(x)))} \tag{4.6}$$

Wobei:

$$z = cos(\frac{1.2268 \cdot sin(\frac{0.9527 \cdot sin(ln|sin(cos(sin(x)))|)}{sin(sin(cos(cos(e^x))))}) + 1.1264 \cdot e^x}{cos(0.6937 \cdot x)}) \tag{4.7}$$

Der Fehler zwischen dem Ausgang des Ziel-Systems und dem Ausgang des von der GP identifizierten Modells ist $9.6 \cdot 10^{-7}$. Die Gl. (4.8) zeigt die von der GP gefundene inverse Funktion ($\widehat{f}^{-1}$).

$$\widehat{x} = \widehat{f}^{-1}(y) = 1.0248 \cdot sin(\frac{1.0092 \cdot y}{e^{sin(sin(0.9683 \cdot y \cdot e^{-1.2269 \cdot |sin(0.7966 \cdot y^2 \cdot e^{-1.0154 \cdot y})|)})}}) \tag{4.8}$$

Die Abb. 4.3 c) zeigt die gesamte Kennlinie, wenn ein Kompensationssystem mit Kennlinie $\widehat{f}^{-1}$ nachgeschaltet ist. Der Fehler zwischen dem rein linearen System und dem von der GP kompensierten System ist $3.9 \cdot 10^{-5}$. In diesem Beispiel findet die GP keine exakte Lösung wie sie Gl. (4.5) zeigt, obwohl sie einfach ist. Das Beispiel zeigt daher eine wichtige Erscheinung, d.h. die GP kann bei der Systemidentifikation kaum eine exakte Lösung finden, aber sie findet sehr häufig eine Annäherungslösung.

Die GP kann kaum eine exakte Lösung finden. Aber sie kann eine Lösung mit hoher Genauigkeit finden. Die GP-Ergebnisse sind normalerweise nur für den Eingangswertebereich der Trainingsdaten gültig.

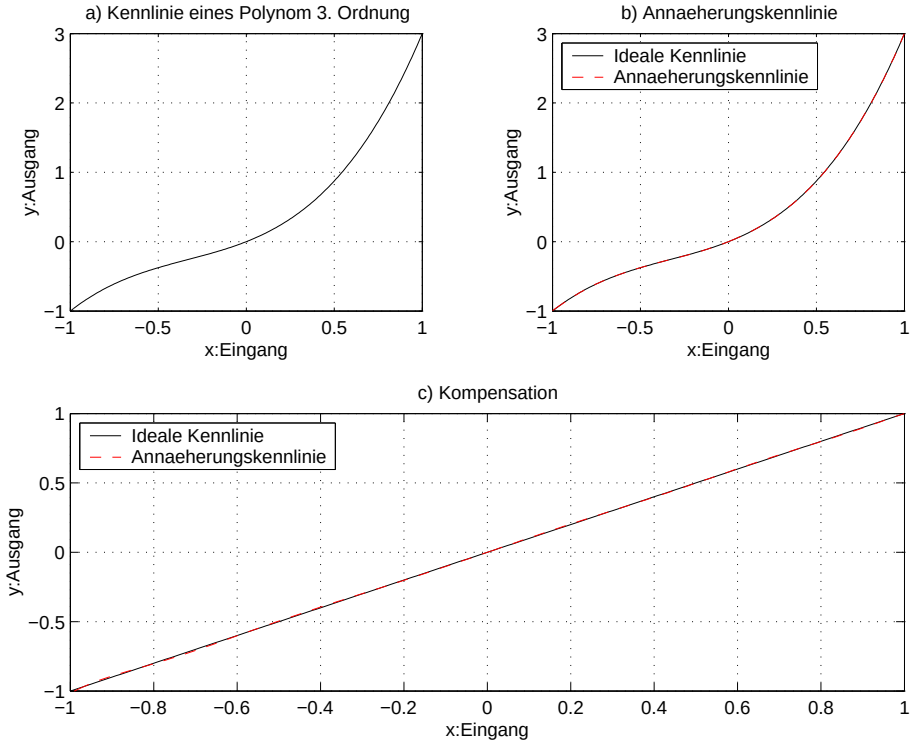


Abbildung 4.3: Identifikation und Kompensation eines NL-Systems mit einer Polynom-Kennlinie 3. Ordnung

4.1.4 Identifikation und Kompensation eines NL-Systems mit einer Polynom-Kennlinie 10. Ordnung

Gl.(4.9) ist eine Polynom-Funktion 10. Ordnung, sie ist die Kennlinienfunktion eines NL-Systems, und Abb. 4.4 a) zeigt die Kennlinie im Eingangsbereich [-1, +1].

$$y = f(x) = 0.6x + 0.5x^2 + 1.2x^3 + 0.5x^4 + x^5 + 0.4x^6 + 1.4x^7 + 0.3x^8 + 1.5x^9 + 0.2x^{10} \tag{4.9}$$

Gl.(4.10) zeigt die von der GP identifizierte Kennlinienfunktion und deren Graphen in Abb. 4.4 b).

$$\widehat{y} = \widehat{f}(x) = (3.1561x + 0.2840e^{(2.4754x)}) \cdot x^4 + (0.8588sin(x) \cdot cos(x) + 0.6577e^{(-0.2122x)}) \cdot x \tag{4.10}$$

Der Fehler zwischen dem Ausgang des Ziel-Systems und dem Ausgang des von der GP identifizierten Modells ist $6.0 \cdot 10^{-4}$. Die Nichtlinearität dieses NL-Systems ist kompensierbar. Die Gl. (4.11) zeigt die von der GP identifizierte inverse Funktion ($\widehat{f}^{-1}$).

$$\widehat{x} = \widehat{f}^{-1}(y) = a_{22}sin(a_{18}sin(a_{17}sin(z + a_{14}y + a_{16}sin(a_{15}y))) + a_{21}sin(a_{20}y + a_{19})) \tag{4.11}$$

Wobei gilt:

$$z = a_{13}sin(a_{11}sin(a_9sin(a_6sin(a_2sin(a_1y) + a_5sin(a_4sin(a_3y))) + a_7y + a_8) + a_{10}y) + a_{12}y) \tag{4.12}$$

und:

$a_1 = 0.3072, a_2 = 2.7494, a_3 = 1.4063, a_4 = 1.0964, a_5 = 1.3034, a_6 = 1.2782, a_7 = 0.0976$ $a_8 = 0.5558, a_9 = 0.4184, a_{10} = 0.7613, a_{11} = 1.9096, a_{12} = 0.6014, a_{13} = 0.2375, a_{14} = 0.2434$ $a_{15} = 0.6693, a_{16} = 0.6836, a_{17} = 1.1491, a_{18} = 0.6606, a_{19} = -0.0866, a_{20} = 0.0662, a_{21} = 0.8527, a_{22} = 1.2898$

Die Abb. 4.4 c) zeigt die gesamte Kennlinie, wenn ein Kompensationssystem mit der Kennlinie $\widehat{f}^{-1}$ nachgeschaltet ist. Der Fehler zwischen dem rein linearen System und dem von der GP kompensierten System ist $3.3 \cdot 10^{-6}$.

Die obigen Experimente zeigen:

1. Unter bestimmten Voraussetzungen ist es möglich, die exakte Lösung zu finden.

2. Normalerweise ist das durch die GP identifizierte System nur eine Annäherung an das Ziel-System.

3. Die Komplexität der GP-Lösung ist nicht in starkem Maße von der Stärke der Nichtlinearität abhängig.

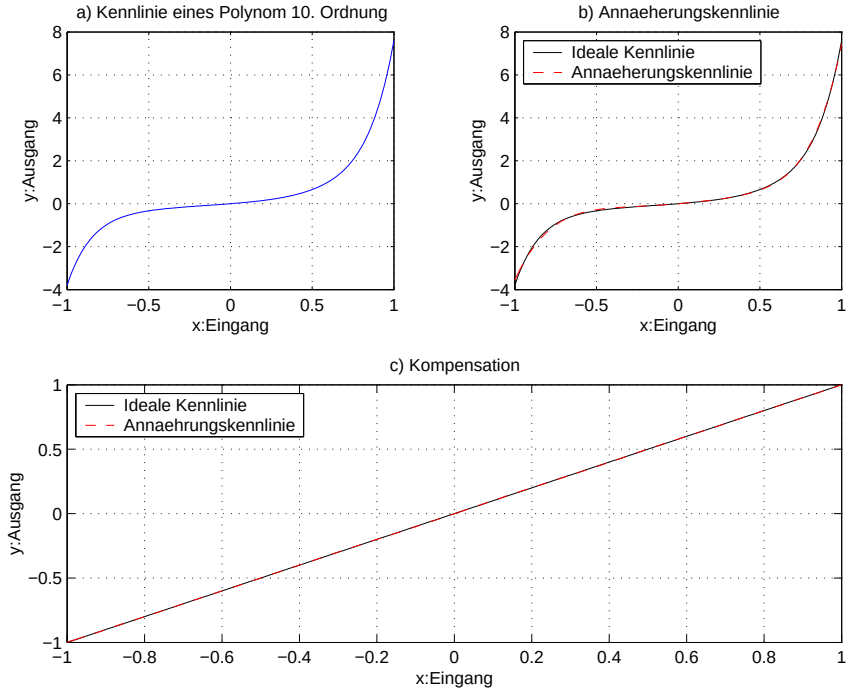


Abbildung 4.4: Identifikation und Kompensation eines NL-Systems mit einer Polynom-Kennlinie 10. Ordnung.

4. Die GP kann die gedächtnislosen nichtlinearen Systeme mit relativ hoher Genauigkeit identifizieren und kompensieren.

5. Die Konstanten-Optimierung spielt eine wichtige Rolle für die erzielte Lösungsgenauigkeit[1].

4.2 Identifikation der gedächtnislosen NL-Systeme mit Rauschen

Durch die obigen Experimente sowie weitere ausgeführte Experimente sind die Fähigkeiten und das Verhalten der GP bei der Systemidentifikation von rauschfreien gedächtnislosen NL-Systeme demonstriert und bewiesen worden. Nun möchten wir das Verhalten der GP bei der Systemidentifikation mit Rauschen untersuchen. In diesem Beispiel wird ein Ziel-System, wie es die Gl. (4.13) zeigt, verwendet. Hier ist zu beachten, dass in diesem Experiment nur Identifikationsprobleme untersucht werden. Bei rauschfreier Situation zeigt die Gl.(4.14) die von der GP gefundenen Annäherungskennlinienfunktion. Der Restfehler ist $1.97 \cdot 10^{-11}$.

[1]Die Annäherungslösung der obigen Experimente sind optimiert worden. Im Kapitel 6 wird eine neue Optimierungsstrategie dargestellt, mit der eine noch höhere Genauigkeit erreicht werden kann.

$$y = f(x) = cos(3.3x) \cdot e^{sin(4.3x)} \tag{4.13}$$

$$\widehat{y} = \widehat{f}(x) = cos(3.3000x) \cdot e^{[sin(4.3000x)+sin(8.3\cdot 10^{-6}\cdot x\cdot sin(e^x)\cdot cos(4.9953\cdot sin(x)))]} \tag{4.14}$$

Bei den Fällen SNR=17.5 dB, 11.4 dB, 5.3 dB und 0 dB zeigt die Abb. 4.5 die von der GP gefundenen Kennlinien. Die Tabelle 4.1 zeigt die entsprechenden Restfehler. Die Abb. 4.6 zeigt die entsprechenden Ausgangsverläufe.

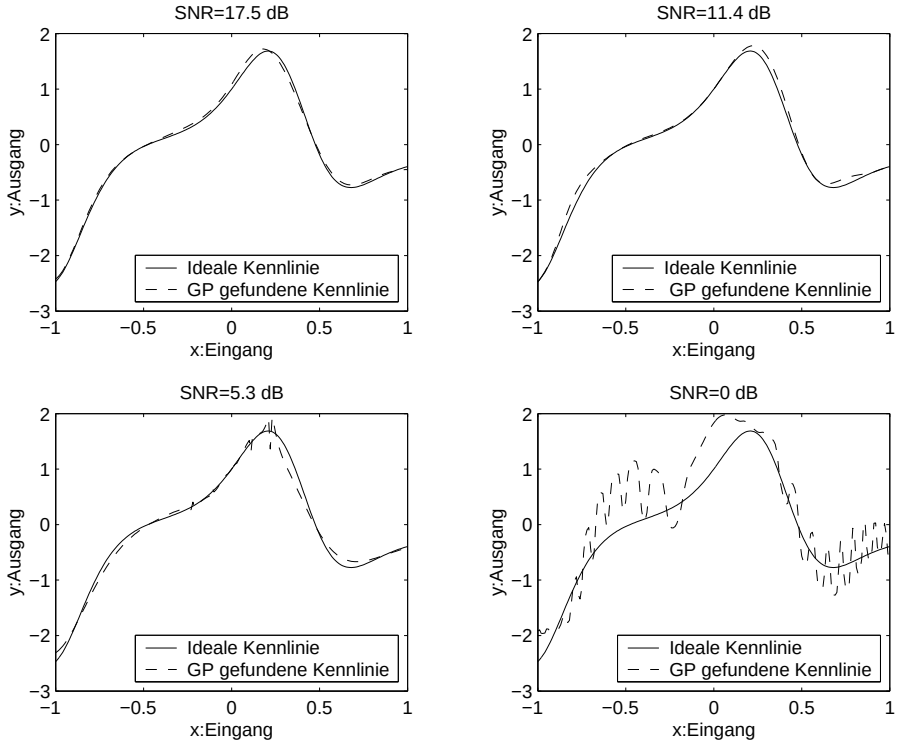


Abbildung 4.5: Die von GP identifizierten Kennlinien bei verschiedenen SNRs

SNR(dB)	∞	17.5	11.4	5.3	0
Restfehler(%)	$1.97 \cdot 10^{-9}$	0.32	0.72	1.19	24.8

Tabelle 4.1: Restfehler bei verschiedenen SNRs

Hier ist zu beachten, dass die 4 Experimente auf der gleichen Länge der Trainingsdaten basieren (150 Datenwerte) und die gleiche standardisierte Fitnessfunktion verwendet wurde. Aus Abb. 4.6 können wir sehen, dass die Ergebnisse der Identifikation ziemlich gut sind, wenn das SNR größer als ca. 5 dB ist. Aber wenn das SNR bei 0 dB liegt, ist das Ergebnis der Identifikation ein großes Problem, d.h. es tritt Overfitting auf. Abb. 4.6 d) zeigt die Ausgangssignale. Im Abschnitt 4.3 werden die Gegenmaßnahmen gegen das Overfitting ausführlich diskutiert. Es wurde

festgestellt: Wenn das Overfitting entsteht, ist es nicht nützlich, die Konstanten-Optimierung zu verwenden, weil bei dieser Situation die Konstanten-Optimierung das Overfitting verstärkt.

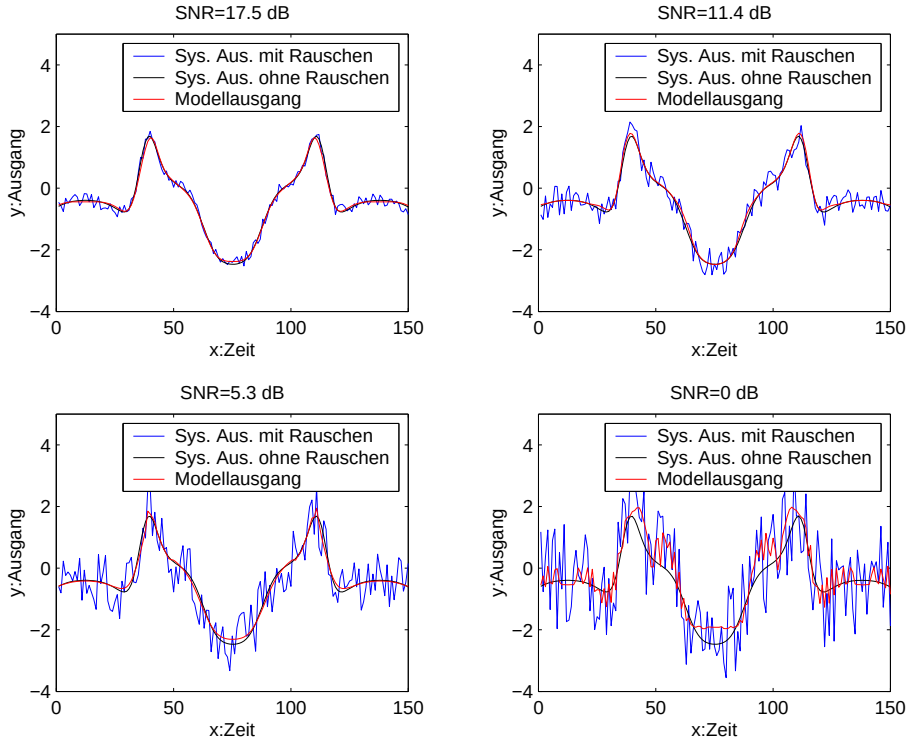


Abbildung 4.6: Die Ausgänge bei verschiedenen SNRs

Beim starken Rauschen erzeugt die GP Overfitting. Wenn SNR>5 dB ist, erzeugt die GP kaum Overfitting. Wenn das Overfitting entsteht, ist es nicht geeignet, die Konstanten-Optimierung zu verwenden.

4.3 Verbesserungsmöglichkeiten

Aus der Abb. 4.6 kann man klar ersehen, dass bei kleinen SNR der Modellausgang eine große Abweichung gegenüber dem Nutzsignal des Systemausgangs besitzt, insbesondere bei SNR=0. Die Ursache ist das vom Modell erzeugte Overfitting, d.h. die GP versucht das identifizierte Modell dem Rauschen anzupassen. Um das Overfitting zu überwinden, werden die folgende Maßnahmen untersucht. Hier ist zu beachten, dass davon ausgegangen wird, dass die Rauschstärke nicht durch Vorverarbeitung (z.B. Filterung) reduziert werden kann.

4.3.1 Verbesserung durch längere Daten

Im Kapitel 2 haben wir als eine wichtige Ursache von Overfitting diskutiert, dass die GP die statistischen Eigenschaften des Rauschens noch nicht vollständig beherrscht, wenn die Trainingsdaten sehr kurz sind. D.h. wenn die GP längere Trainingsdaten nutzen kann, ist es hilfreich, um das Overfitting zu vermeiden. Gl. (2.8) zeigt den theoretischen Zusammenhang zwischen dem Restfehler und der Datenlänge N. Hierbei wird eine 20 fache Datenlänge (20×150=3000 Datenwerte) benutzt. Abb. 4.7 b) zeigt die Kennlinie der von der GP gefundenen besten Lösung. Der Fehler zwischen den Ausgänge des Ziel-Systems und dem identifizierten Systems ist ca.1.6%.

Der Vorteil dieser Gegenmaßnahme ist, dass die Implementierung sehr einfach ist. Man braucht nur mehr Trainingsdaten einzugeben.

Die Nachteile dieser Gegenmaßnahmen sind:

1. Sie ist von vorhandenen beobachteten oder gemessenen Daten abhängig. Bei manchen Situationen ist die Datenlänge fest.

2. Sie benötigt mehr Rechenaufwand und eine längere Trainingsphase.

Mit langen Trainingsdaten wird die Evolutions-geschwindigkeit der GP deutlich langsamer.

4.3.2 Verbesserung durch early-Stop

Im Kapitel 2 haben wir diskutiert, dass es ebenfalls eine wichtige Ursache von Overfitting ist, wenn das Modell eine zu komplizierte Struktur besitzt. Um eine geeignete Struktur zu finden, verwendet man voneinander unabhängige Trainingsdaten und Validierungsdaten. D.h. die GP-Trainingsphase läuft nur bis zu dem Zeitpunkt, wo der Modellfehler der Validierungsdaten sein Minimum erreicht. In diesem Experiment verwenden wir 150 Datenpunkte für das Training, und weitere 150 Datenpunkte für die Validierung. Die gesamten Datenpunkte (für Training und Validierung) sind also 300. Abb. 4.7 a) zeigt die Kennlinie der von GP gefundenen besten Lösung. Der Fehler zwischen dem Ausgang des Ziel-Systems und dem des identifizierten Systems ist ca. 3.6%.

Der Vorteil dieser Gegenmaßnahme ist es, einen kleinen Rechenaufwand zu erreichen.

Die Nachteile dieser Gegenmaßnahme sind:

1. Wenn die gesamten Daten sehr gering sind, ist es problematisch, die Daten in Trainingsdaten und Validierungsdaten einzuteilen.

2. Man braucht eine ausführliche Analyse, um die Unabhängigkeit zwischen den Trainingsdaten und den Validierungsdaten zu gewährleisten.

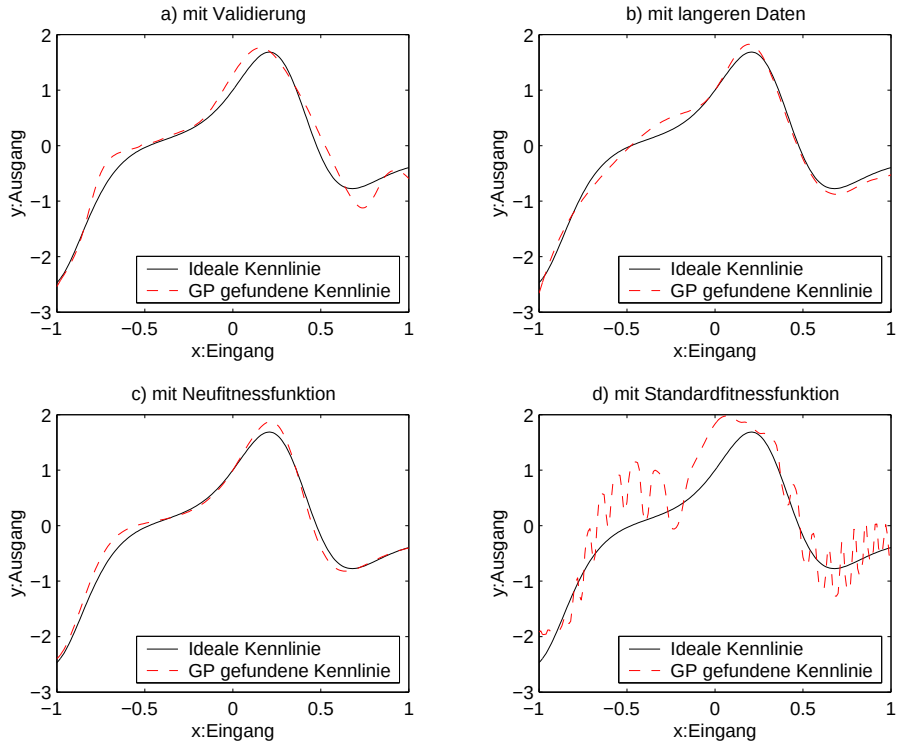


Abbildung 4.7: Identifikation des Ziel-Systems gemäß Gl. 4.13 bei SNR=0. a) bis c) zeigen die Wirkung verschiedener Gegenmaßnahmen gegen das Overfitting. a) Trainingdatenlänge=150, Validierungsdatenlänge=150; b) Trainingsdatenlänge=3000; c) Trainingsdatenlänge=150, mit dynamischer Fitnessfunktion. d) Trainingsdatenlänge=150, ohne Validierung, mit standardisierter Fitnessfunktion

4.3.3 Verbesserung durch Fitnessfunktion

Im Kapitel 3 ist angedeutet, dass die Fitnessfunktion eine wichtige Rolle in der GP spielt. D.h. wenn die GP für die gleiche Aufgabe verschiedene Fitnessfunktionen verwendet, kann die GP möglicherweise verschiedene Ergebnisse bekommen. In diesem Abschnitt wird eine Fitnessfunktion, die auf einer standardisierten Fitnessfunktion basiert und mit AKF/KKF[2] kombiniert ist, entwickelt.

AKF und KKF werden in Gl.(4.15) und (4.16) definiert.

$$l_x(\tau) = \lim_{\mathrm{T}\to\infty} \frac{1}{2\mathrm{T}} \int_{-\mathrm{T}}^{+\mathrm{T}} x(t) \cdot x(t+\tau) dt \tag{4.15}$$

$$l_{xy}(\tau) = \lim_{\mathrm{T}\to\infty} \frac{1}{2\mathrm{T}} \int_{-\mathrm{T}}^{+\mathrm{T}} x(t) \cdot y(t+\tau) dt \tag{4.16}$$

Die AKF besitzt einige wichtige Merkmale:(siehe Abb. 4.8)

[2] Autokorrelationsfunktion/Kreuzkorrelationsfunktion

1. Je unabhängiger die Signalwerte sind, desto schmäler ist die AKF.

2. Für gleichanteilfreie zufällige Signale: Je größer der zeitliche Abstand ist, desto geringer sind die Relationen zwischen den Signalwerte.

3. Für ein weißes Rauschen gilt: AKF=0, wenn $\tau \neq 0$.

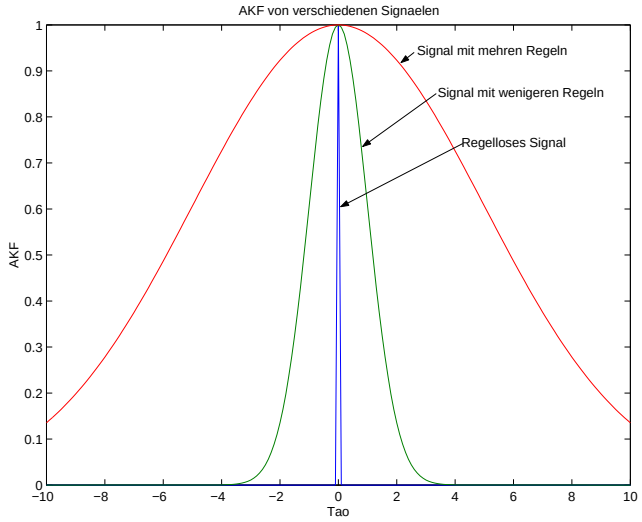


Abbildung 4.8: AKF und die Relation der Signalwerte

Unser Ziel ist es, einen Maßstab(eine Größe) zu finden, mit dem das Overfitting bei der Berechnung der Fitness ermittelt werden kann. Das Prinzip ist, *wenn ein Individuum bei der GP Overfitting erzeugt, wird seine Fitness reduziert.*[3]

Die Ursache des Overfitting ist die Anpassung des identifizierten Modells an die verrauschten Ausgangswerte des Ziel-Systems durch die GP. Weil der Ausgang des Ziel-Systems Rauschen enthält, wird der Ausgang des Modells auch Rauschen enthalten. Durch die Berechnung der AKF des Modellausgangs ist der Rausch-Anteil im Modellausgang vom Nutzsignal unterscheidbar. Gemäß der nachfolgenden Ableitung können wir Gl. (4.17) bekommen. Diese Gleichung zeigt, wenn die GP sich dem Rauschen anzupassen versucht, steigt der Wert von Gl. (4.17) an. Dabei bedeutet:
$\widetilde{n}$: das teilweise gelernte Rauschen–nämlich Overfitting von Typ 2.
$\widetilde{y}$: Systemausgang ohne Rauschen.
$\widehat{y}$: Modellausgang, und $\widehat{y} = \widetilde{y} + \widetilde{n}$.

$$F_1 = l_{\widehat{y}}(0) - \frac{1}{2}[l_{\widehat{y}}(-1) + l_{\widehat{y}}(+1)]$$

$$= \frac{1}{\mathrm{N}} \sum (\widetilde{y}(i) + \widetilde{n}(i))(\widetilde{y}(i) + \widetilde{n}(i)) -$$

[3]Durch Overfitting kann ein Individuum eine bessere Fitness (Gewinn) bekommen. Der Gewinn muss also durch einen Maßstab verkleinert werden.

$$\frac{1}{2N}[\sum(\widetilde{y}(i)+\widetilde{n}(i))(\widetilde{y}(i+1)+\widetilde{n}(i+1))+\sum(\widetilde{y}(i)+\widetilde{n}(i))(\widetilde{y}(i-1)+\widetilde{n}(i-1))]$$

$$= \frac{1}{\mathrm{N}}\sum\widetilde{y}(i)\widetilde{y}(i)+\frac{1}{\mathrm{N}}\sum\widetilde{n}(i)\widetilde{n}(i)-\frac{1}{2\mathrm{N}}\sum\widetilde{y}(i)[\widetilde{y}(i+1)+\widetilde{y}(i-1)]$$

$$\approx \frac{1}{\mathrm{N}}\sum\widetilde{y}(i)\widetilde{y}(i)+\frac{1}{\mathrm{N}}\sum\widetilde{n}(i)\widetilde{n}(i)-\frac{1}{\mathrm{N}}\sum\widetilde{y}(i)\widetilde{y}(i)$$

$$= \frac{1}{\mathrm{N}}\sum\widetilde{n}(i)\widetilde{n}(i)$$

$$F_1 = l_{\widehat{y}}(0)-\frac{1}{2}[l_{\widehat{y}}(-1)+l_{\widehat{y}}(+1)] = \frac{1}{\mathrm{N}}\sum\widetilde{n}(i)\widetilde{n}(i) \tag{4.17}$$

Wenn wir die Gl. (4.17) in die gesamte Fitnessfunktion nach Gl. (4.19) einbeziehen, wie in Gl. (4.20) gezeigt, wird sie das Overfitting von Typ 2 beschränken, weil die Individuen mit Overfitting vom Typ 2 einen schlechten gesamten Fitnesswert besitzen. Wobei F_0 die standardisierte Fitnessfunktion bedeutet. Wenn der Faktor α vergrößert wird, wird die Beschränkungswirkung des Overfitting verstärkt.

Hier ergibt sich ein neues Problem: Obwohl die GP sich den konkreten Werten des Rauschens nicht anzupassen versucht, versucht sie sich dennoch den Kurzzeitmittelwerten des Rauschens anzupassen , d.h. 1. Typ von Overfitting. Um das Overfitting zu überwinden, brauchen wir einen Maßstab (eine Größe), mit dem das Overfitting ausgeschaltet werden kann. Durch die nachfolgende Ableitung können wir Gl. (4.18) bekommen. Diese Gleichung zeigt, wenn die GP sich dem Kurzzeitmittelwert des Rauschens anzupassen versucht, steigt der Wert von Gl. (4.18). Dabei bedeutet:
$\overline{n}$: der von der GP gelernte Kurzzeitmittelwert des Rauschens, nämlich Overfitting von Typ 1.
$\widehat{n}$: geschätztes Rauschen, und $n = \widehat{n}+\overline{n}$.

$$F_2 = l_{\widehat{n}}(0)-\frac{1}{2}[l_{\widehat{n}}(-1)+l_{\widehat{n}}(+1)]$$

$$= \frac{1}{\mathrm{N}}\sum(n(i)-\overline{n}(i))(n(i)-\overline{n}(i))-$$

$$\frac{1}{2\mathrm{N}}[\sum(n(i)-\overline{n}(i))(n(i+1)-\overline{n}(i+1))+(n(i)-\overline{n}(i))(n(i-1)-\overline{n}(i-1))]$$

$$= \frac{1}{\mathrm{N}}\sum n^2(i)-\frac{1}{N}\sum\overline{n}^2(i)+\frac{1}{\mathrm{N}}\sum\overline{n}(i)\frac{1}{2}[\overline{n}(i+1)+\overline{n}(i-1)]$$

$$\approx \frac{1}{\mathrm{N}}\sum n^2(i)-\frac{1}{\mathrm{N}}\sum\overline{n}^2(i)+\frac{1}{\mathrm{N}}\sum\overline{n}^2(i)$$

$$= \frac{1}{\mathrm{N}}\sum n^2(i)$$

$$l_{y\widehat{n}}(0) = \frac{1}{\mathrm{N}}\sum(\widetilde{y}(i)+n(i))(n(i)-\overline{n}(i))$$

$$= \frac{1}{\mathrm{N}}[\sum(\widetilde{y}(i)n(i))-\sum(\widetilde{y}(i)\overline{n}(i))+\sum n^2(i)-\sum(n(i)\overline{n}(i))]$$

$$= \frac{1}{\mathrm{N}} \sum n^2(i) - \frac{1}{\mathrm{N}} \sum (\widetilde{y}(i)\overline{n}(i))$$

$$F_2 = l_{\widehat{n}}(0) - \frac{1}{2}[l_{\widehat{n}}(-1) + l_{\widehat{n}}(+1)] - l_{y\widehat{n}}(0) = \frac{1}{\mathrm{N}} \sum (\widetilde{y}(i)\overline{n}(i)) \tag{4.18}$$

Wenn wir die Gl. (4.18) in die Fitnessfunktion nach Gl. (4.20) einbeziehen, wie dies die Gl. (4.21) zeigt, wird sie das Overfitting beschränken, weil die Individuen mit Overfitting eine schlechte Gesamtfitness besitzen. Wenn der Faktor β vergrößert wird, wird die Beschränkungswirkung verstärkt. Hier ist zu beachten, dass in der Gl. (4.21) der Betrag des Terms F_2 eingeben werden muss, weil es möglich ist, dass der Wert von F_2 negativ wird.

Die kombinierte Fitnessfunktion muss dynamisch verwendet werden. D.h. in verschiedenen Zeitphasen (Generationen) werden verschiedene Fitnessfunktionen verwendet. In unserem Beispiel:

1. Während der Generationen 0–20 wird die Fitnessfunktion, nach Gl. (4.19), verwendet.

2. Während der Generationen 21–40 wird die Fitnessfunktion, nach GL. (4.20), verwendet.

3. Während der Generationen 41–60 wird die Fitnessfunktion, nach Gl. (4.21), verwendet.

$$F = F_0 \tag{4.19}$$

$$F = F_0 + \alpha \cdot F_1 \tag{4.20}$$

$$F = F_0 + \alpha F_1 + \beta \cdot \mid F_2 \mid \tag{4.21}$$

Durch die Dynamisierung der Fitnessfunktion wird die in Abb. 4.7 c) gezeigte Kennlinie von der GP als beste Lösung gefunden. Der Fehler zwischen der Kennlinie des Systems und der von der GP identifizierten Kennlinie ist ca. 1.4%.

Das größte Problem dieser Gegenmaßnahmen ist die Bestimmung der Parameter α und β. Bei diesem Experiment verwenden wir $\alpha = 5$ und $\beta = 8$.

Abb. 4.9 zeigt die Verbesserung der Systemidentifikation durch die Verwendung verschiedener Fitnessfunktionen.

Die wesentliche Verbesserung der Ergebnisse wird innerhalb der ersten 20 Generationen geschehen. In den folgenden 20 Generationen wird eine weitere deutliche Verbesserung eintreten.

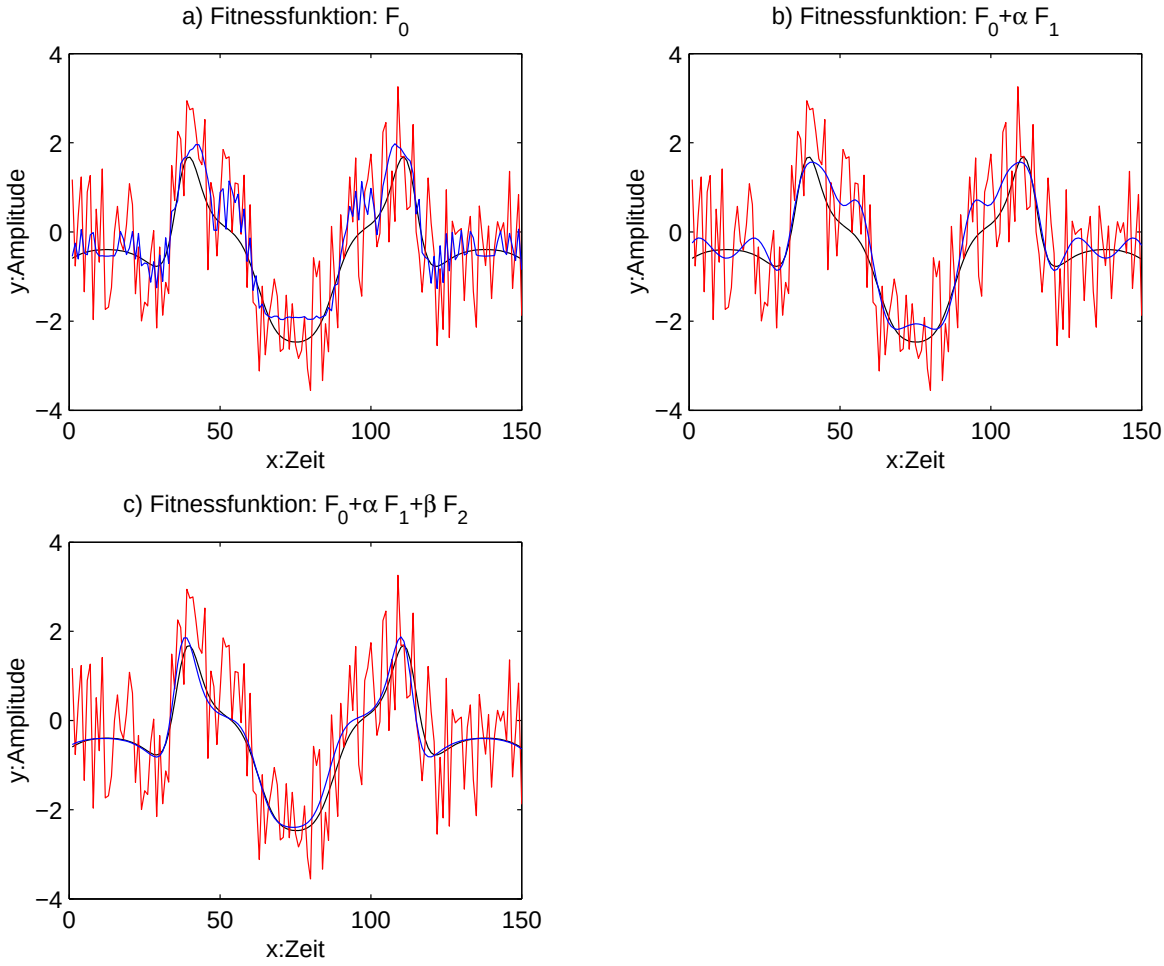


Abbildung 4.9: Verbesserungsprozesse a)—mit standardisierter Fitnissfunktion F_0. b)—mit einer Kombination von Fitnessfunktionen F_0 und F_1. c)—mit einer dynamisierten Kombination von Fitnessfunktionen F_0, F_1 und F_2.

4.4 Nichtlineare Konstanten-Optimierung

Die Stärke der GP besteht darin, dass man mit der GP eine neue Struktur finden kann. Die Schwäche der GP ist es, einen exakten Wert von Parametern zu finden. Mit den Optimierungsalgorithmen kann man aber die Konstanten in der GP weiter optimieren[Panyaworayan02], um die exakten Werte der Konstanten zu erreichen. Konstanten-Optimierung ist allerdings *nicht geeignet* für Anwendungen und Modelle, die Overfitting erzeugen, weil die Konstanten-Optimierung in diesen Fällen das Overfitting weiter verstärkt.

Konstanten-Optimierung ist aber geeignet für Modelle, die kein Overfitting erzeugen, weil die Konstanten-Optimierung in diesem Fall die Genauigkeit des Modells verbessern kann. Konventionelle Optimierungsalgorithmen der nichtlinearen Parameter basieren oft auf die Konjunktions-Gradient, d.h.

1. Der Algorithmus sucht die Parameter nur in der Richtung mit dem größten Konjunktions-Gradienten.

2. Wenn der Konjunktions-Gradient kleiner ist als ein vorgegebener Wert, endet der Algorithmus.

Das größte Problem des nichtlinearen Optimierungsalgorithmus sind die *lokalen Minima oder Maxima.*

Einerseits ist die Wirkung der nichtlinearen Konstanten-Optimierung abhängig von der Struktur, anderseits von den Startwerten bei vorbestimmter Struktur. In[Kafka02] wird eine Optimierungsstrategie gezeigt, d.h. mit zufällig erzeugten Startwerten mehrmals zu optimieren. Die Strategie benötigt eine Voraussetzung, das Modell muss *irgendwann stabil* sein. Für die Modelle mit Rückkopplungen ist dies normalerweise nicht der Fall, d.h. diese Strategie funktioniert dort nicht.

Bei der Suche nach der Struktur sucht die GP gleichzeitig die Rohparameter. Danach werden die sogenannten Rohparameter optimiert. Die Rohparameter sind gute Startwerte für die Optimierungsalgorithmen. In [Panyaworayan02] wird eine derartige Optimierungsstrategie gezeigt, d.h. alle Parameter werden optimiert. Im Kapitel 2 haben wir ein allgemeines Prinzip der Parameterschätztheorie dargestellt, d.h. je größer die Anzahl der Parameter des Modells ist, desto ungenauer ist die Schätzung der Parameter. Unsere Untersuchung zeigt, dass die volle Konstanten-Optimierung nicht die beste Strategie ist. Die wichtigsten Gründe sind:

1. Die volle Konstanten-Optimierung erzeugt mehrere lokale Minima.

2. Die von der GP gefundene Struktur enthält manche strukturische *Intron.*

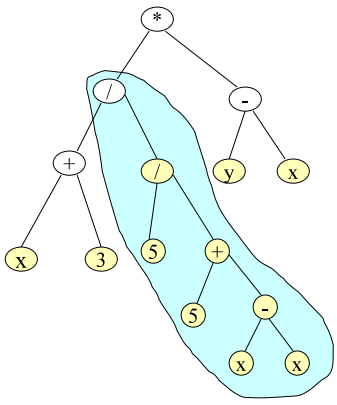


Abbildung 4.10: Die Erscheinung in der GP: Intron, d.h. nutzlose Struktur

Intron ist eine wichtige Erscheinung in der GP. Es bedeutet, dass die GP manche nutzlosen Strukturanteile erzeugt, um bei der Mutation und Kreuzung die wichtigen Strukturanteile im Individuum möglichst nicht zu zerstören. In Abb. 4.10 zeigt z.B.

der dunkle Bereich ein Intron.

Das Intron ist echt nutzlos für die Funktionalität, also nennen wir es reines Intron. Ein reines Intron kann man leicht erkennen und beseitigen. Im Gegensatz dazu erzeugt die GP auch *Quasi-Introns* bei der Systemidentifikation, d.h. das Quasi-Intron besitzt manche Wirkungen für die Funktionalität, aber normalerweise sind seine Wirkungen klein. Die Wirkung vom Quasi-Intron ist eine Ergänzung für die nichtperfekte Hauptstruktur, dabei ist die "nichtperfekte Struktur" so zu verstehen, dass wahrscheinlich die Struktur richtig ist, aber die Parameter nicht richtig sind. Die Erkennung von Quasi-Introns ist keine einfache Aufgabe. Das Quasi-Intron verursacht,

1. Mehrere lokale Minima.

2. und führt zu komplizierteren Strukturen.

Wenn Introns in den GP-Ergebnissen entstehen, kann man diese durch eine Modifikation der Konstanten-Optimierung erkennen und ausschneiden.

Um das Quasi-Intron zu erkennen, wird eine Strategie der teilweisen Konstanten-Optimierung dargestellt. D.h. zunächst wird mit voller Konstanten-Optimierung und gemäß den mathematischen Regeln die Wichtigkeit verschiedener Teile (Konstanten) untersucht, danach werden nur die wichtigsten Teile optimiert. Bei besonderen Fällen kann man auch probieren, ob ein besseres Ergebnis erreicht werden kann, wenn manche nicht wichtigen Teile vernachlässigt werden. Durch diese Strategie kann man ein besseres Ergebnis, sogar perfekte Ergebnisse bekommen. Im Folgenden wird mit einem Beispiel diese Strategie dargestellt.

Die Gl. (4.14) zeigt die von der GP gefundene beste Annäherungsfunktion für die Gl. (4.13). Diese wird umgeschrieben, d.h. die Beschreibung mit vollen Parameter, wie dies die Gl. (4.22) zeigt.

$$\widehat{y} = a_1 \cdot cos(a_2 \cdot x) \cdot e^{[a_3 \cdot sin(a_4 \cdot x) + a_5 \cdot sin(a_6 \cdot x \cdot sin(a_7 \cdot e^{(a_8 \cdot x)}) \cdot cos(a_9 \cdot sin(a_{10} \cdot x)))]} \tag{4.22}$$

Die Konstanten in dieser Gleichung werden gemäß der obigen Strategie mehrmals optimiert. Die Tabelle 4.2 zeigt die Ergebnisse der Optimierung. In dieser Tabelle bedeutet "$\surd$" diese Konstante wird optimiert, "$\times$" diese Konstante wird nicht optimiert.

Aus der Tabelle 4.2 kann man sehen:

1. Wenn die Konstanten a_2 oder a_4 oder a_5 nicht optimiert werden, ist das Ergebnis viel schlechter als bei der vollen Konstanten-Optimierung. Das bedeutet, dass die Konstanten a_2, a_4 und a_5 für die Optimierung wichtig sind.

2. Wenn die Konstanten a_3 oder a_6 oder a_7 oder a_8 oder a_{10} nicht optimiert werden, ist das Ergebnis viel besser als bei der vollen Konstanten-Optimierung. Das bedeutet, dass diese Konstanten für die Optimierung nicht geeignet sind.

Par.	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	a_9	a_{10}	Restfehler
Org. Wert	1.0	3.2	1.0	4.2	1.0	5.0	1.0	1.0	-5.0	1.0	
Opt. Sit.	√	√	√	√	$1.92 \cdot 10^{-2}$	√	√	√	√	√	$8.5 \cdot 10^{-8}$
Opt. Sit.	×	√	√	√	$1.67 \cdot 10^{-2}$	√	√	√	√	√	$5.3 \cdot 10^{-8}$
Opt. Sit.	√	×	√	√	$1.55 \cdot 10^{-2}$	√	√	√	√	√	$4.1 \cdot 10^{-5}$
Opt. Sit.	√	√	×	√	$4.4 \cdot 10^{-10}$	√	√	√	√	√	$9.1 \cdot 10^{-20}$
Opt. Sit.	√	√	√	×	$1.03 \cdot 10^{-1}$	√	√	√	√	√	$1.1 \cdot 10^{-3}$
Opt. Sit.	√	√	√	√	×	√	√	√	√	√	$3.4 \cdot 10^{-5}$
Opt. Sit.	√	√	√	√	$7.36 \cdot 10^{-5}$	×	√	√	√	√	$6.2 \cdot 10^{-11}$
Opt. Sit.	√	√	√	√	$5.17 \cdot 10^{-7}$	√	×	√	√	√	$1.3 \cdot 10^{-12}$
Opt. Sit.	√	√	√	√	$1.79 \cdot 10^{-4}$	√	√	×	√	√	$3.3 \cdot 10^{-11}$
Opt. Sit.	√	√	√	√	$1.90 \cdot 10^{-2}$	√	√	√	×	√	$1.2 \cdot 10^{-7}$
Opt. Sit.	√	√	√	√	$1.0 \cdot 10^{-10}$	√	√	√	√	×	$5.5 \cdot 10^{-21}$
Opt. Sit.	√	√	√	√	0	–	–	–	–	–	$1.2 \cdot 10^{-26}$

Tabelle 4.2: Optimierung mit verschiedenen Konstanten

Durch eingehende Analyse wird herausgefunden, dass der Restfehler stark von der Konstanten a_5 abhängig ist, wenn das Ergebnis sehr gut ist. Und zwar, um ein besseres Ergebnis zu erreichen, muss die Konstante a_5 ziemlich klein sein. Also bietet die Konstante a_5 eine Spur an, um den mit a_5 kombinierten Teil, das Quasi-Intron zu finden. In der Tat, wenn $a_5 = 0$ ist, erreicht die Gl.(4.22) die perfekte Struktur . Und der Restfehler=10^{-26}.

Aus der Tabelle 4.2 kann man sehen, dass die Konstanten a_7 und a_{10} bei der Optimierung eine schlechte Wirkung verursachen. Die Ursache wird durch die Abb. 4.11 verdeutlicht. Aus der Abb. (4.11) kann man klar ersehen, wenn sich beide Konstanten ändern, erzeugt der Restfehler viele lokale Minima. Das ist von nichtlinearen Optimierungsalgorithmen nicht erwünscht. Die Strategie ist also:

1. Vermeiden der Optimierung aller Konstanten, die viele lokale Minima verursachen. Frage: Welche Konstanten verursachen viele Minima? Antwort:
   (a) Untersuchung Stück für Stück wie im obigen Beispiel.
   (b) Vorgaben gemäß den Erfahrungen. Z.B. die Konstanten, die im Tiefen mit sin/cos-Operatoren stehen , verursachen normalerweise mehrere Minima.
   (c) Analyse gemäß den mathematischen Kenntnissen.

2. Analyse des Zusammenhangs zwischen den optimierten Ergebnissen und den optimierten Konstanten.

3. Erkennung der Quasi-Introns gemäß 2. Wenn ein Baumzweig außerdem viele *sin*/*cos*-Operatoren besitzt, sind normalerweise nicht alle *sin*/*cos* Anteile notwendig. Die Grund ist, wenn die GP eine Lösung finden möchte, versucht sie zunächst auf der Ebene der Struktur die Lösung zu verbessern und nicht auf der Ebene der Konstanten. Diese Merkmale verursachen, dass die Lösung der GP sehr komplizierte Strukturen mit einfachen Parametern (z.B. 1) besitzt.

4. Ausschneiden der Quasi-Introns und erneute Optimierung.

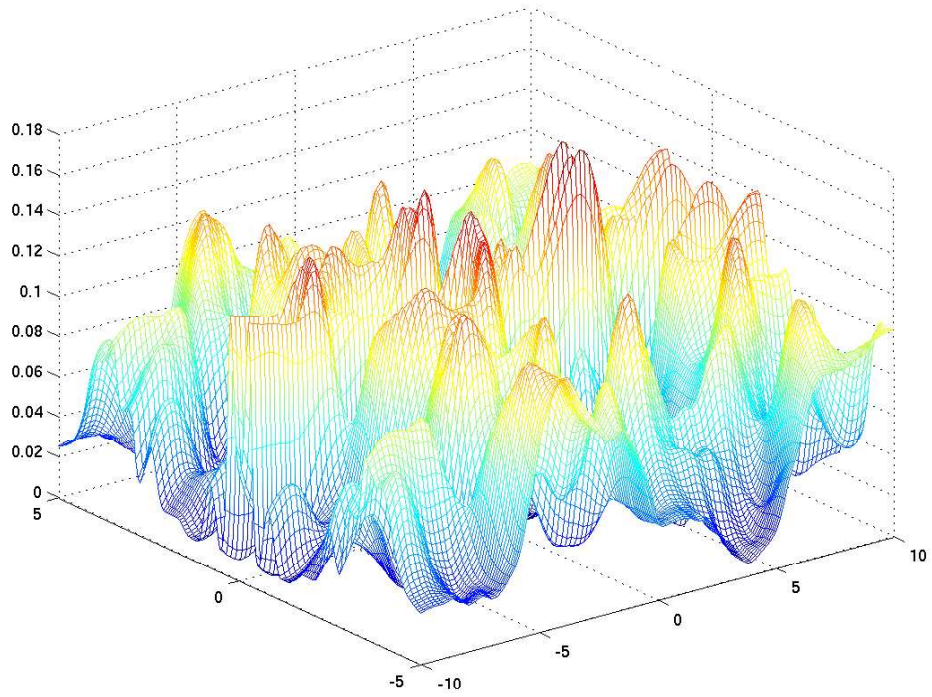


Abbildung 4.11: Landschaft bei der Konstanten-Optimierung

Kapitel 5

Identifikation von Systemen mit Gedächtnis

Ein System mit Gedächtnis ist von solcher Art, dass der Ausgang von vergangenen Eingängen und/oder vergangen Ausgängen abhängig ist. Die einfachste Sorte von solchen Systemen sind die LZI-Systeme. Zur Identifikation eines LZI-Systems stehen viele klassische Verfahren zur Verfügung, wie z.B. AR, MA, ARMA, ARMAX, ARX, OE, BJ, FIR oder IIR Modelle [Ljung87], [Ljung98], [Nelles00].

> Bei der Identifikation von Systemen mit Gedächtnis ist der GP-Suchraum viel größer als der bei Systemen ohne Gedächtnis.

Das nichtlineare System mit Gedächtnis ist ein komplizierter Systemtyp, er weist sowohl Gedächtnis-Verhalten als auch nichtlineares Verhalten auf. Zur Identifikation eines nichtlinearen Systems mit Gedächtnis stehen auch viele klassische und moderne Verfahren zur Verfügung. Die typischen klassischen Verfahren sind z.B. NARX, NARMAX, NOE, NJB, NFIR, NIIR, Volterra-Reihen, Kolmogorov-Polynome usw. [Sjoberg95a, Ljung92, Giancarlo01, Basso02]. Die typischen modernen Verfahren sind z.B. Neuronale Netze(NN), Fussy-Modelle, Neuron-Fussy Modelle, Wavelet Modelle usw.[Sjoberg95b, Pham99, Nelles01, Mallat98, Sureshbabu95]. Alle klassischen und modernen Verfahren gehören zu dem sogenannten Black-Box Methoden. In diesem Kapitel wird die GP zur Identifikation solcher Black-Box Systeme verwendet.

5.1 Identifikation eines LZI-Systems

> Bei der Identifikation von LZI-Systemen kann man durch eine LZI-Grammatik in der GP den Suchraum wesentlich verkleinern.

Als Beispiel wird ein LZI-System, wie es in der Gleichung (5.1) gezeigt wird, verwendet. Dabei haben wir das Vorwissen, dass das Ziel-System ein LZI-System ist. Wenn die GP das Ziel-System identifizieren soll, muss die GP-Suche auf die Form des gesuchten Individuums beschränkt werden, um einen kleineren GP-Suchraum

zu erreichen. Diese Beschränkung wird durch eine LZI-Grammatik in der GP implementiert. Die LZI-Grammatik steuert die GP so, dass jedes Individuum in der GP ein LZI-System ist.

Gleichung des Ziel-Systems:

$$y(i) = 0.25[1.5y(i-1) - 0.7y(i-2) + 0.9y(i-3) - 1.3y(i-4) + 0.3x(i) - 0.13x(i-1) + 2.5x(i-2) + 0.9x(i-3) + 0.4x(i-4) + 0.2x(i-5) + 0.1x(i-6)] \tag{5.1}$$

GP-Lösung:

$$\widehat{y}(i) = \frac{1}{3}\widehat{y}(i-1) - \frac{1}{8}\widehat{y}(i-4) - \frac{1}{8}\widehat{y}(i-5) + \frac{1}{16}\widehat{y}(i-6) + \frac{1}{12}x(i) + \frac{5}{8}x(i-2) + \frac{1}{4}x(i-3) + \frac{1}{8}x(i-5) \tag{5.2}$$

GP-Lösung nach der Konstanten-Optimierung:

$$\widehat{y}(i) = 0.3179\widehat{y}(i-1) - 0.1486\widehat{y}(i-4) - 0.1078\widehat{y}(i-5) + 0.0450\widehat{y}(i-6) + 0.0656x(i) + 0.6038x(i-2) + 0.2766x(i-3) + 0.1328x(i-5) \tag{5.3}$$

Die Gleichung (5.2) zeigt die von der GP gefundene beste Lösung. Der relative Restfehler ist $\varepsilon = 4.9 \cdot 10^{-3}$. Nach der Konstanten-Optimierung (Gleichung (5.3)) ist der relative Restfehler $\varepsilon = 3.2 \cdot 10^{-3}$. Der Verbesserungsgrad der Konstanten-Optimierung in diesem Beispiel ist:

$$\frac{4.9 \cdot 10^{-3}}{3.2 \cdot 10^{-3}} = 1.53$$

Dabei wurden nur die Koeffizienten in der Gleichung (5.2) optimiert. Die Abb. 5.1 zeigt den Identifikationseffekt im Zeitbereich nach der Konstanten-Optimierung.

Das Beispiel zeigt, dass die GP mit LZI-Grammatik ein LZI-System hinreichend gut identifizieren kann. Die Identifikationsgenauigkeit ist relativ hoch. Weil das LZI-System in der Theorie schon gut erforscht und gut gelöst worden ist, hoffen wir dabei nicht, dass die GP eine bessere Lösung bei der Genauigkeit als die traditionellen Verfahren finden kann. In der Tat ist das unmöglich. Durch den Vergleich zwischen der Gleichung (5.1) und (5.2) können wir sehen, dass die GP eine Ersatz-Struktur für das System findet. Mit dieser Ersatz-Struktur wird die Hauptinformation des Systems beschrieben. Das ist ein wichtiges Merkmal der GP. Bei der Systemidentifikation ist es sehr selten, dass die GP die echte Struktur des Simulationssystems findet. Dabei gibt es meistens zwei Möglichkeiten:

1. Die von der GP gefundene Struktur ist einfacher als die echte Struktur und der Fehler zwischen dem System und dem Modell erfüllt die Anforderung der Anwendung.

2. Die von der GP gefundene Struktur ist komplizierter als die echte Struktur und der Fehler zwischen dem System und dem Modell erfüllt die Anforderung der Anwendung.

Die erste Situation entspricht dem vorliegenden Fall.

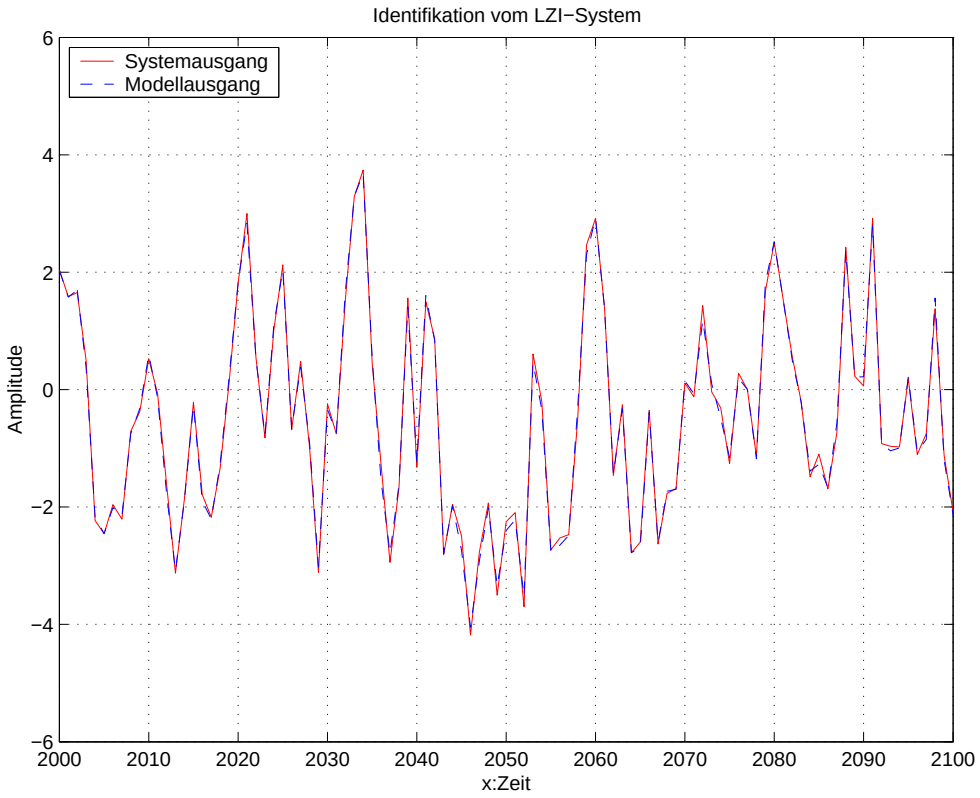


Abbildung 5.1: Identifikation eines LZI-Systems, nach der Gl. (5.1). Anzahl der unabhängigen Durchläufe=40, mit der Konstanten-Optimierung, ohne rekursives GP-Verfahren: $\varepsilon = 3.2 \cdot 10^{-3}$

5.2 Identifikation eines nichtlinearen Systems mit Gedächtnis

Für die Identifikation des allgemeinen nichtlinearen Systems mit Gedächtnis sollte man keine Beschränkung des GP-Suchraums setzen, weil in diesem Fall jede Art der Nichtlinearität möglich ist. In diesem Fall ist die Contextfreie Grammatik (CFG) geeignet.

5.2.1 Identifikation mit Konstanten-Optimierung

Die Stärke der GP ist die Struktur des Modells zu suchen. Exakte Anpassung der Parameter ist die größte Schwäche der GP. Wenn die GP eine geeignete Struktur des Modells gefunden hat, kann man mit traditionellen Verfahren oder modernen Verfahren die Parameter im Modell optimieren. Durch die Optimierungsparameter wird das Verhalten des Modells verbessert. Diese Methode heißt die Konstanten-Optimierung.

Weil die GP normalerweise ein Modell mit nichtlinearen Parametern findet, soll man in diesem Fall die Optimierungs-Verfahren für die nichtlinearen Parameter verwenden. In der Optimierungs-Toolbox von Matlab stehen diese Verfahren zur Verfügung. Der Vorteil dieser Verfahren ist es, dass sie zeitaufwandsarmer sind. Diese Verfahren haben aber einen Nachteil, und zwar zeigen sie lokale Minima. Lokale Minima sind unvermeidbar bei der Optimierung der nichtlinearen Parameter. Dafür gibt es einige Optimierungsstrategien[Nelles01][Kafka02] der nichtlinearen Parameteroptimierung, um möglichst das globale Minimum zu finden.

Um ein globales Minimum zu finden, kann es sein, dass der GA ein besseres Verfahren ist, weil der GA bessere globale Eigenschaft besitzt. Aber der GA ist sehr zeitaufwendig.

Im Experiment 1 wird ein nichtlineares System mit Gedächtnis, wie in der Gleichung (5.4) gezeigt, als Ziel-System verwendet. Dabei wird der Eingangsbereich x auf [-2, +2] beschränkt. Um die Suchfähigkeit der GP mit höherer Zuverlässigkeit zu testen, bieten wir in der GP-Funktionenmenge keine $tan()$-Funktion an. Die Gleichung (5.5) zeigt die von der GP gefundene beste Lösung dieses Systems.

$$y(i) = 0.9y(i-1) - 0.7y(i-2) - 0.2[1 + tan(\frac{x(i-1)}{2})] - 0.5[1 + tan(\frac{x(i-2)}{2})]^2 \quad (5.4)$$

$$\widehat{y}(i) = -\frac{1}{7}x(i-2) + \widehat{y}(i-1) - \frac{37}{56}\widehat{y}(i-2) - \frac{1}{4}e^{x(i-1)} - \frac{1}{8}e^{x(i-3)} - \frac{1}{7}e^{\widehat{y}(i-2)} - \frac{1}{8}sin(sin(e^{x(i-1)})) \quad (5.5)$$

Der relative Restfehler ist $\varepsilon = 8.3 \cdot 10^{-2}$. Nach der Konstanten-Optimierung ist der relative Restfehler $\varepsilon = 3.5 \cdot 10^{-2}$. Der Verbesserungsgrad der Konstanten-Optimierung in diesem Beispiel ist:

$$\frac{8.3 \cdot 10^{-2}}{3.5 \cdot 10^{-2}} = 2.37$$

Die Abb. 5.2 zeigt den Identifikationseffekt im Zeitbereich, dabei ist das Eingangssignal ein Sinussignal mit der Amplitude A=2. Auf dieser Abbildung 5.2 können wir die nichtlineare Verzerrung sofort sehen.

In diesem Beispiel ist die Identifikationsgenauigkeit niedriger als die bei der LZI-Systemidentifikation. In der Tat ist die Identifikation von nichtlinearen Systemen mit Gedächtnis viel schwieriger als die von LZI-Systemen.

Durch den Vergleich zwischen den Gleichungen (5.4) und (5.5) können wir sehen, dass die GP eine Ersatz-Struktur für das System findet. Mit dieser Ersatz-Struktur wird die grundlegende Information des Systems beschrieben. Das Problem ist, dass die von der GP gefundene Ersatz-Struktur nicht ausreichend gut ist, weil die Genauigkeit nach der Konstanten-Optimierung nicht ausreichend hoch ist. In einem späteren Experiment wird das rekursive GP-Verfahren verwendet, um eine bessere Lösung für eine Aufgabe zu finden.

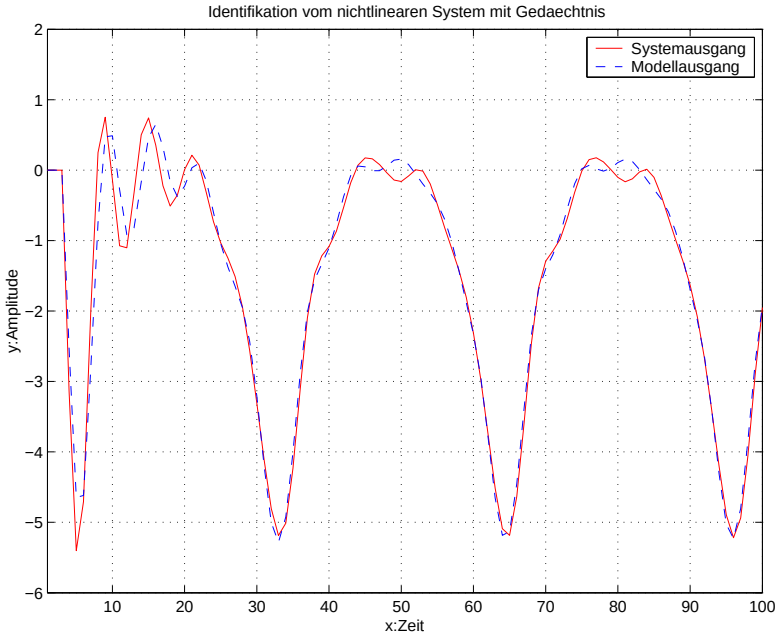


Abbildung 5.2: Identifikation eines nichtlinearen Systems mit Gedächtnis, wie in der Gl. (5.4) gezeigt. Anzahl der unabhängigen Durchläufen=40, mit der Konstanten-Optimierung, ohne rekursives GP Verfahren. $\varepsilon = 3.5 \cdot 10^{-2}$

5.2.2 Identifikation mit rekursivem GP-Verfahren

Das Prinzip der rekursiven GP-Verfahren[Panyaworayan02] ist zweischrittig, zuerst sucht die GP ein Modell, dann werden die Konstanten in diesem Modell optimiert. Falls das Modell die Anforderung nicht erfüllen kann, d.h. der Fehler zwischen dem System und dem Modell größer ist als ein vorgegebener Wert, sucht die GP mit diesem Fehlersignal ein neues Modell. Das neu gefundene Modell ist eine Ergänzung für das zuerst gefundene Modell, dabei können die Konstanten in den beiden Modellen insgesamt optimiert werden. Dieser Prozess kann mehrmals wiederholt werden. Dies heißt das rekursive GP-Verfahren. Im Kapitel 2 wurde dieses Verfahren als eine Strategie der Modellverbesserung beschrieben.

Im Experiment 2 wird ein nichtlineares System mit Gedächtnis, wie in den Gleichungen (5.6) und (5.7) gezeigt, als Ziel-System verwendet.

$$z(i) = 0.25[0.3x(i) - 0.13x(i-1) + 2.5x(i-2) + 0.9x(i-3) + 0.4x(i-4) + 0.2x(i-5) + 0.1x(i-6) + 1.5z(i-1) - 0.7z(i-2) + 0.9z(i-3) - 1.3z(i-4)] \tag{5.6}$$

$$y = 0.6z + 0.5z^2 + 1.2z^3 + 0.5z^4 + z^5 + 0.4z^6 + 1.4z^7 + 0.3z^8 + 1.5z^9 + 0.2z^{10} \tag{5.7}$$

Dabei wird der Eingangsbereich x auf [-1, +1] beschränkt. Um eine höhere Identifikationsgenauigkeit zu erreichen, verwenden wir in diesem Experiment das rekursive GP-Verfahren. Das Verfahren wird bis zur 4. Stufe durchgeführt. Die Gleichung (5.8) zeigt die von der GP gefundene beste Lösung dieses Systems. Der relative Restfehler

ist $\varepsilon = 6.1 \cdot 10^{-2}$. Die Abb. 5.3 zeigt die Veränderung des relativen Restfehlers in den verschiedenen Stufen. Die Verbesserungsgrade des rekursiven GP-Verfahrens in diesem Beispiel sind:

$$\frac{1.45 \cdot 10^{-1}}{6.3 \cdot 10^{-2}} = 2.30$$

Der gesamte Verbesserungsgrad des rekursiven GP-Verfahrens und der Konstanten-Optimierung ist:

$$\frac{1.45 \cdot 10^{-1}}{6.1 \cdot 10^{-2}} = 2.38$$

$$y(i) = y_1(i) + y_2(i) + y_3(i) + y_4(i); \tag{5.8}$$

Wobei:

$$y_1(i) = (Div_9((Mul((Sub((Sqr((Sub((Div_5((x_3))), (x_2))))), (Mul((Sub((Div_3((x_3))), (x_2))), (x_3))))), (Mul((Sub((Sub((x_5), (x_6))), (Sub((x_2), (Sub((x_5), (-1))))))), (Sub((Sub((Sub((Sqr((Mul((Sub((Div_3((x_3))), (x_2))), (Mul((Sub((Div_9((x_3))), (x_2))), (x_3))))))), (x_6))), (y_4))), (Sub((x_6), (-4)))))))))) \tag{5.9}$$

$$y_2(i) = (Add((Add((Add((Add((Add((Add((Add((Div7((x_0))), (Div_6((Sqr((Sqr((Cub((Cub((Sqr((Add((x_2), (Div9((Sqr((x_3)))))))))))))))))), (Div9((Sqr((x_3))))))), (Div_6((Div3((y_4))))))), (Div_6((Div7((y_1))))))), (Div_6((Div9((-1))))))), (Sqr((Sqr((Cub((Cub((Cub((x_3))))))))))), (Div9((Add((x_2), (Div9((Sqr((x_3))))))))))) \tag{5.10}$$

$$y_3(i) = (Div3((Cub((Div_9((Mul((Cub((Mul((Div8((x_5))), (Mul((Sqr((y_2))), (Div_6((x_2))))))))), (Cub((Div_3((Mul((Div_3((Cub((-3))))), (x_4))))))))))))) \tag{5.11}$$

$$y_4(i) = (Cub((Mul((Cub((Sqr((Sqr((Sqr((Add((x_2), (Sqr((Div10((Add((Add((Div10((5))), (Sqr((Add((x_2), (Add((Div10((Cub((Div9((-4))))))), (Div10((Sqr((Add((x_2), (Add((Div10((Div10((x_2))))), (x_5))))))))))))))))), (x_3))))))))))), (x_6)))))) \tag{5.12}$$

Wobei:
$Add(a, b) = a + b$
$Sub(a, b) = a - b$
$Mul(a, b) = a \cdot b$
$Sqr(a) = a^2$
$Cub(a) = a^3$
$Div_k(a) = -\frac{a}{k}$
$Divk(a) = \frac{a}{k}$
$x_j = x(i - j)$
$y_j = y(i - j)$

Aus der Abbildung 5.3 können wir sehen:

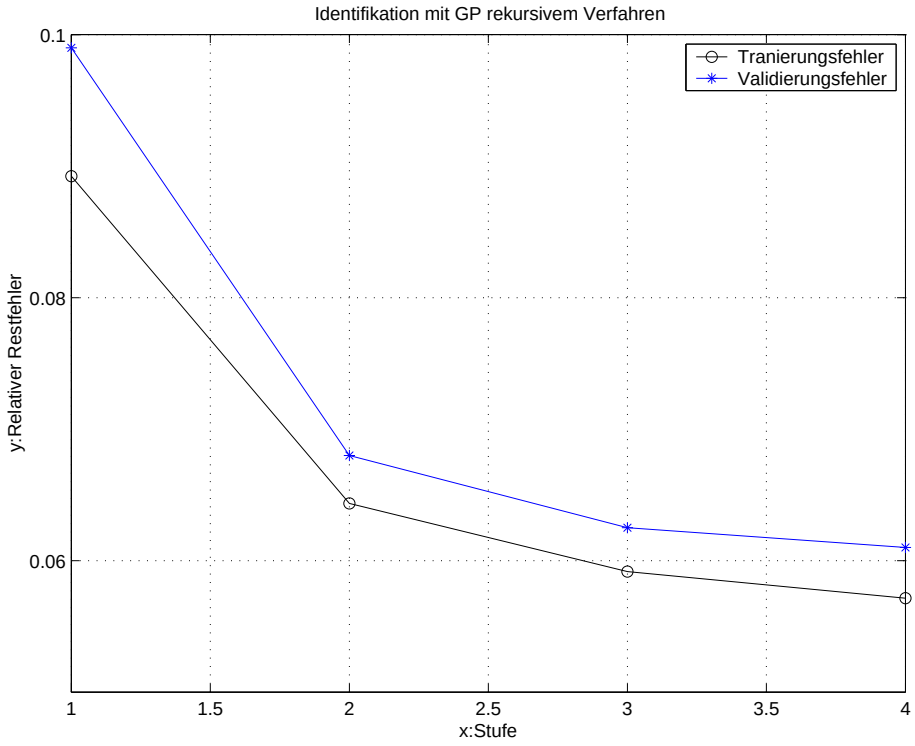


Abbildung 5.3: Relativer Restfehler in Abhängigkeit von der Anzahl der Iterationsstufen bei der Identifikation eines nichtlinearen Systems mit Gedächtnis, wie in Gl. (5.6) und (5.7) gezeigt. Dabei werden das rekursive GP Verfahren und die Konstanten-Optimierung verwendet. $\varepsilon = 6.1 \cdot 10^{-2}$ für die Modell-Validierung.

1. Je mehr Stufen verwendet werden, desto langsamer ist die Abfallgeschwindigkeit des relativen Restfehlers.

2. Die Abfallgeschwindigkeit des Validierungsfehlers ist noch langsamer als die des Trainingsfehlers.

Außerdem wird die Wirkung der Konstanten-Optimierung schwächer, wenn die Stufenanzahl groß wird. Obwohl wir die Komplexität des Modells nicht genau beschreiben können, zeigen die Gleichungen (5.9)–(5.12)[1] ungefähr einen proportionalen Zusammenhang zwischen der Komplexität des gesamten Modells und den Iterations-Stufen. Damit wird deutlich, dass das Verfahren nicht sehr geeignet ist für die Erhöhung der Genauigkeit durch eine große Anzahl von Stufen. Es ist auch selten möglich, eine extrem hohe Genauigkeit zu erreichen, weil es eine Grenze zwischen der Modellverbesserung und der Anzahl der Stufen gibt. Wir können diese Ursache mit der Abb. 5.4 anschaulich zeigen.

[1]Dabei zeigen wir die Gleichung in der originalen Form der GP. Mit einem Formeleditor könnten die Gleichungen (5.9)–(5.12) wesentlich vereinfacht werden.

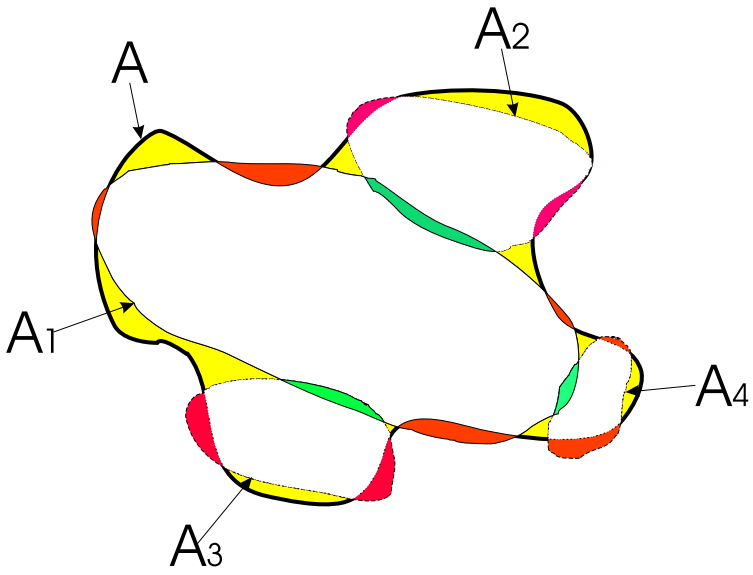


Abbildung 5.4: Veranschaulichung der Genauigkeitsverbesserung durch das rekursive GP-Verfahren.

Dabei bedeutet der Bereich A die perfekte Lösung der Aufgabe. Das Auffinden dieses Bereichs (A) in einem einmaligen GP-Durchlauf ist fast unmöglich . Im rekursiven GP-Verfahren kann es sein, dass die GP in der ersten Stufe nur den Bereich A_1 findet. In der zweiten Stufe findet die GP den Bereich A_2, usw.. Die Komplexität der Restbereiche bzw. der Überlagerungsbereiche ist größer als die Komplexität des Bereichs A. Dies verursacht die verlangsamte Abfallgeschwindigkeit des relativen Restfehlers. Wenn die Anzahl der Stufen groß genug ist, sind die Restbereiche bzw. die Überlagerungsbereiche ähnlich wie ein Rauschen. Also kann die Lösung kaum weiter verbessert werden. Gleichzeitig, wegen der zu hohen Komplexität der Lösung ergeben sich zu viele lokale Minima bei der Konstanten-Optimierung und damit wird die Wirkung der Konstanten-Optimierung kleiner.

Die gleichzeitige Anwendung von der Konstanten-Optimierung und dem rekursiven GP-Verfahren ist daher kritisch. Außerdem ist die Wirkung des rekursiven GP-Verfahrens stark von der Populationsgröße abhängig. D.h. je kleiner die Populationsgröße ist, desto stärker ist die Wirkung des Verfahrens. Aber es ist für komplizierte Aufgaben keine gute Wahl, eine kleine Populationsgröße zu verwenden.

Bei der Identifikation der allgemeinen stark nichtlinearen Systeme mit Gedächtnis kann die GP durch Konstanten-Optimierung und rekursives GP-Verfahren einen Restfehler von ungefähr 10^{-2} erreichen.

5.2.3 Stabilitätsproblem in der Identifikation

In diesem Experiment wird ein nichtlineares System mit Gedächtnis, wie in der Gleichung (5.13) gezeigt, als Ziel-System verwendet. Dabei wird der Eingangsbereich x auf [-3, +3] beschränkt.

$$y(i) = 0.01867 arctan(x(i-1)) + 0.01746 arctan(x(i-2)) + 1.7826 y(i-1) - 0.8187 y(i-2) \quad (5.13)$$

Test 1

In diesem Entwurf verwenden wir die Terminalmenge T=$\{x(i), x(i-1), x(i-2), ..., y(i-1), y(i-2), ...\}$, die Funktionenmenge F=$\{+, -, *, /, sin, cos\}$. Durch 50 unabhängige GP-Durchläufe haben wir *keine* Lösung gefunden, die bei der Systemsimulation[2] stabil ist.

Wenn es in der Terminalmenge T die Rückkopplungs-Terme $y(i-1), y(i-2), ...$ gibt, ist es möglich, dass die GP eine relativ einfache Lösung findet. Im Gegensatz dazu ist es ziemlich schwierig, die Stabilität des Individuums zu steuern, weil die Fitness des Individuums durch den einstufigen Prädiktions-Fehler[3] berechnet wird. Eine Alternative ist es, die Fitness des Individuums durch den Simulationsfehler zu berechnen. Aber unsere Untersuchung zeigt, dass der Simulationsfehler das Stagnieren der Evolution verursacht. Die wichtigste Ursache dafür ist, dass die Simulation den Fehler über die Zeit aufakkumuliert. Ein Modell besteht aus zwei Teilen: Struktur und Parameter. Eine gute Struktur mit falschen Parametern kann ein sehr schlechtes Simulations-Ergebnis erzeugen, insbesondere für nichtlineare Systeme. Die Wirkung der Fehler-Akkumulierung bei der Fitnessberechnung beschränkt also die GP-Suchfähigkeit der Struktur.

Test 2

Die Stabilität der allgemeinen nichtlinearen Systeme ist ein sehr schwieriges Problem. Wegen der Veränderung der Struktur ist es in der GP-Systemidentifikation noch schwieriger. In diesem Test verwenden wir die Terminalmenge:

T=$\{x(i), x(i-1), x(i-2), ...\}$, *ohne* Rückkopplungs-Terme $y(i-1), y(i-2), ...$

Offensichtlich ist es bei diesem Fall unmöglich, dass die GP die Lösung wie die Gleichung (5.13) finden kann. Aber es ist garantiert, dass die Lösung stabil ist, weil die Lösung in diesem Fall ähnlich wie ein Kolmogorov-Polynom ist. Außerdem sind der Simulationsfehler und der einstufige Prädiktionsfehler in diesem Fall identisch.

[2] Siehe Kapitel 2.

[3] Der Unterschied zwischen dem Simulationsfehler und dem einstufigen Prädiktionsfehler siehe Kapitel 2.

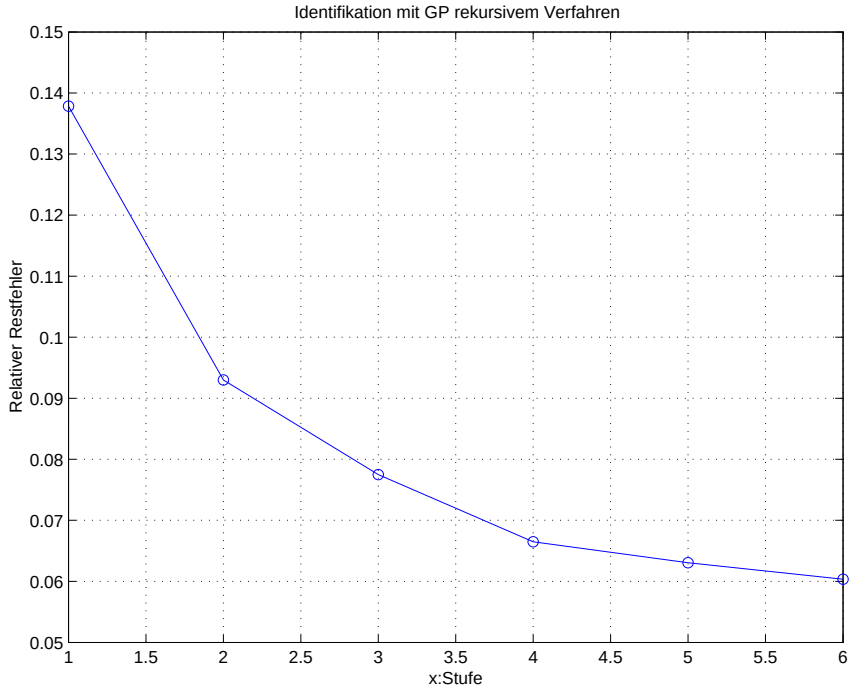


Abbildung 5.5: Relativer Restfehler in Abhängigkeit von der Anzahl der Iterationsstufen bei der Identifikation eines nichtlinearen Systems mit Gedächtnis, wie in Gl. (5.13) gezeigt. Dabei werden das rekursive GP-Verfahren und die Konstanten-Optimierung verwendet. In der Terminalmenge gibt es *keine* Rückkopplungs-Terme. $\varepsilon = 6.0 \cdot 10^{-2}$.

Die Gleichung (5.14) zeigt die von der GP gefundene beste Lösung dieses Systems. Der relative Restfehler ist $\varepsilon = 6.0 \cdot 10^{-2}$. Die Abb. 5.5 zeigt die Veränderung des relativen Restfehlers in verschiedenen Stufen. Dabei wurde das rekursive GP Verfahren bis zur 6. Stufe durchgeführt. Es ergibt sich, dass die Komplexität der gesamten Lösung unterproportional mit der Anzahl der Stufen anwächst. Damit kann man in diesem Fall mehrere rekursive Stufen durchführen.

$$y(i) = y_1(i) + y_2(i) + y_3(i) + y_4(i) + y_5(i) + y_6(i) \tag{5.14}$$

Wobei:

$$y_1(i) = (Div9((Div2((Add((Add((Add((Add((Add((x_6),(x_{10}))),(x_9))),(Add((Div2((Add((Add((x_3),(Add((Add((x_{13}),(Div9((x_8))))),(x_{12}))))),(Div2((Add((Add((x_7),(x_{11}))),(x_{12}))))))))),(x_4))))),(Add((Div2((Add((Add((x_7),(x_{11}))),(x_2))))),(x_5))))),(x_8)))))) \tag{5.15}$$

$$y_2(i) = (Div_7((Div_8((Add((x_{15}),(Add((x_7),(x_{14}))))))))) \tag{5.16}$$

$$y_3(i) = (Div9((Div8((Sub((Sub((x_3), (Div9((Add((Cub((x_{10}))), (Cub((x_8))))))))), (Div8((Sub((Add((Cub((x_4))), (Div9((x_3))))), (Div10((Div8((Cub((x_{16})))))))))))))))) \tag{5.17}$$

$$y_4(i) = (Div9((Div_{10}((Sub((Sub((Sub((Mul((Sqr((x_9))), (Cub((Cub((Sqr((Div_5 ((Div3((Div7((Cub((Div3((Div3((Div_5((Sqr((Div_5((Mul((-6), (Cub((Cub((Div9 ((Cub((Sqr((Div_5((Div3((Div7((Div_5((4)), (x_{17}))), (x_{16}))), (x_1)))))))) \tag{5.18}$$

$$y_5(i) = (Cub((Add((Div_8((Div8((Cub((x_9)))))))), (Div3((Div_2((Div_7((x_{14})))))))))) \tag{5.19}$$

$$y_6(i) = (Cub((Div9((Sub((Div4((x_{14}))), (Div8((Cub((x_7))))))))))) \tag{5.20}$$

Der Vorteil dieses Tests ist es, dass die Lösung der GP bei der Simulation immer stabil ist. Der Nachteil ist, dass die Chance verloren geht, mit der GP eine relativ einfache Lösung zu finden.

5.3 Weitere Verbesserungsmöglichkeit der Suchfähigkeit der GP

Die GP-Suchfähigkeit ist zunächst von den Steuerparametern der GP wie z.B. der Populationsgröße M, der maximalen Anzahl der Generation G_{max} usw. abhängig(siehe Grundlage der GP im Kapitel 3). Durch eine größere Populationsgröße und eine höhere Generationenanzahl kann die GP-Suchfähigkeit erhöht werden. In unseren Untersuchungen wurden M=10000–15000 und $G_{max} = 70$ verwendet[4]. In diesem Abschnitt werden wir eine andere Strategie darstellen, um die Lösungsmöglichkeit der GP weiter zu verbessern.

5.3.1 Polynom GP-Verfahren

Das Polynom GP-Verfahren [Nikolay01] basiert auf der Gruppe der sogenannten Basisfunktionen. Jede Basisfunktion ist z.B. ein Polynom mit zwei Argumenten und höchstens von 2. Ordnung. Dabei besteht die GP-Funktionenmenge aus diesen Basisfunktionen. Die GP-Terminalmenge besteht aus dem aktuellen Wert und den vergangenen Werten des Systemeingangs.

[4]Im Allgemeinen ist das eine relativ große Populationsgröße in der GP.

Beispielsweise erzeugt die GP ein Individuum, wie es in der Abb. 5.6 gezeigt wird. Dabei bedeuten $f_1(v_1, v_2), f_2(v_1, v_2), f_3(v_1, v_2), f_9(v_1, v_2)$ bei diesem Individuum verwendete Basisfunktionen. Jede Basisfunktion hat zwei Argumente v_1 und v_2. v_1 und v_2 entsprechen dem linken und dem rechten Zweig in den Knoten des Baumes. Durch das Einsetzen mehrerer Knoten bildet sich ein Baum. Die Terminal-Blätter sind die Systemeingangssignale. In diesem Beispiel verwendet die GP nur 4 interne Knoten (Basisfunktionen). Die Baum-Struktur ist zwar einfach, aber die Komplexität der von diesem Baum beschriebenen Lösung ist relativ groß.

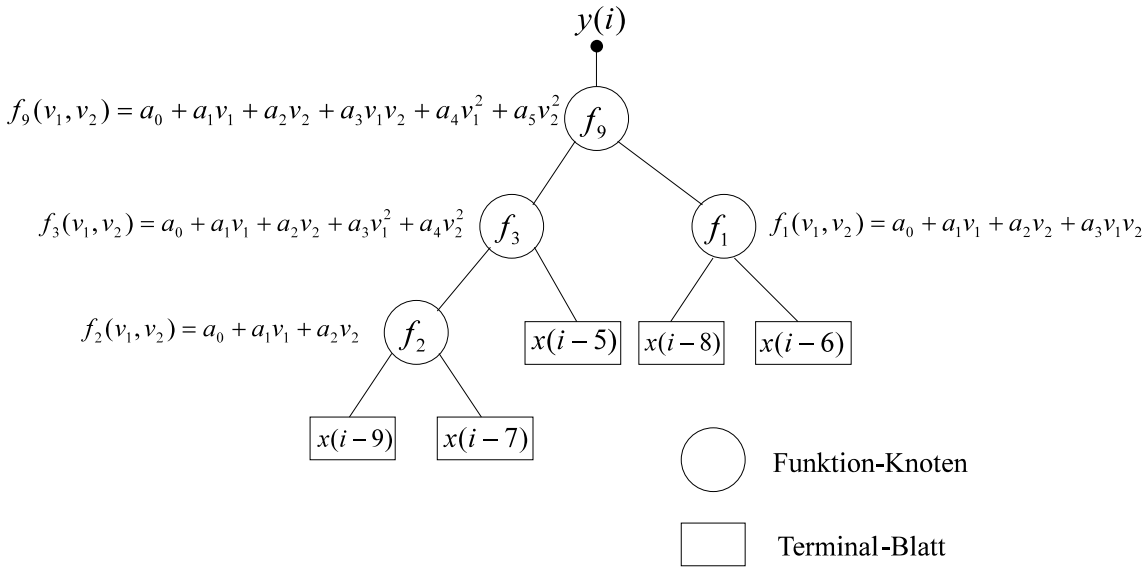


Abbildung 5.6: Veranschaulichung eines Individuums beim Polynom GP-Verfahren.

Die Besonderheiten dieses Verfahrens sind:

1. Mit einer einfachen Struktur kann eine relativ komplizierte Gleichung beschrieben werden.
2. Die Fitnessberechnung ist nicht aufwendig.
3. Das Overfitting kann besser vermieden werden.
4. Die Güte der Ergebnisse ist von der Auswahl der Basisfunktionen abhängig. Aber dies bildet möglicherweise auch einen Vorteil, wenn man eine geeignete Basisfunktionsgruppe für eine Aufgabe bestimmen kann.
5. Die Ergebnisse sind in der Praxis schwierig zu erklären. (Vergleich mit SMOG GP-Verfahren, siehe Abschnitt 5.3.2).

Übrigens kann man das Polynom GP-Verfahren und das rekursive GP-Verfahren gut kombinieren.

5.3.2 SMOG GP-Verfahren

Das SMOG (Struktured MOdel Generator) GP-Verfahren [www01] basiert auf einer Menge von Systembausteinen. Für eine konkrete Aufgabe werden zunächst manche Elemente (Bausteine), die im Modell enthalten sein sollten, vorgegeben. Die GP verbindet diese Elemente intensiv durch das evolutionäre Verfahren, um eine geeignete Modell-Struktur zu erzeugen. Gleichzeitig werden die Parameter in den Modellen (Individuen) optimiert, und die Modelle werden simuliert.

Die Parameter-Optimierung und die Modell-Simulation sind viel zeitaufwendiger als die reine Fitnissberechnung. Es kann daher möglich sein, dass der Simulationsfehler als Fitness des Individuums herausgezogen wird. Dabei ergibt sich noch ein Vorteil, d.h. die gefundene beste Lösung ist stabil in der Simulation, weil die instabilen Modelle in der Simulation auf Grund des schlechteren Fitnesswertes verworfen wurden. Die Parallele Berechnung ist geeignet für die Parameter-Optimierung und die Modell-Simulation. Damit ist die Clusterung ein guter Entwurf für die beiden Aufgaben.

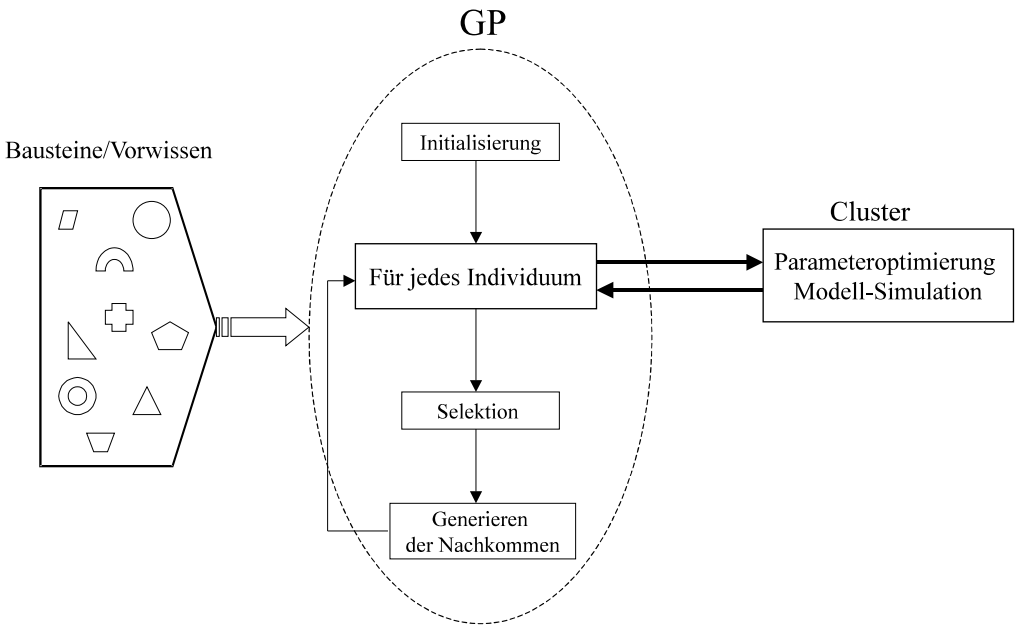


Abbildung 5.7: Veranschaulichung des SMOG GP-Verfahrens.

Die Abb. 5.7 zeigt anschaulich das SMOG GP-Verfahren. Dabei bedeutet die linke Seite die Menge der Bausteine, in der Mitte liegt die allgemeine GP und die rechte Seite zeigt die Parameter-Optimierung und die Modell-Simulation. Dabei können wir sehen, dass die wesentliche Suchfähigkeit von der linken Seite (Vorwissen) abhängt. Nur wenn die geeigneten Bausteine (durch das Vorwissen) für eine konkrete Aufgabe gegeben sind, kann die GP eine gute Lösung finden. Man kann sagen, das SMOG GP-Verfahren ist ein allgemeines Verfahren, wenn die Bausteinemenge

für ein Anwendungsgebiet bekannt ist. Im Allgemeinen sind die folgenden Punkte zu beachten:

1. Wenn man ein genaues Modell finden will, sollten die Bausteine, die im Modell enthalten sein sollen, möglichst genau zu der Aufgabe passen. Die Bausteine, die irrelevant für die Aufgabe sind, sollten möglichst vermieden werden.

2. Um eine Lösung zu finden, sollte die Anzahl der Bausteine umfangreich sein.

Die Vorteile dieses Verfahrens sind:

1. Die von der GP erzeugten Modelle sind in der Praxis möglicherweise erklärbar oder interpretierbar, weil die Bausteine praktische Bedeutungen besitzen.

2. Das von der GP gefundene beste Modell ist in der Simulation stabil.

3. Es ist möglich, dass die GP eine sehr gute Lösung findet, wenn die Bausteine genau zu der Aufgabe passen.

Die Nachteile dieses Verfahrens sind:

1. Die Güte der Ergebnisse ist stark von der Auswahl der Bausteine abhängig.

2. Es kann schwierig sein, die geeigneten Bausteine für eine Aufgabe (Anwendungsgebiet) zu finden.

Übrigens ist dieses Verfahren ein typisches Beispiel für "Der Mensch hilft der GP die Lösung zu finden" (siehe Kapitel 3).

5.4 Zusammenfassung

In diesem Kapitel testen wir durch einige Beispiele die Fähigkeit der GP zur Identifikation eines Systems mit Gedächtnis. In der Allgemeinheit bieten wir dabei kein Vorwissen über das zu identifizierende System an. Das heißt:

1. In der Terminalmenge T gibt es keine besonderen Bausteine über das zu identifizierende System.

2. In der Erzeugungsregel für die Individuen der GP gibt es keine besonderen Beschränkungen.

Im Allgemeinen kann die GP ohne Vorkenntnis über das Ziel-System kaum eine sehr gute Lösung finden. Durch die Anwendung der LZI-Grammatik kann die GP allerdings eine relativ gute Lösung finden. Aber die LZI-Grammatik passt nur für die Identifikation eines LZI-Systems.

Um die GP-Suchfähigkeit zu verbessern, verwenden wir ein rekursives GP-Verfahren und die Konstanten-Optimierung. Die Vorteile und die Nachteile dieser Verfahren

wurden analysiert. Eine weitere Verbesserung der GP-Suchfähigkeit ist durch die Verwendung des Polynom GP-Verfahrens und des SMOG GP-Verfahrens möglich. Ihre Vor- und Nachteile wurden auch vorgestellt.

Durch Experimente und die Analyse der unterschiedlichen GP-Verbesserungsverfahren haben wir festgestellt, dass bei der Identifikation eines nichtlinearen Systems mit Gedächtnis, das Vorwissen über das zu identifizierende System sehr wichtig ist, um ein gutes Ergebnis mit der GP zu erreichen. Das Vorwissen ist eigentlich die Vorkenntniss des Menschen über die konkrete Aufgabe. Dies führt zu der Aussage "Mensch hilft der GP". Darauf aufbauend wird im Kapitel 6 ein Kombinationsalgorithmus entwickelt. Dieser Kombinationsalgorithmus basiert z.B. auf dem Wiener-Modell (Vorwissen) und ist geeignet für die Kompensation von nichtlinearen Verzerrungen. Durch diesen Kombinationsalgorithmus sind die Ergebnisse bei den iterativen Prozessen Schritt für Schritt verbesserbar. Das Wichtigste ist, dass die Komplexität der endgültigen Lösung von der Anzahl der iterativen Stufe unabhängig ist, weil jede iterative Stufe nur eine Zwischenlösung erzeugt. Dieser Kombinationsalgorithmus nutzt den Vorteil der GP und des iterativen Verfahrens. Damit können wir sehr gute Identifikations- und Kompensationsergebnisse erreichen.

Kapitel 6

Kompensation von nichtlinearen Verzerrungen die hinreichend gut durch ein Wiener-Modell beschreibbar sind

Die Kompensation von starken Nichtlinearitäten der mit einem Wiener-Modell beschreibbaren NL-Systeme ist ein praktisch sinnvolles Thema. Dabei gibt es zwei Ziele:

1. Hohe Genauigkeit der Kompensation.
2. Niedriger Rechenaufwand für die Kompensation.

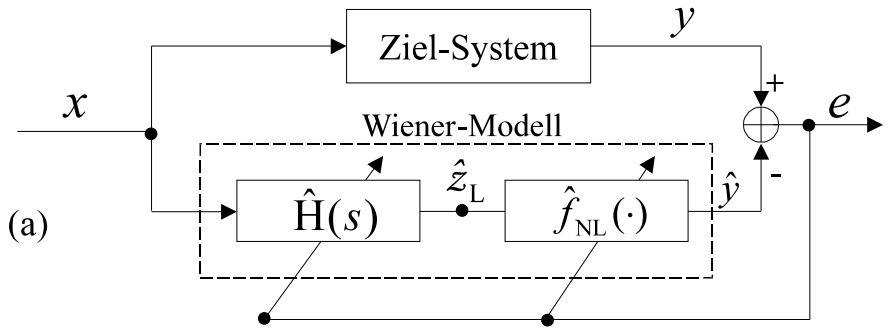
Nur wenn diese zwei Bedingungen gleichzeitig erfüllt werden können, ist der Kompensationsalgorithmus anwendungsfähig.

Um diese zwei Ziele zu erreichen, werden zwei Maßnahmen verwendet:

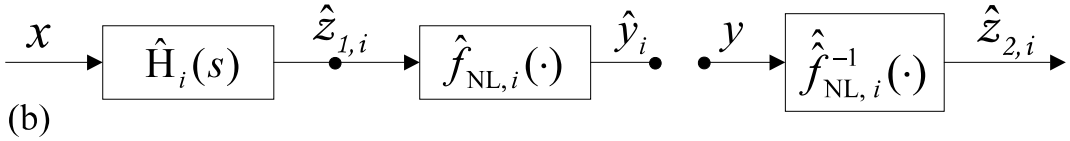
1. Ein neuer Kombinationsalgorithmus wird entwickelt, der im ersten Schritt mit Hilfe der GP eine Systemidentifikation der linearen und nichtlinearen Systemanteile im Wiener-Modell durchführt. Im zweiten Schritt wird ein Kompensationssystem für den NL-Anteil gesucht und unter der Annahme, dass diese Kompensation ideal sei, im dritten Schritt eine erneute Identifikation des verbleibenden linearen Systems mit der GP durchgeführt. Ausgehend von diesem verbesserten linearen Systemanteil, wird erneut der nichtlineare Systemanteil mit der GP identifiziert und erneut ein verbessertes Kompensationssystem gesucht. Aufbauend auf diesem verbesserten NL-System kann man den linearen Anteil erneut identifizieren usw. Je nach Anwendungsfall sind 3 bis 10 Iterationen dieser Art erforderlich.
2. Auffinden einer Methode mit niedrigem Rechenaufwand für die Kompensation ist möglich, wenn man von einer geeigneten Näherung für die Kennlinie des NL-Systemanteils ausgeht, wie z.B. einer Polynomnäherung.

Durch diese zwei Maßnahmen werden die zwei obigen Ziele erreicht.

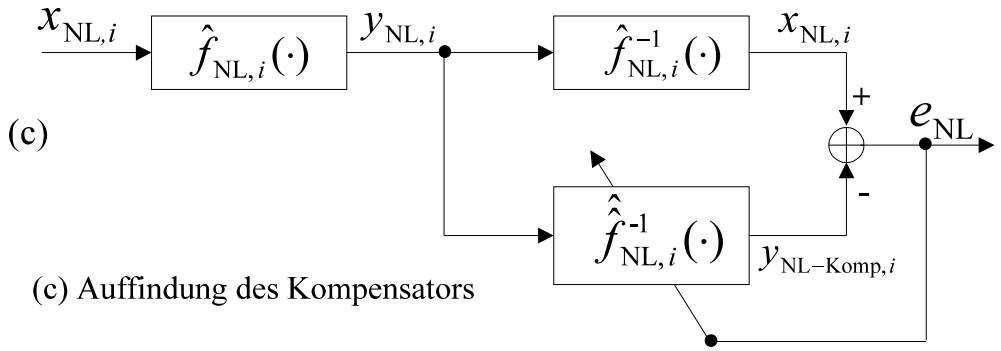
In diesem Kombinationsalgorithmus sind die GP und die Iteration die Basis des ganzen Verfahrens. Die Realisierung der Kombination von der GP-Methode und der Iteration ist das Kernstück des Kombinationsalgorithmus. Die Überwachung der Konvergenz des Kombinationsalgorithmus ist ein wichtiger Bestandteil der folgenden Überlegungen.



(a) Ein Ziel-System wird mit einem Wiener-Modell identifiziert



(b) Identifikation wird durch einen iterativen Prozess durchgeführt



(c) Auffindung des Kompensators

Abbildung 6.1: Veranschaulichung der 3 Schritte des Kombinationsalgorithmus.

Um das Verhalten des Kombinationsalgorithmus genau darzustellen, werden zuerst einige Vereinbarungen und Definitionen getroffen.

Vereinbarungen und Definitionen:

x: Eingangssignal des Ziel-Systems.

x_{NL}: Anregungssignal des NL-Anteils zur Auffindung des Kompensators.

$x_{\mathrm{NL},i}$: Anregungssignal des NL-Anteils zur Auffindung des Kompensators in der $i-ten$ iterativen Stufe.

z_L: ideales LZI-Ausgangssignal des Ziel-Systems.

$\widehat{z}_L$: geschätzter Wert von z_L in der Endstufe.

$\widehat{z}_{1,i}$: geschätztes LZI-Ausgangssignal der $i-ten$ iterativen Stufe.

$\widehat{z}_{2,i}$: kompensierter Ausgang der $i-ten$ iterativen Stufe.

$\widehat{z}$: kompensierter Ausgang in der Endstufe.

$f_{\mathrm{NL}}(\cdot)$: ideale NL-Kennlinienfunktion des NL-Systemanteils.

$\widehat{f}_{\mathrm{NL}}(\cdot)$: geschätzte Funktion von $f_{\mathrm{NL}}(\cdot)$.

$\widehat{f}^{-1}_{\mathrm{NL}}(\cdot)$: ideale Inverse Funktion von $\widehat{f}_{\mathrm{NL}}(\cdot)$.

$\widehat{\widehat{f}^{-1}}_{\mathrm{NL}}(\cdot)$: geschätzte Inverse Funktion von $\widehat{f}_{\mathrm{NL}}(\cdot)$.

$\widehat{f}_{\mathrm{NL},i}(\cdot)$: geschätzte NL-Kennlinienfunktion des NL-Systemanteils in der $i-ten$ iterativen Stufe.

$\widehat{f}^{-1}_{\mathrm{NL},i}(\cdot)$: ideale Inverse-Funktion von $\widehat{f}_{\mathrm{NL},i}(\cdot)$.

$\widehat{\widehat{f}^{-1}}_{\mathrm{NL},i}(\cdot)$: geschätzte Inverse-Funktion von $\widehat{f}_{\mathrm{NL},i}(\cdot)$.

y: Ausgangssignal des Ziel-Systems.

$\widehat{y}_i$: Ausgangssignal des gesamten Modells in der $i-ten$ iterativen Stufe

$\widehat{y}$: Ausgangssignal des gesamten Modells in der Endstufe.

y_{NL}: Ausgangssignal des NL-Anteils gegenüber dem Anregungssignal x_{NL}.

$y_{\mathrm{NL},i}$: Ausgangssignal des NL-Anteils gegenüber dem Anregungssignal $x_{\mathrm{NL},i}$ in der $i-ten$ itrativen Stufe.

$y_{\text{NL-Komp}}$: Ausgangssignal des Kompensators gegenüber dem Anregungssignal x_{NL}.

$y_{\text{NL-Komp},i}$: Ausgangssignal des Kompensators gegenüber dem Anregungssignal $x_{\text{NL},i}$ in der $i-ten$ itrativen Stufe.

$\varepsilon_{L,i}$: relativer Restfehler zwischen idealem LZI-Ausgangssignal und geschätztem LZI-Ausgangssignal der $i-ten$ iterativen Stufe.

Definition: $\varepsilon_{L,i} = \frac{\int [z_L(t) - \widehat{z}_{1,i}(t)]^2 dt}{\int z_L^2(t) dt} \quad ; \quad i = 1, 2, 3...$

$\Delta\varepsilon_{L,i}$: relativer Restfehler zwischen den geschätzten LZI-Ausgangssignalen der $(i-1)-ten$ und der $i-ten$ iterativen Stufe.

Definition: $\Delta\varepsilon_{L,i} = \frac{\int [\widehat{z}_{1,i}(t) - \widehat{z}_{1,i-1}(t)]^2 dt}{\int \widehat{z}_{1,i-1}^2(t) dt} \quad ; \quad i = 2, 3, 4...$

$\varepsilon_{N,i}$: relativer Restfehler zwischen der idealen NL-Kennlinie des NL-Systemanteils und der geschätzten NL-Kennlinie des NL-Systemanteils in der $i-ten$ iterativen Stufe.

Definition: $\varepsilon_{N,i} = \frac{\int [f_{\text{NL}}(x_{\text{NL}}(t)) - \widehat{f}_{\text{NL},i}(x_{\text{NL}}(t))]^2 dt}{\int f_{\text{NL}}^2(x_{\text{NL}}(t)) dt} \quad ; \quad i = 1, 2, 3...$

$\Delta\varepsilon_{N,i}$: relativer Restfehler zwischen den geschätzten NL-Kennlinien des NL-Systemanteils der $(i-1)-ten$ und der $i-ten$ iterativen Stufe.

Definition: $\Delta\varepsilon_{N,i} = \frac{\int [\widehat{f}_{\text{NL},i}(x_{\text{NL}}(t)) - \widehat{f}_{\text{NL},i-1}(x_{\text{NL}}(t))]^2 dt}{\int \widehat{f}_{\text{NL},i-1}^2(x_{\text{NL}}(t)) dt} \quad ; \quad i = 2, 3, 4...$

ε_i: relativer Restfehler zwischen dem Ausgangssignal des Ziel-Systems und dem Ausgangssignal des Modells in der $i-ten$ iterativen Stufe.

Definition: $\varepsilon_i = \frac{\int [y(t) - \widehat{y}_i(t)]^2 dt}{\int y^2(t) dt} \quad ; \quad i = 1, 2, 3...$

ε: relativer Restfehler zwischen dem Ausgangssignal des Ziel-Systems und dem Ausgangssignal des Modells in der Endstufe.

Definition: $\varepsilon = \frac{\int [y(t) - \widehat{y}(t)]^2 dt}{\int y^2(t) dt} \quad ; \quad i = 1, 2, 3...$

$\varepsilon_{KP,i}$: relativer Restfehler zwischen idealem LZI-Ausgang und kompensiertem Ausgang der $i-ten$ iterativen Stufe

Definition: $\varepsilon_{KP,i} = \frac{\int [z_L(t) - \widehat{z}_{2,i}(t)]^2 dt}{\int z_L^2(t) dt} \quad ; \quad i = 1, 2, 3...$

ε_{KP}: relativer Restfehler zwischen idealem LZI-Ausgang und kompensiertem Ausgang in der Endstufe

Definition: $\varepsilon_{KP} = \frac{\int [z_L(t) - \hat{z}(t)]^2 dt}{\int z_L^2(t) dt}$; $i = 1, 2, 3...$

$\varepsilon_{KL,i}$: relativer Restfehler der Kompensation der identifizierten NL-Kennlinie in der $i - ten$ iterativen Stufe

Definition: $\varepsilon_{KL,i} = \frac{\int [x_{\mathrm{NL}}(t) - y_{\mathrm{NL\text{-}Komp},i}]^2 dt}{\int x_{\mathrm{NL}}^2(t) dt}$

ε_{KL}: relativer Restfehler der Kompensation der identifizierten NL-Kennlinie in der Endstufe.

Definition: $\varepsilon_{KL} = \frac{\int [x_{\mathrm{NL}}(t) - y_{\mathrm{NL\text{-}Komp}}(t)]^2 dt}{\int x_{\mathrm{NL}}^2(t) dt}$

R: Rechenaufwand für die Operation M oder P oder exp oder sin oder cos.

Dabei bedeutet:
M: Multiplizeren oder Dividieren; P: Plus oder Minus; exp: exp; sin: sin; cos: cos

Bei den numerischen Berechnungen im Computer wurden die Integrale durch Summen ersetzt.

Wenn der Kombinationsalgorithmus konvergiert, gilt $\widehat{z}_{1,i} \approx \widehat{z}_{2,i} \approx z_L \approx \widehat{z}_L \approx \widehat{z}$.

Die Symbole werden in der Abb. 6.1 veranschaulicht.

6.1 Wiener-Modell

Bei der Kompensation und der Identifikation von unbekannten Systemen passt der Kombinationsalgorithmus zum Wiener-Modell, das in der Abbildung 6.2 gezeigt wird, solange die NL-Kennlinie in diesem Modell monoton ist. Die monotone Eigenschaft der NL-Kennlinie stellt sicher, dass das ganze System kompensierbar ist. Der Kombinationsalgorithmus und das Wiener-Modell sind also für die Kompensation der nichtlinearen Verzerrungen besonders geeignet. Falls die NL-Kennlinie im Wiener-Modell nicht monoton ist, ist das System im Allgemeinen nicht genau kompensierbar. Die wichtigste Voraussetzung für die Anwendung dieses Kombinationsalgorithmus ist, dass die NL-Kennlinie monoton ist.

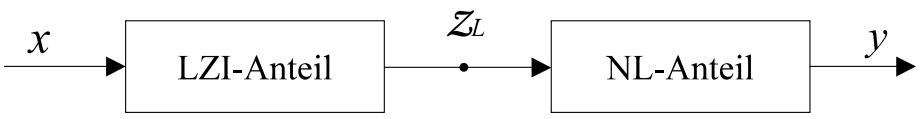


Abbildung 6.2: Zum Kombinationsalgorithmus passendes Wiener-Modell.

Dabei bedeutet

LZI: Linearer ZeitInvarianter Systemanteil im Wiener-Modell.
NL: gedächtnisloser NichtLinearer Systemanteil im Wiener-Modell.

Das ganze System stellt ein nichtlineares System mit endlich langem Gedächtnis dar.

Das Wiener-Modell hat in den folgenden Fällen Bedeutungen:

1. Manche technischen Systeme besitzen eine System-Struktur, die wirklich dem Wiener-Modell entspricht. In diesem Fall ist es definitiv möglich, dass das Wiener-Modell das System darstellen kann.

2. Manche Systeme können mit einem Wiener-Modell gut angenähert werden. In diesem Fall ist eine gegebene Näherungsgenauigkeit erforderlich.

Der Kombinationsalgorithmus passt zu den obigen zwei Situationen, solange die NL-Kennlinie monoton ist.

Im Wiener-Modell (Abb. 6.2) bedeutet x das Eingangssignal des Ziel-Systems, y das Ausgangssignal des Ziel-Systems und z_L das Zwischensignal zwischen dem LZI- und dem NL-Systemanteil. Davon sind x und y beobachtbar und z_L nicht beobachtbar.

Um die Fähigkeit des Kombinationsalgorithmus zu testen, wird in unseren Simulationen ein LZI- und NL-Systemanteil, wie sie in Gleichung (6.1) und (6.2) beschrieben werden, verwendet. Die zwei Gleichungen sind für den Test zwar zufällig gewählt, aber im Allgemeinen können nur stabile Systeme betrachtet werden. Weitere Einschränkungen sind nicht erforderlich.

1. Der LZI-Systemanteil besitzt 11 Koeffizienten und verschiedene Arten der Rückkopplungen, und zwar Negative und Positive.

2. Der NL-Systemanteil ist kompensierbar im Eingangswertebereich [-2, +2].

3. Das Ziel-System ist kompensierbar im Ausgangswertebereich [-5.2, +5.2].

$$z_L(i) = 0.18 \cdot [0.3x(i) - 0.13x(i-1) + 2.5x(i-2) + 0.9x(i-3) + 0.4x(i-4) + 0.2x(i-5) + 0.1x(i-6) + 1.5z_L(i-1) - 0.7z_L(i-2) + 0.9z_L(i-3) - 1.3z_L(i-4)] \tag{6.1}$$

$$y = z_L + 0.8z_L^2 + 0.4z_L^3 - 0.2z_L^4 \tag{6.2}$$

Im Allgemeinen sind die positiven Rückkopplungen der hohen Ordnung ein "gefährlicher" Term. "Gefährlich" bedeutet dabei, falls ein System mit bestimmten Rückkopplungen besonders beschränkt wird, ist es möglich, dass das System ein

chaotisches Verhalten zeigt und zu einem Chaos-System wird oder dass das System nicht stabil ist. Ein Chaos-System oder ein unstabiles System ist nicht kompensierbar. In unserem Beispiel treten keine Chaos-Eigenschaften auf und das Ziel-System muss stabil sein, weil es unser Ziel ist, die nichtlinearen Verzerrungen zu kompensieren.

Da der Eingangswertebereich die Eigenschaft des nichtlinearen Ziel-Systems stark beeinflusst, wird in unseren Simulationen der Eingangswertebereich des nichtlinearen Ziel-Systems beschränkt, damit der Ausgangswertebereich des Ziel-Systems auf [-5.2, +5.2] beschränkt werden kann.

6.2 Äquivalentes Wiener-Modell

Das Wiener-Modell besteht aus zwei Teilen, nämlich dem LZI-Anteil und dem statischen NL-Anteil. Der Kombinationsalgorithmus möchte die zwei Teile vom Ziel-System abspalten. Der Kombinationsalgorithmus kann nur ein äquivalentes Wiener-Modell identifizieren.

Abb. 6.3 veranschaulicht das äquivalente Wiener-Modell, dabei ist k ein Koeffizient. Für beliebige k bleibt das Wiener-Modell unverändert, solange $k \neq 0$ ist. Der Kombinationsalgorithmus kann den Koeffizienten k des Ziel-Systems nicht eindeutig identifizieren.

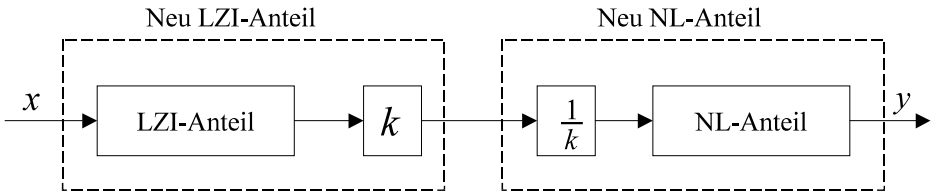


Abbildung 6.3: Die Identifikation des LZI-Anteils und des NL-Anteils im Wiener-Modell ist ohne andere zusätzliche Information, wie z.B. Amplitudengang von LZI-Anteil, nicht eindeutig möglich.

6.3 Erfahrungen für die Anwendbarkeit des Kombinationsalgorithmus

Die Identifikation des Ziel-Systems ist die Basis der nichtlinearen Systemkompensation bei der GP. Nur wenn der NL-Anteil des Ziel-Systems hinreichend genau

identifiziert wird, ist es möglich, dass der NL-Anteil gut kompensiert werden kann.

Der Kombinationsalgorithmus führt zunächst zur Identifikation des Ziel-Systems gemäß einem Wiener-Modell. In diesem Wiener-Modell gibt es zwei Teile, d.h. einen LZI-Anteil und einen NL-Anteil. Beim Kombinationsalgorithmus muss die GP einen LZI-Systemanteil und einen NL-Systemanteil identifizieren, damit der Kombinationsalgorithmus weiter ablaufen kann. Ansonsten funktioniert der Kombinationsalgorithmus nicht .

Durch unsere Untersuchungen haben wir in den Experimenten gefunden: (siehe Kapitel 4 und 5)

- Mit der GP ist es effektiv, ein LZI-System zu identifizieren.
- Mit der GP ist es ziemlich effektiv, ein gedächtnisloses NL-System zu identifizieren sowie zu kompensieren.

Diese beiden Experiment-Ergebnisse erfüllen die Anforderungen dieses Kombinationsalgorithmus.

Obwohl man nicht allgemein beweisen kann, dass die GP ein Ziel-System sicher identifizieren kann, so zeigen die durchgeführten manigfaltigen Simulationen doch, dass die GP in hohem Maße als praxistauglich bezeichnet werden kann.

Weil der Kombinationsalgorithmus durch einen Iterations-Prozess das Ziel-System identifiziert, benötigt er am Anfang einen Initial-*"Wert"*, und zwar einen Initial-LZI-Systemanteil oder einen Initial-NL-Systemanteil. Dazu werden drei Entwürfe betrachtet, wobei jeder Entwurf zu verschiedenen Systemsituationen passt.

6.4 Drei Realisierungsentwürfe des Kombinationsalgorithmus

Im Folgenden werden drei Realisierungsentwürfe und ihre Effekte dargestellt. Jeder Entwurf besitzt verschiedene Voraussetzungen und passt zu verschiedenen Systemeigenschaften und Anwendungsbedingungen.

6.4.1 Entwurf 1: Kompensation von starken NL-Systemen mit schwachem NL-Bereich

Für dieses Experiment machen wir folgende Annahmen:

- Man kann den Eingangswertebereich des Testsignals steuern bzw. eine Auswahl treffen.
- Das Ziel-System enthält einen schwach nichtlinearen Bereich.
- Man weiß etwa, wo der schwach nichtlineare Bereich des Ziel-Systems liegt.

Bei dem folgenden Kombinationsalgorithmus wird die Systemkompensation und die Systemidentifikation in drei Stufen eingeteilt:

Realisierung des Kombinationsalgorithmus

Erste Stufe: Der Initial-LZI-Anteil wird vom ganzen Ziel-System identifiziert. Dies wird im schwach nichtlinearen Bereich mit der LZI-Grammatik durchgeführt. Und zwar durch ein stark begrenztes Testsignal, das im schwach nichtlinearen Bereich liegt. In diesem Bereich bestimmt der LZI-Systemanteil das Verhalten des ganzen nichtlinearen Systems.

Zweite Stufe: Der Initial-NL-Anteil wird vom ganzen Ziel-System identifiziert. Dies wird im stark nichtlinearen Bereich durchgeführt. Und zwar, durch ein grosses Testsignal, das im stark nichtlinearen Bereich liegt. Hier dominiert der NL-Systemanteil das ganze System.

Dritte Stufe: Der identifizierte LZI- und NL-Anteil werden durch einen Iterationsprozess verbessert. Dies wird als ein iterativer Prozess im stark nichtlinearen Bereich durchgeführt.

Der komplette Kombinationsalgorithmus für diesen Entwurf gemäß Abb. 6.4 erfolgt in den folgenden Iterationsschritten:

1. durch die LZI-Grammatik mit x und y wird der Initial-LZI-Anteil identifiziert.

Identifikation im schwach nichtlinearen Bereich

2. mit dem identifizierten LZI-Anteil und x wird z_1 berechnet.

3. mit dem berechneten z_1 und y wird der Initial-NL-Anteil identifiziert.

Identifikation im stark nichtlinearen Bereich

4. mit dem identifizierten NL-Anteil wird NL^{-1} identifiziert.

Kompensation von NL

5. mit dem identifizierten Kompensationssystem NL^{-1} und y wird z_2 berechnet.

6. Es wird $z_1 = z_2$.

7. wenn der Restfehler zwischen dem Systemausgang und dem Modellausgang kleiner als ein vorgegebener Wert ist, stoppt der Iterationsprozess. Wenn der Restfehler zwischen dem Systemausgang und dem Modellausgang nicht kleiner als ein vorgegebener Wert ist, wird der Algorithmus mit Schritt 8 und 9 fortgesetzt.

8. durch die LZI-Grammatik mit x und z_1 wird der LZI-Anteil wieder identifiziert.

Korrektur des LZI-*Anteils*

9. mit z_1 und z_2 wird der Amplitudengang des LZI-Anteils angepasst.

der Amplitudengang des LZI-*Anteils wird verbessert*

10. zurück zu 2.

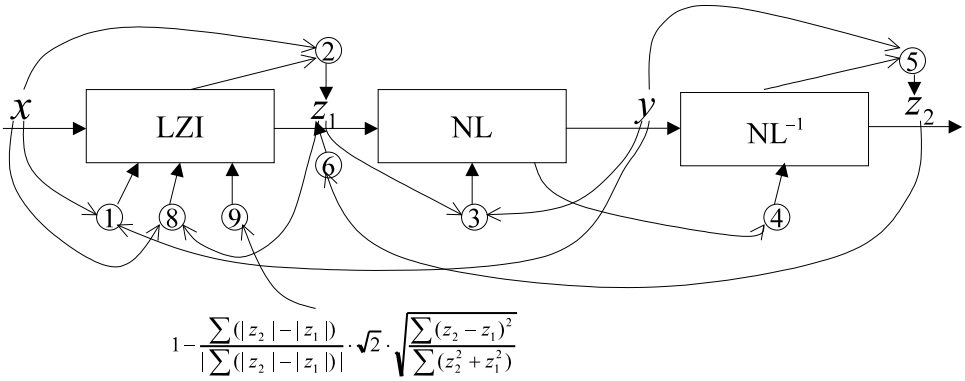


Abbildung 6.4: Beschreibung des Kombinationsalgorithmus

Dabei ist $1-\frac{\sum(|z_2|-|z_1|)}{|\sum(|z_2|-|z_1|)|}\cdot\sqrt{2}\cdot\sqrt{\frac{\sum(z_2-z_1)^2}{\sum(z_2^2+z_1^2)}} = 1\pm\sqrt{2}\cdot\sqrt{\frac{\sum(z_2-z_1)^2}{\sum(z_2^2+z_1^2)}}$ ein Faktor für die LZI-Amplitudengangsanpassung (AA) in der entsprechenden Iteration, um die Konvergenzgeschwindigkeit zu beschleunigen. Dieser Faktor soll auf den Bereich [0.9, 1.1] beschränkt werden. D.h. die LZI-Amplitudengangsanpassung besitzt maximal eine Anpassungsfähigkeit von 10%.

Konvergenzprozess und Genauigkeit des Kombinationsalgorithmus

Die Abbildung 6.5 zeigt den Konvergenzprozess des Kombinationsalgorithmus im NL-Systemanteil. Aus dieser Abbildung kann man ersehen, dass die NL-Kennlinie des Modells nur in der 3. iterativen Stufe auf die NL-Kennlinie des Ziel-Systems hin konvergiert.

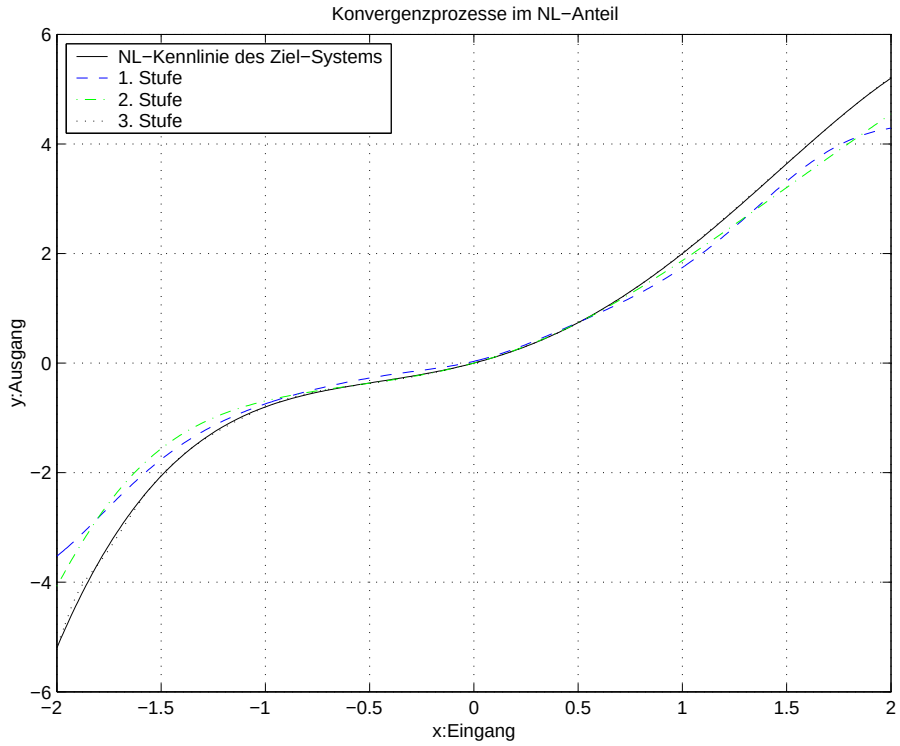


Abbildung 6.5: Konvergenzprozess im NL-Anteil des Modells

Die Abbildung 6.6 zeigt den Kompensationseffekt des Ziel-Systems in der Endstufe. Aus dieser Abbildung kann man sehen, dass der Ausgang des LZI-Anteils des Ziel-Systems und der kompensierte Ausgang fast identisch sind. Der normierte Restfehler ist $5.32 \cdot 10^{-4}$ mit unabhängigen Testdaten.

Für den Kombinationsalgorithmus ist die Kompensationsgenauigkeit (ε_{KL}) der NL-Kennlinie des Modells sehr wichtig. Die Genauigkeit (ε_{KL}) bestimmt die Kompensationsgenauigkeit (ε_{KP}) der Nichtlinearität des Ziel-Systems. Allerdings beschreiben die beiden Genauigkeiten verschiedene Eigenschaften. Nur wenn die Kennlinie des NL-Anteils des Ziel-Systems hinreichend genau identifiziert und kompensiert wird, ist es möglich, die Nichtlinearität des gesamten Ziel-Systems gut zu kompensieren.

Die Abbildungen 6.7, 6.8 und 6.9 zeigen den Kompensationseffekt der NL-Kennlinie des Modells in der Endstufe. Davon zeigen die Abbildungen 6.7 und 6.8 den Kom-

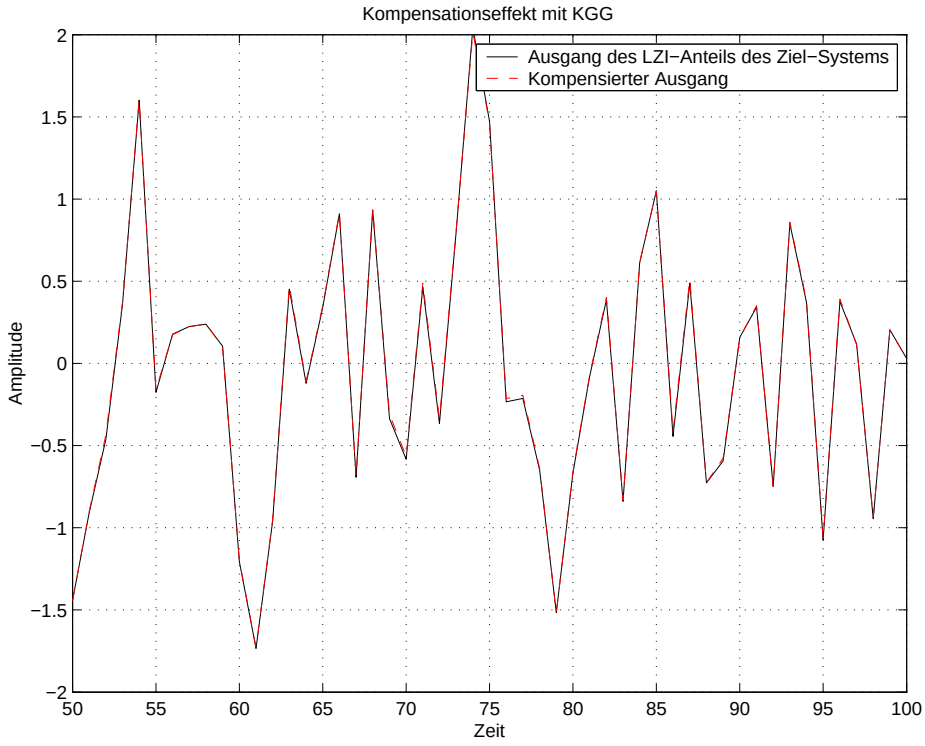


Abbildung 6.6: Kompensationseffekt mit KGG

pensationseffekt der NL-Kennlinie des Modells in den lokalen Eingangswertebereichen [-0.8, +0.5], [-2, -0.8] und [+0.5, +2] nach der Konstanten-Optimierung. Dabei entsprechen die Eingangswertebereiche [-0.8, +0.5], [-2, -0.8] und [+0.5, +2] den Ausgangswertebereichen des Ziel-Systems [-0.5556, +0.7325], [-5.2, -0.5556] und [+0.7325, +5.2].

Aus diesen Abbildungen 6.7 und 6.8 kann man sehen, dass der Kompensationseffekt der NL-Kennlinie des Modells im ganzen Bereich wahrscheinlich sehr schlecht ist. Aber in einem bestimmten Bereichsanteil ist er sehr gut. Durch die Auswahl des besonderen Bereichs der verschiedenen Kompensationsfunktionen setzt man eine Kompensationsgleichungsgruppe (KGG) zusammen. Die Abbildung 6.9 zeigt den Kompensationseffekt von der KGG. Durch die KGG sowie die abschnittsweise Optimierung (AO), die im folgenden Abschnitt ausführlich erläutert werden, kann man immer ziemlich hohe Kompensationsgenauigkeiten für die NL-Kennlinie des Modells erreichen. In diesem Beispiel besteht die KGG aus den Gleichungen (6.3) und (6.4).

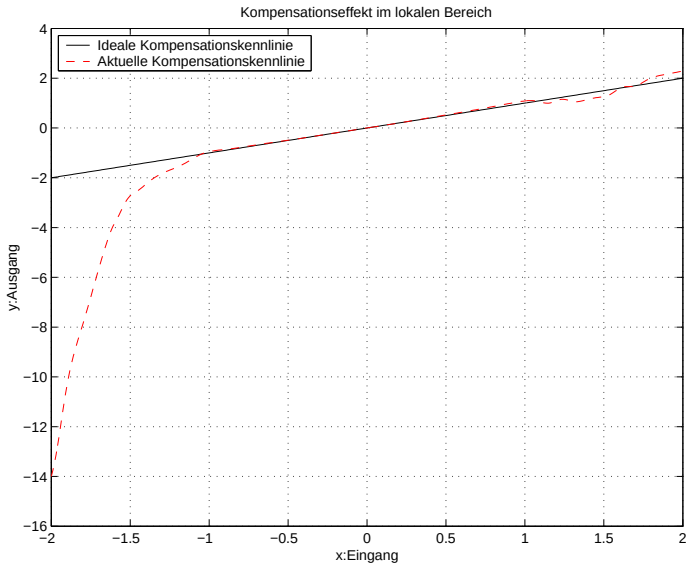


Abbildung 6.7: Kompensationseffekt der NL-Kennlinie des Modells mit Gl. (6.3) im Eingangswertebereich [-0.8, +0.5].

Kompensationsfunktionen und Rechenaufwand

Die in der Endstufe identifizierte KGG der NL-Kennlinie des Modells ist in den Gleichungen (6.3) und (6.4) gezeigt.

$$\widehat{z} = a_1 sin \frac{a_2 y}{a_3 cos(a_4 y^2) + a_5 cos[a_6 sin(\frac{a_7}{y} + a_8)] + a_9 cos(a_{10} y + a_{11}) + a_{12}}, y \in [-0.6, 0.7] \tag{6.3}$$

$$\widehat{z} = a_1 sin\{a_2 e^{a_3 sin(a_4 y)} cos[a_5 cos(a_6 e^{a_7 cos(a_8 y + a_9)})]\} + \frac{a_{10} y}{a_{11} sin(a_{12} y) + a_{13}} + a_{14}, y : \text{sonst} \tag{6.4}$$

An diesen Gleichungen kann man klar sehen, dass der Rechenaufwand sehr groß ist. Die Tabelle 6.3 zeigt den genauen Rechenaufwand. Dabei werden die Koeffizienten für die Gleichungen (6.3) und (6.4) in der Tabelle 6.1 gezeigt.

Auswertungen des Entwurfs 1

Vorteile:

- Ziemlich sichere und schnelle Konvergenz.
- Hohe Genauigkeit.

Nachteile:

- Es werden zwei Testsignale benötigt.
- Es ist Vorwissen erforderlich zur Bestimmung des schwachen NL-Bereichs.

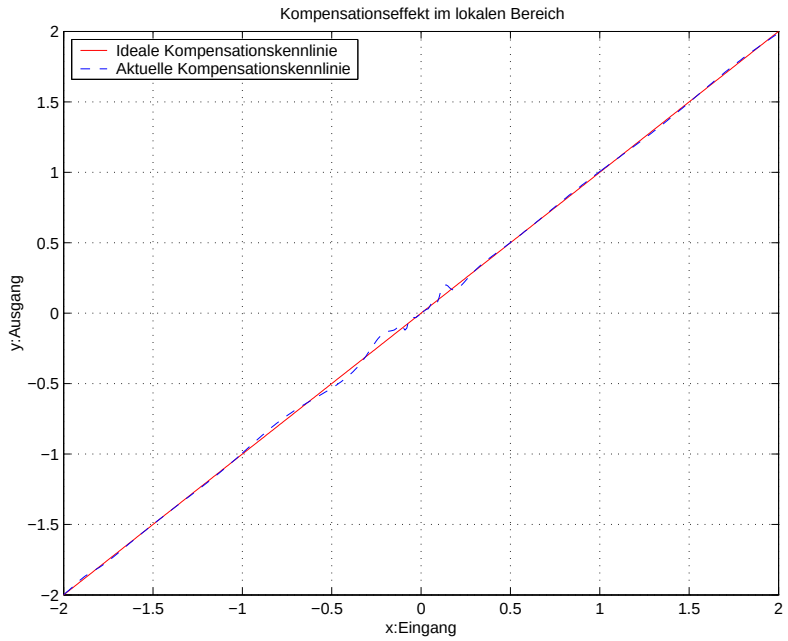


Abbildung 6.8: Kompensationseffekt der NL-Kennlinie des Modells mit Gl. (6.4) in Eingangswertebereichen [-2, -0.8] und [+0.5, +2]

Par	Gl. (6.3)		Gl. (6.4)		KGG.
	opt. im GB	opt. im TB_1	opt. im GB	opt. TB_2	
a_1		2.7133		1.1295	
a_2		0.6235		1.2547	
a_3		0.1850		0.4238	
a_4		0.1762		1.4132	
a_5		0.5493		0.6584	
a_6		1.9621		2.0608	
a_7		0.6535		1.2004	
a_8		5.3658		1.2882	
a_9		1.0844		1.9088	
a_{10}		0.6484		0.8736	
a_{11}		-9.1144		0.8598	
a_{12}		2.5943		0.3123	
a_{13}				1.1857	
a_{14}				-1.0687	
ε_{KL}		$7.50 \cdot 10^{-4}$		$3.32 \cdot 10^{-5}$	$6.34 \cdot 10^{-5}$
ε_{KP}					$5.32 \cdot 10^{-4}$
Bemerkung: GB=[-6,6]; TB_1=[-0.5556,0.7325]; TB_2=[-6,-0.5556]$\bigcup$[0.7325,6]					

Tabelle 6.1: Koeffizienten und relativer Fehler für den Entwurf 1

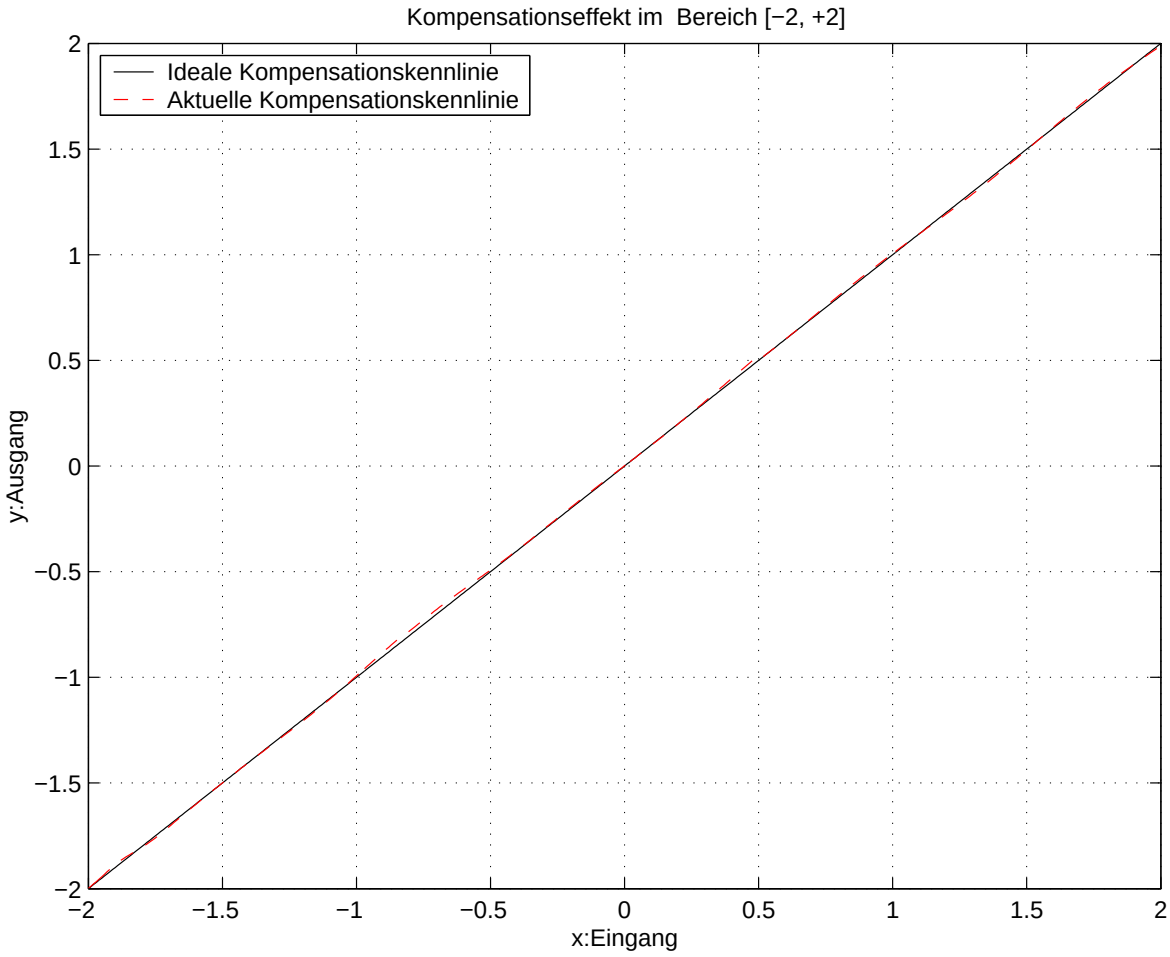


Abbildung 6.9: Kompensationseffekt der NL-Kennlinie des Modells mit KGG im Eingangswertebereich [-2, +2]

6.4.2 Entwurf 2: Kompensation der starken NL-Systeme ohne Kenntnis des schwachen NL-Bereichs

In diesem Experiment wird angenommen, dass die Nichtlinearität des Ziel-Systems nicht sehr stark ist.

Der Kombinationsalgorithmus wird hier in drei Stufen eingeteilt:

Realisierung des Kombinationsalgorithmus

Erste Stufe: Der Initial-LZI-Anteil wird vom ganzen Ziel-System identifiziert. Dies wird im gesamten Eingangswertebereich durch die LZI-Grammatik durchgeführt.

Zweite Stufe: Der Initial-NL-Anteil wird vom ganzen Ziel-System identifiziert. Dies wird im gesamten Eingangswertebereich durchgeführt.

Dritte Stufe: Die identifizierten LZI- und NL-Anteile werden durch den Iterationsprozess verbessert. Dies wird als ein iterativer Prozess im gesamten Eingangswertebereich durchgeführt.

Der komplette Kombinationsalgorithmus für dieses Experiment enthält folgende Schritte:

1. Durch die LZI-Grammatik mit x und y wird der Initial-LZI-Anteil identifiziert.
2. mit dem identifizierten LZI-Anteil und x wird z_1 berechnet.
3. Mit dem berechneten z_1 und y wird der NL-Anteil identifiziert.
4. Mit dem identifizierten NL-Anteil wird NL^{-1} identifiziert.
5. Mit dem identifizierten NL^{-1} und y wird z_2 berechnet.
6. Es wird $z_1 = z_2$.
7. Wenn der Restfehler zwischen dem Systemausgang und dem Modellausgang kleiner als ein vorgegebener Wert ist, stoppt der Iterationsprozess. Wenn der Restfehler nicht kleiner als ein vorgegebener Wert ist, wird der Algorithmus mit Schritt 8 und 9 fortgesetzt.
8. durch die LZI-Grammatik mit x und z_1 wird der LZI-Anteil erneut identifiziert.
9. mit z_1 und z_2 wird der Amplitudengang des LZI-Anteils angepasst.
10. zurück zu Schritt 2.

Konvergenzprozess und Genauigkeit des Kombinationsalgorithmus nach dem 2. Entwurf

Die Abbildungen 6.10, 6.11, 6.12, 6.20, 6.21, 6.22, 6.23 und 6.24 zeigen den Konvergenzprozess dieses Kombinationsalgorithmus.

Die Abbildungen 6.13, 6.14, 6.15, 6.16 und 6.17 zeigen den Kompensationseffekt der NL-Kennlinie des Modells in der Endstufe.

Die Abbildung 6.18 zeigt den Kompensationseffekt der Nichtlinearität des Ziel-Systems.

Die Abbildung 6.19 zeigt den Identifikationseffekt des Ziel-Systems im Zeitbereich.

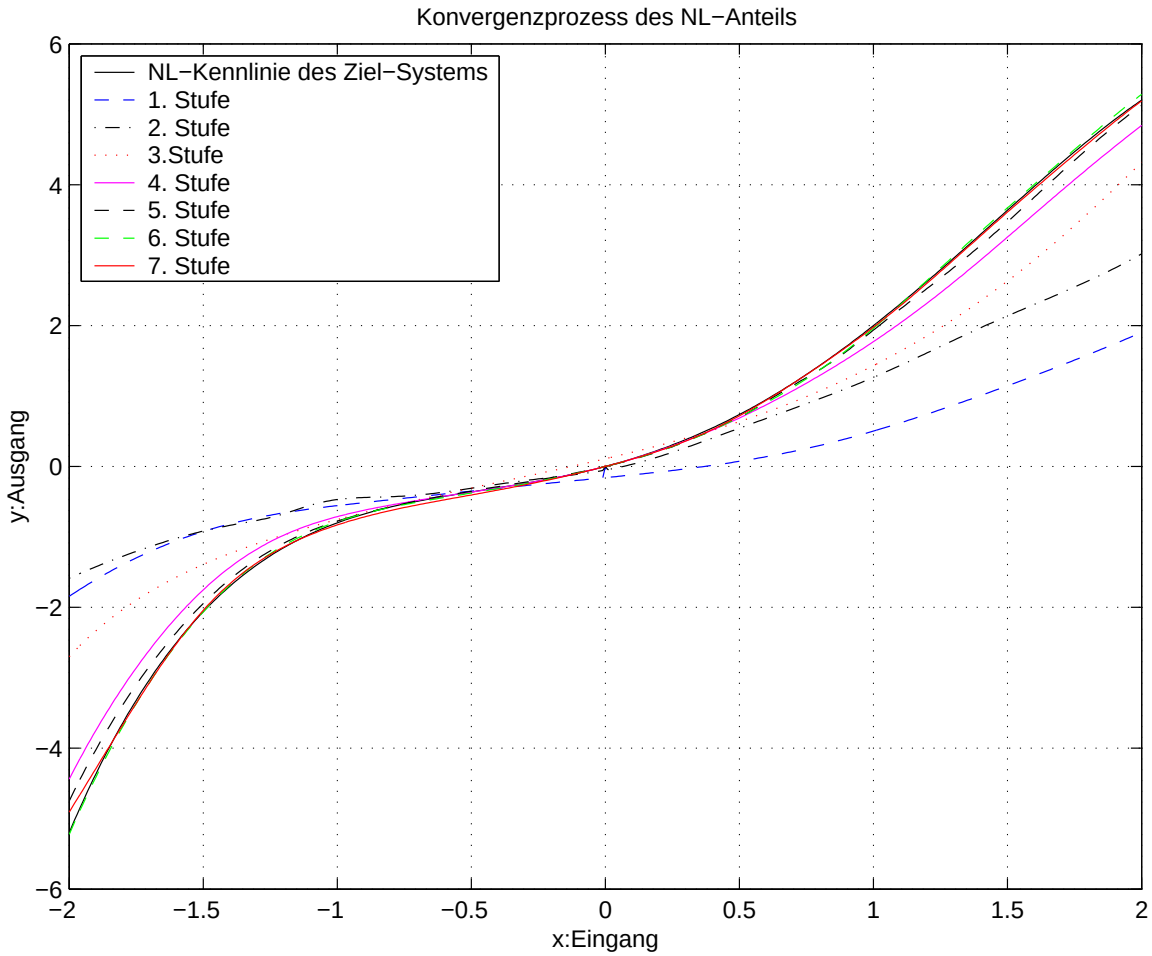


Abbildung 6.10: Konvergenzprozess des NL-Systemanteils nach dem 2. Entwurf

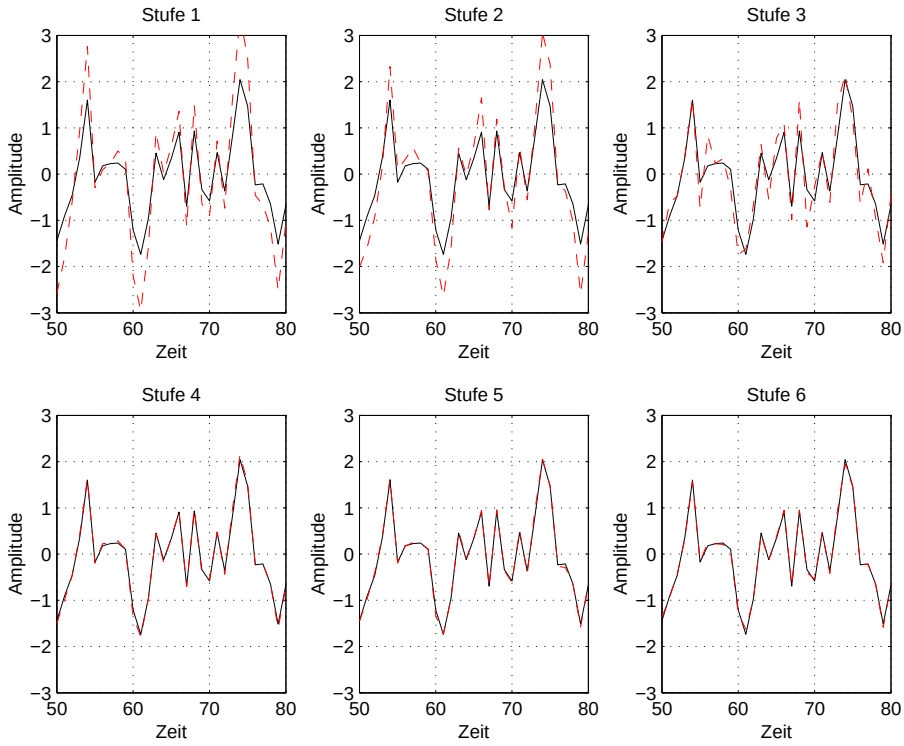


Abbildung 6.11: Konvergenzprozess des LZI-Systemanteils nach dem 2. Entwurf

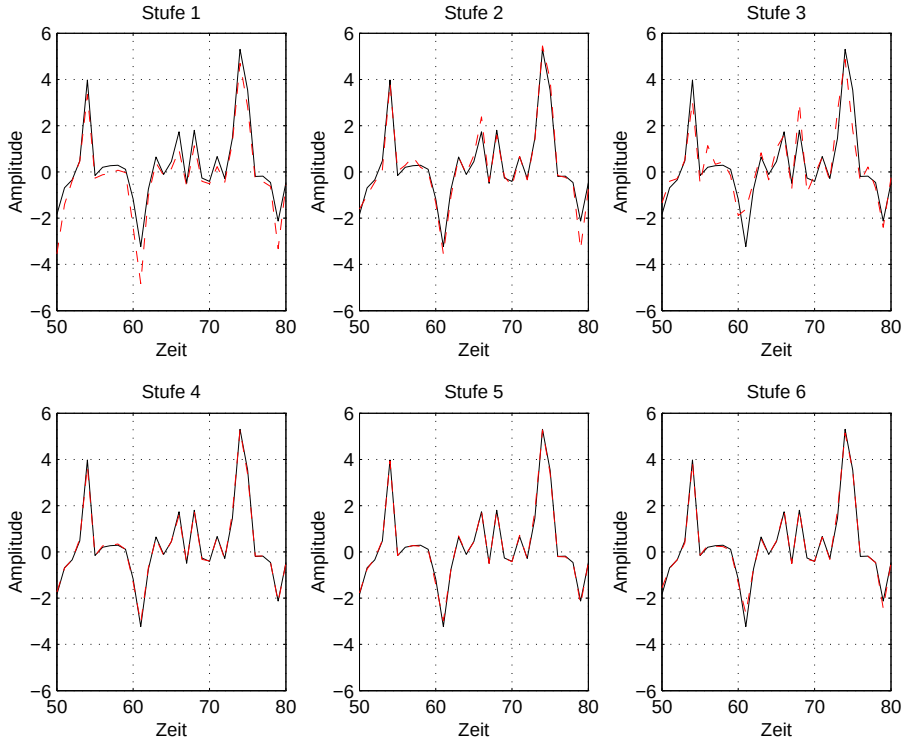


Abbildung 6.12: Konvergenzprozess des ganzen Systems nach dem 2. Entwurf

Konvergenzkriterium

Die Abbildungen 6.10, 6.20,6.21, 6.22, 6.23 und 6.24 zeigen die Konvergenzprozesse des Kombinationsalgorithmus.

In der praktischen Anwendung sind die relativen Restfehler $\varepsilon_{L,i}$ und $\varepsilon_{N,i}$ nicht berechenbar, allerdings haben diese beiden relativen Restfehler eine große theoretische Bedeutung.

Die relativen Restfehler $\Delta\varepsilon_N(i)$ und $\Delta\varepsilon_L(i)$ bilden eine gute Überwachung für die GP-Durchläufe. Das Konvergenzkriterium ist, dass ε_i gegen Null oder einen Konstanten konvergiert.

Kompensationsgleichungsgruppe (KGG) und ihr Rechenaufwand

Der in der Endstufe identifizierte LZI- und NL-Anteil des Modells sind in den Gleichungen (6.5) und (6.6) dargestellt.

$$\widehat{z}_L(i) = \frac{1}{18}x(i)+\frac{4}{9}x(i-2)+\frac{5}{8}x(i-3)+\frac{1}{12}x(i-4)+\frac{1}{9}x(i-5)-\frac{2}{27}\widehat{z}_L(i-4)-\frac{1}{9}\widehat{z}_L(i-5) \tag{6.5}$$

$$\widehat{y} = \frac{\widehat{z}_L \cdot e^{-[sin(cos(1.5299\widehat{z}_L-18.3586))]}}{cos(cos(\widehat{z}_L)) \cdot cos(sin(e^{[sin(cos(cos(sin(e^{[cos(\widehat{z}_L)]})+\widehat{z}_L-9))+0.9894)]}))} \tag{6.6}$$

Die KGG vom NL-Anteil des Modells ist in den Gleichungen (6.7) und (6.8) gezeigt. Aus diesen Gleichungen (6.7) und (6.8) kann man klar sehen, dass der Rechenaufwand für die Kompensation sehr groß ist. Die Tabelle 6.3 zeigt den genauen Rechenaufwand. Die Tabelle 6.2 zeigt die Parameter in den Gleichunngen (6.7) und (6.8) sowie die Kompensationsgenauigkeit.

$$\widehat{z} = a_1 sin[a_2 sin(a_3 y) + \frac{a_4}{e^{a_5 e^{a_6 y}}}] + a_7 sin[a_8 sin(a_9 y)] + a_{10} y + a_{11}; y \in [-0.3776, 0.7] \tag{6.7}$$

$$\widehat{z} = a_1 y e^{a_2 cos[a_3 cos[a_4 sin[a_5 sin(a_6 y e^{a_7 cos(\frac{cos(a_8 cos[a_9 sin(a_{10} y)+a_{11} cos(a_{12} y)])}{a_{13} y})}) + a_{14} sin(a_{15} sin(a_{16} y))] + a_{17}] + a_{18} cos(a_{19} y)]} \tag{6.8}$$

Auswertung

Vorteile:

- Sichere Konvergenz.
- Hohe Genauigkeit.

Nachteile:

- Konvergenzgeschwindigkeit ist relativ langsam.

Koeffizienten und relativer Fehler für den Entwurf 2					
Par	Gl. 6.7		Gl. 6.8		KGG.
	opt. im GB	opt. im TB_1	opt. im GB	opt. im TB_2	
a_1	0.4099	0.3688	1.0079	1.3721	
a_2	2.4594	0.9943	0.9883	-1.3019	
a_3	0.4101	-0.4273	0.9638	0.7741	
a_4	8.9909	7.2154	1.3327	1.1321	
a_5	1.3038	1.7952	1.0821	1.5888	
a_6	0.9261	0.7773	1.0969	0.3496	
a_7	0.1286	0.2902	1.1278	-0.7800	
a_8	1.6825	1.4800	1.6705	1.6543	
a_9	0.7208	1.1936	2.1051	1.4493	
a_{10}	0.3863	0.8132	0.1240	0.8040	
a_{11}	-0.2735	-0.3425	0.6993	1.4187	
a_{12}			-0.0084	0.8427	
a_{13}			0.4423	0.4600	
a_{14}			1.1146	3.5579	
a_{15}			0.3244	0.2356	
a_{16}			1.0145	0.9240	
a_{17}			-6.0972	-6.0039	
a_{18}			1.0081	0.8798	
a_{19}			0.4021	0.4325	
ε_{KL}	$2.75 \cdot 10^{-4}$	$6.54 \cdot 10^{-6}$	$2.29 \cdot 10^{-4}$	$3.70 \cdot 10^{-5}$	$3.65 \cdot 10^{-5}$
ε_{KP}					$3.86 \cdot 10^{-4}$
Bemerkung: GB=[-6, +6]; TB_1=[-0.3776, +0.7]; TB_2=[-6,-0.3776]$\bigcup$[+0.7, +6]					

Tabelle 6.2: Koeffizienten und relativer Fehler für den Entwurf 2

Rechenaufwand					
	Gl. (6.3)	Gl. (6.4)	Gl. (6.7)	Gl. (6.8)	Gl. (6.22)
P	5	4	4	4	4
M	11	13	10	21	11
sin/cos	5	6	4	12	
exp	0	2	2	2	

Tabelle 6.3: Rechenaufwand

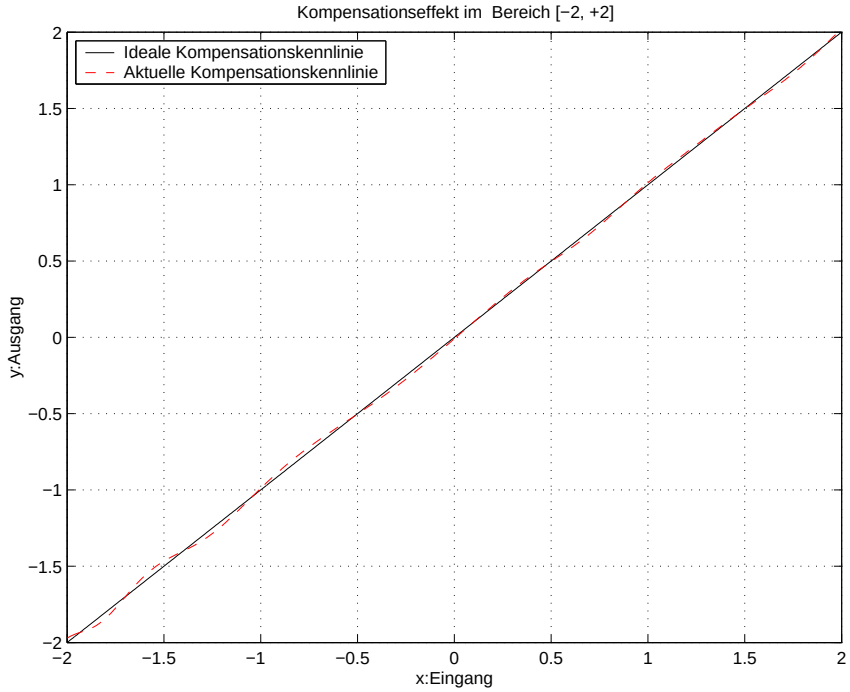


Abbildung 6.13: Kompensationseffekt der NL-Kennlinie des Modells nach Gl. (6.7) im Bereich[-2, +2]

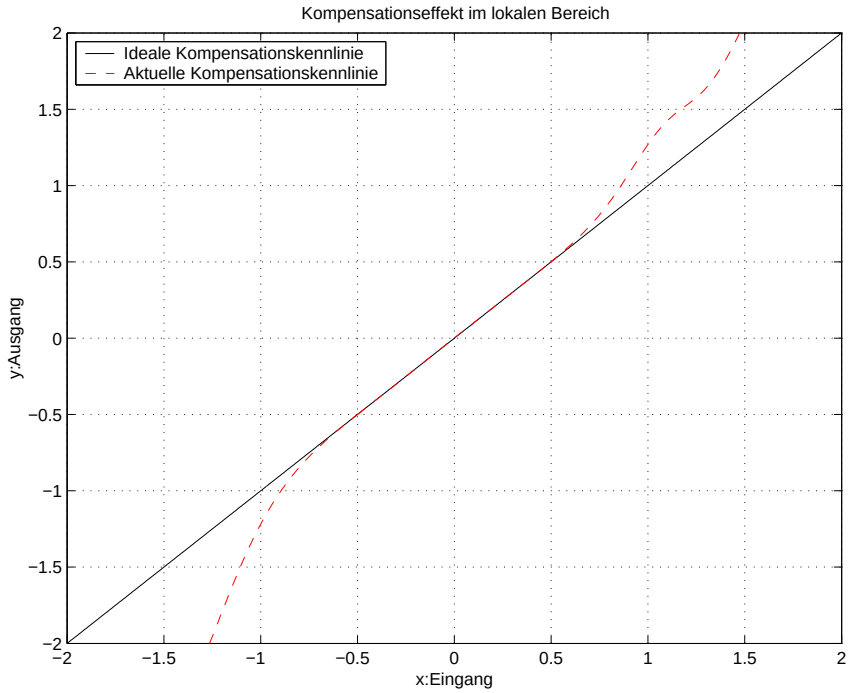


Abbildung 6.14: Kompensationseffekt der NL-Kennlinie des Modells nach Gl.(6.7) im Bereich [-0.3776, +0.7]

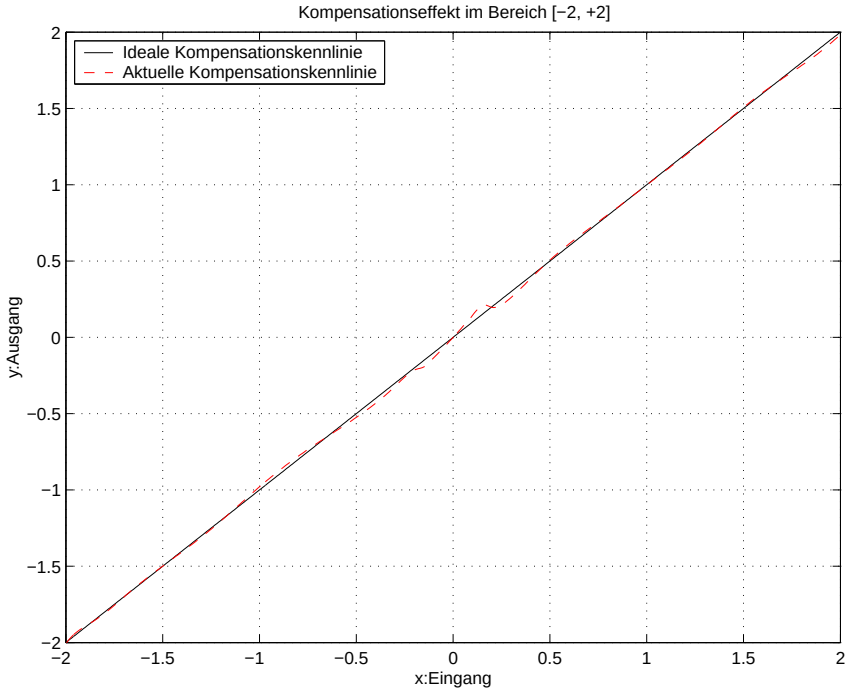


Abbildung 6.15: Kompensationseffekt der NL-Kennlinie des Modells nach Gl.(6.8) im Bereich [-2, +2]

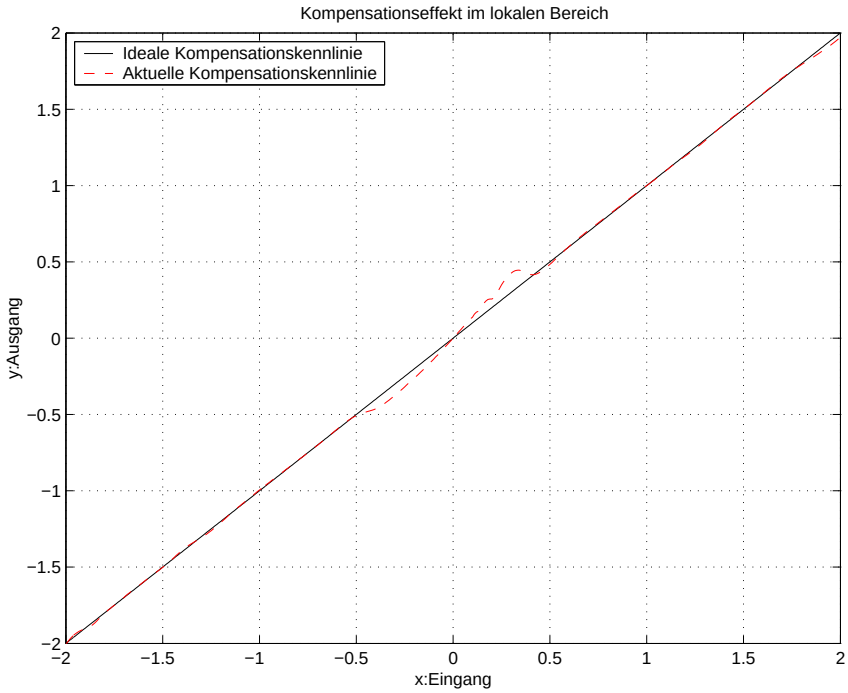


Abbildung 6.16: Kompensationseffekt der NL-Kennlinie des Modells nach Gl.(6.8) im Bereich [-2, -0.3776]$\bigcup$[+0.7, +2]

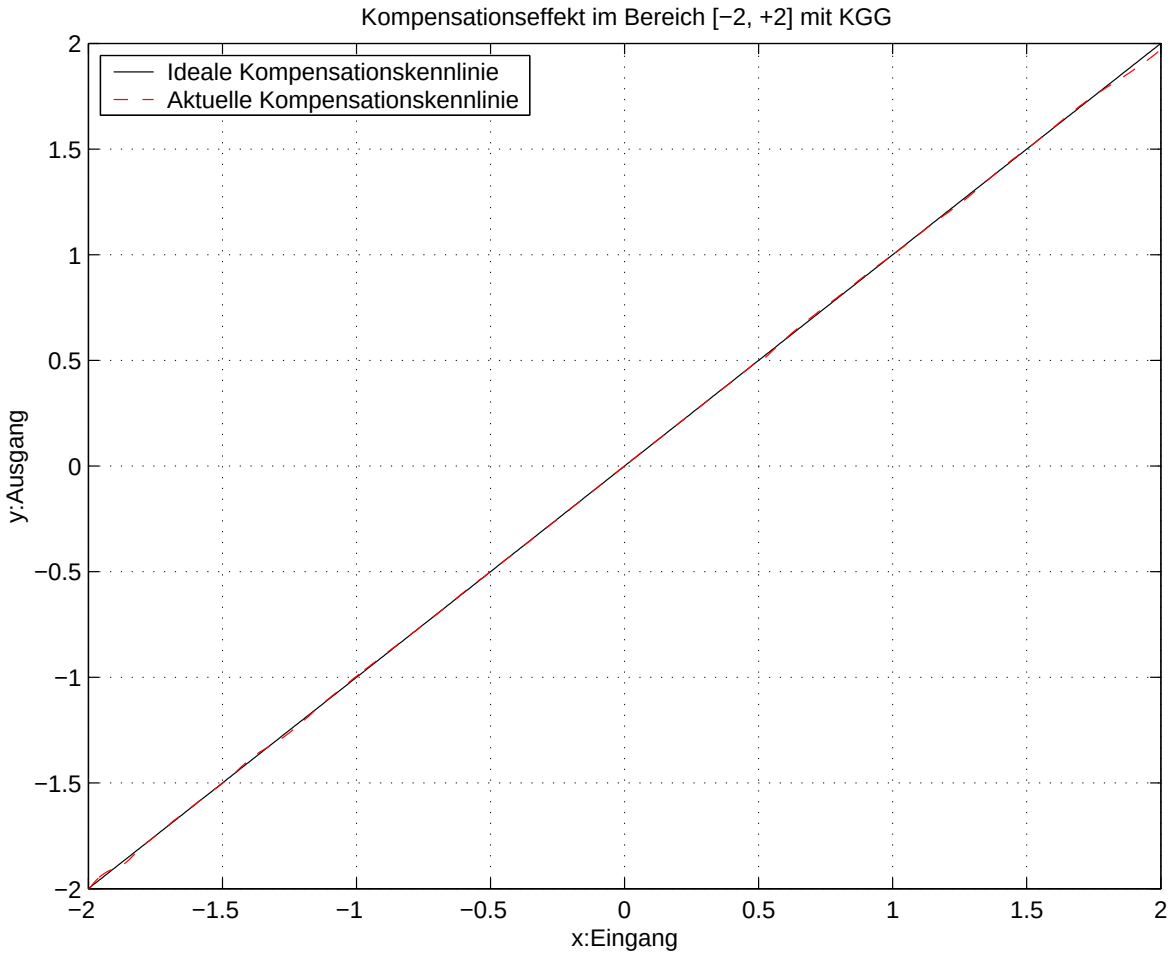


Abbildung 6.17: Kompensationseffekt der NL-Kennlinie des Modells nach KGG im Bereich [-2, +2]

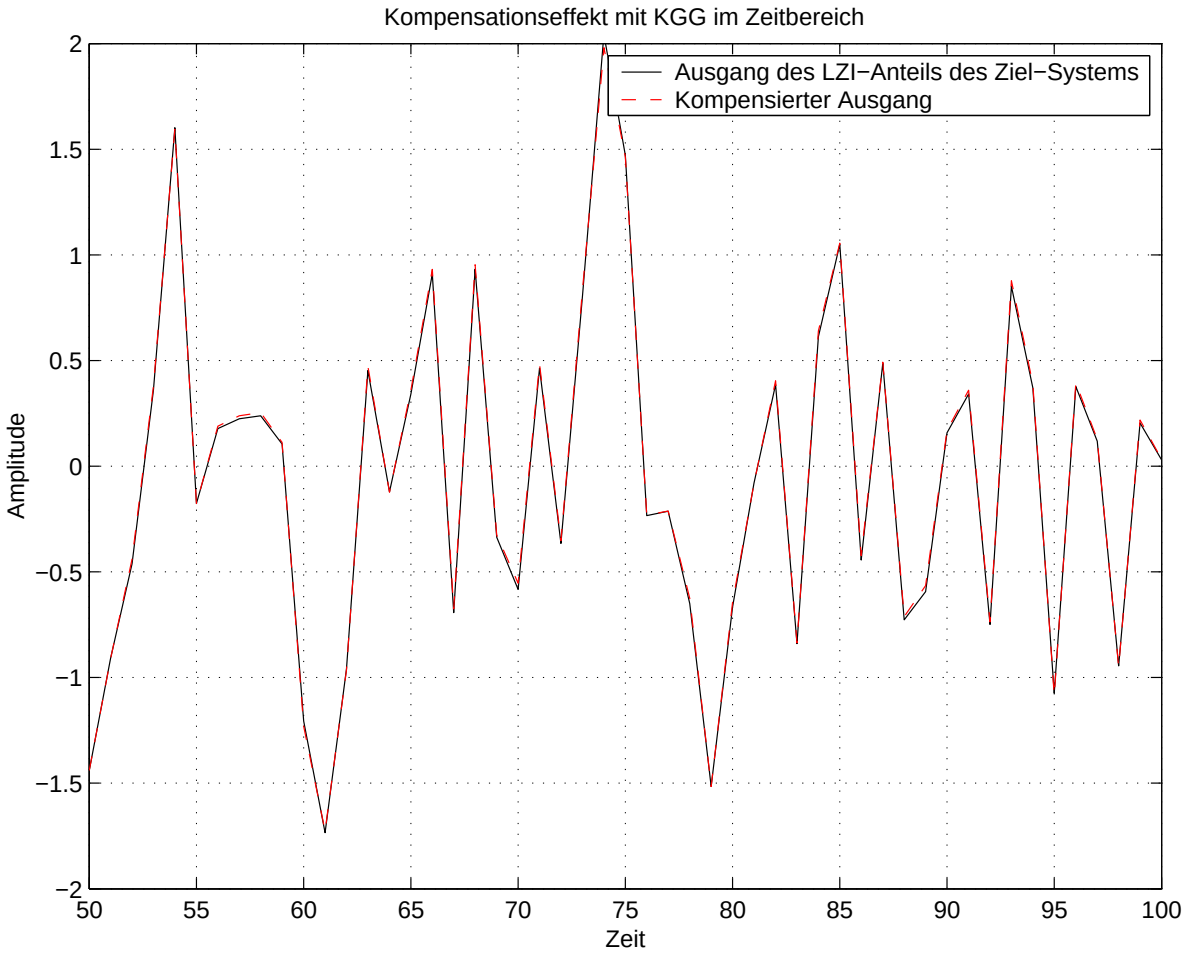


Abbildung 6.18: Kompensationseffekt mit der KGG im Zeitbereich

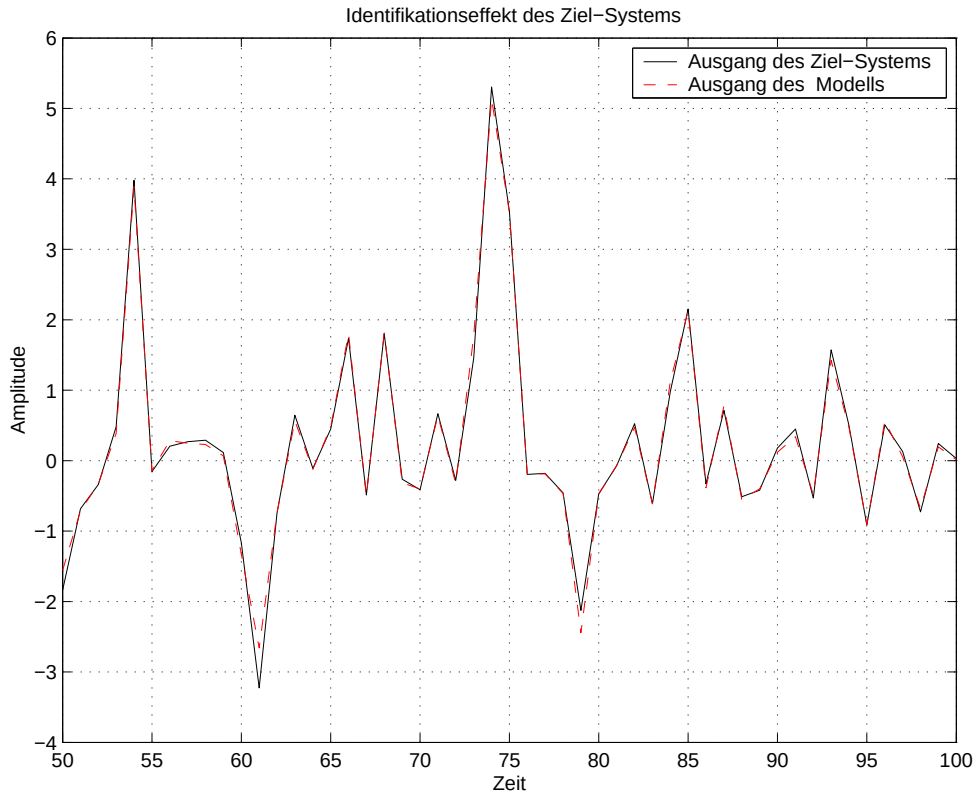


Abbildung 6.19: Identifikationseffekt des ganzen Systems im Zeitbereich

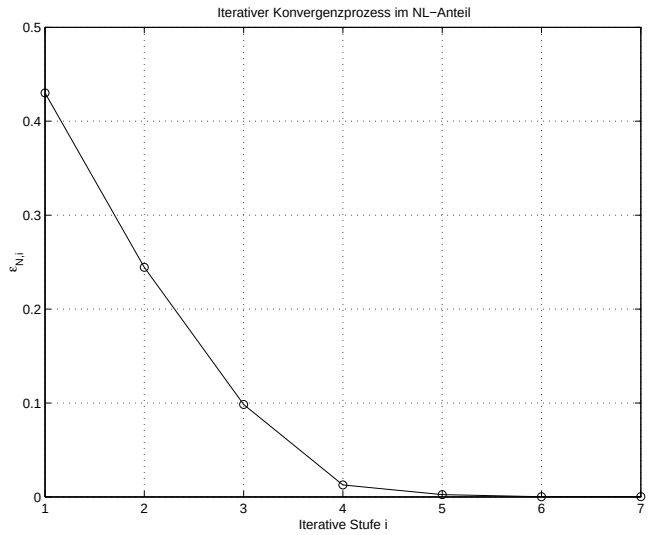


Abbildung 6.20: Konvergenzprozess im NL-Anteil

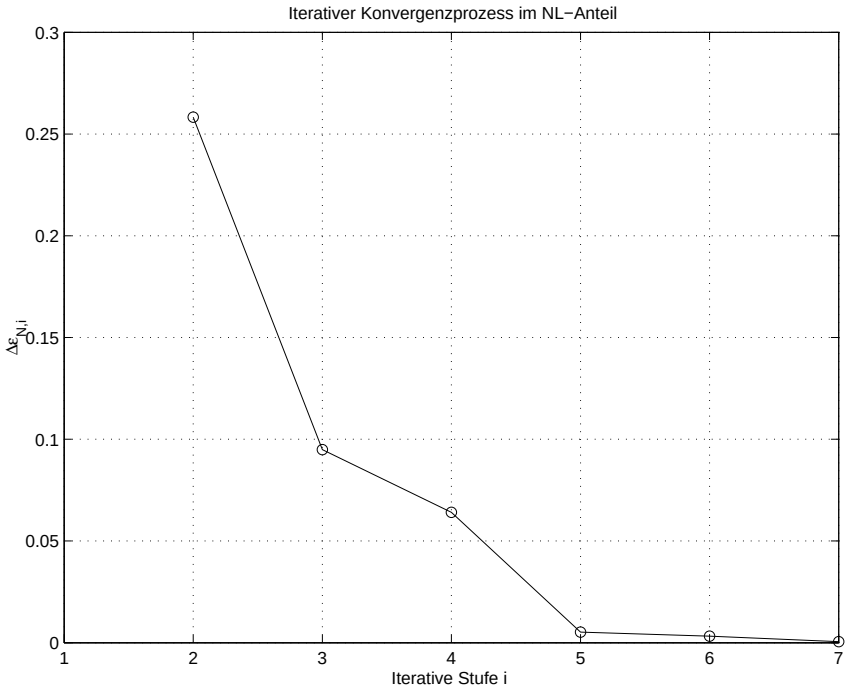


Abbildung 6.21: Konvergenzprozess im NL-Anteil

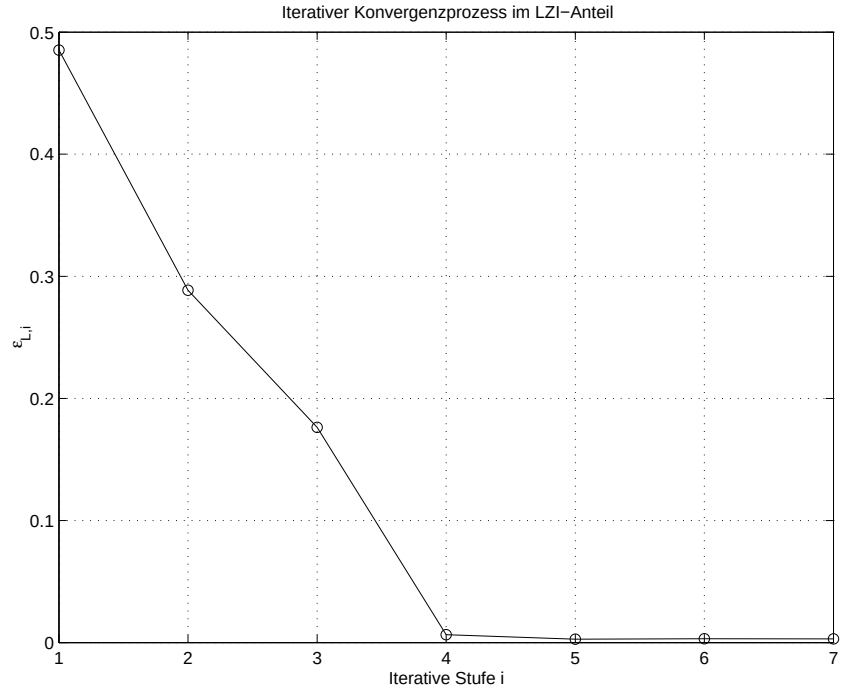


Abbildung 6.22: Konvergenzprozess im LZI-Anteil

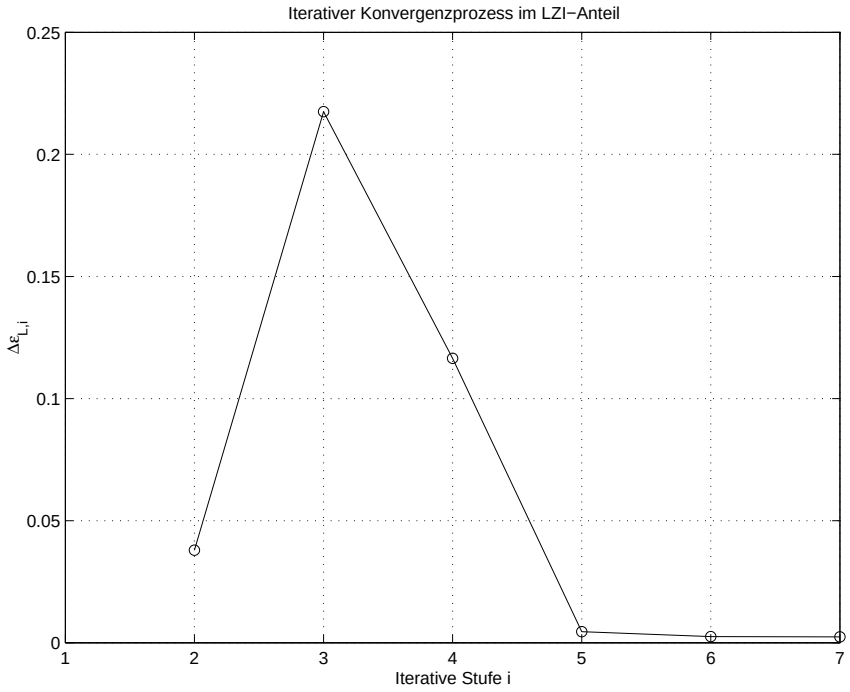


Abbildung 6.23: Konvergenzprozess im LZI-Anteil

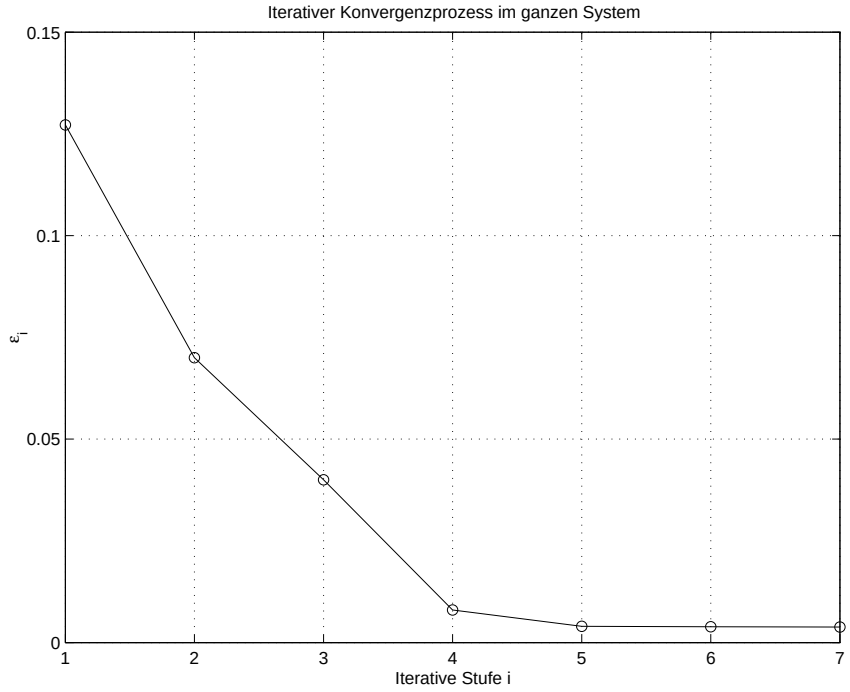


Abbildung 6.24: Konvergenzprozess im ganzen System

6.4.3 Entwurf 3: Kompensation von starken NL-Systemen ohne schwachen NL-Bereich

Die zwei vorgegebenen Beispiele sind erfolgreich unter der Voraussetzung, dass das identifizierte Modell einen echten LZI-Anteil enthält. Dies bedeutet, dass das Ziel-System mit einer Volterra-Reihe beschrieben werden kann, wobei der LZI-Anteil 1. Ordnung von Null verschieden ist. D.h. im statischen NL-Systemanteil soll ein linearer Bestandteil enthalten sein. Allerdings ist es auch dann möglich, dass das Wiener-Modell, wie es in Abbildung 6.2 (auf Seite 108) gezeigt wird, ein kompensierbares System ist, wenn es im statischen NL-Anteil keinen linearen Bestandteil gibt. Obwohl diese Situation nicht sehr sinnvoll für die praktische Anwendung der Kompensation ist, ist sie sinnvoll für theoretische Systeme. In dieser Situation ist das Ziel-System ein rein nichtlineares System.

Um die Situation umzustellen, wird der 3. Entwurf des Kombinationsalgorithmus im Folgenden dargestellt.

Für dieses Experiment wird angenommen, dass man etwa weiß, wie die Änderungstendenz der NL-Kennlinie ist. z.B. ähnlich wie x^3 .

Jetzt werden die Kompensation und die Identifikation der rein nichtlinearen Systeme in drei Stufen aufgeteilt:

Realisierung des Algorithmus

Erste Stufe: Bestimmung eines angenommenen Initial-NL-Anteils für den Iterationsprozess. z.B. x^3.

Zweite Stufe: Der Initial-LZI-Anteil wird vom ganzen Ziel-System durch die LZI-Grammatik identifiziert.

Dritte Stufe: Der identifizierte LZI-Anteil und der angenommene Initial-NL-Anteil werden durch einen Iterationsprozess verbessert.

Der komplette Kombinationsalgorithmus für diesen Entwurf lautet:

1. Bestimmung eines angenommenen Initial-NL-Anteils für den Iterationsprozess. z.B. x^3.
2. mit dem Initial-NL-Anteil wird NL^{-1} identifiziert.
3. Mit dem identifizierten NL^{-1}-Anteil und y wird z_2 berechnet.
4. Es wird $z_1 = z_2$ gesetzt.
5. Mit x und z_1 sowie der LZI-Grammatik wird der Initial-LZI-Anteil identifiziert.

6. Mit x und dem identifizierten LZI-Anteil wird z_1 berechnet.

7. Mit z_1 und y wird der NL-Anteil erneut genauer identifiziert.

8. Wenn der Restfehler zwischen dem Systemausgang und dem Modellausgang kleiner als ein vorgegebener Wert ist, stoppt der Iterationsprozess. Wenn der Restfehler nicht kleiner als ein vorgegebener Wert ist, kehrt der Algorithmus zurück zu 2.

Konvergenzprozess und Genauigkeit des Kombinationsalgorithmus

In diesem Experiment verwenden wir den LZI-Systemanteil, wie er in Gleichung (6.9) gezeigt wird, und den NL-Systemanteil, wie er in Gleichung (6.10) gezeigt wird.

$$z_L(i) = 0.18 \cdot [0.3x(i) - 0.13x(i-1) + 2.5x(i-2) + 0.9x(i-3) + 0.4x(i-4) + 0.2x(i-5) + 0.1x(i-6) + 1.5z_L(i-1) - 0.7z_L(i-2) + 0.9z_L(i-3) - 1.3z_L(i-4)] \tag{6.9}$$

$$y = 0.2z_L^2 + 2z_L^3 + 0.1z_L^4 \tag{6.10}$$

Dabei können wir sehen, dass es in der Gleichung (6.10) gar keinen linearen Bestandteil gibt. Wenn das ganze Ziel-System mit einer Volterra-Reihenentwicklung beschrieben wird, fehlt die Volterra-Reihenentwicklung des linearen Teils. Das ist ein extra stark nichtlineares System aus der Sicht der Volterra-Reihenentwicklung. Und zwar ist es ein rein nichtlineares System.

Allerdings ist das System kompensierbar aus der Sicht des Wiener-Modells , solange der Eingangswertebereich des NL-Systemanteils auf [-4, +4] beschränkt bleibt, weil in diesem Eingangswertebereich die NL-Kennlinie monoton ist.

Die Abb. 6.25 zeigt den Konvergenzprozess des Kombinationsalgorithmus im NL-Anteil. Dabei können wir sehen, dass die identifizierte NL-Kennlinie des Modells nicht ganz auf die NL-Kennlinie des Ziel-Systems hin konvergiert. Darüber haben wir im Abschnitt 6.2 mit der Abbildung 6.3 diskutiert. Die in der Endstufe identifizierte NL-Kennliniefunktion ist in der Gleichung (6.11) gezeigt. Der normierte Restfehler zwischen dem Ausgang des Ziel-Systems und dem Ausgang des Modells ist $\varepsilon = 4.9 \cdot 10^{-4}$. Im Zeitbereich zeigt die Abb. 6.26 den gesamten Identifikationseffekt. Außerdem zeigt die Abb. 6.27 die Anpassung der identifizierten NL-Kennlinie mit einem Faktor $k = 0.8506$.

$$\widehat{y} = \widehat{z}_L^2 \cdot [a_1\widehat{z}_L + a_2 cos(\frac{(a_3 sin(a_4\widehat{z}_L) + a_5\widehat{z}_L) \cdot cos(sin(cos(sin(a_6\widehat{z}_L cos(a_7\widehat{z}_L) + a_8\widehat{z}_L^2))))}{\widehat{z}_L})] \tag{6.11}$$

Um die Gleichung (6.11) zu kompensieren, identifiziert die GP die KGG, wie sie in Gleichungen (6.12) und (6.13) gezeigt werden.

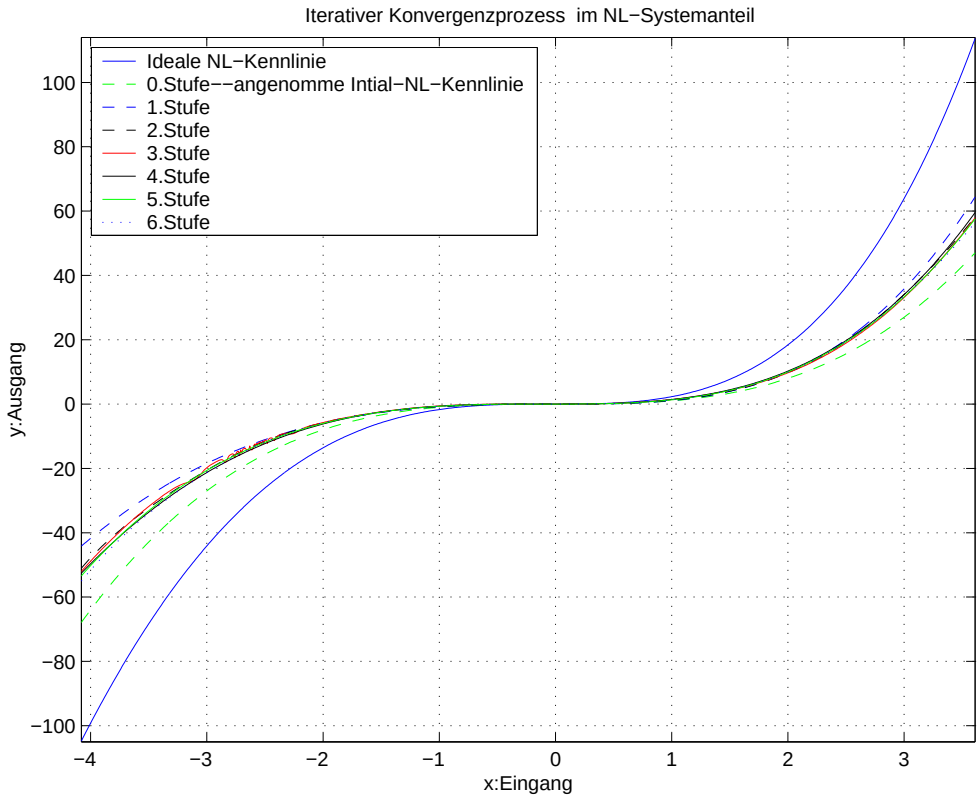


Abbildung 6.25: Konvergenzprozess des NL-Systemanteils

$$\widehat{z} = \frac{a_1 y + a_2 e^{a_3 y} + a_7 sin(a_6 e^{(a_4 x e^{a_5 y})})}{a_8 + a_9 y} + a_{14} cos(a_{13} e^{(a_{12} y + a_{11} sin a_{10} y)}) \qquad (6.12)$$

$$\widehat{z} = a_{13} cos(a_{12} exp(a_{11} exp(a_3 exp(a_2 exp(\frac{a_1}{y})) + a_8 exp(a_7 exp(a_5 exp(\frac{1}{a_4 y}))) + a_8 y + a_{10}))) \qquad (6.13)$$

Durch abschnittsweise Konstanten-Optimierung (AO) wird die KGG optimiert. Dabei optimieren wir die Gleichung (6.12) in den Ausgangswertebereichen [-55, -0.06], [-0.06, $-1.6 \cdot 10^{-3}$] und [$-1.6 \cdot 10^{-3}$, 0] des Ziel-Systems. Wir optimieren die Gleichung (6.13) in den Ausgangswertebereichen [0, +0.1905], [+0.1905, +1.337] und [+1.337, +56.5] des Ziel-Systems. Abb. 6.28 zeigt die Kompensationseffekte der NL-Kennlinie des Modells im Eingangswertebereich [-4.08, +3.61] mit den Gleichungen (6.12) und (6.13). Der normiete Restfehler ist $\varepsilon_{KL} = 3.2 \cdot 10^{-6}$.

Obwohl die Gleichungen (6.12) und (6.13) sehr gute Kompensationseffekte für die NL-Kennlinie des Modells erreicht haben, ist der Rechenaufwand für beide Gleichungen sehr groß.

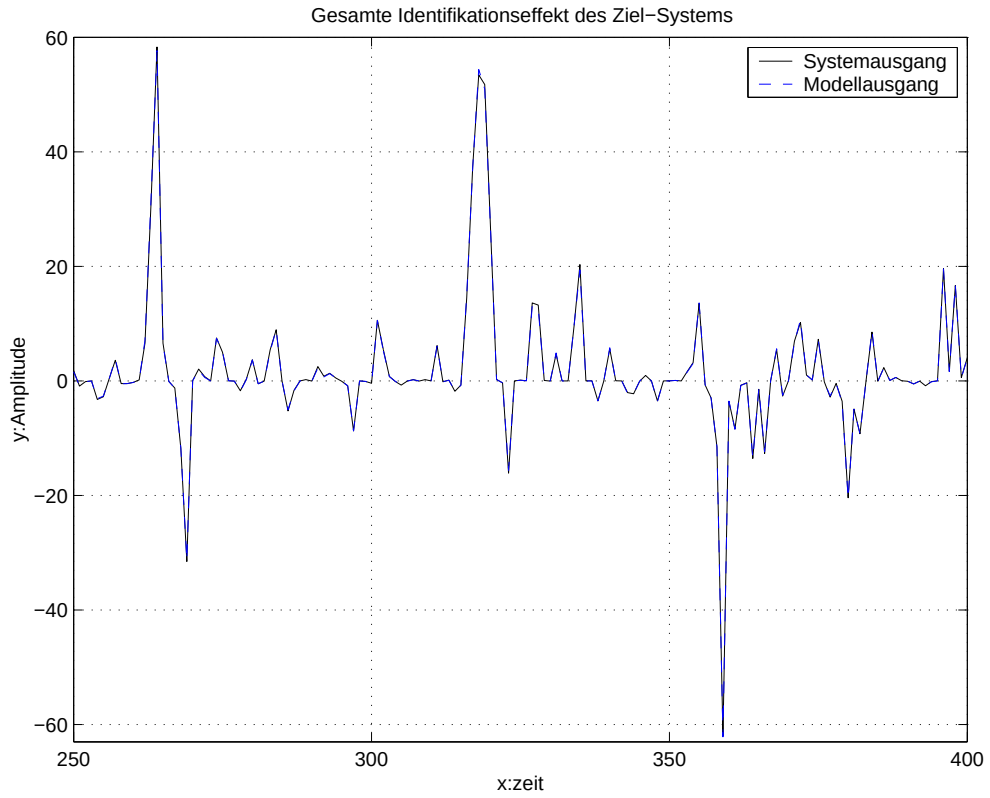


Abbildung 6.26: Systemausgang und Simulationsverlauf des Modells

Um den Rechenaufwand zu verrigern, verarbeiten wir die Ergebnisse mit dem Quotient aus zwei Polynomen weiter. Am Ende bekommen wir dann die Polynom-KGG, wie sie in den Gleichungen (6.14), (6.15) und (6.16) gezeigt sind.

$$\widehat{z} = \frac{a_6y^6 + a_5y^5 + a_4y^4 + a_3y^3 + a_2y^2 + a_1y + a_0}{b_6y^6 + b_5y^5 + b_4y^4 + b_3y^3 + b_2y^2 + b_1y + b_0}; y \in [-55, 0] \tag{6.14}$$

$$\widehat{z} = \frac{a_3y^3 + a_2y^2 + a_1y + a_0}{b_3y^3 + b_2y^2 + b_1y + b_0}; y \in [0, +9.933] \tag{6.15}$$

$$\widehat{z} = \frac{a_7y^7 + a_6y^6 + a_5y^5 + a_4y^4 + a_3y^3 + a_2y^2 + a_1y + a_0}{b_6y^6 + b_5y^5 + b_4y^4 + b_3y^3 + b_2y^2 + b_1y + b_0}; y \in [+9.933, +56.5] \tag{6.16}$$

Dabei ist die Gleichung (6.14) für den Ausgangswertebereich [-55, 0] des Ziel-Systems geeignet. Die Gleichung (6.15) ist geeignet für den Ausgangswertebereich [0, +9.933] des Ziel-Systems. Die Gleichung (6.16) ist geeignet für den Ausgangswertebereich [+9.933, +56.5] des Ziel-Systems. Der normierte Restfehler ist $\varepsilon_{KL} = 3.7 \cdot 10^{-6}$. Dabei wird die Gleichung (6.15) für die Ausgangswertebereiche [0, +0.049] und [+0.049, +9.933] optimiert.

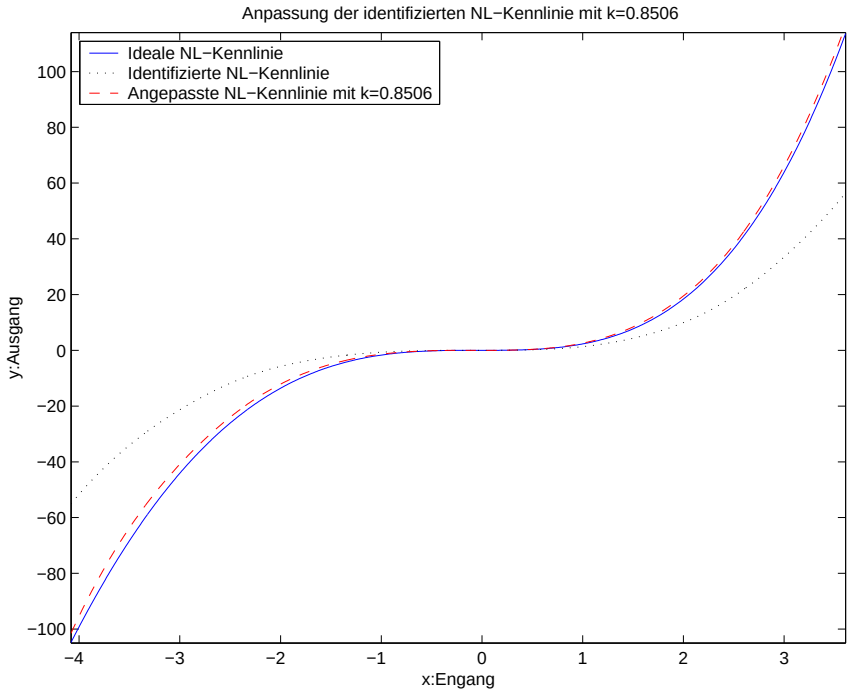


Abbildung 6.27: Anpassung der NL-Kennlinie des Modells mit einem Faktor $k = 0.8506$. Diese Anpassung hat keinen Einfluss auf die Genauigkeit des Modells. Sie verändert auch nicht die Nichtlinearität des Modells .

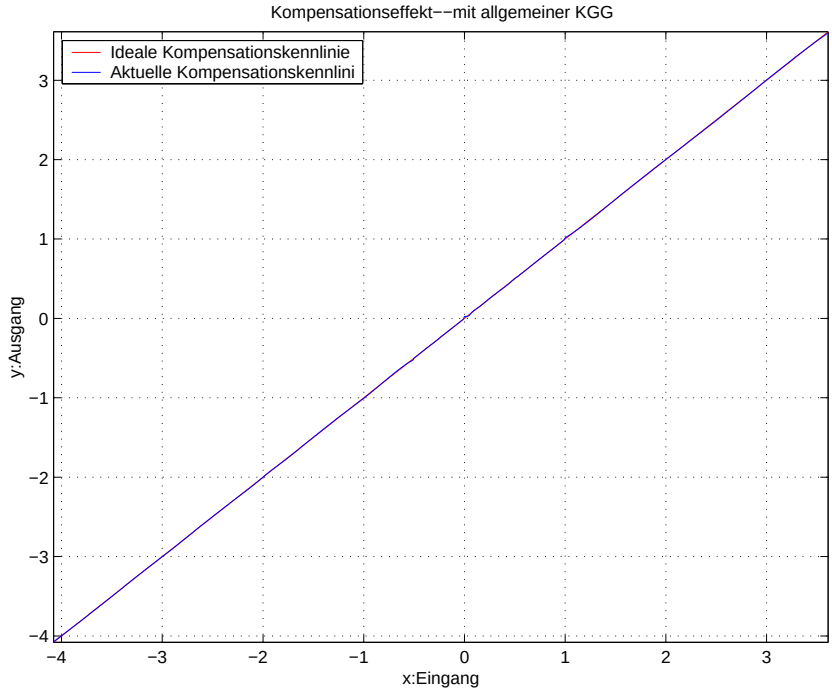


Abbildung 6.28: Kompensationseffekt der NL-Kennlinie des Modells mit KGG.

6.5 Vergleich der drei Experimente

Durch die obigen Experimente kann man sehen, dass der Kombinationsalgorithmus unter bestimmten Bedingungen für die drei Typen von Ziel-Systemen passt. Um die drei Situationen besser zu verstehen, sind in Tabelle 6.4 die Performance und die Bedingungen zusammengefasst.

Vergleich der drei Experimente			
	Experiment 1	Experiment 2	Experiment 3
Konvergenz-möglichkeit	definitiv	ziemlich sicher	abhängig von der Wahl des Initial-NL-Anteils
Rechen-aufwand	niedrig	mittel	abhängig von der Wahl des Initial-NL-Anteils
Genauigkeit	hoch	hoch	hoch
Initial-Anteil	LZI-Anteil	LZI-Anteil	NL-Anteil
Voraus-setzungen	1. monotone NL-Kenn. 2. zwei Testsignale 3. Modell enthält LZI-Anteil	1. monotone NL-Kenn. 2. ein Testsignal 3. Modell enthält LZI-Anteil	1. monotone NL-Kenn. 2. ein Testsignal 3. Modell enthält keinen LZI-Anteil
Passende Struktur	Wiener-Modell	Wiener-Modell	Wiener-Modell
Nichtlinearität	ziemlich stark	stark	sehr stark
mögliche Probleme	Bestimmen des schwachen NL-Bereichs	Konvergenz-Geschwindigkeit	Auswahl des Initial-NL-Teils

Tabelle 6.4: Vergleich der drei Experimente

6.6 Einige Hinweise für die Anwendung des Kombinationsalgorithmus

Um den Kombinationsalgorithmus erfolgreich anzuwenden, müssen folgende Hinweise beachtet werden:

- Bei der Identifikation des NL-Anteils sollte das Vorwissen über das Ziel-System genutzt werden. z.B. geglättete und monotone Eigenschaften des NL-Systemanteils.
- Der NL^{-1}-Anteil jeder Stufe muss ziemlich genau identifiziert werden und dessen Eingangswertebereich muss mit dem Ausgangswertebereich des Ziel-Systems übereinstimmen.
- Der Trainingsdatenwertebereich sollte den Eingangswertebereich der tatsächlichen Anwendung abdecken.

- Bei jeder Identifikation- oder Kompensationsstufe wird die Konstanten-Optimierung verwendet.

- Die Amplitudengangsanpassung des LZI-Anteils soll beschränkt werden. z.B. auf einen Wert kleiner als 10%.

- Wenn die GP keine globale Lösung im ganzen Bereich für die Kompensation des NL-Anteils des Modells findet, wird die KGG verwendet.

- Die Polynom-KGG hat große Bedeutung für die Kompensation in der Endstufe. Die Polynom-KGG kann nur den Rechenaufwand verringern, sie kann die Genauigkeit nicht erhöhen. Durch die abschnittsweise Optimierung (AO) kann die Polynom-KGG etwa die gleiche Genauigkeit wie die allgemeine Beschreibungsform erreichen.

Achtung: *Die Lösung der GP ist sehr sensitiv hinsichtlich des Wertebereichs des Eingangssignals, im Allgemeinen ist die Eigenschaft des nichtlinearen Systems vom Bereich des Eingangssignals abhängig.*

6.7 Reduzierung des Rechenaufwands

Im Wiener-Modell ist der Rechenaufwand stark vom statischen NL-Anteil abhängig. Obwohl die Lösung der GP sehr hohe Genauigkeiten erreicht, ist es sehr oft der Fall, dass die Lösung kompliziert und rechenaufwendig ist, wenn keine Beschränkungen für die Funktionsmenge gesetzt werden. In den praktischen Anwendungen ist es erwünscht, eine elegante und nicht rechenaufwendige Lösung zu finden.

Um eine elegante und nicht rechenaufwendige Lösung zu bekommen, kann man die Formen der GP-Lösungen beschränken. Weil die echte Form der Lösung eines Problems nicht bekannt ist, sind die besonderen und rationellen Beschränkungen schwierig. Außerdem verringert eine ungeeignete Beschränkung die Suchfähigkeit der GP.

Wenn alle Ableitungen an der Stelle x=0 existieren, kann eine Funktion $f(x)$ mit einer Tailor-Reihe, wie sie in der Gl. (6.17) beschrieben wird, entwickelt werden.

$$f(x) = \sum_{i=0}^{\infty} a_i x^i \tag{6.17}$$

In praktischen Anwendungen kann man die höheren Ordnungen vernachlässigen, damit der Rechenaufwand relativ klein wird. Aber unglücklicher Weise ist es so, dass die Suchfähigkeit der GP stark verringert wird, wenn in der GP diese reduzierte Form verwendet wird.

Dabei ist zu beachten, *dass die Beschreibungsfähigkeit von einem Modell und die Suchfähigkeit der GP mit diesem Modell zwei unterschiedliche Begriffe sind.* Die Suchfähigkeit der GP ist immer kleiner als die Beschreibungsfähigkeit des zu Grunde liegenden Modells. Wenn ein Problem durch mehrere Modellformen beschrieben werden kann, ist die Suchfähigkeit der GP für diese Beschreibungsformen unterschiedlich. Unsere Untersuchung zeigt, dass die Form (6.17) für die GP nicht sehr geeignet ist.

Um den Nachteil der Form (6.17) zu überwinden, wird in dieser Arbeit ein Quotient aus zwei Polynomen verwendet, wie er in der Gleichung (6.18) gezeigt wird. Offenbar ist die Gleichung (6.18) eine Obermenge von der Gleichung (6.17) hat aber auch noch relativ kleinen Rechenaufwand.

$$f(x) = \frac{\sum_{i=0}^{\infty} a_i x^i}{\sum_{j=0}^{\infty} b_j x^j} \tag{6.18}$$

Im Allgemeinen ist die Suchfähigkeit der GP mit der Gleichungsform (6.18) auch viel schwächer als mit der allgemeinen Beschreibungsform, aber sie ist viel stärker als mit der Gleichung (6.17). Durch abschnittsweise Optimierung (AO) kann man die Genauigkeit erhöhen.

Nach unseren Untersuchungen wird gezeigt, dass *bei der Konstanten-Optimierung die GP-Lösung sehr oft nur für bestimmte Bereiche eine sehr gute Optimierungs-Wirkung erreichen kann.* Deshalb verwenden wir die Abschnittsweise Optimierung (AO). Die AO bedeutet dabei:

1. Der Eingangswertebereich des Kompensators wird *geeignet* eingeteilt.

2. Der Kompensator wird in den eingeteilten Bereichen getrennt optimiert.

Durch die Anwendung der Gleichungsform (6.18) für den Kompensator und die AO kann man eine relativ aufwandsarme Kompensationsgleichungsgruppe (KGG) mit großer Genauigkeit finden. Da die Methode kompliziert ist, kann man sie nur in der Endstufe des Kombinationsalgorithmus verwenden. Das Ziel ist es, dafür zu sorgen dass die endgültige Lösung für die Anwendung einen akzeptablen Rechenaufwand besitzt. Im Gegensatz dazu ist die Komplexität der iterativen Zwischen-Lösungen irrelevant im Vergleich zur Komplexität der endgültigen Lösung.

Für die AO-Methode ist es wichtig, die geeigneten lokalen Optimierungsbereiche zu erkennen. Diese AO-Methode ist ziemlich gut geeignet für die Kompensation des NL-Anteils des Modells. Im Abschnitt 6.8 werden wir die Methode und das Prinzip zur Erkennung der lokalen Optimierungsbereiche an Hand eines Beispiels ausführlich darstellen. Im Allgemeinen ist es nicht schwierig, die Genauigkeit der Kompensation der NL-Kennlinie des Modells mit dem Quotient aus zwei Polynomen und der AO-Methode bis hin zu einem Restfehler von 10^{-4} zu erreichen. In dieser Arbeit wird diese Methode als die Abschnittsweise Optimierung (AO) in mehren lokalen

Bereichen mit dem Quotient aus zwei Polynomen genannt.

Im folgenden werden einige Experimente gemacht, um die AO-Methode besser zu verdeutlichen.

6.7.1 GP-Suchfähigkeit mit dem Quotienten aus zwei Polynomen

Um die Suchfähigkeit der GP mit dem Quotienten aus zwei Polynomen zu testen, wird eine NL-Kennlinienfunktion, wie sie in der Gleichung (6.19) gezeigt wird, verwendet. Im Eingangswertebereich [-1, +1] veranschaulicht die Abbildung 6.29 die Kennlinie.

$$f(x) = e^{sin(4.3x)} \cdot cos(3.3x) \tag{6.19}$$

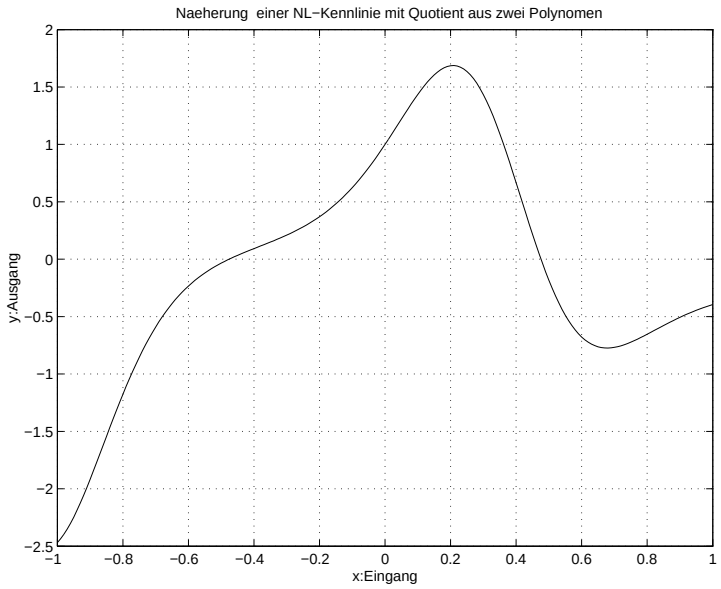


Abbildung 6.29: Original-NL-Kennlinie des Zielsystems nach Gl. (6.19)

Die Funktion ist nur für diesen Test vorgesehen, und hat keinen direkten praktischen Bezug. Sie erfüllt aber dafür die folgenden 3 Punkte:

1. Sie enthält keinen Bestandteil x^n.
2. Sie ist nicht monoton im Eingangswertebereich [-1, +1].
3. Sie ist nicht symmetrisch.

Diese Eigenschaften oder diese Komplexität der NL-Kennlinie sind ausreichend für die Kompensation des NL-Anteils durch das Modell. Wenn die GP mit einem Quotient aus zwei Polynomen diese Gleichung gut näheren kann, ist es ziemlich

wahrscheinlich, dass die GP andere noch einfachere NL-Kennlinien auch gut näheren kann.

Zunächst hat die GP die Lösung, wie sie in Gleichung (6.20) gezeigt wird, identifiziert. Die Abb. 6.30 veranschlicht den Näherungseffekt gemäß Gl. (6.20).

$$\widehat{f}(x) = \frac{\sum_{i=1}^{13} a_i x^{i-1}}{a_{15}x + a_{14}} \tag{6.20}$$

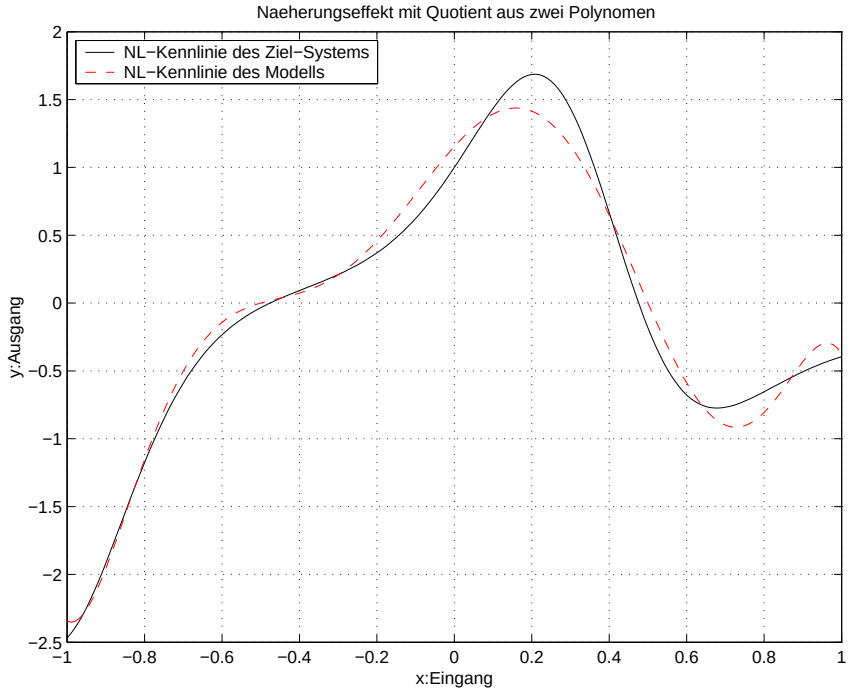


Abbildung 6.30: Identifikation eines Zielsystems gemäß Gl. (6.19): Von der GP identifizierte Kennlinie vor der Konstanten-Optimierung im Bereich [-1, +1]

Diese Lösung wird gemäß der nachfolgenden Schritte verarbeitet:

1. Konstanten-Optimierung im ganzen Bereich [-1, +1].
2. Abschnittsweise Optimierung (AO) im Teilbereich [-1, +0.2].
3. Abschnittsweise Optimierung (AO) im Teilbereich [+0.2, +1].
4. Bilden des Bereichs [-1, +1] mit den Teilbereichen [-1, +0.2] und [+0.2, +1].

Die Abbildungen 6.31, 6.32, 6.33 und 6.34 veranschaulichen die Verarbeitungseffekte der AO. Die Tabelle 6.5 zeigt die Koeffizienten, die Genauigkeit und den Rechenaufwand. Besonders veranschaulicht die Abbildung 6.35 die Veränderung der

	org. im [-1,1]	opt. im [-1,1]	AO. im [-1,0.2]	AO. im [0.2,1]	AO. im[-1,1]
a_1	-3322	-1.0000	-1.0000	-1.0000	
a_2	-7995	-3.1563	-0.2375	-8.0449	
a_3	19137	6.1628	12.2876	-9.0622	
a_4	54060	25.0761	24.1269	138.2702	
a_5	-31570	-12.2730	-7.9947	-85.9946	
a_6	-100065	-58.8466	-42.1902	-257.9031	
a_7	36307	17.3821	14.8651	157.7909	
a_8	45300	38.4608	17.7062	179.3654	
a_9	-13788	-8.7654	-2.7638	-29.7574	
a_{10}	7440	9.7009	-0.7539	-19.4668	
a_{11}	-2864	-1.4472	1.2818	-83.5041	
a_{12}	-2880	-13.1322	-0.2933	-56.2438	
a_{13}	960	1.9121	-6.4044	75.9130	
a_{14}	-2880	-0.2232	-0.2499	-0.3257	
a_{15}	960	0.1529	0.9585	0.1060	
ε	$1.70 \cdot 10^{-2}$	$5.08 \cdot 10^{-3}$	$4.15 \cdot 10^{-5}$	$1.48 \cdot 10^{-4}$	$7.38 \cdot 10^{-5}$
R	14M+13P	14M+13P	14M+13P	14M+13P	14M+13P

Tabelle 6.5: Koeffizienten, Genauigkeit und Rechenaufwand für die Gl. (6.20) mit einem Quotient aus zwei Polynomen bei verschiedener Konstanten-Optimierung

Genauigkeit bzw. des relativen Fehlers.

Durch das Beispiel kann man sehen, dass es nicht schwierig ist, eine Genauigkeit von $7.38 \cdot 10^{-5}$ zu erreichen, wobei der Rechenaufwand $(14M+13P)$ bei dieser hohen Genauigkeit akzeptabel ist. In diesem Beispiel ist die abschnittsweise Konstanten-Optimierung (AO) sehr wichtig. Der Gewinnfaktor der AO gegenüber der originalen GP-Lösung ist:

$$\frac{1.7 \cdot 10^{-2}}{7.38 \cdot 10^{-5}} = 230$$

Der Gewinnfaktor der Konstanten-Optimierung gegenüber der originalen GP-Lösung ist:

$$\frac{1.07 \cdot 10^{-2}}{5.08 \cdot 10^{-3}} = 3.4$$

Der Gewinnfaktor der AO gegenüber der Konstanten-Optimierung ist:

$$\frac{230}{3.4} = 68$$

Über die AO-Methode wird in dem folgenden Abschnitt 6.8 noch ausführlicher diskutiert. Das Beispiel zeigt eine wichtige Folgerung: es ist möglich, dass für die Näherung einer komplizierten Kennlinie mit dem Quotient aus zwei Polynomen zwei

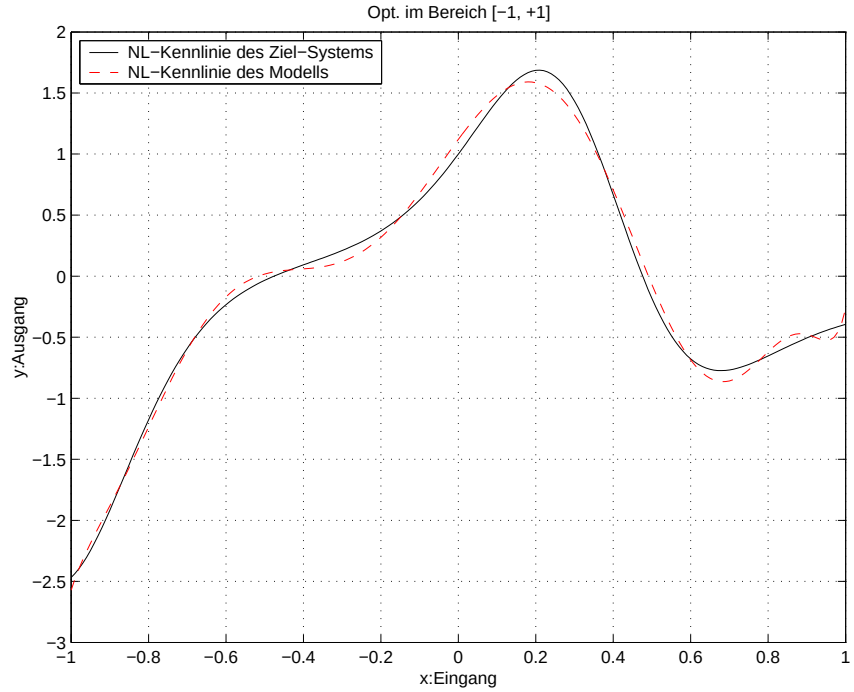


Abbildung 6.31: Identifikation eines Zielsystems gemäß Gl. (6.19): Von der GP identifizierte Kennlinie nach der Konstanten-Optimierung im Bereich [-1, +1]

Bedingungen, nämlich die höhere Genauigkeit und ein niedrigerer Rechenaufwand, gleichzeitig erfüllbar sind.

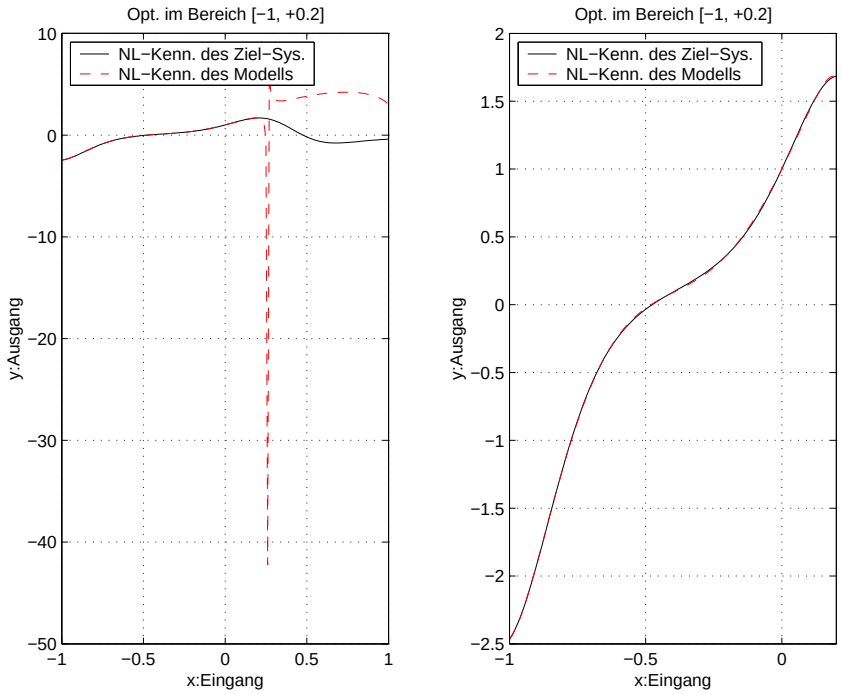


Abbildung 6.32: Identifikation eines Zielsystems durch AO im Bereich [-1, +0.2]

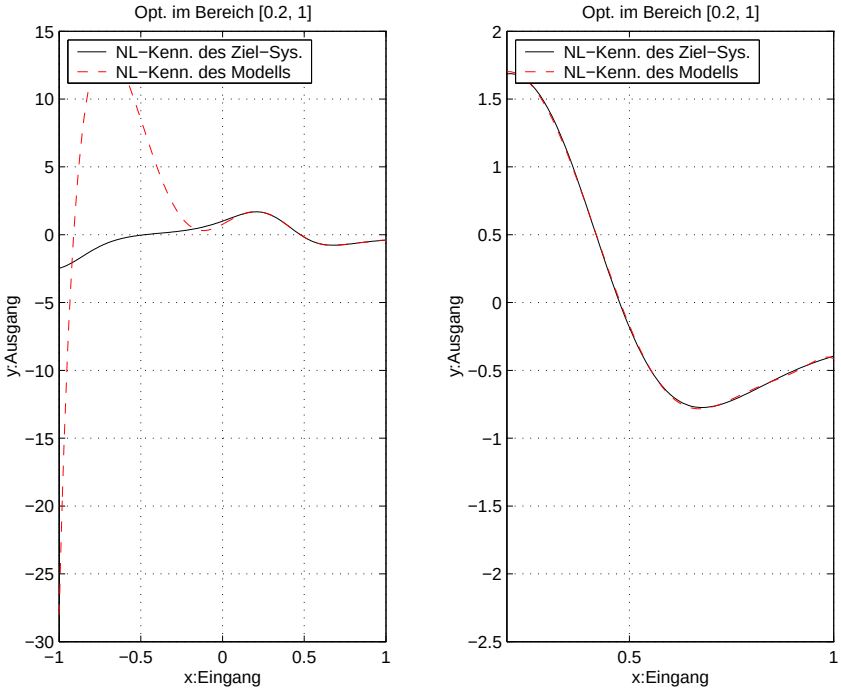


Abbildung 6.33: Identifikation eines Zielsystems durch AO im Bereich [+0.2, +1]

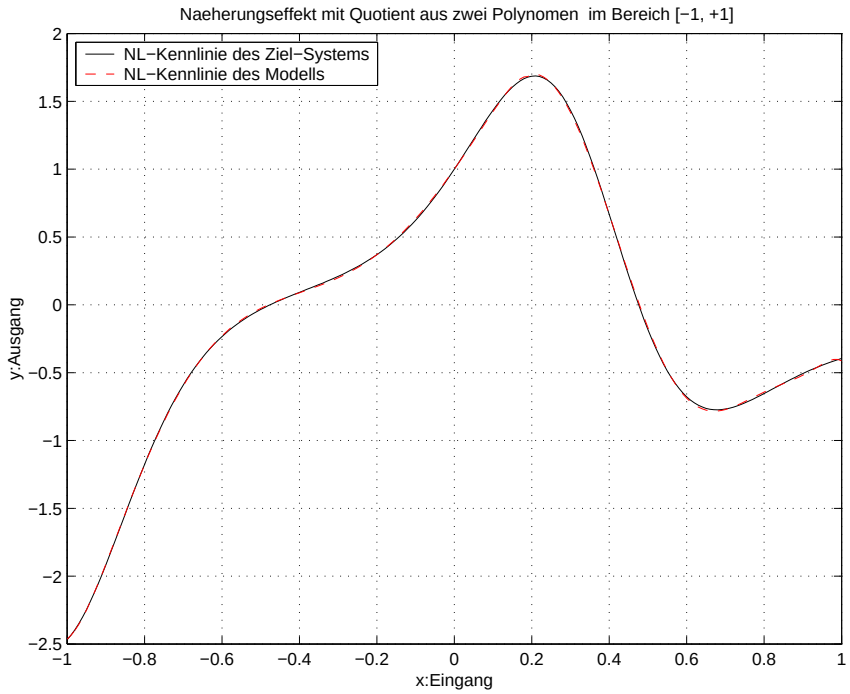


Abbildung 6.34: Identifikation eines Zielsystems gemäß Gl. (6.19) durch AO im gesamten Bereich [-1, +1]

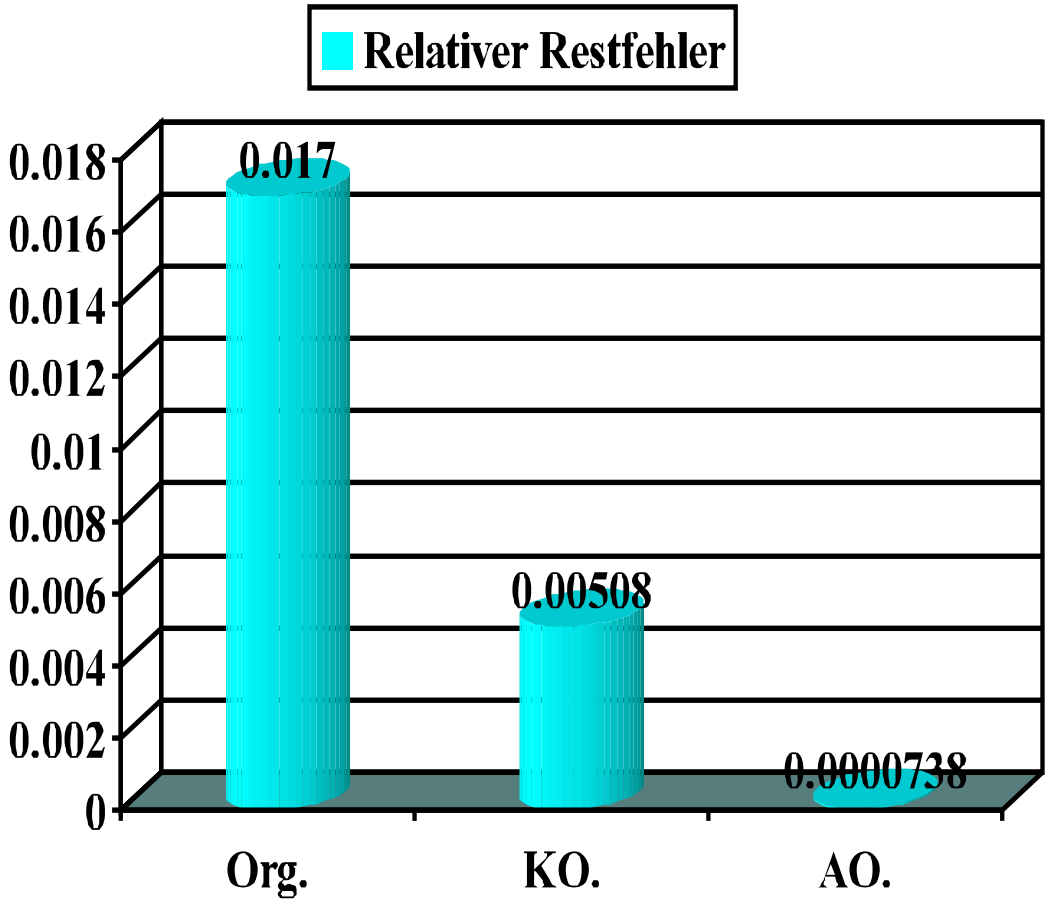


Abbildung 6.35: Vergleich des relativen Fehlers zwischen der originalen GP-Lösung mit dem Quotienten aus zwei Polynomen, der Konstanten-Optimierung und der abschnittsweisen Optimierung (AO) gemäß Gl.(6.20). Dabei bedeutet Org. die originale GP-Lösung, KO steht für Konstanten-Optimierung und AO bedeutet abschnittsweise Optimierung.

6.7.2 Vereinfachung der Lösung des Experiments 2 mit dem Quotienten aus zwei Polynomen und AO

Mit Hilfe des Quotienten aus zwei Polynomen und der AO, die im obigen Beispiel verwendet wurde, werden die Lösung aus dem Experiment 2 wieder verarbeitet. Sie wird durch die GP vereinfacht. Die Gleichung (6.21) ist die Kennlinienfunktion des NL-Anteils nach dem Wiener-Modell. Die Gleichung (6.22) ist die Kennlinienfunktion des Kompensators.

$$\widehat{y} = 1.0026\widehat{z}_L + 0.7711\widehat{z}_L^2 + 0.3999\widehat{z}_L^3 - 0.1907a_4\widehat{z}_L^4 \tag{6.21}$$

$$\widehat{z} = \frac{a_1 y^3 + a_2 y}{a_3 y^3 + a_4 y^2 + a_5 y + a_6} \tag{6.22}$$

Die Tabelle 6.6 zeigt die Koeffizienten, die Genauigkeit und den Rechenaufwand der Gleichung (6.22). Die Kompensationsgenauigkeit (ε_{KP}) $5.06 \cdot 10^{-4}$ von der Gleichung (6.22) ist ein bisschen niedriger als die Kompensationsgenauigkeit $3.86 \cdot 10^{-4}$ der Gleichungen (6.7) und (6.8). Die kleine Differenz der Genauigkeit ist nicht wichtig. Offenbar ist die Kompensationsfunktion (6.22) viel einfacher und viel rechenaufwandsärmer als die Kompensationsfunktionen (6.7) und (6.8).

Koeffizienten, relativer Fehler und Rechenaufwand					
Par.	org. im GB	opt. im GB	opt im TB_1	opt im TB_2	KGG nach AO
a_1	-18	-17.4423	-50.7562	3.8597	
a_2	-108	-107.9088	-24.6355	-139.7240	
a_3	1	1.2951	17.1718	-0.5980	
a_4	-54	-51.7418	-49.4631	24.9605	
a_5	-30	-36.5536	-26.7261	-106.8960	
a_6	-81	-92.1343	-25.9684	-128.3652	
ε_{KL}	$2.40 \cdot 10^{-3}$	$9.88 \cdot 10^{-4}$	$1.51 \cdot 10^{-5}$	$9.95 \cdot 10^{-6}$	$1.25 \cdot 10^{-5}$
ε_{KP}					$5.06 \cdot 10^{-4}$
R	7M+4P	7M+4P	7M+4P	7M+4P	7M+4P
Bemerkung: GB=[-6, +6], TB_1=[-6, 0], TB_2=[0, +6]					

Tabelle 6.6: Koeffizienten und Genauigkeit für die Gleichung (6.22) bei verschiedener Konstanten-Optimierung

Ein Vergleich mit der Gleichung (6.6) zeigt, dass die Gleichung (6.21) fast eine identische Genauigkeit erreicht, sie aber viel einfacher und viel rechenaufwandsärmer ist. Tatsächlich besitzt die Gleichung (6.21) auch die gleiche Struktur, die das Ziel-System besitzt. Es gibt nur einen kleinen Unterschied in den Koeffizienten.

Um den Effekt der nichtlinearen Kompensation zu veranschaulichen, wird eine Einton-Analyse verwendet. Die Abbildung 6.36 zeigt jeweils einen Ausschnitt des Eingangssignals, des verzerrten Ausgangssignals und des entzerrten Ausgangssignals.

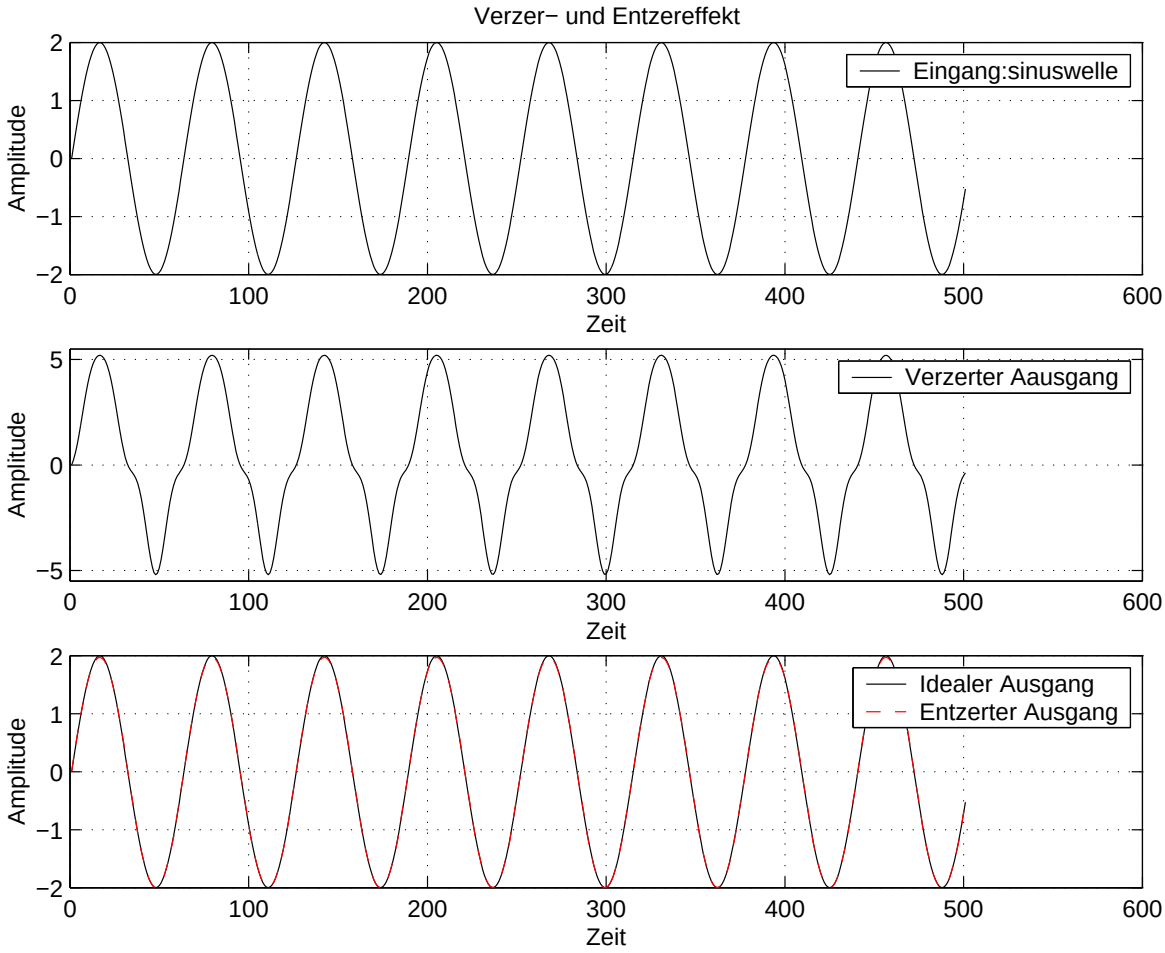


Abbildung 6.36: Verzerrungs - und Entzerrungseffekt bei einem nichtlinearen System gemäß Gl.(6.22) mit sinusförmiger Anregung

6.8 Weitere Diskussionen über den Kombinationsalgorithmus

Der vorgestellte Kombinationsalgorithmus basiert auf folgenden Punkten:

1. der GP,
2. dem Iterationsverfahren,
3. dem Wiener-Modell,
4. den Basissuchfähigkeiten der GP.

Dabei bedeuten die Basissuchfähigkeiten, dass unter den akzeptierbaren und realisierbaren Bedingungen (wie z.B. eine Populationsgröße von 7000 bis 20000 sowie 50 bis 70 Generationen für jeden Durchlauf), in begrenzter Anzahl von unabhängigen Durchläufen (wie z.B. 50-100), findet die GP fast immer eine ziemlich gute Lösung für ein Problem. Diese Basissuchfähigkeit der GP wurde in Kapitel 4 und 5 durch einige Experimente demonstriert.

Bevor der Kombinationsalgorithmus konvergiert, ist es möglich, dass der NL-Anteil der Zwischenergebnisse nicht monoton ist. Modelle mit nicht monotonem NL-Anteil sind aber nicht gültig. Das heißt, bei der Selektion müssen nicht nur der Fitnesswert eines Individuums, sondern auch dessen Monotonie berücksichtigt werden.

In diesen Experimenten, besonders aber im Experiment 3, tauchen viele Zwischenergebnisse auf, die zwar kleinere Fitnesswerte besitzen aber nicht monoton sind. Eine nicht monotone Eigenschaft der NL-Kennlinie ist in der Kompensation und daher auch bei der Selektion ausgeschlossen.

Um eine Lösung der GP mit monotoner NL-Kennlinie zu bekommen, ist es eine gute Methode die Fitnessfunktion zu modifizieren. In der Fitnessberechnung wird eine Folge von Gewichten eingefügt. Im Experiment 3 wird das Gewicht $G_i = \frac{1}{|y(i)|+1}, i = 1, 2, 3, ...$ verwendet. Gemäß unserer Untersuchungen ist es viel besser mit Gewicht als ohne Gewicht bei der Fitnessberechnung zu arbeiten, um eine monotone NL-Kennlinie zu finden.

Die Abschnittsweise Optimierung (AO), die in diesem Kombinationsalgorithmus verwendet wird, spielt eine große Rolle bei der Erhöhung der Genauigkeit. In der AO ist es wichtig, die geeigneten Abschnitts-Bereiche zu trennen. Allerdings ist die Trennung auch nicht exakt. Gemäß unserer Untersuchung durch die Beobachtung des originalen GP-Ergebnisses und der optimierten GP-Ergebnisse im ganzen Bereich kann man die Trennungslinie der Bereiche aus der Erfahrung bestimmen. Nach unseren Erfahrungen sollen folgende Kriterium herangezogen werden:

1. Bereiche, die in der globalen Optimierung gut sind.
2. Bereiche, die in der globalen Optimierung flach sind.
3. Bereiche, die im originalen GP-Ergebnis besser sind.
4. Bereiche, die im originalen GP-Ergebnis flach sind.
5. Nullpunkt der Kennlinie.
6. Grenzpunkte und Grenzwerte.

Wenn der Eingangswertebereich nicht sehr groß ist, kann man normalerweise durch die obigen Beobachtungen die Trennungslinie feststellen. Wenn der Eingangswertebereich sehr groß ist, gibt es die Möglichkeit, dass man mit der GP abschnittsweise identifiziert und kompensiert.

Um die AO-Methode zu demonstrieren, wird ein Beispiel aus einer der iterativen Stufen im Experiment 3 betrachtet. Die Abbildung 6.37 zeigt eine von der GP gefundene NL-Kennlinie des Modells in einer iterativen Zwischenstufe.

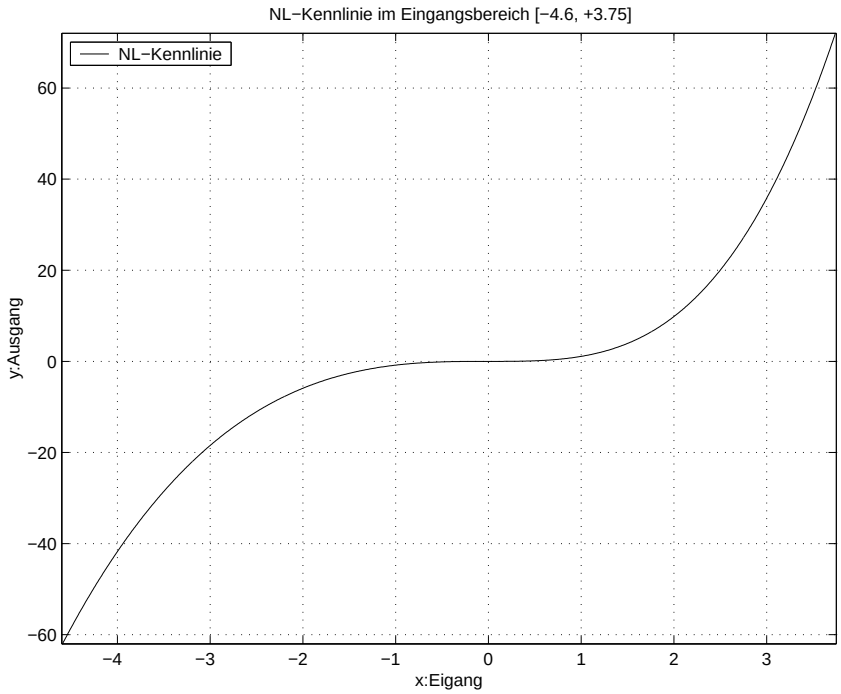


Abbildung 6.37: NL-Kennlinie in einer iterativen Stufen

Um den Ausgangswertebereich des Ziel-Systems anzupassen, liegen die Eingangswerte des NL-Anteils im Bereich von [-4.6, +3.75]. Durch 80 unabhängige GP-Durchläufe wird schließlich die beste Lösung, wie sie in der Abbildung 6.38 gezeigt

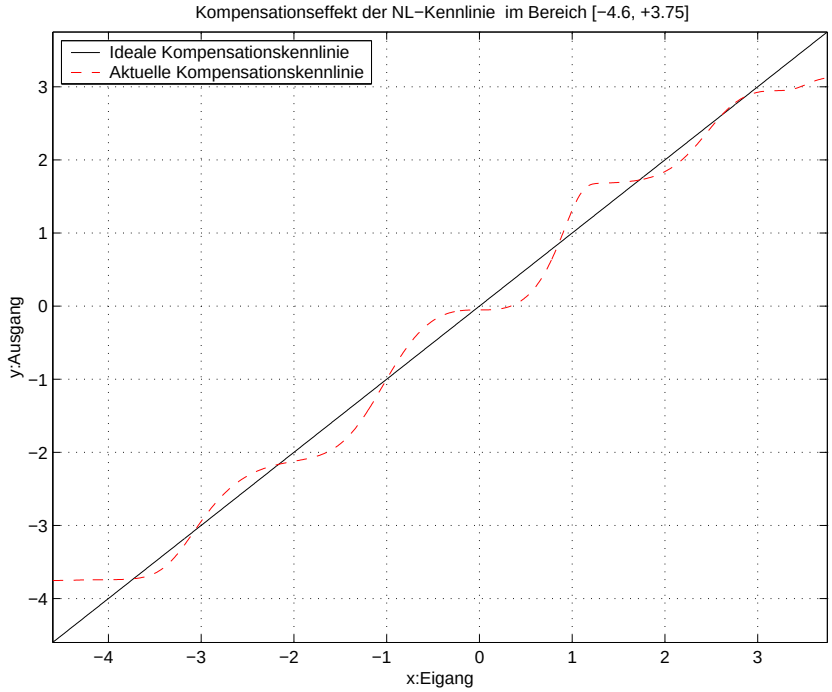


Abbildung 6.38: Kompensationseffekt der GP-Lösung im Bereich [-4.6, +3.75] vor der Konstanten-Optimierung

wird, identifiziert.

Die Abbildung 6.39 veranschaulicht den Kompensationseffekt des NL-Anteils des gefundenen Modells im Bereich [-4.6, +3.75] nach der Konstanten-Optimierung.

Aus dieser Abbildung kann man sehen, dass der Kompensationseffekt nicht gut ist, aber sich das Ergebniss außerhalb [-1, +1] im Vergleich mit dem Ergebnis vor der Konstanten-Optimierung verbessert hat. Das bedeutet, dass $x = \pm 1$ möglicherweise der Trennungspunkt für die Teilbereiche ist. Danach wird die Lösung in den Bereichen [-4.6, -1], [-1, +1] und [+1, +3.75] optimiert. Die Abbildungen 6.40, 6.41 und 6.42 veranschaulichen den Kompensationseffekt in den 3 verschiedenen Bereichen.

Aus den drei Abbildungen kann man sehen, dass der Kompensationseffekt in den Bereichen [-4.6, -1.4] und [+1.1, +3.75] nach der AO sehr gut ist. Aber sie ist nicht gut im Bereich [-1.4, +1.1]. Dann wird die Lösung in den Teilbereichen [-1.4, 0] und [0, +1.1] optimiert, weil der Ursprung normalerweise ein möglicher Trennungspunkt ist. Der Kompensationseffekt ist trotzdem nicht gut. Das bedeutet, dass die Struktur aus der GP-Lösung im Bereich [-1.4, +1.1] nicht geeignet ist.

Um eine gute Lösung im Bereich [-1.4, +1.1] zu finden, werden neue GP-Durchläufe nur im Bereich [-1.4, +1.1] durchgeführt. Dann wird eine Lösung im Bereich [-1.4, 1.1], wie sie in der Abbildung 6.43 gezeigt wird, gefunden.

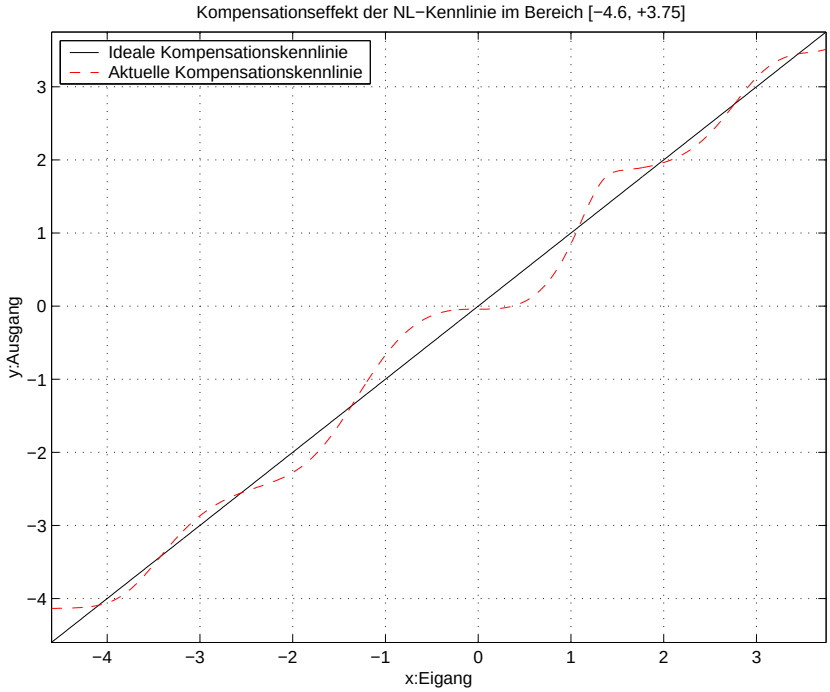


Abbildung 6.39: Kompensationseffekt der GP-Lösung nach der Konstanten-Optimierung im Bereich [-4.6, +3.75]

Die Abbildung 6.44 veranschaulicht den Kompensationseffekt im Bereich [-1.4, +1.1] nach der Konstanten-Optimierung. Aus der Abbildung 6.44 ist deutlich zu ersehen, dass der Ursprung ein Trennungspunkt ist. Deshalb wird die Konstanten-Optimierung in den Teilbereichen [-1.4, 0] und [0, +1.1] getrennt durchgeführt.

Die Abbbildungen 6.45 und 6.46 veranschaulichen den Kompensationseffekt in den Bereichen [-1.4, 0] und [0, +1.1] nach der Konstanten-Optimierung.

Am Ende wird der gesamte Kompensationseffekt, wie er in Abbildung 6.47 gezeigt wird, von den Teilbereichen in Abbildungen 6.40, 6.41, 6.45 und 6.46 zusammengebildet. Aus der Abbildung 6.47 kann man weiter ersehen, dass der Kompensationseffekt im gesamten Bereich [-4.6, +3.75] deutlich besser ist als vor der AO.

Falls das Problem durch die vorgegebene AO (Abschnitts-Optimierung) noch nicht gelöst wird, gibt es noch eine weitere Methode, nämlich AI (Abschnitts-Identifikation) und AK (Abschnitts-Kompensation):

1. Der ganze Bereich wird in verschiedene kleine Bereiche eingeteilt.

2. Die GP sucht die Lösung für jeden kleinen Bereich.

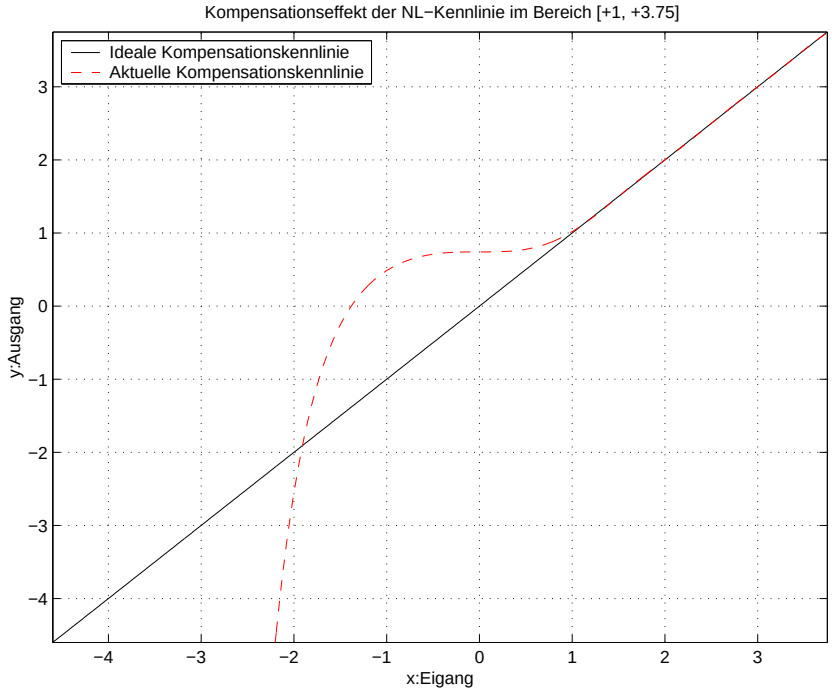


Abbildung 6.40: Kompensationseffekt der GP-Lösung nach der AO im Bereich [+1, +3.75]

Diese Methode ist zwar sicher, die Lösung aber normalerweise nicht günstig für die Anwendung, weil man in den praktischen Anwendungsfällen einen großen Aufwand betreibt:

1. Man braucht einen Entscheider für die Auswahl der Teilbereich-Lösungen.
2. Man braucht mehrere Implementierungen für die verschiedenen Teilbereich-Lösungen.

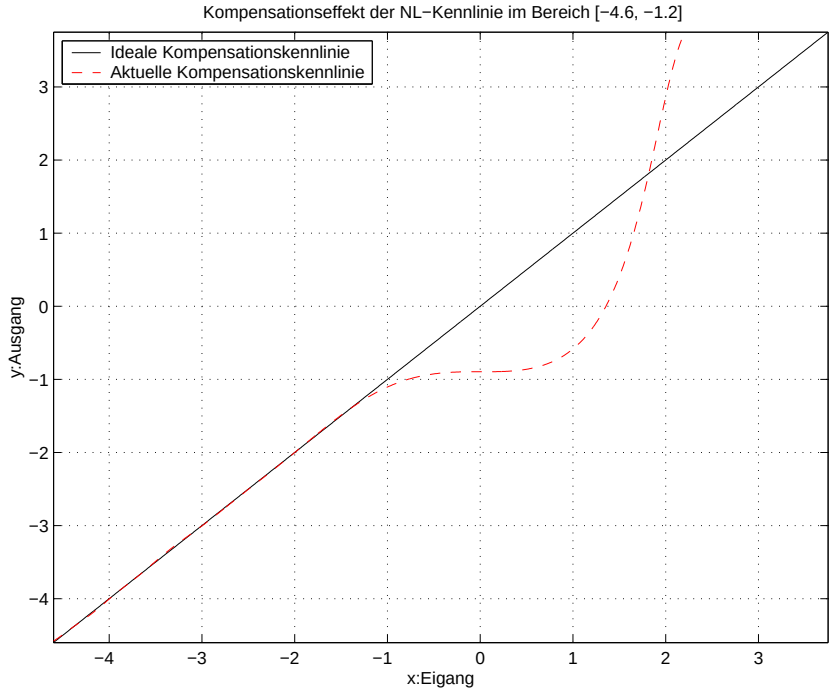


Abbildung 6.41: Kompensationseffekt der GP-Lösung nach der AO im Bereich [-4.6, -1.2]

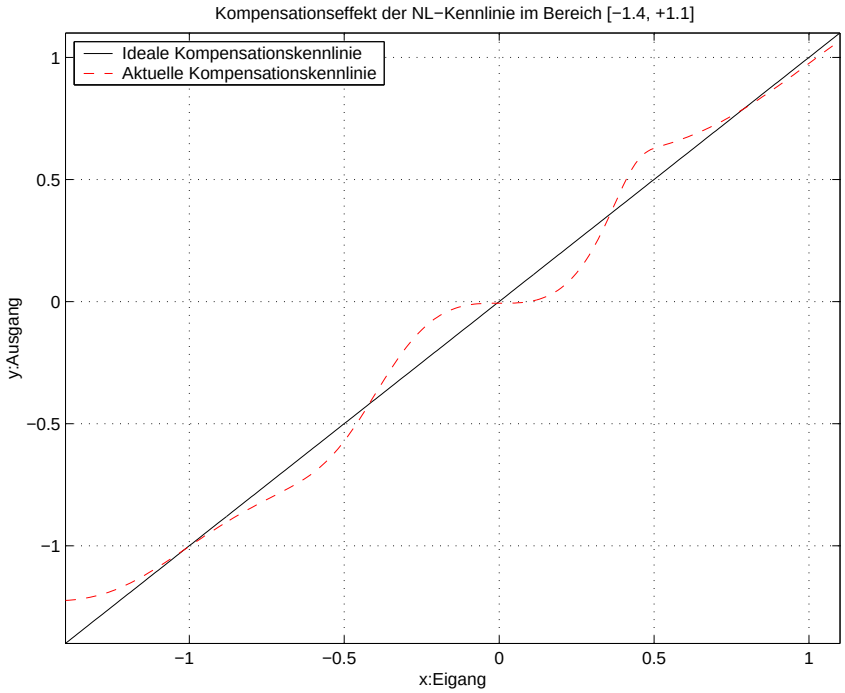


Abbildung 6.42: Kompensationseffekt der GP-Lösung nach der AO im Bereich [-1.4, +1.1]

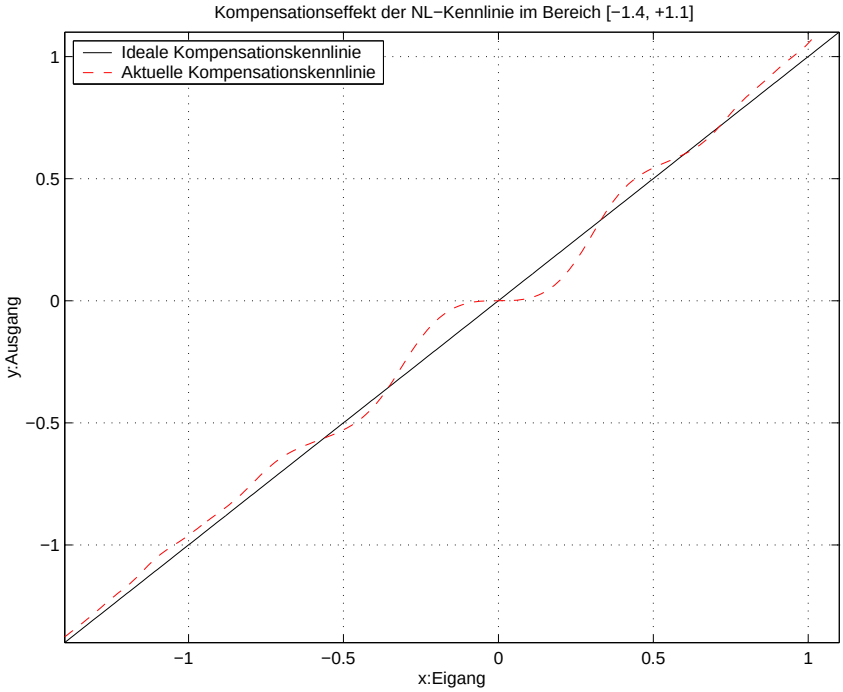


Abbildung 6.43: Getrennt gesuchte GP-Lösung vor der Konstanten-Optimierung im Bereich [-1.4, +1.1]

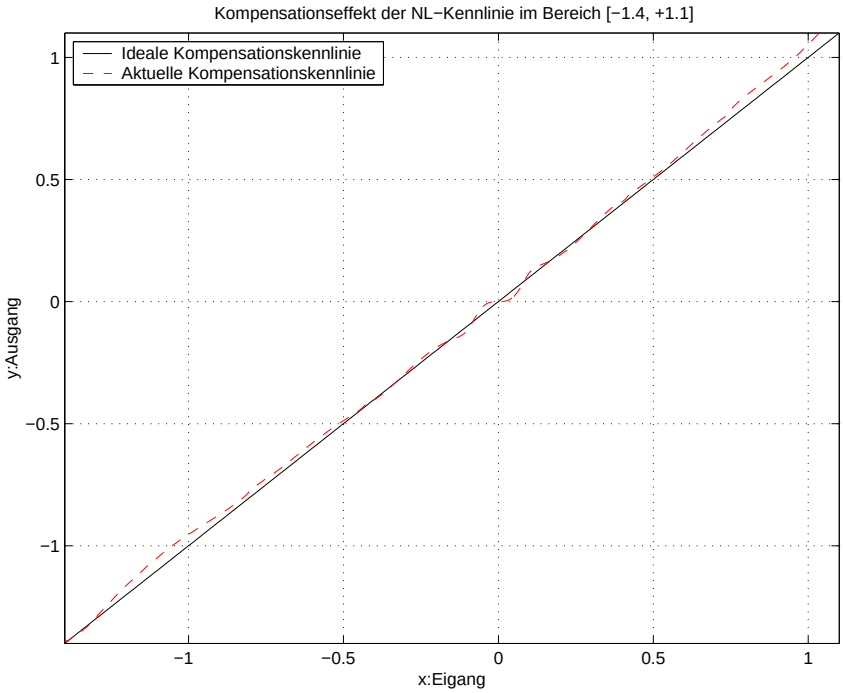


Abbildung 6.44: Kompensationseffekt aus der getrennt gesuchten GP-Lösung nach der Konstanten-Optimierung im Bereich [-1.4, +1.1]

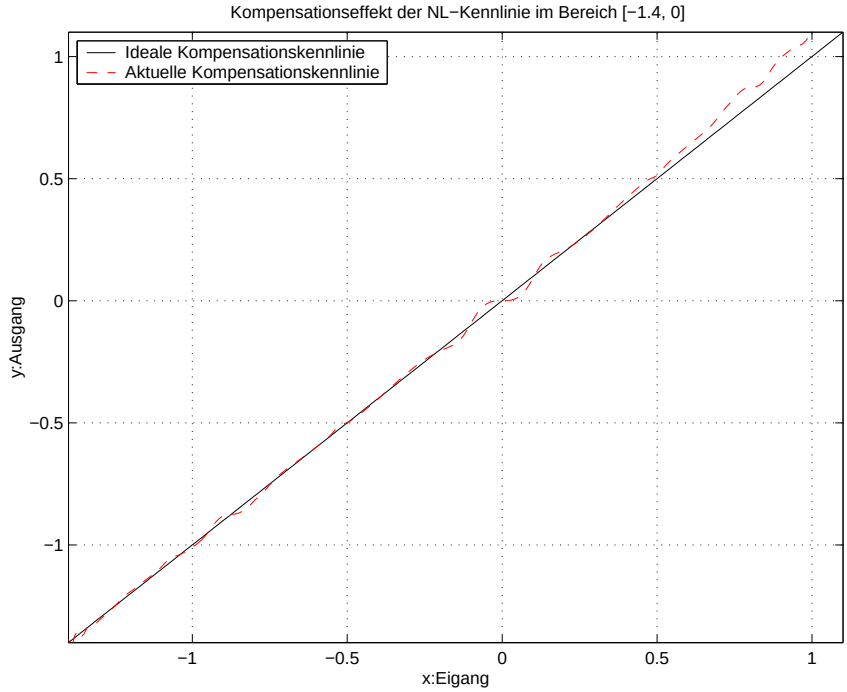


Abbildung 6.45: Kompensationseffekt aus der getrennt gesuchten GP-Lösung nach der AO im Bereich [-1.4, 0]:

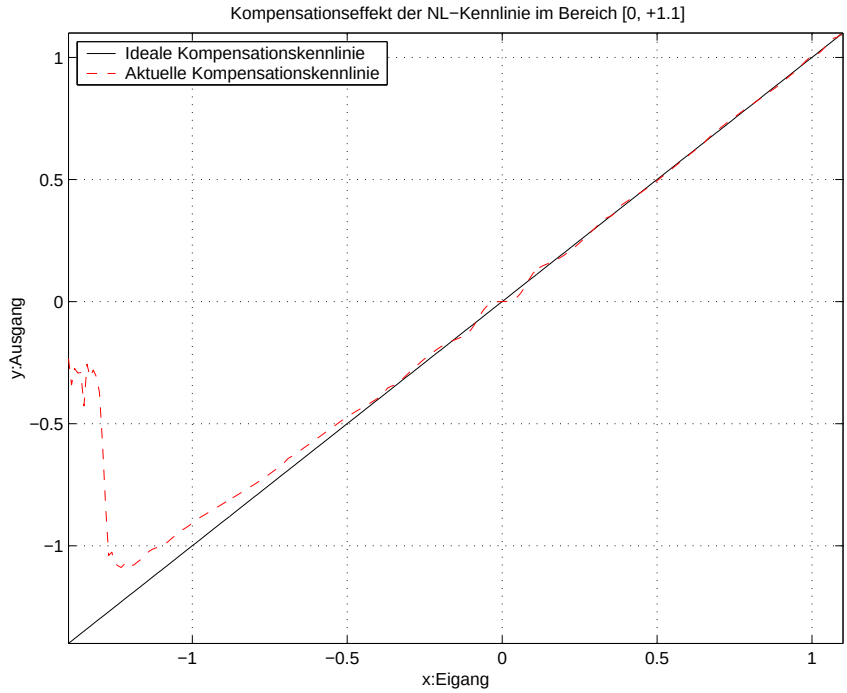


Abbildung 6.46: Kompensationseffekt aus der getrennt gesuchten GP-Lösung nach der AO im Bereich [0, +1.1]

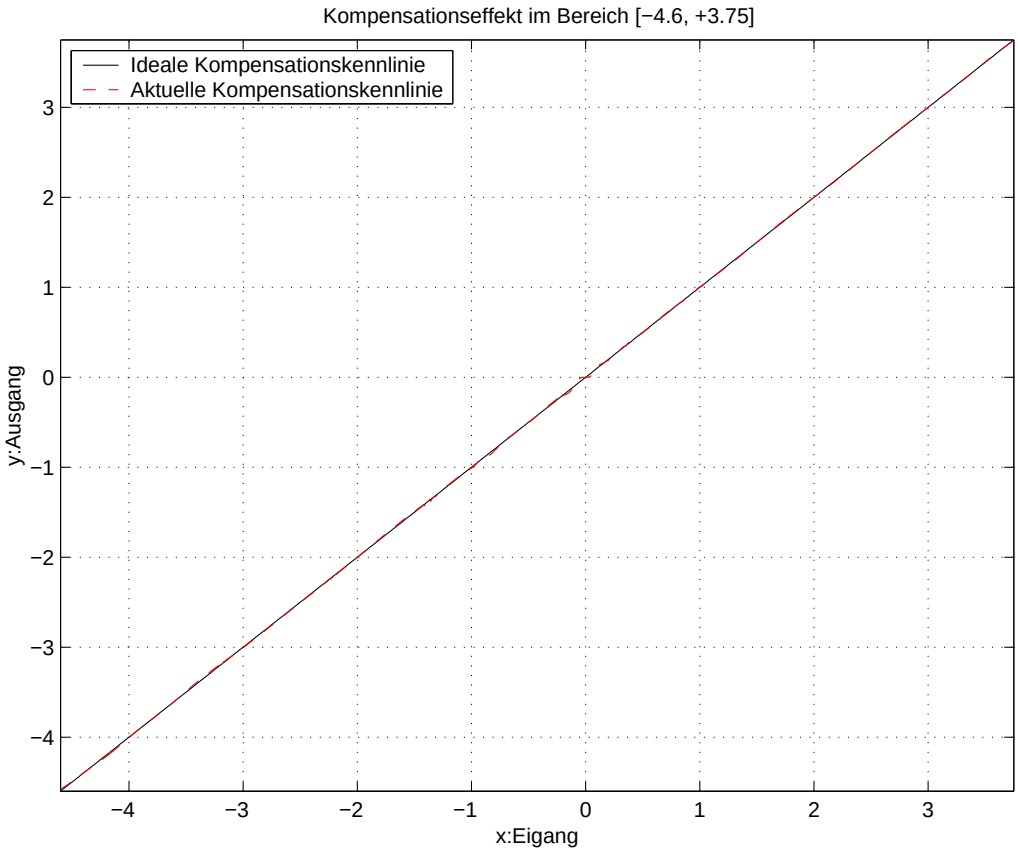


Abbildung 6.47: Kompensationseffekt im gesamten Bereich [-4.6, +3.75] mit KGG und AO

Kapitel 7

Anwendungsbeispiele

In diesem Kapitel sollen die Möglichkeiten, die beim Einsatz von rekursiven GP-Verfahren und dem Kombinationsalgorithmus in nichtlinearen Systemen entstehen, mit praktischen Beispielen demonstriert werden. Zunächst wird gezeigt, wie man die Modellierung eines realen nichtlinearen Systems (hier ein Lautsprecher) durch ein rekursives GP-Verfahren durchführen kann, anschließend wird gezeigt, wie die nichtlinearen Verzerrungen des Lautsprechers kompensiert werden können. In weiteren Beispielen wird demonstriert, wie man ein kompliziertes System durch ein Wiener-Modell annähren kann. D.h. es geht um die Frage, wie man bei der Modellierung der nichtlinearen Systeme durch den Kombinationsalgorithmus die Modellkomplexität reduzieren kann.

7.1 Identifikation und Kompensation eines Lautsprechers mit schwacher Nichtlinearität

In diesem Beispiel wird die Möglichkeit zur Identifikation und Kompensation von Nichtlinearitäten in einem Lautsprecher unter Anwendung der GP untersucht.

Für die Untersuchung werden zunächst Messsignale des Lautsprechers aufgenommen. Die Messsignale wurden als digitale Dateien zur Auswertung bereitgestellt.

7.1.1 Identifikation des Lautsprechers (Ziel-System)

Die Aufgabe der Identifikation ist es, gemäß den gemessenen Daten des Lautsprechers ein Modell zu erstellen. Dafür gibt es viele Verfahren, hier wird ein rekursives GP-Verfahren (siehe Kapitel 5) verwendet. Der Kombinationsalgorithmus ist in diesem Beispiel für die Identifikation und Kompensation des Lautsprechers nicht geeignet, weil der Lautsprecher mit einem Wiener-Modell nicht hinreichend gut nachgebildet werden kann.

Trainingsdaten

Von den gemessenen Lautsprechersignalen werden 3000 Ein- und Ausgangssignalpunkte ausgewählt. Die 3000 Daten werden als Trainingsdaten in der GP verwendet.

Terminalmenge und Funktionsmenge

Zur Identifikation des Lautsprechers ist die Terminalmenge:

$$\mathrm{T} = \{x(i-1), x(i-2), ..., x(i-O_{max}), y(i-1), y(i-2), ..., y(i-O_{max}), \pm 1, \pm 2, ...\pm 10\}$$

Dabei ist $x(\cdot)$ das Eingangssignal des Lautsprechers, $y(\cdot)$ ist das Ausgangssignal des Lautsprechers und O_{max} ist die geschätzte maximale Gedächtnislänge des Systems. In diesem Beispiel wird $O_{max} = 10$ angenommen. Gemäß den Ergebnissen der GP kann man O_{max} abschätzen. D.h wenn für einen initialen Wert von O_{max} die GP in den Individuen sehr oft den Term $x(i - O_{max})$ oder $y(i - O_{max})$ enthält, sollte der Wert von O_{max} vergrößert werden, umgekehrt wird O_{max} reduziert.

Zur Identifikation des Lautsprechers ist die Funktionenmenge wie folgt definiert:

$$\mathrm{F} = \{+, -, *, /, sin, cos, exp\}$$

Fitnessfunktion

Zur Identifikation des Lautsprechers ist die Rohfitness r die Summe des Quadrates der Differenz zwischen dem Systemausgang und dem Modellausgang. Daraus folgt:

$$r_1 = \sum_{i=O_{max}+1}^{3000} (y(i) - \widehat{y}_1(i))^2 \tag{7.1}$$

Dabei ist $y(i)$ das Ausgangssignal des Lautsprechers. $\widehat{y}_1(i)$ ist das Ausgangssignal des in der ersten Stufe identifizierten Modells. Die Fitnessfunktion ist nur gültig für die erste Stufe des rekursiven GP-Verfahrens. Für die Fitnessfunktion der weiteren Stufen gilt:

$$r_j = \sum_{i=O_{max}+1}^{3000} (\Delta y(i) - \widehat{y}_j(i))^2 \tag{7.2}$$

Wobei $\Delta y(i) = y(i) - \sum_{k=1}^{j-1} \widehat{y}_k(i)$. $\widehat{y}_k(i)$ ist das Ausgangssignal des Modellanteils, der in der $k-ten$ Stufe gefunden wurde. Diese Fitnessfunktion ist nur gültig für die $j-te$ Stufe des rekursiven GP-Verfahrens, d.h. für $j > 1$.

Einstellung der GP-Parameter und Abbruchkriterium:

Die Einstellungen der GP-Parameter sind in der Tabelle 7.1.1 dargestellt. Der einmalige GP-Lauf wird nur gestoppt, wenn der Wert für die maximalen Generationenanzahl G_{max} erreicht ist.

GP-Parameter	Einstellung
Anzahl der GP-Läufe RUN_{max}	20
Populationsgröße M	10000
Max. Generation G_{max}	60
Kreuzungswahrscheinlichkeit p_c	0,9
Reproduktionswahrscheinlichkeit p_r	0,1
Mutationswahrscheinlichkeit p_m	0,3
Max. Tiefe der Anfangspopulation $D_{initial}$	25
Max. Tiefe während des restl. Laufes $D_{created}$	75
Initialisierungsmethode	Half-ramping
Selektionsverfahren	Fitnessproportional
Elitismus	Ja

Tabelle 7.1: Einstellungen des GP-Durchlaufes für die Identifikation und Post-Linearisierung des Lautsprechers

Simulationsergebnisse

Die Abb. 7.1 zeigt den Zusammenhang zwischen dem normierten Restfehler und der Anzahl der rekursiven Stufen. Dabei kann man sehen, dass der Simulationsfehler nicht weiter abnimmt, wenn die 11. rekursive Stufe erreicht ist. Der relative Restfehler ist dann $1.7 \cdot 10^{-3}$. Die Abb. 7.2 zeigt den Simulationsverlauf des Lautsprechers im Zeitbereich.

7.1.2 Post-Linearisierung

Das Ziel der Post-Linearisierung ist es, durch einen Post-Linearisierer die nichtlinearen Verzerrungen des Ziel-Systems zu kompensieren und die lineare Übertragungsfunktion des Ziel-Systems beizubehalten. Der Post-Linearisierer ist offenbar ein nichtlineares System, um ihn aufzufinden, benötigt die GP das Ausgangssignal des "LZI-Systemanteils" des Ziel-Systems und das Ausgangssignal des Ziel-Systems (siehe Kapitel 2). Die beiden Signale bilden die Trainingsdaten zur Auffindung des Post-Linearisierers.

Trainingsdaten

Mit obigem identifizierten Modell des Lautsprechers wird der LZI-Systemanteil abgespalten. Mit dem Eingangssignal des Lautsprechers und diesem abgespalteten

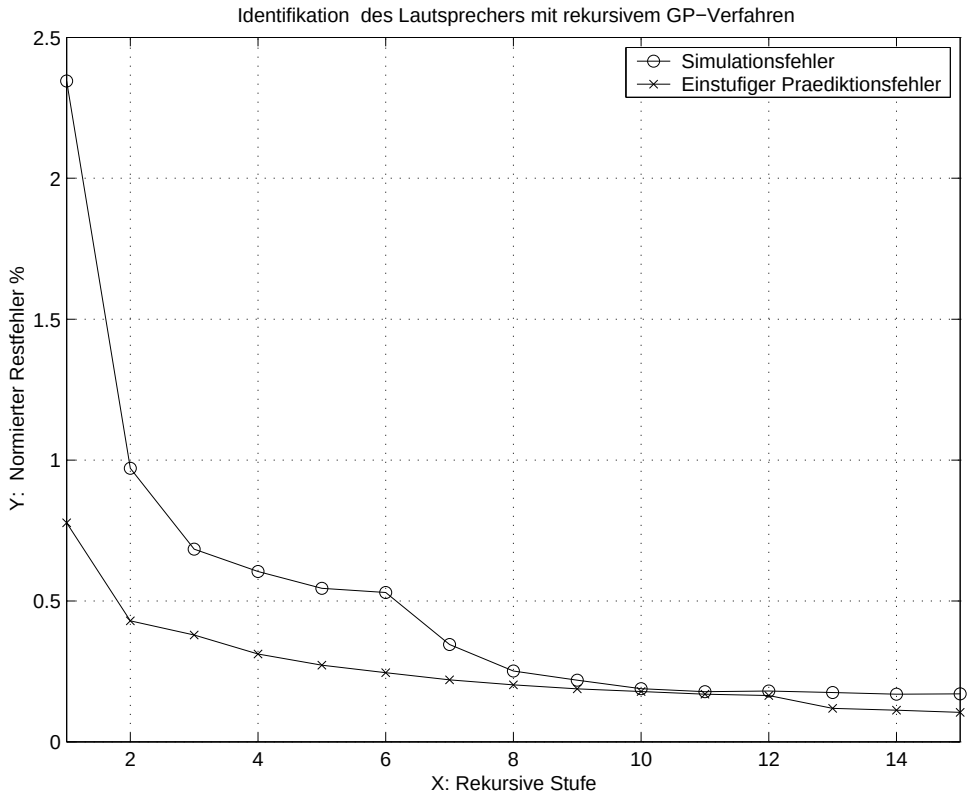


Abbildung 7.1: Identifikation des Lautsprechers mit einem rekursiven GP-Verfahren

LZI-Systemanteil wird das Ausgangssignal des LZI-Systemanteils des Lautsprechers berechnet. Dieses Ausgangssignal mit $2T_0$ Verzögerung bildet das Ausangssignal zur Auffindung des Post-Linearisierers. Das Ausgangssignal des Lautsprechers bildet das Eingangssignal zur Auffindung des Post-Linearisierers(siehe Kapitel 2: Aufbau zur Auffindung des Post-Linearisiers). Von diesen beiden Ein- und Ausgangssignalen werden 3000 Trainingsdaten zur Auffindung des Post-Linearisierers erzeugt.

Terminalmenge

Während der Auffindung des Post-Linearisierers mit Rückkopplungs-Term tritt das gleiche Problem, nämlich das Stabilitätsproblem, wie es in Kapitel 5 besprochen wurde, auf. Um dieses Problem zu vermeiden, wird die GP-Terminalmenge keinen Rückkopplungs-Term enthalten. Die Terminalmenge ist also:

$$\mathrm{T} = \{x_{PL}(i-1), x_{PL}(i-2), ..., x_{PL}(i-O_{max}), \pm 1, \pm 2, ... \pm 10\}$$

Dabei ist O_{max} die geschätzte maximale Gedächtnislänge des Post-Linearisierers, für die in diesem Beispiel $O_{max} = 30$ gilt. Gemäß den Zwischenergebnissen der GP kann man O_{max} noch modifizieren. x_{PL} ist das Eingangssignal zur Auffindung des Post-Linearisierers, nämlich $x_{PL} = y$. y_{PL} ist das linearisierte Ausgangssignal des Lautsprechers.

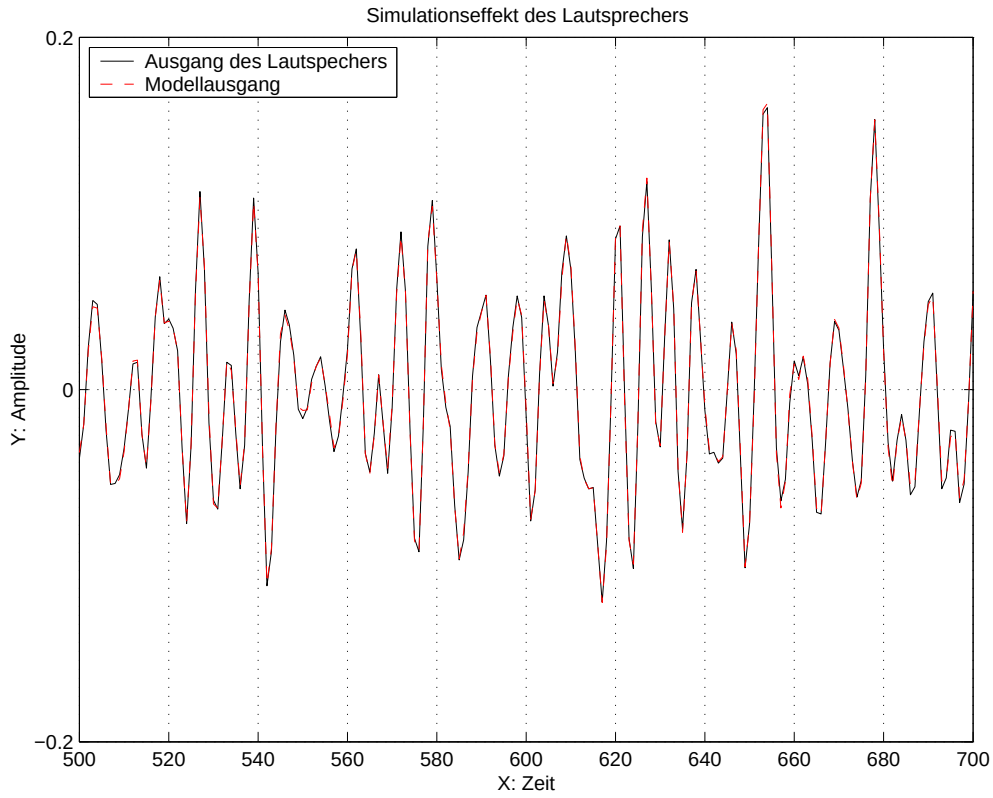


Abbildung 7.2: Simulationsverlauf des Systemausgangs des Lautsprechers und des Ausgangs des Modells

Der Vorteil von der Terminalmenge ohne Rückkopplungs-Term ist der, dass der Post-Linearisierer stabil läuft.

Weitere Bedingungen und Einstellungen des GP-Laufes

Die Funktionenmenge, die Fitnessfunktion, die Einstellung der GP-Parameter und das Abbruchkriterium bleiben unverändert wie bei der Systemidentifikation.

Simulationsergebnisse

Die Abb. 7.3 zeigt den Zusammenhang zwischen dem normierten Restfehler und der Anzahl der rekursiven Stufen bei der Auffindung des Post-Linearisierers.

Dabei kann man sehen, dass der normierte Restfehler sehr langsam abnimmt. Die Anzahl der rekursiven Stufen ist viel größer als die Anzahl der Stufen bei der Identifikation des Lautsprechers. Die Ursache hierfür ist, dass in der Terminalmenge keine Rückkopplungs-Terme enthalten sind. Deshalb ist das Modell wesentlich umfangreicher. Der normierte Restfehler ist $2.0 \cdot 10^{-3}$.

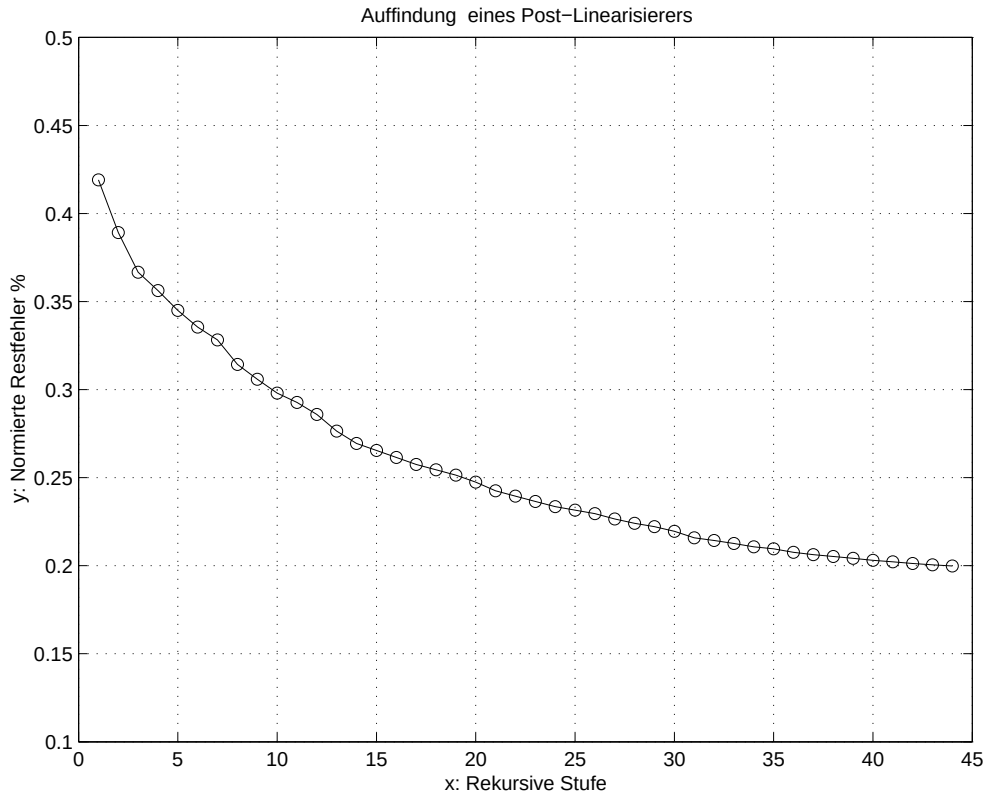


Abbildung 7.3: Auffindung des Post-Linearisierers mit rekursivem GP-Verfahren

7.2 Reduktion der Modellkomplexität durch Systemidentifikation nach dem Wiener-Modell

In der modernen Steuerungstechnik sowie anderen Gebieten spielen nichtlineare Steuerungen bzw. die nichtlineare Modellierung eine zunehmend wichtigere Rolle. In diesen Fällen ist eine rechenaufwandsarme Modellierung wünschenswert, um die Steuerung oder nichtlineare Signalverarbeitung in Echtzeit implementieren zu können. Das Wiener-Modell ist ein rechenaufwandsarmes nichtlineares Modell. In den folgenden Beispielen wird demonstriert, wie ein nichtlineares Ziel-System, das in der Steuerungstechnik oder Nachrichtentechnik möglicherweise verwendet wird, durch den Kombinationsalgorithmus mit einem Wiener-Modell approximiert werden kann.

7.2.1 Beispiel 1

Ziel-System und Trainingsdaten

Das nichtlineare Ziel-System in diesem Beispiel wird durch die Abb. 7.4 veranschaulicht. Dabei zeigen die Gleichungen (7.3)–(7.8) die entsprechenden $\mathrm{H}_1(s)$–$\mathrm{H}_6(s)$.

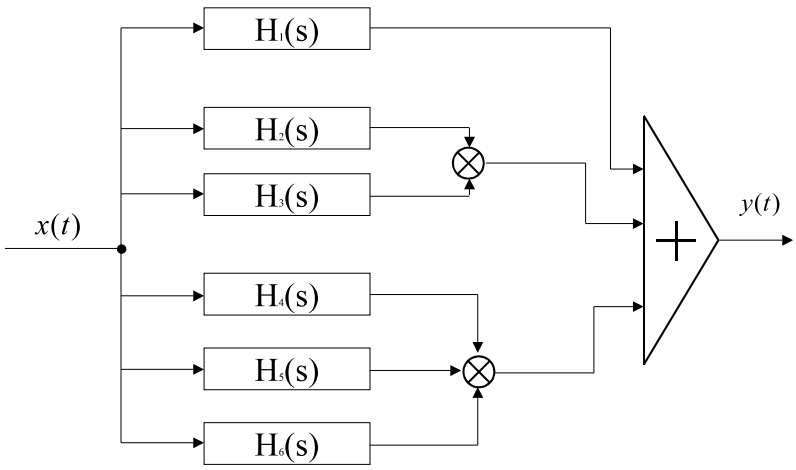


Abbildung 7.4: Ziel-System 1 für Beispiel 1

$$\mathrm{H}_1(s) = \frac{0.25s + 0.4}{0.24s^5 + 0.28s^4 + 0.33s^3 + 0.365s^2 + 0.43s + 1} \tag{7.3}$$

$$\mathrm{H}_2(s) = \frac{0.036s + 0.048}{0.12s^5 + 0.12s^4 + 0.12s^3 + 0.12s^2 + 0.12s + 1} \tag{7.4}$$

$$\mathrm{H}_3(s) = \frac{0.025s + 0.03}{0.1s^2 + 0.1s + 1} \tag{7.5}$$

$$\mathrm{H}_4(s) = \frac{0.315}{0.45s + 1} \tag{7.6}$$

$$\mathrm{H}_5(s) = \frac{0.3055}{0.47s^2 + 0.47s + 1} \tag{7.7}$$

$$\mathrm{H}_6(s) = \frac{0.192s + 0.36}{0.48s^3 + 0.48s^2 + 0.48s + 1} \tag{7.8}$$

Das Eingangssignal x ist ein weißes Rauschen und dessen Werte sind auf [-1.75, +1.75] beschränkt. Die Ausgangssignalwerte liegen im Bereich [-5.0, +5.0]. Zur Identifikation des Ziel-Systems werden 2500 Trainingsdaten verwendet.

Terminalmenge und Funktionsmenge

Durch den Kombinationsalgorithmus wird das nichtlineare Ziel-System in zwei Teile, nämlich den LZI-Anteil und den statischen (gedächtnislosen) NL-Anteil, aufgeteilt. Die zwei Anteile werden durch die GP iterativ identifiziert. Zur Identifikation jedes Anteils gibt es also eine eigene Terminalmenge und eine eigene Funktionenmenge.

Zur Identifikation des LZI-Anteils verwendet die GP die Terminalmenge:

$$\mathrm{T}_{\mathrm{LZI}} = \{x(i-1), x(i-2), ..., x(i-O_{\mathrm{L}max}), z(i-1), z(i-2), ..., z(i-O_{\mathrm{L}max}), \pm 1, \pm 2, ...\pm 10\}$$

Dabei ist $z(\cdot)$ das Ausgangssignal des LZI-Anteils. $O_{\mathrm{L}max}$ ist die Gedächtnislänge des LZI-Anteils, die in diesem Beispiel zu $O_{\mathrm{L}max} = 15$ angenommen wird. Gemäß den Zwischenergebnissen der GP kann man $O_{\mathrm{L}max}$ modifizieren.

Zur Identifikation des LZI-Anteils verwendet die GP die Funktionenmenge:

$$\mathrm{F}_{\mathrm{LZI}} = \{+, -, *, /\}$$

In der Funktionenmenge gibt es zwar die Operatoren Multiplizieren (*) und Dividieren (/), aber diese beiden Operatoren sind nur für die Operationen zwischen den Konstanten bzw. den Konstanten und den Variablen (außer Konstant/Variable) gültig und nicht gültig für die Operationen zwischen den Variablen . Dies wird durch die LZI-GP-Grammatik automatisch berücksichtigt. In der LZI-GP-Grammatik sind nur die linearen Kombinationen zwischen den Variablen erlaubt.

Zur Identifikation des statischen NL-Anteils verwendet die GP die Terminalmenge:

$$\mathrm{T}_{\mathrm{NL}} = \{z, \pm 1, \pm 2, ... \pm 10\}$$

und die Funktionenmenge:

$$\mathrm{F}_{\mathrm{NL}} = \{+, -, *, /, sin, cos, exp\}$$

Dabei gelten die Operationen für alle Variablen und Konstanten.

Fitnessfunktion

Zur Identifikation des LZI-Anteils gilt für die Rohfitness $r_{\mathrm{LZI},1}$ der $1-ten$ iterativen Stufe:

$$r_{\mathrm{LZI},1} = \sum_{O_{\mathrm{L}max}+1}^{2500} (y(i) - \widehat{z}_{1,1}(i))^2 \tag{7.9}$$

Dabei ist $y(i)$ das Ausgangssignal des Ziel-Systems. $\widehat{z}_{1,1}(i)$ ist das Ausgangssignal des in der ersten iterativen Stufe geschätzten LZI-Anteils. Diese Rohfitnessfunktion ist *nur* für die erste iterative Stufe gültig. Für die Rohfitness der $j-ten$ iterativen Stufe gilt:

$$r_{\mathrm{LZI},j} = \sum_{i=O_{\mathrm{L}max}+1}^{2500} (\widehat{z}_{2,j-1}(i) - \widehat{z}_{1,j}(i))^2 \tag{7.10}$$

Dabei ist $\widehat{z}_{1,j}(i)$ das Ausgangssignal des in der $j-ten$ iterativen Stufe geschätzten LZI-Anteils. $\widehat{z}_{2,j-1}(i)$ ist das Ausgangssignal des NL-Kompensators in der $j-1-ten$ iterativen Stufe. Diese Rohfitnessfunktion ist gültig für die $j-te$ iterative Stufe, und $j > 1$.

Zur Identifikation des statischen NL-Anteils ist die Rohfitness $r_{\mathrm{NL},j}$ der $j-ten$ iterativen Stufe die Summe der gewichteten Quadrate der Differenz, um möglichst glatte Eigenschaften für die statische NL-Kennlinie zu erhalten. Daraus folgt:

$$r_{\mathrm{NL},j} = \sum_{i=O_{\mathrm{L}max}+1}^{2500} (\widehat{f}_j(\widehat{z}_{1,j}(i)) - y(i))^2 \cdot \frac{1}{1 + \|y(i)\|} \tag{7.11}$$

Dabei ist $\widehat{f}_j(\widehat{z}_{1,j})$ das Ausgangssignal des in der $j-ten$ iterativen Stufe geschätzten NL-Anteils. y ist das Ausgangssignal des Ziel-Systems.

Zur Auffindung des Kompensators des statischen NL-Anteils in der $j-ten$ iterativen Stufe benötigt die GP für den statischen NL-Anteil ein *neues* Anregungssignal $x_{\mathrm{NL},j}$, wie es in Gl. (7.12) gezeigt wird.

$$x_{\mathrm{NL},j}(i) = [-a_j, ..., -0.03, -0.02, -0.01, 0.00, 0.01, 0.02, 0.03, ..., b_j] \tag{7.12}$$

Dabei ist $-a_j$ die Untergrenze und b_j die Obergrenze von $x_{\mathrm{NL},j}$. $-a_j$ und b_j *müssen* dem Ausgangswertebereich des Ziel-Systems y angepasst sein, ansonsten ist ein schlechtes Ergebnis vom Kompensator zu erwarten.

Zur Auffindung des Kompensators für den statischen NL-Anteil ist die Rohfitness $r_{\mathrm{NL}^{-1},j}$ der $j-ten$ iterativen Stufe die Summe der gewichteten Quadrate der Differenz zwischen dem Ausgangssignal $\widehat{y}_{Komp,j}$ des Kompensators und dem Eingangssignal $x_{\mathrm{NL},j}$ des statischen NL-Anteils. Damit kann die Kompensationsgenauigkeit in der nahe Ursprungs verbessert werden. Daraus folgt:

$$r_{\mathrm{NL}^{-1},j} = \sum_{i=1}^{\mathrm{N}_{Komp,j}} (x_{\mathrm{NL},j}(i) - \widehat{y}_{Komp,j}(i))^2 \cdot \frac{1}{1 + \|x_{\mathrm{NL},j}(i)\|} \tag{7.13}$$

Dabei ist $x_{\mathrm{NL},j}$ das neue Anregungssignal des statischen NL-Anteils für die $j-te$ iterative Stufe. $\widehat{y}_{Komp,j}$ ist das Ausgangssignal des Kompensators. $\mathrm{N}_{Komp,j}$ ist die Datenlänge von $x_{\mathrm{NL},j}$. Diese Datenlänge kann für verschiedene iterative Stufen unterschiedlich sein.

Einstellung der GP-Parameter und Abbruchkriterium

Die Einstellungen der GP-Parameter sind in der Tabelle 7.2.1 dargestellt. Der einmalige GP-Lauf wird nur gestoppt, wenn die Anzahl der maximalen Generationen G_{max} erreicht ist.

GP-Parameter	Einstellung
Anzahl der GP-Läufe RUN_{max}	40
Populationsgröße M	10000
Max. Generation G_{max}	60
Kreuzungswahrscheinlichkeit p_c	0.9
Reproduktionswahrscheinlichkeit p_r	0.1
Mutationswahrscheinlichkeit p_m	0.3
Max. Tiefe der Anfangspopulation $\mathrm{D}_{initial}$	25
Max. Tiefe während des restl. Laufes $\mathrm{D}_{created}$	75
Initialisierungsmethode	Half-ramping
Selektionsverfahren	Fitnessproportional
Elitismus	Ja

Tabelle 7.2: Einstellungen des GP-Durchlaufes für Beispiel 1

Simulationsergebnisse

Durch den Kombinationsalgorithmus (bis zur 8. iterativen Stufe) wird das nichtlineare Ziel-System mit einem Wiener-Modell nachgebildet.

Die Gleichung (7.14) zeigt den endgültigen LZI-Anteil des Wiener-Modells, und die Gleichung (7.15) zeigt die endgültige Kennlinienfunktion des statischen NL-

Anteils des Wiener-Modells.

$$\widehat{\mathrm{H}}(s) = \frac{0.2305 - 0.0937s^2 - 0.0352s^3 + 0.0078s^7}{1 - 0.1250s^6} \tag{7.14}$$

$$\begin{aligned}\widehat{f}_{\mathrm{NL}}(x) = a_{14}x \cdot exp(a_{13}sin(a_{12}exp(a_{11}cos(a_{10}x)))+ \\ a_9exp(a_7exp(a_5sin(a_4exp(a_3x \cdot exp(a_2cos(a_1x)))) + a_6) + a_8))\end{aligned} \tag{7.15}$$

Der normierte Restfehler zwischen dem Ausgang des Ziel-Systems und dem Ausgang des Wiener-Modells ist $1.55 \cdot 10^{-2}$. Dabei kann man feststellen, dass der Rechenaufwand von der Komplexität des statischen NL-Anteils abhängig ist.

Die Gleichung (7.15) kann mit der GP durch einen Quotienten aus zwei Polynomen noch vereinfacht werden. Die Gleichung (7.16) zeigt die vereinfachte Kennlinienfunktion. Die Modellierungsgenauigkeit ist etwa gleich wie bei der Kennlinienfunktion gemäß Gl. (7.15).

$$\widehat{f}_{\mathrm{NL}}(x) = -0.1942x^{13} + 1.1045x^3 + 1.9826x + \frac{15.6632x^3}{(2.3630x^{10} + 5.2795)^2} \tag{7.16}$$

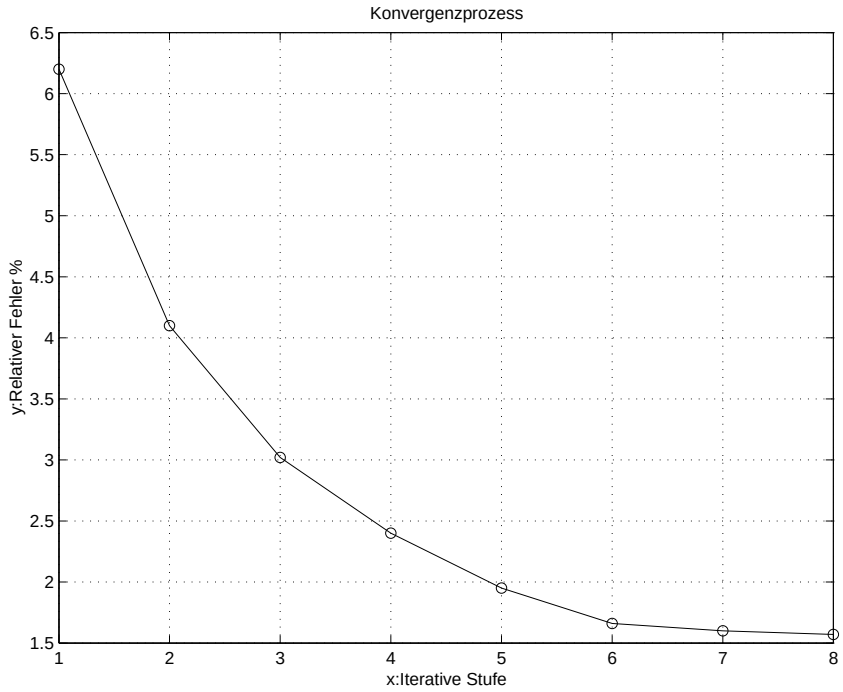


Abbildung 7.5: Konvergenzprozess für Beispiel 1

Die Abb. 7.5 zeigt den Konvergenzverlauf des Kombinationsalgorithmus. Dabei kann man sehen, dass der normierte Restfehler ab der 7. bis 8. iterativen Stufe

kaum noch verbessert wird. Das bedeutet, dass der Kombinationsalgorithmus an dieser Stelle schon fast konvergiert. Der normierte Restfehler ($1.55 \cdot 10^{-2}$) ist höher als der normierte Restfehler, der in Kapitel 6 bei der Simulation des Algorithmus erreicht wurde. Der wichtigste Grund dafür ist, dass das Wiener-Modell nur eine beschränkte Beschreibungsvollständigkeit für das allgemeine nichtlineare System hat.

Die Abb. 7.6 veranschaulicht die Kennlinie der Gleichung (7.16).

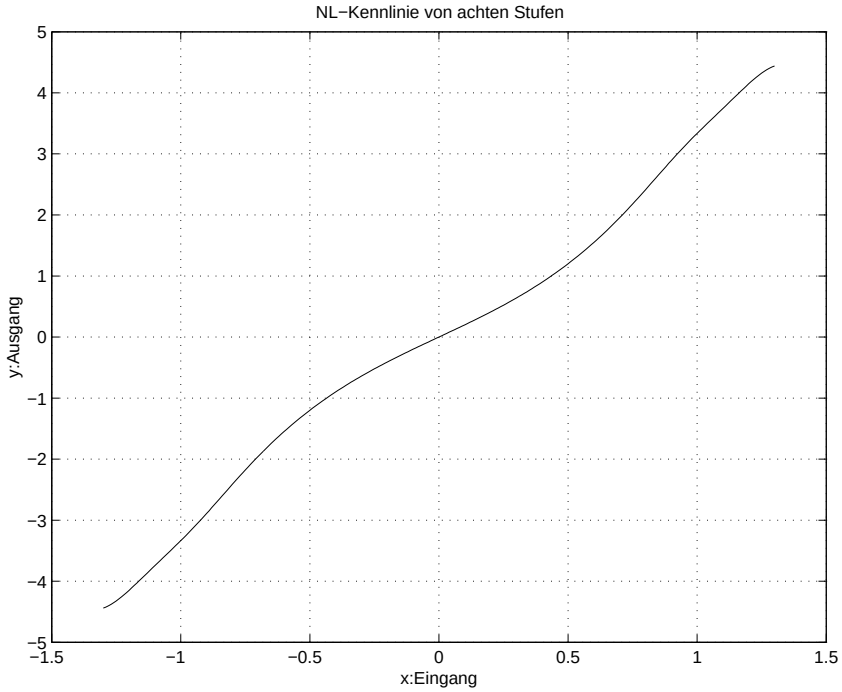


Abbildung 7.6: Identifizierte Kennlinie des NL-Anteils beim Wiener-Modell für Beispiel 1

Die Abb. 7.7 zeigt einen Ausschnitt des Ausgangs des Ziel-Systems und den Simulationsverlauf des mit dem Wiener-Modell modellierten Systems.

Der normierte Restfehler zwischen dem Ziel-Systemausgang und dem Modellausgang gemäß Gl. (7.16) ist $1.55 \cdot 10^{-2}$. Im Gegensatz dazu beträgt der normierte Restfehler mit einem linearen Modell $9.0 \cdot 10^{-2}$. Die beiden Modelle besitzen aber einen vergleichbaren Rechenaufwand. Der Gewinnfaktor der Genauigkeit des Wiener-Modells gegenüber dem linearen Modell ist:

$$\frac{9.0 \cdot 10^{-2}}{1.55 \cdot 10^{-2}} = 5.8$$

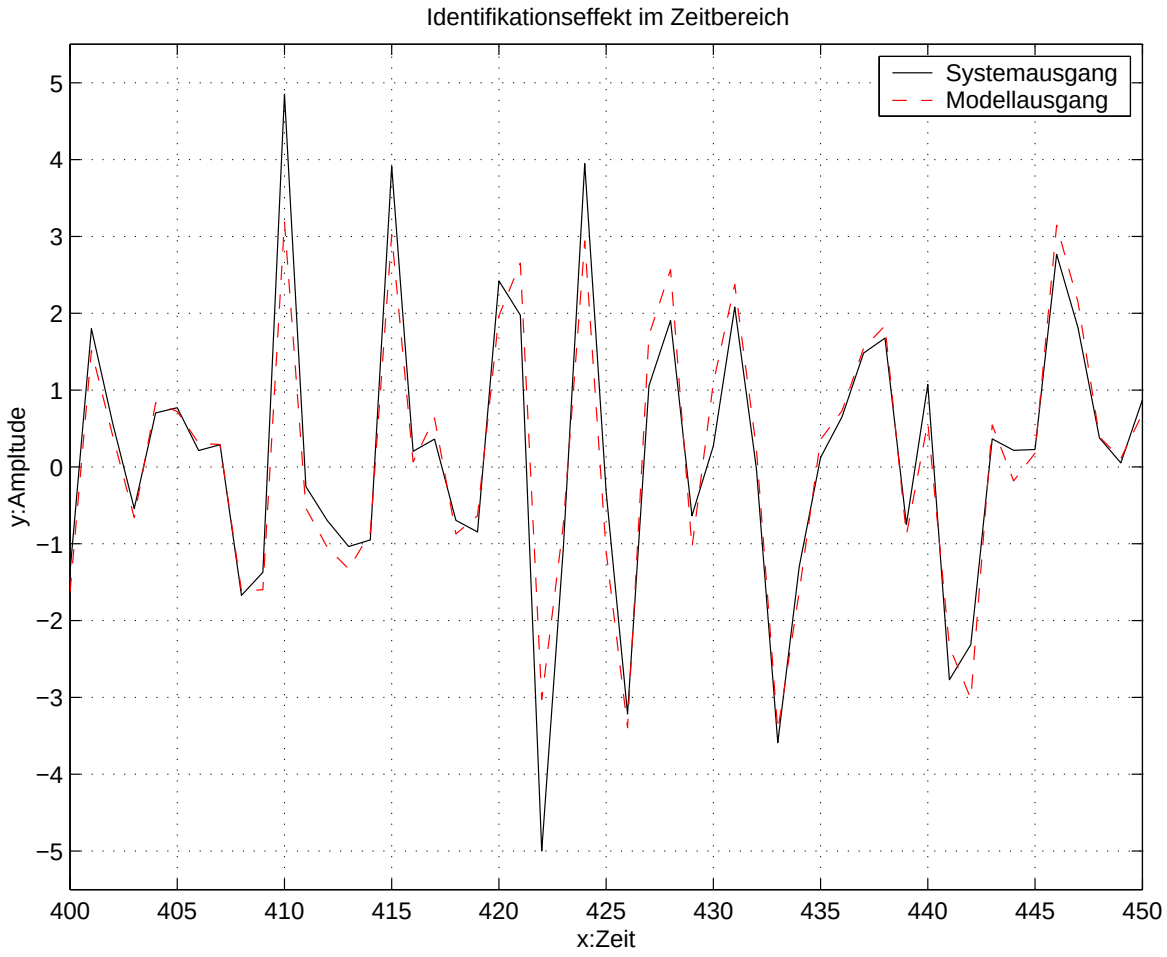


Abbildung 7.7: Systemausgang und Simulationsverlauf des Wiener-Modells für Beispiel 1

7.2.2 Beispiel 2

Ziel-System und Trainingsdaten

Das nichtlineare Ziel-System veranschaulicht die Abb. 7.8. Dabei zeigen die Gleichungen (7.17) und (7.18) die entsprechenden $\mathrm{H}_1(s)$ und $\mathrm{H}_2(s)$ aus Abb. 7.8. Die Gleichungen (7.19) und (7.20) zeigen die entsprechenden Kennlinienfunktionen von NL_1 und NL_2 aus Abb. 7.8.

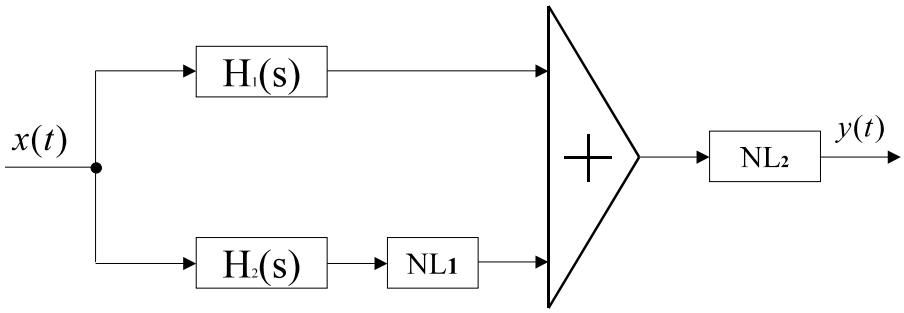


Abbildung 7.8: Ziel-System 2 für Beispiel 2

$$\mathrm{H}_1(s) = \frac{0.42s^2 + 0.49s + 0.56}{0.03s^2 + 0.4s + 1} \tag{7.17}$$

$$\mathrm{H}_2(s) = \frac{0.5s + 0.6}{0.1s^2 + 0.1s + 1} \tag{7.18}$$

$$f_{\mathrm{NL}_1}(x) = \frac{1 - e^{-0.8x}}{1 + e^{-0.8x}} \tag{7.19}$$

$$f_{\mathrm{NL}_2}(x) = \frac{1 - e^{-1.5x}}{1 + e^{-1.5x}} \tag{7.20}$$

Das Eingangssignal x ist ein weißes Rauschen, dessen Augenblickswerte auf [-1.75, +1.75] beschränkt sind. Das Ausgangssignal liegt ca. auf (-1.0, +1.0), weil NL_2 ein Begrenzer ist. Zur Identifikation des Ziel-Systems werden 3000 Trainingsdaten verwendet. Die anderen Bedingungen und Einstellungen des GP-Laufes sind identisch mit Beispiel 1.

Konvergenzprozess

Die Abb. 7.9 zeigt den Verlauf des normierten Restfehlers zwischen dem Systemausgang und dem Modellausgang abhängig von der Anzahl der iterativen Stufen. Dabei läuft der Kombinationsalgorithmus nur bis zur 4. Stufe, danach ändert sich der Restfehler kaum noch. Der normierte Restfehler bei den Trainingsdaten ist $1.1 \cdot 10^{-3}$.

Simulationsergebnisse

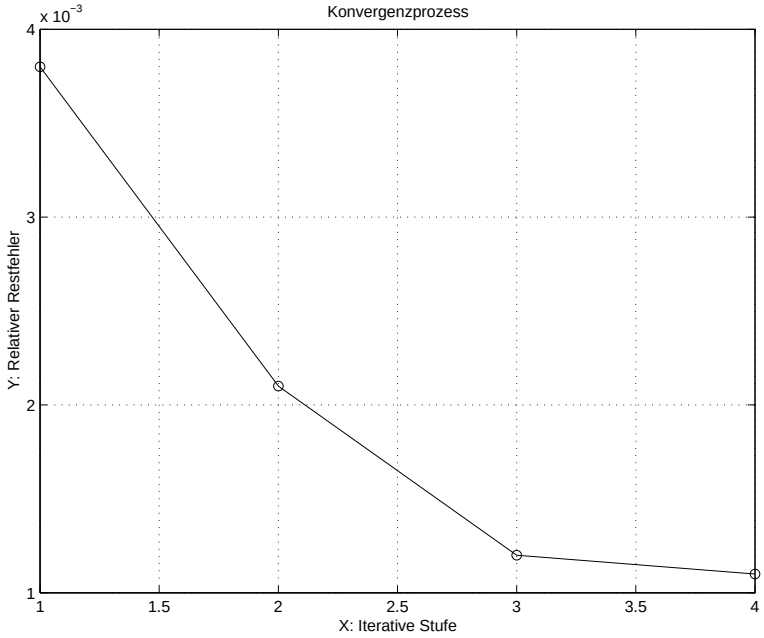


Abbildung 7.9: Konvergenzverlauf des Restfehlers für Beispiel 2

Durch den Kombinationsalgorithmus (nur bis zur 4. iterativen Stufen) wird das nichtlineare Ziel-System mit einem Wiener-Modell nachgebildet.

Die Gleichung (7.21) zeigt den endgültigen LZI-Anteil des Wiener-Modells, und die Gleichung (7.22) zeigt den endgültigen statischen NL-Anteil des Wiener-Modells.

$$\widehat{\mathrm{H}}(s) = \frac{0.009577s^4 - 0.04666s^3 + 0.1955s^2 + 0.3329s + 0.4375}{0.216s + 1} \tag{7.21}$$

$$\widehat{f}_{\mathrm{NL}}(x) = 1.0621sin(1.2531sin(1.0870sin(0.9360x))) \tag{7.22}$$

Der normierte Restfehler zwischen dem Ausgang des Ziel-Systems und dem Ausgang des Wiener-Modells ist bei den Trainingsdaten $1.1 \cdot 10^{-3}$.

Bei der Simulation eines nichtlinearen Systems ist normalerweise das Modell nur für den Eingangswertebereich, der kleiner oder gleich dem Eingangswertebereich des Trainingssignals ist, gültig. In der Testphase wird das Modell mit 100000 zufällig generierten Testdaten (weißes Rauschen) und unterschiedlichen Wertebereichen des Eingangssignals getestet. Das Test-Ergebnis wird in Abb. 7.10 gezeigt.

Aus dieser Abbildung (Abb. 7.10) kann man sehen:

1. Das Modell passt auch im Eingangswertebereich [-3, +3], obwohl der Eingangswertebereich der Trainingsdaten auf [-1.75, +1.75] beschränkt war. Das zeigt eine häufig anzutreffende Erscheinung, nämlich dass die GP aus unvollständi-

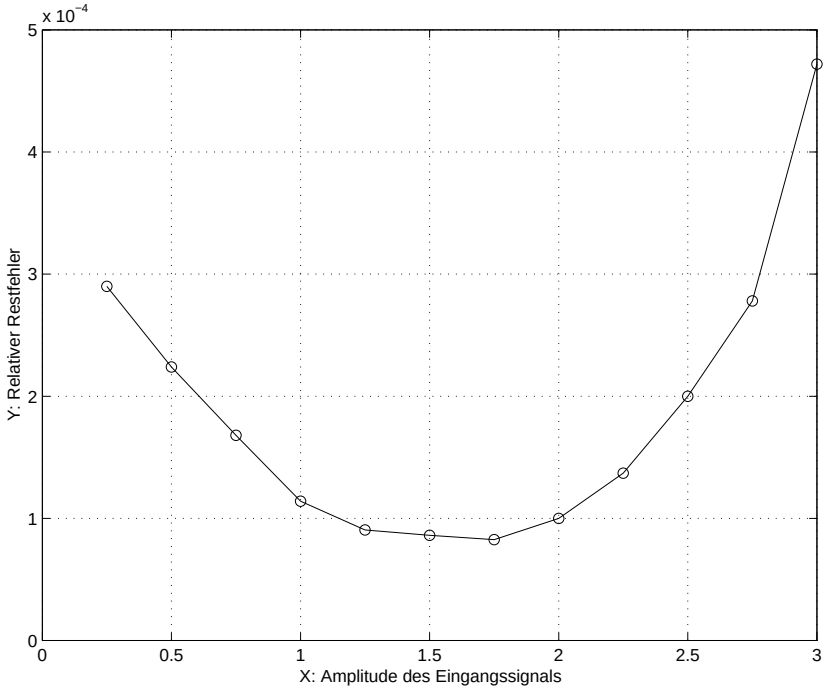


Abbildung 7.10: Test verschiedener Eingangswertebereiche für Beispiel 2

gen Trainingsdaten das richtige Modell des Ziel-Systems finden kann. Das gilt aber nicht immer.

2. Der kleinste normierte Restfehler wurde gemessen, wenn das Eingangssignal in dem Bereich [-1.75, +1.75] liegt. Der kleinste Restfehler ist $8.26 \cdot 10^{-5}$. Er ist sogar kleiner als der Restfehler bei den Trainingsdaten $1.1 \cdot 10^{-3}$. Der wichtigste Grund hierfür ist, dass die Testphase viel länger als die Trainingsphase ist.

Wenn das Ziel-System nur mit einem linearen Modell nachgebildet wird, ist der normierte Restfehler $2.3 \cdot 10^{-2}$. Der Gewinn der Genauigkeit gegenüber dem linearen Modell ist:

$$\frac{2.3 \cdot 10^{-2}}{8.26 \cdot 10^{-5}} = 278$$

Das Ergebnis ist viel besser als im Beispiel 1. Der Grund liegt darin, dass das Ziel-System in diesem Beispiel mit einem Wiener-Modell ziemlich genau nachgebildet werden kann.

Die Abb. 7.11 zeigt den Systemausgang und den simulierten Modellausgang.

Die Abbildung 7.12 veranschaulicht die Kennlinien von NL_1, NL_2, und NL-Wiener-Modell.

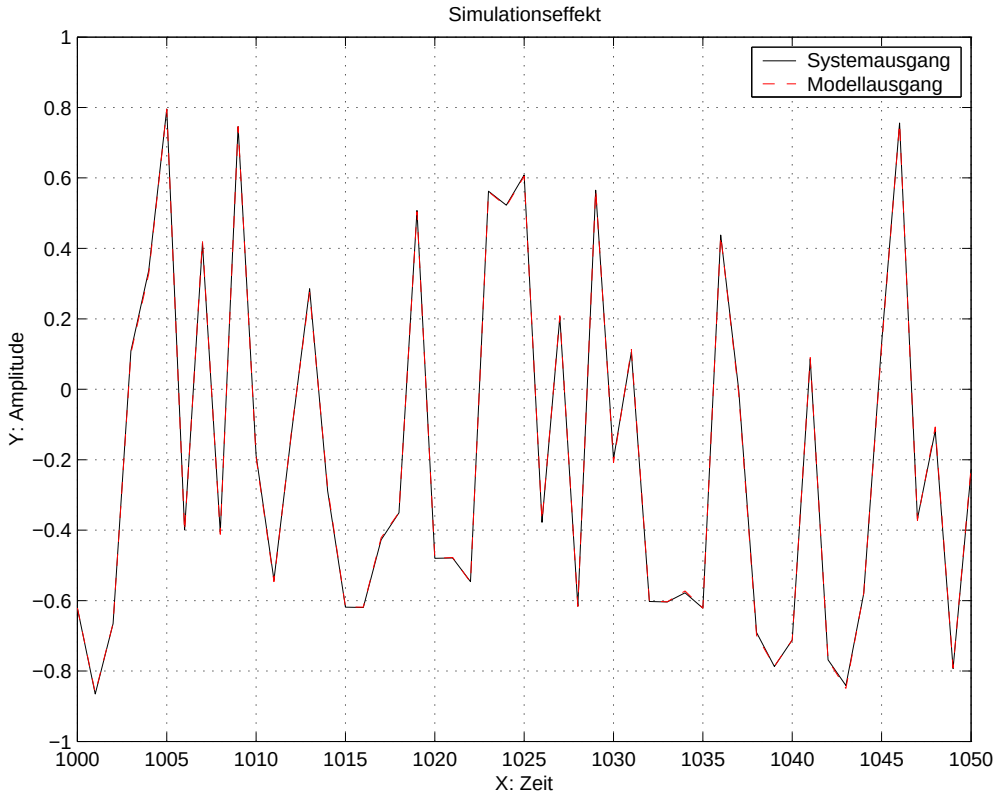


Abbildung 7.11: Systemausgang und Simulationsverlauf des Modells für Beispiel 2

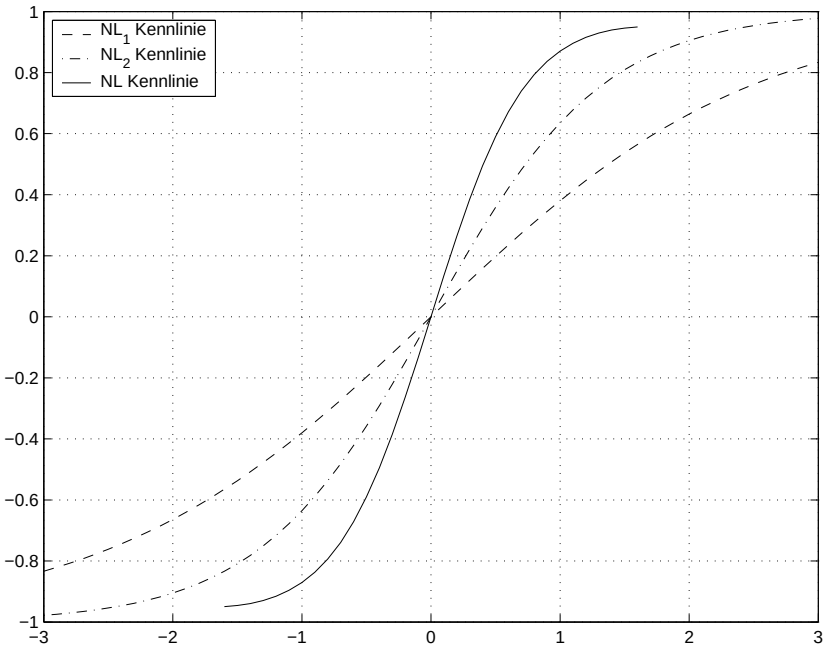


Abbildung 7.12: Kennlinien für Beispiel 2: NL_1, NL_2 und NL-Wiener-Modell

7.2.3 Beispiel 3

Ziel-System und Trainingsdaten

Das nichtlineare Ziel-System wird in der Abb. 7.13 veranschaulicht. Dabei werden die Übertragungsfunktionen der Systeme LZI_1 bzw. LZI_2 in Gl. (7.23) bzw. Gl. (7.24) dargestellt. Die Gl. (7.25) zeigt die entsprechende Kennlinienfunktion von NL_0.

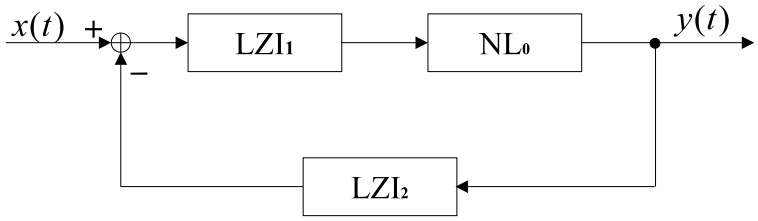


Abbildung 7.13: Ziel-System 3 für Beispiel 3

$$\mathrm{H}_1(s) = \frac{0.66s^2 + 0.77s + 0.88}{0.03s^2 + 0.4s + 1} \tag{7.23}$$

$$\mathrm{H}_2(s) = \frac{0.075s + 0.09}{0.1s^2 + 0.1s + 1} \tag{7.24}$$

$$f_{\mathrm{NL}_0}(x) = \frac{1 - e^{-2x}}{1 + e^{-2x}} \tag{7.25}$$

Das Eingangssignal x ist ein weißes Rauschen, das auf den Bereich [-1.75, +1.75] beschränkt ist. Die Ausgangssignale liegen im Bereich (-1.0, +1.0), weil NL_0 eine Begrenzungskennlinie ist. Zur Identifikation des Ziel-Systems werden 3000 Trainingsdaten verwendet. Die anderen Bedingungen und Einstellungen des GP-Laufes sind identisch mit Beispiel 1.

Simulationsergebnisse

Durch den Kombinationsalgorithmus (bis zur 5. iterativen Stufen) hat die GP ein Wiener-Modell für das Ziel-System identifiziert.

Die Gleichung (7.26) zeigt den identifizierten LZI-Anteil, und die Gleichung (7.27) zeigt den identifizierten gedächtnislosen NL-Anteil.

$$\widehat{\mathrm{H}}(s) = \frac{-0.01s^5 + 0.07251s^4 + 0.001867s^3 + 0.3407s^2 + 0.2713s + 0.4472}{0.07724s^3 + 0.232s^2 + 0.1845s + 1} \tag{7.26}$$

$$\widehat{f}_{\mathrm{NL}}(x) = 1.0556586sin(1.6150585sin(sin(1.150875x))) \tag{7.27}$$

Der normierte Restfehler zwischen dem Ausgang des Ziel-Systems und dem Ausgang des Wiener-Modells ist bei den Trainingsdaten $8.1 \cdot 10^{-4}$.

Mit 100000 zufällig generierten Testdaten (weißes Rauschen) wird das Modell für unterschiedliche Wertebereiche des Eingangssignals getestet. Das Test-Ergebnis ist in Abb. 7.14 dargestellt.

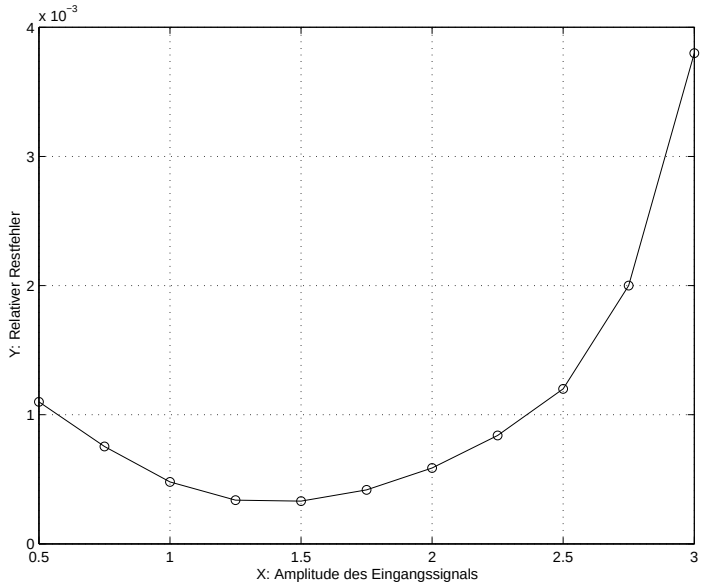


Abbildung 7.14: Test verschiedener Eingangswertebereiche für Beispiel 3

Aus dieser Abb. 7.14 kann man ersehen:

1. Das Modell passt gut für den Eingangswertebereich [-3, +3], obwohl der Eingangswertebereich der Trainingsdaten auf [-1.75, +1.75] beschränkt war.

2. Der kleinste normierte Restfehler wurde gemessen, wenn das Eingangssignal auf den Bereich [-1.5, +1.5] beschränkt ist. Er liegt bei $3.31 \cdot 10^{-4}$. Wenn das Eingangssignal auf den Bereich [-1.75, +1.75] erweitert wird, steigt der normierte Restfehler auf $4.18 \cdot 10^{-4}$ leicht an.

Wenn das Ziel-System nur mit einem linearen Modell nachgebildet wird, ist der normierte Restfehler $5.5 \cdot 10^{-2}$. Daher ergibt sich für den Gewinnfaktor der Genauigkeit des Wiener-Modells gegenüber dem linearen Modell:

$$\frac{5.5 \cdot 10^{-2}}{4.18 \cdot 10^{-4}} = 132$$

Das Ergebnis ist viel besser als im Beispiel 1 aber schlechter als im Beispiel 2.
Die Abb. 7.15 zeigt den Systemausgang und den simulierten Modellausgang.
Die Abb. 7.16 veranschaulicht die Kennlinien von NL_0 und NL-Wiener-Modell.

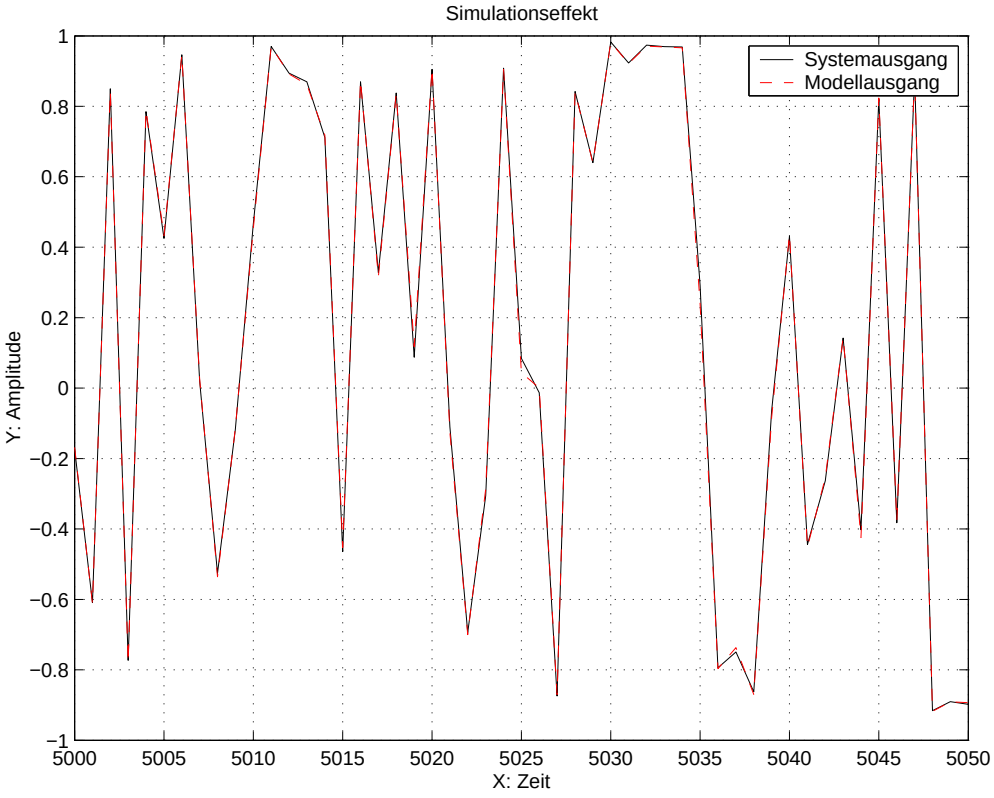


Abbildung 7.15: Systemausgang und Simulationsverlauf des Modells für Beispiel 3

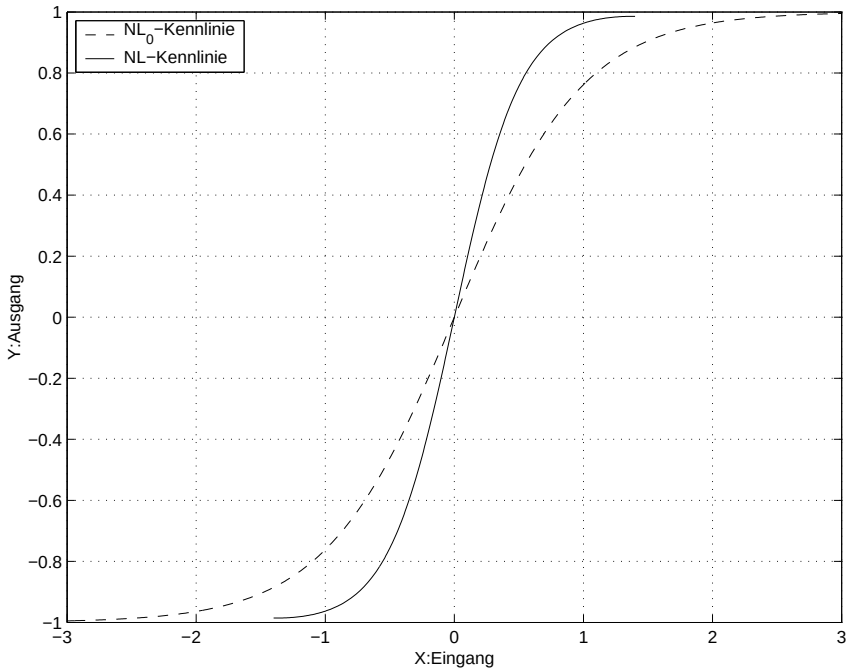


Abbildung 7.16: Kennlinien für Beispiel 3: NL_0, NL-Wiener-Modell.

7.2.4 Beispiel 4

Ziel-System und Trainingsdaten

Das Blockschaltbild des Ziel-Systems ist gleich wie beim Beispiel 2 (Abb. 7.8). Dabei werden die Übertragungsfunktionen der LZI_1- bzw. LZI_2-Systeme gemäß Gl. (7.28) bzw. Gl.(7.29) verändert. Die Gleichungen (7.30) und (7.31) zeigen die entsprechenden Kennlinienfunktionen von NL_1 und NL_2. Die Ausgangssignalleistung von NL_1 ist etwa 10% von der des LZI_1-Systems.

$$\mathrm{H}_1(s) = \frac{0.42s^2 + 0.49s + 0.56}{0.03s^2 + 0.4s + 1} \tag{7.28}$$

$$\mathrm{H}_2(s) = \frac{0.3375s + 0.405}{0.1s^2 + 0.1s + 1} \tag{7.29}$$

$$f_{\mathrm{NL}_1}(x) = 0.6x^3 + 0.03x^5 \tag{7.30}$$

$$f_{\mathrm{NL}_2}(x) = \frac{1 - e^{-2.5x}}{1 + e^{-2.5x}} \tag{7.31}$$

Das Eingangssignal x ist weißes Rauschen und es ist auf den Bereich [-1.75, +1.75] beschränkt. Die Ausgangssignale liegen im Bereich (-1.0, +1.0), weil NL_2 eine Begrenzungskennlinie ist. Zur Identifikation des Ziel-Systems werden 3000 Trainingsdaten verwendet. Die anderen Bedingungen und Einstellungen des GP-Laufes sind identisch mit Beispiel 1.

Simulationsergebnisse

Durch den Kombinationsalgorithmus (bis zur 5. iterativen Stufen) hat die GP ein Wiener-Modell für das Ziel-System identifiziert.

Die Gleichung (7.32) zeigt den identifizierten LZI-Anteil, und die Gleichung (7.33) zeigt den identifizierten gedächtnislosen NL-Anteil.

$$\widehat{\mathrm{H}}(s) = \frac{0.4544 + 0.3760s + 0.2739s^2 - 0.0296s^3 - 0.0004s^4}{1 + 0.3150s} \tag{7.32}$$

$$\widehat{f}_{\mathrm{NL}}(x) = 0.9890sin(1.4190sin(1.2075x)) \tag{7.33}$$

Der normierte Restfehler zwischen dem Ausgang des Ziel-Systems und dem Ausgang des Modells ist $3.7 \cdot 10^{-4}$ bei den Trainingsdaten.

Mit 100000 zufällig generierten Testdaten (weißes Rauschen) wird das Modell in unterschiedlichen Wertebereichen des Eingangssignals getestet. Das Test-Ergebnis wird in Abb. 7.17 gezeigt.

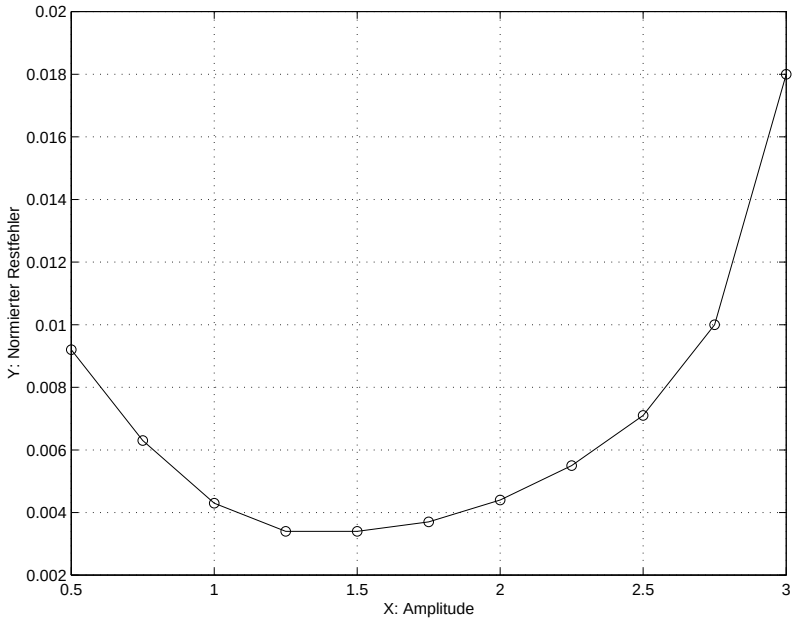


Abbildung 7.17: Test unterschiedlicher Eingangswertebereiche für Beispiel 4

Aus Abb. 7.17 ist zu ersehen:

1. Das Modell passt auch für den Eingangswertebereich [-2.75, +2.75], obwohl der Eingangswertebereich der Trainingsdaten auf [-1.75, +1.75] beschränkt ist. Wenn der Wertebereich des Eingangssignals außerhalb [-2.75, +2.75] liegt, nimmt der normierte Restfehler stark zu.

2. Der kleinste normierte Restfehler wird gemessen, wenn das Eingangssignal im Bereich [-1.25, +1.25] liegt. Der kleinste normierte Restfehler ist $3.4 \cdot 10^{-3}$. Wenn das Eingangssignal im Bereich [-1.75, +1.75] liegt, steigt der normierte Restfehler leicht auf $3.7 \cdot 10^{-3}$ an.

Wenn das Ziel-System nur mit einem linearen Modell nachgebildet wird, ist der normierte Restfehler $4.9 \cdot 10^{-2}$. Der Gewinnfaktor der Genauigkeit des Wiener-Modells gegenüber dem linearen Modell ist daher:

$$\frac{4.9 \cdot 10^{-2}}{3.7 \cdot 10^{-3}} = 13$$

Das Ergebnis ist besser als im Beispiel 1 aber wesentlich schlechter als in den Beispielen 2 und 3.

Die Abb. 7.18 zeigt den Systemausgang und den simulierten Modellausgang.

Die Abb. 7.19 veranschaulicht die Kennlinien von NL_1, NL_2 und NL-Wiener-Modell.

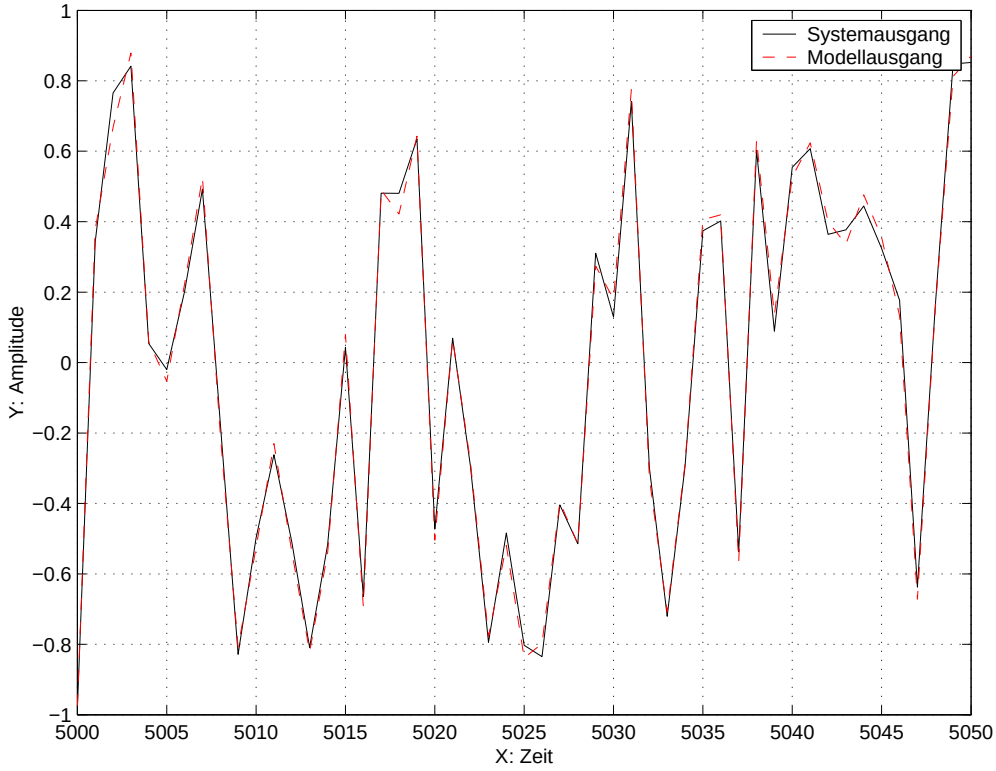


Abbildung 7.18: Systemausgang und Simulationsverlauf des Modells für Beispiel 4

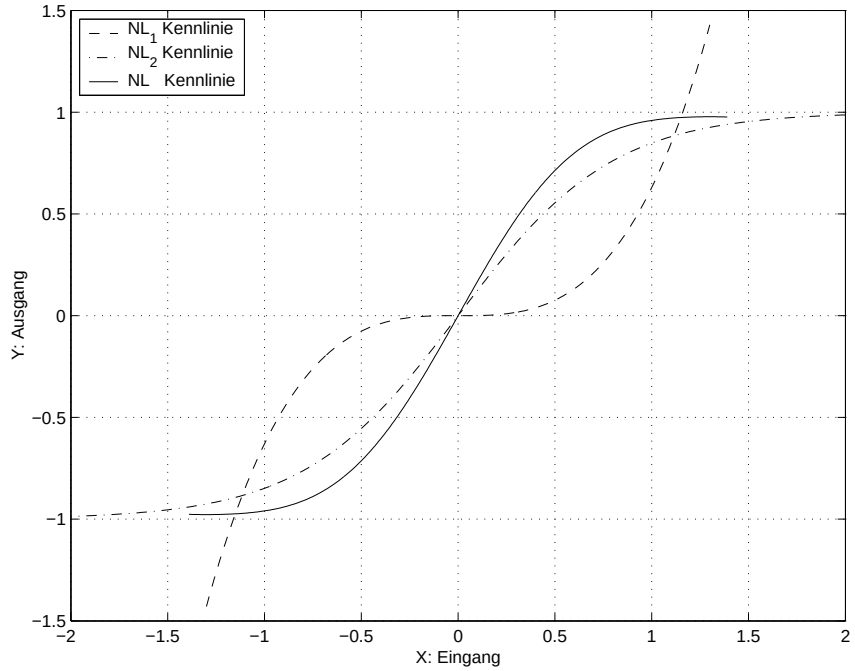


Abbildung 7.19: Kennlinien für Beispiel 4: NL_1, NL_2 und NL-Wiener-Modell.

Die Tabelle 7.3 zeigt einen Überblick über die Ergebnisse der Beispiele.

	Lautsprecher	Beisp. 1	Beisp. 2	Beisp. 3	Beisp. 4
ε	$1.7 \cdot 10^{-3}$	$1.55 \cdot 10^{-2}$	$8.26 \cdot 10^{-5}$	$4.18 \cdot 10^{-4}$	$3.7 \cdot 10^{-3}$
Gewinnfaktor	3.3	5.8	278	132	13
Konvergenz-Geschwindigkeit	–	8	4	5	5
Wiener-Modell-Beschreibbarkeit	–	nicht gut	sehr gut	sehr gut	gut
Verfahren	rekur. GP	KA	KA	KA	KA
Ziel-System	Lautsprecher	Abb.7.4	Abb.7.8	Abb.7.13	Abb.7.8

Tabelle 7.3: Vergleich der dargestellten Beispiele

7.3 Beschreibungsgenauigkeit des Wiener-Modells für NL-Systeme

Die Beschreibungsgenauigkeit des Wiener-Modells für ein konkretes nichtlineares System wird durch den kleinsten normierten Restfehler δ , wie er in der Gleichung(7.34) dargestellt ist, definiert.

$$\delta = \min_{\mathrm{W} \subset \Omega} \frac{1}{\sum_{i=0}^{\mathrm{N}} y^2(i)} \sum_{i=0}^{\mathrm{N}} [y(i) - \widehat{y}_{\mathrm{W}}(i)]^2 \tag{7.34}$$

Dabei gilt:

$y(i)$ ist das Ausgangssignal des nichtlinearen Ziel-Systems ohne Rauschen.
$\widehat{y}_{\mathrm{W}}(i)$ ist das Ausgangssignal des Wiener-Modells ohne Rauschen.
Ω ist die Menge aller Wiener-Modelle mit beliebigen LZI- und NL-Anteilen.

Alle nichtlinearen Ziel-Systeme kann man auf die drei folgenden Teilmengen aufteilen:

1. $\delta = 0$. In diesem Fall spricht man davon, dass das Wiener-Modell das Ziel-System exakt beschreiben kann. Das ist ein Sonderfall, d.h. die physikalische Struktur des Ziel-Systems ist wirklich ein Wiener-Modell.

2. $0 < \delta < \varepsilon$. Dabei ist ε ein vorgegebener Wert für die Modellierungsgenauigkeit. In diesem Fall spricht man davon, dass das Wiener-Modell das Ziel-System gut annäheren kann.

3. $\delta > \varepsilon$. In diesem Fall spricht man davon, dass das Wiener-Modell das Ziel-System nicht beschreiben kann.

Offensichtlich bildet δ die Obergrenze der Genauigkeit des Kombinationsalgorithmus für ein beliebiges nichtlineares Ziel-System. Das heißt, dass die Genauigkeit des vom Kombinationsalgorithmus identifizierten Modells die Beschreibungsgenauigkeit des Wiener-Modells nicht überschreiten kann.

Von den Testergebnissen in der Tabelle 7.3 können wir die folgenden Schlussfolgerungen ziehen:

1. Der Kombinationsalgorithmus ist geeignet für die Systemidentifikation, wenn das zu identifizierende System mit einem Wiener-Modell exakt oder hinreichend gut beschrieben werden kann.

2. Die Genauigkeit des Kombinationsalgorithmus ist von der Beschreibungsgenauigkeit des Wiener-Modells stark abhängig.

3. Die Genauigkeit des Kombinationsalgorithmus ist um einen wesentlichen Faktor (5–300) besser als die Genauigkeit des rein linearen Modells mit vergleichbarem Rechenaufwand.

4. Der Kombinationsalgorithmus kann sehr hohe Genauigkeiten erreichen, wenn der NL-Systemanteil des Ziel-Systems ein Sättigungssystem ist. Er bildet daher eine wichtige Ergänzung für konventionelle Volterra-Modelle. (Sättigungssysteme kann man durch ein Volterra-Modell nicht gut beschreiben.)

5. Wenn der Kombinationsalgorithmus ein kompliziertes Modell für den statischen NL-Anteil findet, kann die GP durch eine Post-Verarbeitung (auch eine Systemidentifikation) die NL-Kennlinie auch durch einen Quotienten aus zwei Polynomen darstellen, ohne dass ein wesentlicher Verlust für die gesamte Genauigkeit entsteht.

6. Der Kombinationsalgorithmus ist nicht geeignet für nichtlineare Ziel-Systeme, deren Nichtlinearität zwar schwach ist aber deren δ relativ groß ist. Der Lautsprecher ist ein Beispiel dafür. In diesem Fall ist das rekursive GP-Verfahren eine gute Ergänzung. Die Lösung aus dem rekursiven GP-Verfahren ist aber normalerweise kompliziert.

Kapitel 8

Ausblick

Die Behandlung nichtlinearer Systeme ist zunehmend weiter ins wissenschaftliche bzw. praktische Interesse gerückt. Aber der Übergang von linearer Systemtheorie zur nichtlinearen Systemtheorie ist immer schwierig. Theoretisch braucht die GP keine Vorkenntnisse über das behandelte nichtlineare System sondern nur Tranings-daten aus dem behandelten System, um automatisch eine Lösung bieten zu können. Dies bildet ein *riesiges Potential* der GP in der Behandlung der *unbekannten oder teilweise unbekannten* nichtlinearen Systeme.

Im Wesentlichen hat diese Arbeit die Identifikations- bzw. Kompensationsprobleme der mit einem Wiener-Modell hinreichend gut beschreibbaren Systeme gelöst. Mit einem Quotienten aus zwei Polynomen kann der NL-Anteil vereinfacht werden. Jedoch ist der Quotient aus zwei Polynomen bestimmt nicht die eleganteste (einfachste) Lösung für den NL-Anteil. Beispielsweise ist die Funktion $asin(bx)$ viel eleganter als ihr äquivalenter Quotient aus zwei Polynomen.

Theoretisch hat die GP das Potential, eine elegante Lösung zu finden. Allerdings ist diese Fähigkeit wegen der Erscheinung von Introns beschränkt. In Kapitel 4 haben wir eine Methode, die Post-Verarbeitung, vorgestellt, durch die man Introns erkennen und ausschneiden kann. Die Post-Verarbeitung ist in dieser Methode die Modifikation der Konstanten-Optimierung. Es lohnt sich in den praktischen Anwendungen, die Vereinfachung der GP-Lösung weiter zu untersuchen.

Eine weitere Untersuchungsmöglichkeit ist es, mit einer Kette von LZI- und NL-Anteilen den Kombinationsalgorithmus zu erweitern, wie es in Abb. 8.1 gezeigt wird. Mindestens sollte der Kombinationsalgorithmus auf das Wiener-Hammerstein-Modell (L-N-L Modell) oder N-L-N Modell erweiterbar sein. Wenn die Kette jedoch noch länger wird, ist wahrscheinlich eine genauere Untersuchung der Konvergenz-problematik notwendig.

Eine weitere Anwendungsmöglichkeit für diesen Kombinationsalgorithmus sind Anwendungen, in denen die nutzbare Nichtlinearität des Ziel-Systems beibehalten wird und die unbrauchbaren Nichtlinearitäten des Ziel-Systems kompensiert werden.

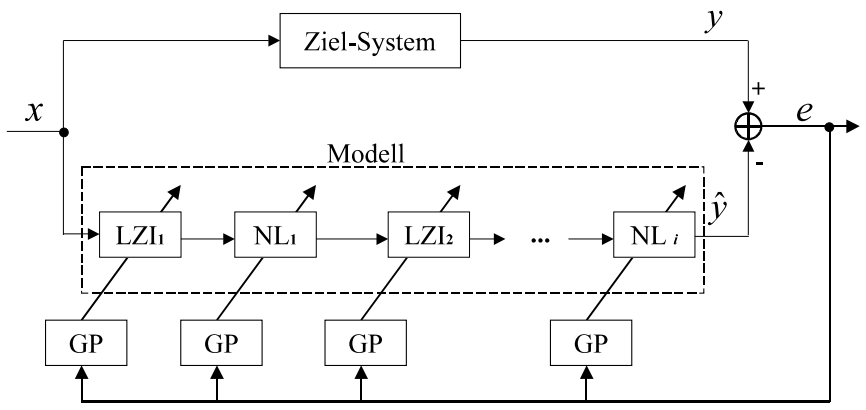


Abbildung 8.1: Erweiterung des Kombinationsalgorithmus

Die Abb. 8.2 zeigt ein Beispiel, in dem die Nichtlinearität der 2. Ordnung beibehalten und die Nichtlinearitäten höherer Ordnung kompensiert werden.

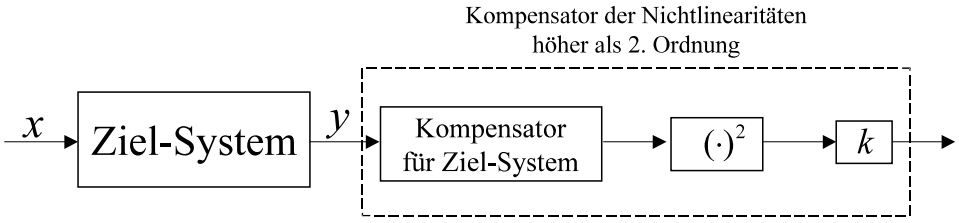


Abbildung 8.2: Eine weitere Anwendungsmöglichkeit des Kombinationsalgorithmus

Am Ende dieser Arbeit wird eine Idee für das Forschungsgebiet der Systemidentifikation vorgestellt, die GP und die neuronalen Netze (NN) zu kombinieren. Die GP besitzt eine ziemlich starke Struktursuchfähigkeit. Bei vorgegebener NN-Struktur haben die Trainingsalgorithmen von NN sehr starke Lernfähigkeit für die numerischen Verbindungs-Gewichte in NN zwischen den einzelnen Neuronen. Eigentlich bildet das Gewichtentraining die Kernarbeit verschiedener Trainingsalgorithmen in NN. Die Schwäche von NN ist es, dass sie nicht automatisch vershiedene neuronale Netze (Strukturen) erzeugen können. Mit Hilfe der GP kann diese Schwäche überwunden werden. Abb. 8.3 veranschaulicht einen Implementierungsblock.

Dabei erzeugt die GP zahlreiche unterschiedliche neuronale Netze, von denen jedes einem Idividuum in der GP-Population entspricht. Danach wird jedes NN mit Hilfe des Trainingsalgorithmus optimiert. Es ist möglich, dass die GP eine bessere NN-Struktur findet.

In diesem Fall ist es sehr wahrscheinlich, dass die Grafikbasierte GP besser geeignet ist als die Baumbasierte GP.

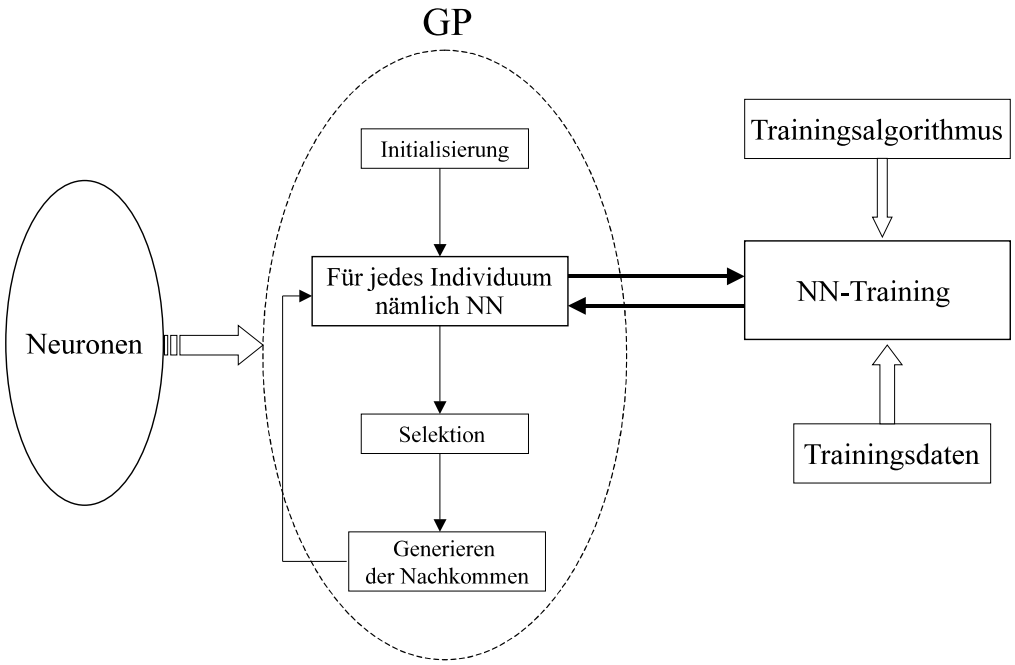


Abbildung 8.3: Kombination zwischen der GP und dem NN

A. Verzeichnis der Abkürzungen und Symbole

A.1. Abkürzungen

AA	Amplitudengangsanpassung
ADF	Automatisch definierte Funktion
AI	Abschnittsweise Identifikation
AK	Abschnittsweise Kompensation
AO	Abschnittsweise Optimierung
AR	AutoRegressive
ARMA	AutoRegressive Moving Average
ARMAX	AutoRegressive Moving Average with Exogenous Input
ARX	AutoRegressive with Exogenous Input
BJ	Box-Jenkins
CTF-GP	Genetische Programmierung mit kontextfreier Grammatik.
EA	Evolutionärer Algorithmus
EP	Evolutionäre Programmierung
ES	Evolutionäre Strategie
FIR	Finite Impulse Response Filter
GA	Genetischer Algorithmus
GB	Ganzer Bereich für die Optimierung
GP	Genetische Programmierung
IIR	Infinite Impulse Response Filter
KGG	Kompensationsgleichungsgruppe
KO	Konstanten-Optimierung
LZI	Lineare zeitinvariante Systeme
LZV	Lineare zeitvariante Systeme
MIMO	Multiple Input Multiple Output
MISO	Multiple Input Single Output
NARX	Nichtlineare ARX
NARMAX	Nichtlineare ARMAX
NFIR	Nichtlineare FIR

NIIR	Nichtlineare IIR
NBJ	Nichtlineare BJ
NOE	Nichtlineare OE
NN	Neuronales Netz
NL	Gedächtnislose zeitinvariante NichtLineare Systeme
NLV	Gedächtnislose zeitvariante NichtLineare Systeme
OE	Output Error
SISO	Single Input Single Output
SMOG	Struktured MOdel Generator
SNR	Signal-Rausch-Verhältnis
TB	Teilbereich für die Optimierung

A.2. Symbole und ihre Bedeutung

$a(i,k)$:	Adjustierte Fitness.
$r(i,k)$:	Rohfitness.
$r_{\mathrm{LZI},j}$:	Rohfitness für die Identifikation des LZI-Systemanteils in der $j-ten$ iterativen Stufe.
$r_{\mathrm{NL},j}$:	Rohfitness für die Identifikation des NL-Systemanteils in der $j-ten$ iterativen Stufe.
r_{NL}^{-1}, j :	Rohfitness für die Identifikation der Kompensationsfunktion des NL-Systemanteils in der $j-ten$ iterativen Stufe.
$s(i,k)$:	Standardisierte Fitness
$n(i,k)$:	Normalisierte Fitness
x :	Eingangssignal des Ziel-Systems
y :	Ausgangssignal des Ziel-Systems
$\widehat{y}$:	Ausgangssignal des Modells
z_L :	Ausgangssignal des LZI-Anteils
z :	Ausgangssignal des Kompensators
e :	Fehlersignal
e_y :	Fehlersignal des Modells
e_z :	Fehlersignal des Kompensators
D :	Tiefe des Individuums
$\mathrm{D}_{createt}$:	Maximale Tiefe des Individuums
$\mathrm{D}_{initial}$:	Maximale Tiefe des Individuums bei der Initialisierung
M :	Populationsgröße
G :	Anzahl der Generationen
G_{max} :	Maximale Anzahl der Generationen
p_c :	Kreuzungswahrscheinlichkeit
p_m :	Mutationswahrscheinlichkeit
p_r :	Reproduktionswahrscheinlichkeit
p_{ep} :	Selektionswahrscheinlichkeit im inneren Knoten
p_{ip} :	Selektionswahrscheinlichkeit im Endknoten
S :	Das Startsymbol der kontextfreien Grammatik

z_L: ideales LZI-Ausgangssignal des Ziel-Systems.

$\widehat{z}_L$: geschätzter Wert von z_L in der Endstufe.

$\widehat{z}_{1,i}$: geschätztes LZI-Ausgangssignal der $i-ten$ iterativen Stufe.

$\widehat{z}_{2,i}$: kompensierter Ausgang der $i-ten$ iterativen Stufe.

$\widehat{z}$: kompensierter Ausgang in der Endstufe.

$f_{\mathrm{NL}}(\cdot)$: ideale NL-Kennlinienfunktion des NL-Systemanteils.

$\widehat{f}_{\mathrm{NL}}(\cdot)$: geschätzte Funktion von $f_{\mathrm{NL}}(\cdot)$.

$\widehat{f}^{-1}_{\mathrm{NL}}(\cdot)$: ideale Inverse Funktion von $\widehat{f}_{\mathrm{NL}}(\cdot)$.

$\widehat{\widehat{f}^{-1}}_{\mathrm{NL}}(\cdot)$: geschätzte Inverse Funktion von $\widehat{f}_{\mathrm{NL}}(\cdot)$.

$\widehat{f}_{\mathrm{NL},i}(\cdot)$: geschätzte NL-Kennlinienfunktion des NL-Systemanteils in der $i-ten$ iterativen Stufe.

$\widehat{f}^{-1}_{\mathrm{NL},i}(\cdot)$: ideale Inverse-Funktion von $\widehat{f}_{\mathrm{NL},i}(\cdot)$.

$\widehat{\widehat{f}^{-1}}_{\mathrm{NL},i}(\cdot)$: geschätzte Inverse-Funktion von $\widehat{f}_{\mathrm{NL},i}(\cdot)$.

y: Ausgangssignal des Ziel-Systems.

$\widehat{y}_i$: Ausgangssignal des gesamten Modells in der $i-ten$ iterativen Stufe

$\widehat{y}$: Ausgangssignal des gesamten Modells in der Endstufe.

y_{NL}: Ausgangssignal des NL-Anteils gegenüber dem Anregungssignal x_{NL}.

$y_{\mathrm{NL},i}$: Ausgangssignal des NL-Anteils gegenüber dem Anregungssignal $x_{\mathrm{NL},i}$ in der $i-ten$ itrativen Stufe.

$y_{\text{NL-Komp}}$: Ausgangssignal des Kompensators gegenüber dem Anregungssignal x_{NL}.

$y_{\text{NL-Komp},i}$: Ausgangssignal des Kompensators gegenüber dem Anregungssignal $x_{\mathrm{NL},i}$ in der $i-ten$ itrativen Stufe.

$\varepsilon_{L,i}$: relativer Restfehler zwischen idealem LZI-Ausgangssignal und geschätztem LZI-Ausgangssignal der $i-ten$ iterativen Stufe.

$\Delta\varepsilon_{L,i}$: relativer Restfehler zwischen den geschätzten LZI-Ausgangssignalen der $(i-1)-ten$ und der $i-ten$ iterativen Stufe.

$\varepsilon_{N,i}$: relativer Restfehler zwischen der idealen NL-Kennlinie des NL-Systemanteils und der geschätzten NL-Kennlinie des NL-Systemanteils in der $i-ten$ iterativen Stufe.

$\Delta\varepsilon_{N,i}$: relativer Restfehler zwischen den geschätzten NL-Kennlinien des NL-Systemanteils der $(i-1)-ten$ und der $i-ten$ iterativen Stufe.

ε_i: relativer Restfehler zwischen dem Ausgangssignal des Ziel-Systems und dem Ausgangssignal des Modells in der $i-ten$ iterativen Stufe.

ε: relativer Restfehler zwischen dem Ausgangssignal des Ziel-Systems und dem Ausgangssignal des Modells in der Endstufe.

$\varepsilon_{KP,i}$: relativer Restfehler zwischen idealem LZI-Ausgang und kompensiertem Ausgang der $i-ten$ iterativen Stufe

ε_{KP}: relativer Restfehler zwischen idealem LZI-Ausgang und kompensiertem Ausgang in der Endstufe

$\varepsilon_{KL,i}$: relativer Restfehler der Kompensation der identifizierten NL-Kennlinie in der $i-ten$ iterativen Stufe

ε_{KL}: relativer Restfehler der Kompensation der identifizierten NL-Kennlinie in der Endstufe.

R: Rechenaufwand für die Operation M oder P oder exp oder sin oder cos.

Literaturverzeichnis

[Bershad99] N. Bershad, P. Celka, and J.M. vesin, "Stochastic analysis of gradient adaptiv identification of nonlinear systems with memory for Gaussian data and noisy input and output measurements," *IEEE Trans. Signal Processing,* vol. 47, pp.675–689, Mar. 1999

[Bershad00] N. Bershad, P. Celka, and J.M. vesin, "Analysis of stochastic gradient tracking of time-varying polynomial wiener systems," *IEEE Trans.* Vol. 48, pp. 1676–1686, June 2000

[Banzhaf98] W. Banzhaf, P. Nordin, R.E. Keller and F.D. Francone, *"Genetic programming an introduction: On the automatic evolution of computer programs and its applications,"* Morgan Kaufmann Publischers, Inc., San Francisco, California, 1998

[Baeck97] T.Bäck, D.Fogel, Z.Michalewics, Handbook of Evolutionary Computation, Oxford University Press, Oxford 1997.

[Basso02] M.Basso, F.Bencirenni, L.Ciarrê, S.Groppi, and G. Zappa, Experience With NARX Model identification of an industrial Power Plant Gas Turbine. Proceeding of 41st IEEE Conference On Decision and Control, Las Vegas, Nevada USA, December 2002

[Chen90] S.Chen, S.A.Billings, C.F.N.Cowan, and P.M.Grant, Practical Identification of NARMAX Models using radial basis functions, Int. J. Control, 52(6):1327-1350,1990

[Eykhoff77] Pieter Eykhoff, *"System Identification: Parameter and State Estimation,"* Wiley and Sons, New York, 1977

[Frank97] W. Frank, *"Aufwandsarme Modellierung und Kompensation nichtlinearer Systeme auf der Basis von Voltera-Reihen,"* Vorgelegte Dissertation, Fakultät für Elektrotechnik und Informationstechnik, Universität der Bundeswehr München, 1997

[Frank95] W.Frank, Compensation of Linear and Nonlinear sensor Distortions, Kongressband Sensor 95, Nürnberg, Germany, p.889-892,1995

[Fogel00] D.B.Fogel: What is evolutionary computation?, IEEE Spectrum, Vol.37, No.2, p.26-32, Feb., 2000

[Freemann98] J.J. Freemann: A Lineare Representation for GP using Context Free Grammars, Proceedings of the third Annual Conference on Genetic Programming 1998, Morgan Kaufmann, p.72-77, 1998

[Ghogho97] M. Ghogho, S. Meddeb, and J. Bakkoury, "Identification of time-varying nonlinear channels using polynomial filters," in *Proc,IEEE Workshop Nonlinear Signal and Image Processing.* Ann Arbor, MI, 1997

[Giannakis01] G.B. Giannakis, E. Serpedin, *"A bibliography on nonlinear system identification," Signal Processing,* Vol.81, No.3, pp.533–580,2001

[Giancarlo01] Giancarlo Ferrari-Trecate, Giuseppe De Nicolao, NARX Models: Optimal Parametric Approximation of Nonparametric Estimators, Proceeding of the American Control Conference, Arlington, VA June 25-27,2001

[Hunter86] I.W. Hunter, M.J. Korenberg, *"The identification of nonlinear biological systems: Wiener and Hammerstein cascade models,"* Biol. Cybern., vol.55, pp.135–144,1986

[Haykin96] S. Haykin, *"Adaptive filter theory,"* 3rd ed., Prentice-Hall, Englewood Cliffs, New Jersey, 1996

[Hörner96] H.Hörner, Ein Kern für genetisches Programmieren, Diplomarbeit, Wirtschaftsuniversität Wien, 1996

[Iba96] H. Iba, T. Sato, and H. de Garis, *"System identification approach to genetic programming,"* in *Proceedings of the 1th IEEE Conference on Evolutionary Computation.* Piscataway, NJ:IEEE Press, Vol. 1, pp.401–406, 1996

[Isermann74] R.Isermann, *"Prozessidentifikation: Identifikation und Parameterschätzung dynamischer Prozesse mit diskreten Signalen,"* Springer-Verlag, Berlin, 1974

[Koza92] J. R. Koza, *"Genetic programming: On the programming of Computers by means of natural selection,"* MIT Press, Cambridge, MA, 1992

[Koza94] J. R. Koza, *"Genetic programming II: Automatic discovery of reusable programs,"* MIT Press, Cambridge, MA, 1994

[Koza99] J. R. Koza, *"genetic programming III: Darwinian invention and problem solving,"* San Mateo, CA: Morggan Kaufmann, 1999

[Korenberg86] M.J. Korenberg, I.W.Hunter, The Identification of Nonlinear Biological Systems: LNL Cascade Models, Biol. Cybern., Vol.55, p.125-134, 1986

[Kafka02] V. S. Kafka, *"Rekursive Strukturen auf Voltera-Basis zur aufwandsarmen Darstellung und Entzerrung von nichtlinearen Systemen,"* Vorgelegte Dissertation, Fakultät für Elektrotechnik und Informationstechnik, Universität der Bundeswehr München, 2002

[Korenberg86] M.J. Korenberg, L.D. Paarmann, *"The Identification of nonlinear biological systems: LNL cascade models,"* Biol. Cyber., vol.55, pp. 125–134, 1986

[Ljung87] L.Ljung System Identification: Theory for the User. Prentice-Hall, Englewood Cliffs, NJ. 1987

[Ljung92] L.Ljung and T.Glad, Modelling of dynamic Systems, Prentice Hall, Englewood Cliffs, NJ. 1992

[Ljung98] L.Ljung, MATLAB System Identification Toolbox, Version 4.0. The MATHWORK Inc., Natick, MA, 1998

[Mathews96] V.J. Mathews, *Adaptive Volterra Filters Using Orthogonal Structures,"* IEEE Signal Processing Letters, Vol. 3, No. 12, pp. 307-309, December 1996

[Mathews00] V.J. Mathews and G.L. Sicuranza, Polynomial Signal Processing. New York: John Wiley & Sons, 2000

[Möller92] D. Möller, Modellbildung, Simulation und Identifikation dynamischer Systeme, Springer-Verlag, Berlin, 1992

[Mallat98] S.Mallat, A Wavelet Tour of Signal Processing, Academic Press, 1998

[Nikolay01] Nikolay Y. Nikolaev and Hitoshi Iba, "Regularization approach to inductive genetic programming," *IEEE Trans. Evolutionary Computation,* vol. 5, No. 4, August 2001.

[Nelles01] O.Nelles, *"Nonlinear System Identification: From classical approaches to neural networks and fussy models, "* Springer-Verlag, Berlin, 2001

[Nowak99] R.D. Nowak, R.G.Baraniuk, "Wavelet-based transformation for nonlinear signal processing," IEEE Trans. Signal Processing, Vol. 47, No.7, pp.1852-1865, July 1999

[Nikolaev00] N. Nikolaev und H. Iba, Inductive Genetic Programming of Polynomial Learning Networks, IEEE Symposium on Combinations of Evolutionary Computation and Neural Networks, p.158-167, 2000

[Prakriya95] S.Prakriya and D. Hatzinakos, "Blind identification of LTI–ZMNL–LTI nonlinear channel models," *IEEE Trans. Signal Processing,"* vol. 43, pp. 3007–3017, Dec,1995

[Panyaworayan02] W. Panyaworayan, *"Prädiktion von chaotischen Zeitreihen mit rekursiver genetischer Programmierung und Konstantenoptimierung,"* Doktorarbeit, Fakultät für Elektrotechnik und Informationstechnik, Universität der Bundeswehr München, 2002

[Pham99] D.T. Pham und X.Liu, Neural Networks for Identification, Predication and Control, Springer-Verlag London, 4. ed., 1999

[Marenbach95] P. Marenbach, K.D. Bettenhausen, and B.Cuno. Selbstorganisierende Generierung Strukturierter Prozessmodelle. *Automatisierungstechnik,* 43(6):277-288, 1995.

[Marenbach96] P. Marenbach, K.D. Bettenhausen, and S. Freyer. Signal path-oriented approach for generation of dynamic process models. In *Conference on genetic programming,* pp. 327-332, Stanford, 1996.

[Schetzen80] M. Schetzen, *"The Volterra and Wiener theories of nonlinear systems,"* Wiley and Sons, New York, 1980

[South96] M.South, C. Bancroft, M.J. Willis und M.T. Tham, System Identification Via Genetic Programming, International Conference on Control 96, UKACC, Conference Publication No.427, Vol.2, p. 912-917, 1996

[Sjoberg95a] J.Sjoberg, Q.Zhang, L.Ljung, A.Benveniste, B.Delyon, D.Y. Glorennec, H.Hjalmarsson, and A.Juditsky, Nonlinear black-box Modelling in System Identification: a unifield Overview. Automatica, 31(12):1691–1724,1995

[Sjöberg95b] J.Sjöber, Non-Linear System Identification with Neural Networks, Printed in Sweden by Linköpings Tryckeri, 1995

[Surenbabu95] N.Surenbabu, J.Farrel. Wavelet Based System Identification for Nonlinear Control Applications, Proceedings of IEEE Control und Decision Conference. pp.236-241, 1995

[Wang01] L. Wang, K.K. Teo and Z. Lin, Predicting Time Series with Wavelet Packet Neural Networks, Proceedings of the 2001 International Joint Conference on Neural Networks, Vol.3, Washington, D.C. p.1593-1597, 2001

[Weinbrenner97] T.Weinbrenner, GPC++ –Genetic Programming C++ Class Library, Diplomarbeit, Institut für electromechanische Konstruktion, TU Darmstadt, 1997

[Wang94] L.Wang, Adaptive Fussy Systems and Control: Design and Stability Analysis. Prentice-Hall, Englewood Cliffs, NJ. 1994

[www01] http://www.rt.e-technik.tu-darmstadt.de/ eva/GP/smog.html

[Zhu99] Y.C.Zhu Parametric Wiener Modell identification for Control. Proceedings of 14th IFAC world Congress, July 5-9, 1999, Beijing.

[Zhu02] Y.C.Zhu Estimation of an N-L-N Hammerstein-Wiener Modell, Proceedings of 15th IFAC world Congress, July 21-26, 2002, Barcelona.