Lehrstuhl für Realzeit-Computersysteme

Aufgabenorientierte Kopplung von Sensoren mit unterschiedlichen Abtasteigenschaften

Christian Robl

Vollständiger Abdruck der von der Fakultät für Elektrotechnik und Informationstechnik der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktors der Ingenieurwissenschaften (Dr.-Ing.)

genehmigte Dissertation.

Vorsitzender: Univ.-Prof. Dr.-Ing. A. W. Koch

Prüfer der Dissertation:

1. Univ.-Prof. Dr.-Ing. G. Färber
2. Univ.-Prof. Dr.-Ing. G. Reinhart

Die Dissertation wurde am 25.01.00 bei der Technischen Universität München eingereicht und durch die Fakultät für Elektrotechnik und Informationstechnik am 28.06.00 angenommen.

Vorwort

Die vorliegende Dissertation entstand während meiner Tätigkeit als wissenschaftlicher Mitarbeiter am Lehrstuhl für Realzeit-Computersysteme der Technischen Universität München im Teilprojekt 5 *„Mikropositioniereinrichtung für die Präzisionsmontage"* des Forschungsverbundes Mikrosystemtechnik FORMIKROSYS der Bayerischen Forschungsstiftung.

Ein besonderer Dank gilt meinem Doktorvater, Herrn Prof. Dr.-Ing. G. Färber, für das in mich gesetzte Vertrauen und die mir gewährten Freiheiten. Die mit ihm geführten fachlichen Diskussionen trugen wesentlich zum Gelingen dieser Arbeit bei. Bedanken möchte ich mich auch bei Prof. Dr.-Ing G. Reinhart für die gute Projektleitung im Teilprojekt 5 sowie für das Interesse an meiner Arbeit.

Ich möchte mich bei meinem Kollegen Darius Burschka bedanken, der mir in zahlreichen Diskussionen wertvolle Hinweise und Tipps geben konnte. Im weiteren gilt mein Dank Robert Huber und Martin Orehek für die fachliche Unterstützung in regelungstechnischen Fragen. Bedanken möchte ich mich ebenfalls bei Alexa Hauck für die Korrektur meiner Arbeit sowie bei Tom Hopfner für die gute UNIX-Betreuung. Ich möchte mich bei allen Mitarbeitern des Lehrstuhls für Realzeit-Computersysteme für die gute Arbeitsatmosphäre bedanken.

Ein weiterer Dank geht an Michael Höhn vom iwb für die gute Zusammenarbeit im Teilprojekt 5 sowie für die Bereitstellung von Messequipment und Versuchsaufbauten.

Dank bin ich auch meinen Diplomanden schuldig. Im besonderen möchte ich hier Thomas Andreas und Gerhard Englberger nennen, die wichtige Beiträge zur vorliegenden Arbeit geleistet haben.

Mein ganz besonderer Dank gilt meinen Eltern und meiner Frau Evi, die mich während der ganzen Zeit uneingeschränkt unterstützt haben.

Inhaltsverzeichnis

Zusammenfassung

In der vorliegenden Arbeit werden Verfahren zur Sensorkopplung auf Signal-Level-Ebene für Regelungszwecke erarbeitet und in einer MatLab basierten Toolbox realisiert und verwaltet. Die zugrunde liegende Systemarchitektur unterteilt die Sensorkopplung in drei funktionelle Blöcke: Synchronisation, Sensordatenvorverarbeitung und Sensordatenfusion. Der Multisensor-Regelkreis verhält sich durch die Sensorkopplung mit der vorgestellten Architektur wie der klassische 1-Sensorregelkreis.

Für die zeitlich korrekte Kopplung verschiedener Sensoren, vor allem bei unterschiedlichen Abtastraten, ist die entscheidende Eigenschaft der Sensoren, wie und vor allem wann neue Daten zur Verfügung gestellt werden. Es werden anhand dieser Eigenschaften vier verschiedene Sensortypen klassifiziert: Zeitkontinuierlicher zyklischer, zeitdiskreter zyklischer, zeitdiskreter asynchroner und zeitkontinuierlich asynchroner Sensor. Für jeden dieser Sensortypen wird ein Sensormodell entwickelt, das neben dem Übertragungsverhalten auch Fehlermodelle enthält.

Für die drei funktionellen Blöcke werden Verfahren als SimuLink-Blöcke implementiert, wobei klassische und neuentwickelte Verfahren zum Zuge kommen. Ein Vergleich der Verfahren stellt deren Leistungsfähigkeit unter Beweis. Bei der Sensordatenvorverarbeitung wird das Hauptaugenmerk auf Verfahren der zweifachen Integration gelegt.

Das zur Spezifikation und Simulation verwendete Werkzeug ist SimuLink, die Simulationsumgebung von MatLab. Eine Erweiterung von SimuLink ermöglicht die Verwendung des gleichen Blockdiagramms zur Softwaresimulation und zur automatischen C-Codegenerierung für die Hardware-in-the-Loop-(HIL)-Simulation. Normalerweise werden hierzu zwei unterschiedliche Blockdiagramme benötigt, was leicht zu Konsistenzproblemen führt.

Die Zuordnung der Hardware-Ein- und Ausgabekomponenten zu den Sensoren bzw. Aktoren erfolgt im Blockdiagramm über Hardware-Interfaces, die sich bei der Software-Simulation neutral verhalten. Bei der Umschaltung der Simulationsart von Softwaresimulation zu Codegenerierung wird das Blockdiagramm automatisch in die andere Darstellung konvertiert, wobei alle Blöcke der Strecke entfernt werden. Die Hardware-Interfaces, eine Art Gerätetreiber, werden als MatLab-S-Functions in C realisiert, man kann dabei wegen der offenen MatLab-Struktur beeinflussen, ob die Gerätetreiber als Funktionsaufruf oder „inline“ in den generierten C-Code eingefügt werden. Es werden Hardware-Interfaces für Standard-I/O-Komponenten sowie für Interrupt getriggerte Eingabekomponenten entwickelt.

Die Methodik zur Sensorkopplung wird mit Hilfe der erarbeiteten Verfahren und der entstandenen Toolbox am Reglerentwurf für eine Multisensor-Mikropositioniereinrichtung evaluiert. Es werden drei verschiedene Sensoren mit unterschiedlicher Dynamik und unterschiedlichem Abtastverhalten (Lasersensoren, Beschleunigungssensoren und ein Bildverarbeitungssystem) zur Korrektur von dynamischen und statischen Störgrößen eingesetzt. Das mit den Methoden dieser Arbeit neuentwickelte Regelungssystem wird mit dem mit klassischen Methoden entworfenen ursprünglichen Rege-

lungssystem verglichen. Es zeigt sich, dass sich der Entwurf des Multisensor-Regelungssystems bei besserer Leistungsfähigkeit wesentlich vereinfacht. Die entwickelten Sensormodelle bewirken, dass die Softwaresimulationsergebnisse nahezu 1:1 auf das reale System übertragen werden können, was eine Vereinfachung des Reglerentwurfs darstellt. Die meist begrenzte Ressource „Hardware“ (z.B. Industrieroboter) ist beim Entwurf und beim Tunen des Regelungssystems nicht mehr unbedingt notwendig. Die Grundlage hierfür bildet die gewissenhafte Systemidentifikation der Strecke, wie das Beispiel der Mikropositioniereinrichtung (MPE) zeigt.

1 Einleitung

1.1 Problemstellung

Fortschritte in der Sensor- und Mikrosystemtechnik haben bewirkt, dass neue Sensoren, die kleiner, billiger und genauer sind, verfügbar werden und damit neue Anwendungsbereiche geschaffen werden konnten. Beispielhaft sind hier Beschleunigungssensoren zu nennen, die nun in großen Stückzahlen in der Automobilindustrie für die Crash-Erkennung der Airbags (mehrere Stück pro Airbag) eingesetzt werden. Man ist dadurch heute in der Lage, in vielen Bereichen der Automatisierungstechnik, der autonomen mobilen Robotersysteme und der Automobilindustrie kostengünstige Multisensor-Systeme aufzubauen. Dies bedeutet natürlich auch, dass sich die Anforderungen an den Regelungsentwurf und der dazugehörigen Signalverarbeitung erhöhen, wobei aber eine bessere Regelbarkeit und/oder höhere Genauigkeit erzielt werden kann. Es können dadurch Systeme realisiert werden, die mit einem konventionellen 1-Sensor-System nicht regel- bzw. steuerbar gewesen wären.

Der klassische Regelungsentwurf sieht z. B. nach [11] folgende Schritte vor:

1. Modellierung oder Identifikation der Regelstrecke
2. Auswahl und Anordnung der Stellglieder und Sensoren („Control configuration")
3. Aufstellen des Regelgesetzes, Reglerentwurf und Software-Simulation
4. Implementierung des Reglers auf dem Zielsystem
5. Test des Gesamtsystems

In der Forschung liegen die Schwerpunkte eindeutig bei 1. und 3., wie man auch deutlich an der Vielzahl der vorhandenen Methoden zum Regelungsentwurf (PID-, LQR-, H∞-, Fuzzy-, Neuro-Regler) und zur Systemidentifikation (Korrelations- und Spektral-Verfahren, ARX-, ARMAX-, „Output-Error" und Box-Jenkins-Modelle) erkennen kann. Es ist mittlerweile sogar zu einem Forschungsschwerpunkt geworden zu bestimmen, welche Methode für welche Problemstellung am geeignetsten ist [12]. Den größten Einfluss auf die Regelbarkeit und Leistung des Regelungssystems jedoch hat die Wahl und Anordnung der Aktoren und Sensoren (2.).

Es ergibt sich somit folgende Problemstellung:
Für ein gegebenes System, bei dem die Regelstrecke sowie das zeitdiskrete Regelungskonzept und die Stellglieder vorgegeben sind, soll ein optimales Sensorkonzept gefunden werden, mit dem eine Regelung des Gesamtsystems unter den gegebenen Spezifikationen möglich wird oder es soll die Aussage getroffen werden können, welche Spezifikationen eingeschränkt bzw. verändert werden müssen, um eine Regelung zu ermöglichen.

Zur Lösung des Problems gilt es folgende Punkte zu klären:

- Welche Arten von Sensoren hat man zur Verfügung?
- Wie muss man die Sensoren charakterisieren und modellieren, damit man sie untereinander vergleichen kann?
- In welche funktionalen Blöcke können die einzelnen Schritte zur Kopplung von Sensoren unterteilt werden (z. B. Synchronisation, Adaption und Fusion)?
- Welche Architektur ergibt sich daraus bei Regelungssystemen?
- Wie beschreibt man das Gesamtsystem, um eine Stabilitäts- und Funktionsprüfung mittels Simulation zu ermöglichen? Welche Effekte treten dabei auf?
- Wie kann man unter Verwendung der Simulationsergebnisse automatisch das Regelungssystem auf dem Zielsystem implementieren?

1.2 Sensorfusionsarten im Vergleich

Bei der Kopplung von Sensoren in Multisensor-Systemen unterscheidet man grundsätzlich zwischen Sensor-Integration und Sensor-Fusion. Werden bei der Sensor-Integration verschiedene Sensoren für verschiedene Aufgaben eingesetzt, so werden bei der Sensorfusion Daten von verschiedenen Sensoren zu einer gemeinsamen Darstellung kombiniert (z. B. Erstellen einer Karte mit CCD-Kamera- und Laser-Scanner-Daten).

Die zu fusionierenden Sensordaten können redundante, zusätzliche aber auch widersprüchliche Informationen enthalten. Durch die Fusion von Sensoren unterschiedlicher Bearbeitungszeit ergeben sich durch deren unterschiedliche Messzeitpunkte zeitgenauere Informationen. Da nach [62] die Sensordatenfusion immer eine Verbesserung der Sensordaten und eine Erhöhung der Verfügbarkeit darstellt, können ungenauere und billigere, dafür aber mehrere Sensoren verwendet werden. In [88] wird schließlich der Beweis erbracht, dass bei optimaler Sensordatenfusion die Qualität der fusionierten Sensordaten nie schlechter ist als die der nicht fusionierten.

Es gibt 3 verschiedene Arten der Sensor-Fusion:

1. „Zeitliche Fusion“: Zusammenfassung von zeitlich aufeinanderfolgenden Daten eines Sensors zu einem einzigen Datum (z.B. Tracking oder Motion-Stereo)
2. „Redundante Fusion“: Zusammenfassung von Daten von Sensoren gleichen Typs, um eine Verbesserung der Datenqualität sowie eine höhere Verfügbarkeit zu erreichen.
3. „Multimodale bzw. Multisensor-Fusion“: Zusammenfassung von Daten verschiedener Sensoren mit verschiedenen Messprinzipien.

In [37] erfolgt eine Klassifikation der Sensorfusion in 5 Abstraktionsstufen, wobei bei steigendem Abstraktionsniveau (1-5) der Umfang der zu fusionierenden Daten abnimmt.

1. Bei der Fusion auf Signal-Level-Ebene werden ein- oder mehrdimensionale Signale mit geringer Informationsdichte aber dafür mit hoher Datenrate fusioniert.

Eine zeitliche Synchronisation der Sensoren ist erforderlich. Durch die Fusion wird eine Reduktion der Varianz der Sensordaten ($\sim 1/\sqrt{n}$ bei unabhängigen Sensoren) erreicht. Als Methoden zur Fusion kann üblicherweise bei gleichartigen Sensoren der gewichtete Mittelwert oder bei dynamischen Sensordaten, die nicht unbedingt von gleichartigen Sensoren stammen müssen, der Kalman-Filter angewendet werden. Eine weitere Alternative stellt die Methode der "Bayesian Estimation Using Consensus Sensors" [61] dar, die hauptsächlich zur Fusion von redundanten Sensoren entwickelt wurde.

2. Bei der Fusion auf Pixel-Level-Ebene werden mehrere Bilder kombiniert, um eine höhere Leistungsfähigkeit bei Bildverarbeitungsaufgaben zu erreichen. Als Methoden zur Fusion werden logische Filter (z. B. AND-Filter), mathematische Morphologie, Bild-Algebra oder „Simulated Annealing" verwendet.
3. Bei der Feature-Level-Fusion werden aus Bild- oder Signaldaten extrahierte Merkmale zeitlich oder räumlich fusioniert. Die Merkmale (z. B. Kanten, Flächen und Eckpunkte) werden dabei durch ihre geometrischen Eigenschaften wie Form, Lage und Position beschrieben. Dadurch wird eine reduzierte Verarbeitungszeit, eine höhere Genauigkeit und eine Erweiterung bei den Eigenschaften der extrahierten Merkmale erreicht. Verwendete Methoden sind unter anderem "Tie Statistic", "Gauss-Markov Estimation" und der erweiterte Kalman-Filter.
4. Die Symbol-Level-Fusion erlaubt die Kombination von Sensordaten auf einer hohen Abstraktionsebene. Es werden Symbole bzw. Objekte mit den dazugehörenden Detektions-Unsicherheiten fusioniert um eine Erhöhung der Wahrscheinlichkeit für das jeweilige Objekt zu erreichen. Fusions-Methoden sind zum Beispiel "Bayesian Estimation", "Dempster-Shafer Evidental Reasoning", "Production Rules" mit Konfidenzmaß und Fuzzy Logic [34].
5. Mit der Behaviour-Level-Fusion wird versucht, die Reaktion eines Systems auf eine Anregung zu verbessern, indem verschiedene Reaktionsmuster zum Beispiel mit Fuzzy-Reglern kombiniert werden.

Es kann zwischen drei Arten der Darstellung der fusionierten Daten unterschieden werden:

1. räumlich: 2D- oder 3D-Kartenerstellung
2. zeitlich: a) sequentiell: Daten des genaueren Sensors nur wenn nötig („Sensor Selection")
   b) parallel: Ergebnis der Multisensor- oder redundanten Fusion
3. räumlich und zeitlich: Ergebnis der multimodalen Fusion wie zum Beispiel im 4D-Ansatz von Dickmanns [113], [23].

Die hauptsächlichen Anwendungsgebiete der Sensordatenfusion sind:

- Objekterkennung
- Autonome mobile Multisensor-Robotersysteme
- Industrielle Anwendungen
  - Materialverarbeitung

    - Teileherstellung
    - Qualitätskontrolle
    - Montage
- Militärische Anwendungen
    - Zielerkennung und –verfolgung
    - Raumfahrt
    - Inertiale Navigation

1.3 Aufgabenstellung

Der Entwurf von Regelungssystemen sowie deren Simulation und Implementierung erfolgt heute üblicherweise mit sogenannten Computer-Aided-Control-System-Design (CACSD) Tools. Das zur Zeit am weitesten verbreitete Werkzeug dieser Art ist MatLab / SimuLink mit den dazugehörigen Toolboxen.

Ziel dieser Arbeit ist die Entwicklung von Verfahren zur Sensorkopplung, die durch ihre Implementierung in einer MatLab basierten Toolbox den Entwurf von multimodalen Multisensor-Regelsystemen vereinfachen. Es wird dabei nur die Kopplung der Sensoren auf Signal-Level-Ebene betrachtet, wie es bei den meisten inneren Regelkreisen auch der Fall ist. Es wird davon ausgegangen, dass die notwendigen Systembeschreibungen der Strecke und der Stellglieder sowie das Regelkonzept vorgegeben ist. Da heute die meisten Regelsysteme auf Mikrocontrollern oder eingebetteten Systemen realisiert werden, sollen in dieser Arbeit nur zeitdiskrete Regelungssysteme betrachtet werden.

Mit der Problemstellung von 1.1 leiten sich folgende Arbeitspunkte ab, die innerhalb der vorliegenden Arbeit gelöst werden.

- Entwicklung einer geeigneten modularen Systemarchitektur, die es ermöglicht, verschiedene Sensorkonstellationen und Signalverarbeitungsalgorithmen ohne großen Aufwand zu realisieren und die als Struktur für die Implementierung in einer Toolbox verwendet werden kann (Kapitel 3).
- Klassifikation der verschiedenen Sensoren anhand der für den zeitdiskreten Regelungsentwurf notwendigen Eigenschaften und Aufstellen von Simulationsmodellen für die einzelnen Sensorklassen (Kapitel 3.1).
- Bereitstellung und Entwicklung von Algorithmen zur Sensorvorverarbeitung, vornehmlich zur Offsetkorrektur, Differentiation und (Doppel)-Integration (Kapitel 3.2).
- Entwicklung, Vergleich und Implementierung von Algorithmen zur Synchronisation (Kapitel 3.3) und Fusion (Kapitel 3.4) von Sensordaten.
- Simulation von beispielhaften Multisensorsystemen (Kapitel 3.5).
- Ermöglichen der automatischen Implementierung auf dem Zielsystem des spezifizierten Regelungssystems anhand des Software-Simulations-Blockdiagramms und Entwicklung von Hardware-Interface-Templates für die Anbindung der verschiedenen Sensorklassen (Kapitel 3.6).

- Beispielhafte Implementierung eines Multisensorsystems: Regelung einer Mikropositioniereinrichtung, die über drei verschiedene Sensoren verfügt (Kapitel 4).

2 Stand der Technik

2.1 Beschreibung und Simulation von zeitdiskreten Systemen

Die von Kalman 1959 aufgestellte Theorie von Abtastsystemen [53] stellt auch heute noch die Grundlage für die Beschreibung von abgetasteten zeitdiskreten Regelungssystemen dar. Es wird darin der Regelkreis in zeitkontinuierliche dynamische Elemente (z. B. Strecke) und zeitdiskrete dynamische Elemente (z. B. Regler) unterteilt. Die zeitdiskrete Abtastung wird mit Abtast– und Haltegliedern dargestellt, wobei die folgenden Arten der Abtastung möglich sind und durch die Eigenschaften des Abtast- und Halteglieds bestimmt werden:

- Konventionelle Abtastung: Die Abtastung erfolgt in äquidistanten Schritten und alle Abtastglieder arbeiten synchron mit der gleichen Abtastrate.
- Asynchrone Abtastung: Die Abtastung erfolgt in äquidistanten Schritten und alle Abtastglieder haben die gleiche Abtastrate, wobei ein zeitlicher Versatz der Abtastzeiten vorliegt.
- Multiorder Abtastung: Die Abtastung erfolgt in nicht konstanten, aber periodisch sich wiederholenden Abtastschritten. Alle Abtastglieder arbeiten synchron.
- Multirate Abtastung: Jedes Abtastglied hat seine eigene konstante Abtastrate.
- Totzeitbehaftete Abtastung: Die Abtastglieder stellen den mit einer konstanten Abtastrate bestimmten Wert erst nach einer konstanten Totzeit zur Verfügung.
- Zufällige Abtastung: Die Abtastzeit ist eine Zufallsvariable.

Für alle diese Abtastsysteme (und auch für Mischformen daraus) können mit dieser Theorie die Systemgleichungen in Zustandsform aufgestellt werden und damit kann die Stabilität des Regelsystems bestimmt werden. In [5] wird diese Theorie aufgegriffen, um weitere Stabilitätskriterien sowie das Nyquist Kriterium und das Übertragungsverhalten für Regelschleifen mit verschiedenen Abtastraten abzuleiten. Die Abtastraten dürfen allerdings nur das Vielfache einer Basisabtastrate sein und die Abtastung muss synchron erfolgen.

Die Theorie der Abtastregelungen für konventionelle und asynchrone Abtastsysteme ist in Standardwerken wie z. B. [1] dargelegt. Multirate Abtastsysteme können geschlossen nur dann behandelt werden, wenn das Verhältnis der Abtastraten rational ist. Ein irrationales Verhältnis lässt sich nach [1] mit der vorhandenen Theorie praktisch nicht realisieren, so dass für diese Systeme nur die Simulation bleibt. Boyd [12] versucht mit dem Ansatz der konvexen Optimierung festzustellen, ob es für ein gegebenes System und bei gegebener Konfiguration ein Regelgesetz gibt, das die geforderte Spezifikation einhält bzw. wie die Spezifikationen oder auch die Konfiguration geändert werden müssen, um noch ein Regelgesetz erhalten zu können. Letztendlich kommt er zu dem

Schluss, dass eine analytische Beschreibung eines Regelsystems nur im Spezialfall (linear und zeitinvariant oder konvex) möglich ist.

Der zeitdiskrete Regelungsentwurf kann nach [84] entweder als Emulation des kontinuierlichen Systems oder - bei ausreichend hoher Abtastrate (>10-25fache der höchsten Signalfrequenz) - direkt im Zeitdiskreten erfolgen. Beide Varianten stellen jeweils nur Näherungen des realen Systems mit den daraus resultierenden Effekten (z. B. Aliasing) dar. In [117] und den dort weiterführenden Referenzen werden deshalb direkte Regelungsentwürfe für gemischte diskrete und kontinuierliche (=hybride) Abtastsysteme mit konstanten Abtastraten vorgestellt und in eine MatLab basierte Toolbox zusammengefasst [33].

Allgemein kann gesagt werden, dass hybride Abtastsysteme mit beliebigem Abtastverhalten eine wachsende Bedeutung erhalten, aber es existieren zu ihrer formalen, mathematischen Beschreibung und Analyse nur erste Ansätze. Methodiken zum gezielten Entwurf solcher Systeme liegen nur für Spezialfälle vor. Gegenstand des Schwerpunktprogramms KONDISK der DFG [56] ist seit 1995 die Grundlagenforschung auf dem Gebiet der kontinuierlich-diskreten technischen Systeme im Hinblick auf die Analyse ihres Gesamtverhaltens und den systematischen, integrierten Entwurf ihrer Systemkomponenten.

Auch bei MODELICA [28], eine Sprache zur Spezifikation und Simulation von physikalischen Systemen, fehlt noch die zeitdiskrete Komponente. Hier ist man gerade bemüht, die Interaktion von diskreten und kontinuierlichen Systemen zu integrieren.

MatLab kann beim Entwurf eines Regelungssystems mit der Control System Toolbox [68] oder mit der Robust Control Toolbox [21] entweder komplett kontinuierliche oder komplett zeitdiskrete Systeme handhaben. Abhilfe schafft hier die Verwendung des dazugehörigen Simulationswerkzeugs SimuLink [72], das zumindest hybride Systeme simulieren kann. Neben SimuLink gibt es weitere Werkzeuge dieser Art, z. B. LabView RT [51], MatrixX [67] und Asket SD [103]. Da MatLab / SimuLink am weitesten verbreitet ist und auch in der Industrie eine starke Rolle spielt [108], wird in dieser Arbeit auf MatLab / SimuLink aufgebaut. Es ist auch bestens geeignet, die Simulationsergebnisse für die automatische Implementierung auf dem Zielsystem zu verwenden. Es kann hierfür für gewisse Zielsysteme (z.B. VxWorks, DOS oder Windows) der eigene C-Codegenerator (Realtime Workshop) oder für zusätzliche Zielsysteme auch Entwicklungssysteme anderer Anbieter wie z.B. dSPACE [39] oder Opal RT [80] verwendet werden. Es ist eine übliche Methode, MatLab durch Hinzufügen von anwenderspezifischen Toolboxen an seine Gegebenheiten anzupassen, wie z.B. die schon erwähnte Sampled Data Control Toolbox [33] oder die nD-Control System Toolbox [116]. Dies wird durch die offene und plattformunabhängige (außer Mex-Files) Architektur von MatLab / SimuLink unterstützt. Es fehlt aber noch eine passende Sensorfusion Toolbox, bei der die automatische Umwandlung der simulierten Hardware zu Hardware-Interfaces bei der automatischen C-Codegeneration möglich ist.

2.2 Architektur von Multisensorsystemen

Die in [66] vorgestellte allgemeine Architektur zur Multisensor Fusion besteht aus vier Stufen:

1. Interpretation: Die Messwerte werden anhand von physikalischen Modellen und Fehlermodellen interpretiert. Ein Feedback der 4. Stufe wählt die zu verwendenden Sensoren aus.
2. Fusion: Hier werden die Messwerte mit verschiedenen Verfahren fusioniert. Um konsistente Informationen zu erhalten, gibt es einen Feedback von der 4. Stufe, die die zu fusionierenden Sensordaten auswählt.
3. Schätzung: Anhand der fusionierten Daten werden die Systemzustände geschätzt.
4. Sensor Management & Planung: Anhand von a priori Wissen und den aktuellen Zuständen werden die Sensoren und Fusionsverfahren ausgewählt. Von hier aus gehen die Daten zur nächsten Stufe der Verarbeitung (z. B. Regelung).

Diese Einheit kann in einer zentralisierten Anordnung, bei der alle Sensoren an dieser einen Fusionseinheit hängen, betrieben werden. Es ist aber auch eine hierarchische Anordnung denkbar, bei der eine Untermenge von Sensoren an lokalen Fusionseinheiten hängen und deren Daten dort verarbeitet werden. Die lokalen Einheiten schicken ihre Daten an die übergeordnete globale Fusionseinheit, die ihrerseits die Daten der lokalen Einheiten fusioniert. Eine dritte Möglichkeit stellt eine dezentrale Anordnung dar, bei der jeder Sensor an einer eigenen Fusionseinheit hängt, die die verarbeiteten Daten aller anderen oder einem ausgewählten Teil der Fusionseinheiten bekommt. Diese Einheit schickt dann ihrerseits die fusionierten Daten an die betreffenden anderen Einheiten.

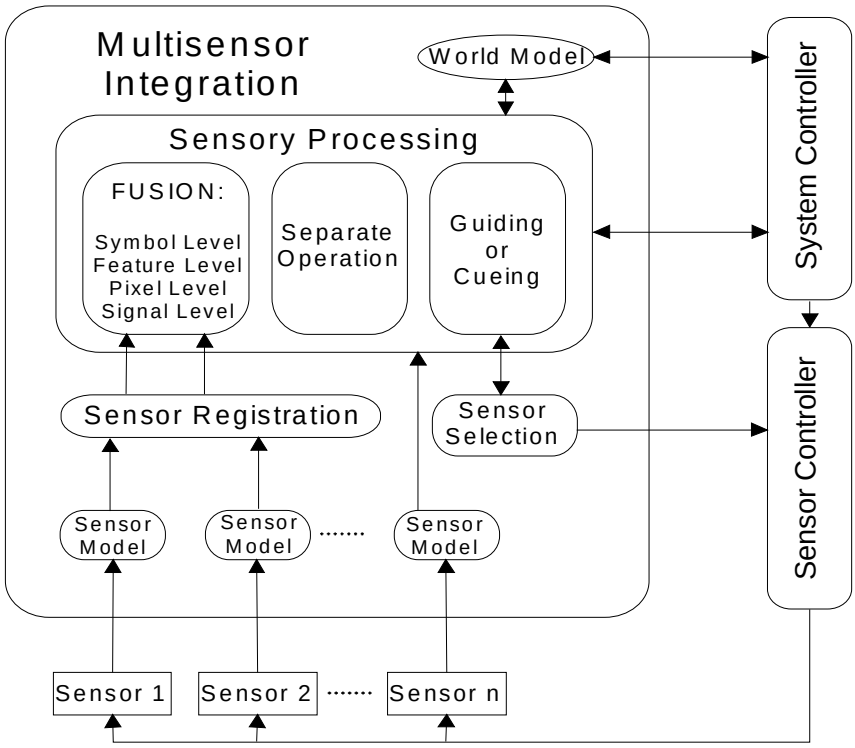

Bild 2.2.1: Systemarchitektur nach Luo und Kay

Luo und Kay stellen in [62] eine Architektur zur Multisensor Fusion und Integration vor (siehe Bild 2.2.1). Basierend auf Sensor- und Fehlermodellen wird die Qualität der einzelnen Sensordaten bewertet. Im Sensor-Registration Block werden die Sensordaten räumlich und zeitlich aneinander angepasst, bevor die Daten fusioniert werden. Im Weltmodell werden die Systemzustände und a priori Wissen abgelegt. Diese Architektur lässt im Rahmen einer Sensorintegration unabhängige Sensoren zu. Der Sensor Selection Block dient der Auswahl der im Moment zu verwendenden Sensoren. Dies stellt die verallgemeinerte Version der Architektur von Luo et al. [61] (siehe Bild 2.2.2) dar.

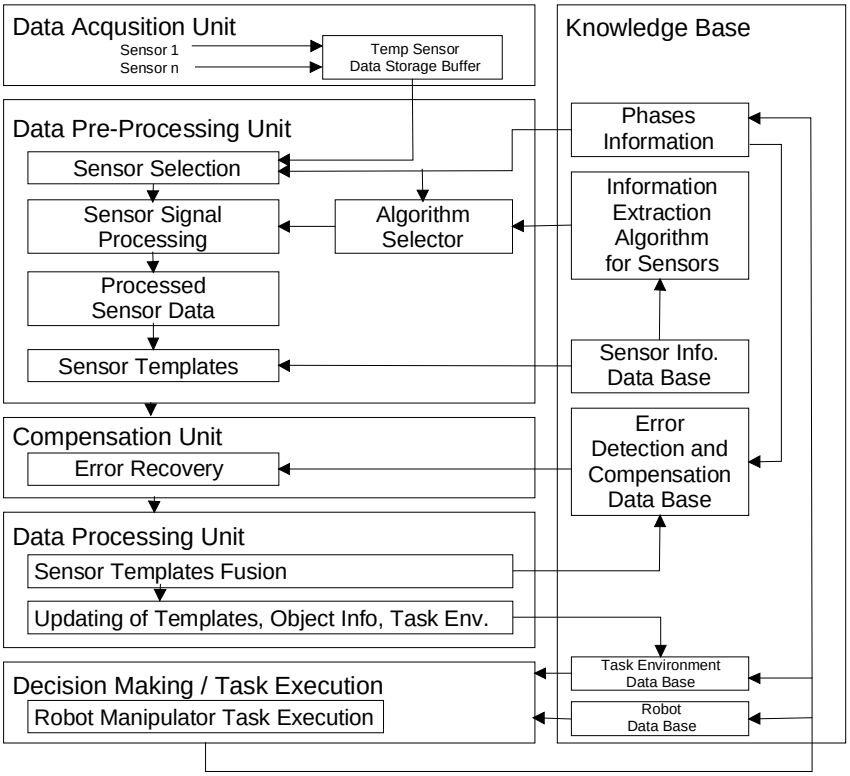

Bild 2.2.2 : Systemarchitektur nach Luo et al.

Durrant-Whyte präsentiert in [27] einen hierarchischen Ansatz zur räumlichen Sensorfusion, bei dem einzelne Sensoren zu sogenannten Sensorteams zusammengefasst werden. Innerhalb eines Teams kommunizieren die Sensoren untereinander, um die Informationen festzulegen, welche an andere Teams weitergegeben werden. Ein Sensorteam kann auch als logischer Sensor bezeichnet werden.

In [29] ist ein Ansatz zu sehen, der sich speziell mit der Fusion bei verschiedenen Taktraten beschäftigt. Es wird jedoch festgelegt, dass die Abtastraten ein rationales Verhältnis aufweisen müssen. Für diesen Spezialfall kann zudem die Stabilität und die Performance des Systems angegeben werden. Eine objektorientierte Architektur, Instrumented-Logical-Sensor (ILS) System genannt, wird in [22] präsentiert. Die einzelnen Objekte, ein ILS-Modul, beschreiben Algorithmen und definieren Ein- und Ausgänge. An ILS-Modulen können entweder Sensoren oder wieder ILS-Module hängen. Zur Kommunikation zwischen den Modulen sind feste Strukturen definiert.

Zur sequentiellen Sensorfusion gibt es eine Vielzahl von Architekturen, die sich meist auf den Anwendungsbereich autonome mobile Systeme beschränken. In [20] und [47] nimmt ein sogenannter Perception Controller alle Daten auf und entscheidet anhand des Sensors, der das beste Ergebnis liefert, welche Daten an die nächste Schicht weitergegeben werden. In [54] wird ein neuronales Netz für diese Aufgabe verwendet.

Die Just-in-time-Architektur aus [26] verwendet im Normalfall den ungenaueren schnelleren Sensor und schaltet nur bei Bedarf auf den langsameren aber genaueren Sensor um, wobei aber das System verlangsamt wird.

Alle diese Architekturen wurden hauptsächlich zur räumlichen Datenfusion bei autonomen mobilen Systemen entwickelt und sind deshalb zur zeitlichen Fusion nur bedingt geeignet. Sie sind zum Teil im Abstraktionsniveau zu hoch, um zur Sensorfusion auf Signal-Level-Ebene verwendet werden zu können.

Zeitbereich / Räumlicher Bereich	aktueller Zeitpunkt	zeitlich lokale differentielle Umgebung	lokale Zeitintegrale	erweiterte lokale Zeitintegrale		globale Zeitintegrale
aktuelle Position	„hier und jetzt“ lokale Messungen	zeitliche Veränderung (aufgrund der Rauschverstärkung vermieden)	1-Schritt Übergangsmatrix	- -		- -
räumlich lokale differentielle Umgebung	Kanten und Krümmungen	“”	Transition von Merkmalparametern	Merkmalshistorie		- -
lokale räumliche Integrale	Objektzustand Merkmalsverteilung Form	Bewegung „dynamische Modelle“	Zustandstransition	Vorhersagen im Nahbereich Objektzustandshistorie		sparse Vorhersagen
Manöverraum von Objekten	lokale Situation	Führungsgröße für effizienten Regler	(1-Schritt Prädiktion der Situation)	Mehrschrittprädiktion der Situation Manöverüberwachung		- -
Auftragsraum von Objekten	aktuelle globale Situation	- -	- -	Überwachung		Auftragsüberwachung

Tabelle 2.2.1 : Struktur 4D-Ansatz

Die Kombination von räumlicher und zeitlicher Datenfusion stellt der 4D-Ansatz von Dickmanns [23] dar. Der 4D-Zustandsraum wird durch die drei räumlichen Koordinaten und die Zeit aufgespannt. Der 4D-Ansatz sieht die integrale Verwendung von

- dynamischen Bewegungsmodellen unter Beachtung von Totzeiten und dem Reglerausgang
- 3D-Modellen für visuell messbare Merkmale
- perspektivisch projizierten Modellen

- prädizierte Fehlerrückkopplung im 4D-Zustandsraum

vor.

In Tabelle 2.2.1 ist die dem 4D-Ansatz zugrundeliegende Struktur dargestellt, dabei wird zwischen verschiedenen integralen und differentiellen Darstellungen auf verschiedenen Stufen unterschieden. Ausgehend von der Position zum aktuellen Zeitpunkt (links oben) wird nach rechts die zeitliche Darstellung variiert (Differentiation, Einschrittintegration, Prädiktion) und nach unten die Abstraktionsebene der räumlichen Darstellung erhöht.

2.3 Kalman-Filter

Der Kalman-Filter ist in der Regelungstechnik zu einer Standardmethode, sei es zur Zustandsschätzung, Synchronisation oder Fusion, avanciert. Es sollen hier kurz die verschiedenen Varianten des Kalman-Filters vorgestellt werden, die mathematische Betrachtung erfolgt in 3.3.2.3.

Der kontinuierliche zeitvariante Kalman-Filter [82] ist für lineare Systemgleichungen die Grundlage der weiteren Varianten. Es werden meist die Gleichungen für nicht korrelierte Störgrößen (weißes Rauschen) verwendet, im allgemeinen Fall kann durch Hinzunahme einer weiteren Kovarianzmatrix auch korreliertes weißes Rauschen gehandhabt werden. Für zeitinvariante Systemgleichungen ergibt sich der kontinuierliche zeitinvariante Kalman-Filter [82], [77] und für zeitdiskrete Realisierungen gibt es jeweils das diskrete Pendant, den diskreten zeitvarianten Kalman-Filter [62], [82], [68] und den diskreten zeitinvarianten Kalman-Filter [82], [68]. Es gibt nach [65] zwei verschiedene Möglichkeiten, die Messwertgleichungen des Kalman-Filters zu aktualisieren, die mathematisch korrekte und die numerisch stabilere. Nichtlineare Systemgleichungen können mit dem nichtlinearen erweiterten Kalman-Filter in zeitdiskreter [82] oder zeitkontinuierlicher Version [102] behandelt werden. Erweiterter Kalman-Filter kann auch bedeuten, dass unbekannte Systemparameter durch Hinzunahme von Pseudo-Systemzuständen [113] modelliert werden.

Sind die Anfangsparameter des Kalman-Filters nicht bekannt oder verändert sich das System unbekannt im Laufe der Zeit, so kann nach [62] ein adaptiver Kalman-Filter verwendet werden. Die Filterparameter werden dabei mit Standardverfahren (eine Übersicht findet sich in [107]) oder mit Fuzzy-Methoden [95] angepasst.

Eine weitere Darstellung ist die U-D-Kovarianz-Faktorisations-Form [62], bei der die Systemmatrizen zur einfacheren mathematischen und stabileren numerischen Handhabung in eine obere Dreiecksmatrix und eine Diagonalmatrix zerlegt werden [36].

In [55] wird ein komplexer Kalman-Filter vorgestellt, bei dem 2D-Variablen in einer komplexen Beschreibung ($z = x + i \cdot y$) zusammengefasst werden. In dieser Variante des Kalman-Filters werden dadurch zwei separate Kalman-Filter mittels der komplexen Beschreibung zusammengefasst.

Durch Auftrennen der Systemgleichungen bei der Sensordatenfusion kann der Kalman-Filter in unabhängige, für jeden Sensor separat bestimmbare Teile zerlegt werden. Die Aktualisierung der Systemzustände erfolgt dann wieder zusammengefasst.

Diese 2-stufige Variante wird paralleler Kalman-Filter [40] genannt und ist besonders gut für Multiprozessor-Systeme geeignet.

Der Kalman-Filter kann auch in der inversen Kovarianzform dargestellt werden. Diese Realisierung wird Informationsfilter [78], [13], [66] genannt und ist für alle Varianten des Kalman-Filters möglich.

Der Kalman-Filter kann nur stochastische Störgrößen berücksichtigen. In [38] wird eine Erweiterung vorgestellt, bei der auch beschränkte Unsicherheiten mit einem „Set theoretic estimation"-Ansatz behandelt werden können.

2.4 Sensormodelle

Durrant-Whyte beschreibt in [27] eine Technik, um Sensoren und die Informationen zu modellieren, die sie liefern. Das resultierende Modell besteht aus drei Teilmodellen.

1. Observation Model auch Measurement Model: beschreibt die Eigenschaften der Messwerte
2. Dependency Model: beschreibt die Abhängigkeit des Sensors von Informationen anderer Sensoren
3. State Model: beschreibt wie die Messwerte durch die Sensoranordnung und interne Zustände beeinflusst werden (Kalibrierung)

In [94] wird das Measurement Model für den mit 1MHz getakteten Beschleunigungssensor ADXL50 von Analog Devices aus den physikalischen Differentialgleichungen abgeleitet und als SimuLink-Blockdiagramm realisiert. Es wird nur das ideale Modell ohne Störungen und Rauscheinflüsse abgeleitet.

Mit Fehlermodellen und Formfiltern kann dies, wie in [75] dargelegt, berücksichtigt werden. Man erweitert die Zustände des idealen Measurement Models so, dass man nur noch deterministische Signale und weißes Rauschen als Eingangsgrößen hat und fügt ein sogenanntes Fehlermodell hinzu. Das so erweiterte Modell (State augmentation) stellt dann das reale Sensormodell dar. Kann man Störgrößen nicht mit weißem Rauschen beschreiben, so muss man sie sich über mit weißem Rauschen gespeiste Formfilter generieren. Diese Formfilter sind Bestandteil des Fehlermodells. Beispielhaft werden Formfilter für

- weißes Rauschen
- Bias
- Braun'sche Bewegung
- Braun'sche Bewegung mit Zufallskonstanten
- Markovprozeß 1. Ordnung
- Markovprozeß 2. Ordnung

hergeleitet.

In [7] wird darauf aufbauend das Drift-Offset Fehlermodell für ein Gyroskop abgeleitet und in [66] das Fehlermodell für einen Sonarsensor. Physikalische Sensormodelle

dieser Struktur für einen akustischen Sensor und eine Kamera werden zum Beispiel in [88] hergeleitet und für Sensorfusionsalgorithmen verwendet.

2.5 Synchronisation

Im DSP-Blocksatz [69] und der Signal Processing Toolbox [71] von MatLab sind bereits einige Verfahren enthalten, die zur Synchronisation von Sensordaten verwendet werden können. Das Halteglied 1. Ordnung (siehe 3.3.1.2) ist davon das einzige, das keine zusätzliche Verzögerung verursacht. Der FIR-Rate-Converter-Block [32] kann die Abtastrate im Verhältnis $\frac{k}{l}$ wandeln, erzeugt hierbei einen Delay von k-Werten. Ähnlich arbeitet der Resample-Block, der jedoch an den Randbereichen ungenau und somit für schritthaltendes Resampling ungeeignet ist. Mit dem Intfilt-Block [79] oder dem Interp-Block [49] kann die Abtastrate nur um einen Integerwert erhöht werden. Es entsteht dabei eine Totzeit von n Werten bei einem Filter n-ter Ordnung (Defaultwert: $n=7$).

In [5] werden bei der Abtastung mit unterschiedlichen Abtastraten die Abtastwerte im Verhältnis $\frac{k}{l}$ der Abtastraten zusammengefasst. Die jeweils zusammengefassten Werte stellen die synchronisierten Werte dar. Dieses Verfahren kann allerdings nur eingesetzt werden, wenn die resultierende um k bzw. l niedrigere Abtastrate noch zur Regelung ausreicht.

Der Kalman-Filter und seine Varianten sind ebenfalls zur Synchronisation geeignet. In [29] wird die langsamste Abtastrate mit einem einfachen Kalman-Filter synchronisiert. Die rekursive Zustandsschätzung mit dem Kalman-Filter in [8] lässt nur konstante, aber unterschiedliche Abtastraten zu. Die Synchronisation von asynchronen Messungen mittels „Forward Propagation“ wird in [13] mit einem Informationsfilter (linear oder nicht linear) durchgeführt. Um jedoch die Prädiktion auch für Zeiten zwischen den Abtastraten berechnen zu können ist eine kontinuierliche Variante notwendig. Eine 1-Schritt „Forward Propagation“ von Videodaten wird in [118] mit einem erweiterten Kalman-Filter realisiert. Unterstützende Messungen, z. B. von einem Beschleunigungssensor, sind notwendig.

In [41] erfolgt die Synchronisation zwischen zwei Abtastraten durch die Interpolation mit einem linearisierten Beobachter, an dessen Stelle auch ein Lünberger Beobachter oder Kalman-Filter treten könnte. Ackermann stellte in [1] Gleichungen auf, mit denen die Werte zwischen den Abtastschritten bei diskreten Systemen berechnet werden können. Es erfordert jedoch die Kenntnis der zeitkontinuierlichen Zustandsdarstellung. Man erhält damit die synchronisierten Werte mit einer Totzeit von einem Abtastschritt. Bei einem direkten diskreten Regelungsentwurf, bei dem die Strecke nur in diskreter Darstellung bekannt ist, kann dieses Verfahren nicht angewandt werden. Hier greift die „Lifting“ Methode von [117] und der darin enthaltenen weiterführenden Referenzen, bei der ein direkter Entwurf von gemischt kontinuierlich diskreten Systemen durch Bestimmung der Werte zwischen den Abtaststellen vorgestellt wird.

In Dickmanns 4D-Ansatz [23] werden die Totzeiten in den dynamischen Modellen des erweiterten Kalman-Filters berücksichtigt. Somit können mehrere Rückführungsschlei-

fen mit unterschiedlichen Abtastraten (inertiale Sensoren: 500Hz, Vision: 25Hz und GPS: 1Hz) verwendet werden.

Bei der Realisierung von diskreten Regelungssystemen erhält man aufgrund der Anbindung der realen Aktoren und Sensoren Totzeiten. Der diskrete Regler an sich verursacht durch das Halteglied 0-ter Ordnung eine Totzeit von einem Abtastschritt [84]. Das Problem des Aliasing bei Abtastsystemen erfordert meist vor der Abtastung einen Anti-Aliasing Filter, dessen Phasenlage eine weitere Totzeit bewirkt. Es entsteht daraus die Notwendigkeit, die Totzeiten zu kompensieren oder mit zu modellieren. Nur in Sonderfällen, z. B. [65], bei denen das Regelsystem robust genug ist, können die Totzeiten vernachlässigt werden.

In [4] werden totzeitbehaftete Messdaten mit einem Kalman-Filter synchronisiert. Dies funktioniert jedoch nur, wenn die Eigenwerte der Systemmatrix A echt stabil sind, ansonsten muss der Fehler, der durch die Totzeit entsteht, kompensiert werden (siehe 3.3.2.3). Die Totzeit kann nach [96] durch Einfügen von Pseudozuständen ins Systemmodell im Kalman-Filter berücksichtigt werden. Diese Methode erfordert aber eine genaue Kenntnis der Totzeit, zudem muss diese unbedingt konstant sein. Bei robusten einfachen Systemen wie z. B. [9] führt sogar die ungefähre Kenntnis der Totzeit mit dieser Methode zum Ziel.

In [14] wird die Totzeit mit einem adaptiven Kompensator, dem Smith Dead-time compensator, vor dem eigentlichen Regler eliminiert. Der Kompensator arbeitet nach der Methode der kleinsten Quadrate und erfordert das Regelgesetz analytisch sowie ein Referenzsignal. Dieses Verfahren ist also nur in Sonderfällen, z. B. Trackinganwendungen, einsetzbar. Ein weiterer Kompensator wird in [86] vorgestellt. Hier wird versucht, die Totzeit, die sich durch ein Halteglied 0-ter Ordnung ergibt, zu kompensieren. Da dieser Zero-Order-Hold-Kompensator differentielle Eigenschaften hat und somit das vorhandene Rauschen verstärkt, bringt dieses Verfahren trotz der Totzeitkompensation keine Verbesserung der Genauigkeit.

Bei bekannter und konstanter Totzeit kann diese auch im Regler kompensiert werden, wie es in [101] vorgeschlagen wird. Eine weitere Einschränkung ist, dass bei mehreren Sensoren mit dieser Methode alle die gleiche Totzeit besitzen müssten.

Wie schon weiter oben angemerkt, ist das Halteglied 1. Ordnung das einzige Verfahren mit dem eine Totzeitkompensation möglich ist und das auch in MatLab standardmäßig vorhanden ist.

2.6 Sensordaten-Vorverarbeitung

Außer den einfachen Standardverfahren, wie Lookup-Tabellen, Anti-Aliasing-Filter, Skalierung oder Koordinatentransformation gibt es weitere Verfahren, die sich speziell der Problematik der Sensorvorverarbeitung widmen. Hier ist hauptsächlich die Anpassung der physikalischen Einheiten durch Differentiation und Integration sowie die Kalibrierung und Offsetkorrektur gemeint.

Um die Rauscheinflüsse der numerischen Differentiation zu vermindern, wird in [9] ein Tiefpass nachgeschaltet und zur Unterdrückung von Störfrequenzen Notch-Filter eingesetzt. Anstelle der numerischen Differentiation verwendet man weniger rauschemp-

findliche IIR- und FIR-Filter [83], [48] und adaptive FIR-Filter [111]. Bei der numerischen Integration verwendet man statt den bekannten Standardverfahren Rechteckregel, Trapezregel und Keppler'sche Fassregel [100] in [83] ebenfalls einen FIR-Filter. Alle diese Verfahren haben das Problem, dass ein Offset am Integratoreingang zur Instabilität führt, bzw. dass die unbekannten Anfangszustände der Integratoren neue Offsets bewirken. In [6] behilft man sich in der Art, dass die Anfangsbedingungen mit anderen Messungen bestimmt werden. Die Verschiebung des Integratorpoles von der Stabilitätsgrenze in Richtung des stabilen Bereiches wird in [83] vorgeschlagen. Durch Verwendung einer Feedforward-Kalman-Filter-ähnlichen Struktur werden in [118] die Werte eines Beschleunigungssensors 2-fach integriert. In [75] wird zur Doppelintegration eine Fehler-Feedback-Kalman-Filterstruktur, die aus den Fehlerdifferentialgleichungen des Systems abgeleitet wurde, verwendet.

Die Standard-Integratoren können bei einem offsetfreien Eingangssignal wiederverwendet werden. In [83], [6] und [98] versucht man mit mäßigem Erfolg, den Offset mit Hochpässen filtern zu können. Probleme sind zum einen die Phasenlage und zum anderen der lange Einschwingvorgang. Eine weitere Alternative bietet der Bias-Beobachter aus [16], bei dem man aber auch mit langen Einschwingzeiten kämpfen muss. Dadurch, dass der Kalman-Filter mittelwertsfrei schätzt, kann nach [14] die Offsetunterdrückung mit einem Feedforward-Kalman-Filter im Regler erfolgen. Dies funktioniert jedoch nur, wenn man zusätzliche Einschränkungen trifft, wie z. B. in [24]. Hier wird bei der Navigation eines Fahrzeuges mit Beschleunigungssensordaten davon ausgegangen, dass die Geschwindigkeiten orthogonal zur Bewegungsrichtung Null sein müssen.

Durch eine gute Kalibrierung kann der Offset vermindert werden, ganz kann man ihn so nie unterdrücken, da der Offset in der Regel sich über die Zeit verändert. Den Verfahren zur Offsetunterdrückung wird aber damit insofern geholfen, dass ihre Einschwingzeiten aufgrund des kleineren Sprungs (es wird initial ein Offset von Null angenommen) kürzer werden. Bei Beschleunigungsensoren ist die Kalibrierung aufgrund der relativ hohen Erdbeschleunigung besonders wichtig, wie [57] zeigt. In [114] wird ein Verfahren zur hochgenauen Kalibrierung von 3D-Beschleunigungssensoren vorgestellt, in dem versucht wird, die Montagefehler zu bestimmen. Ein andere Methode ist die in [50] vorgeschlagene Parameterschätzung anhand des gemessenen Offsets mit einem zeitvarianten Kalman-Filter.

Bildverarbeitungsverfahren, die als Ergebnis einen Wert auf Signal-Level liefern[1], werden in dieser Arbeit zu den (komplexen) Sensoren gerechnet. Beispielhaft seien hier nur einige Algorithmen genannt.

- Template Matching zur Bestimmung des Versatzes zweier Objekte [10]
- Konturfolger [64], [91] zur Exploration und Navigation in Innenräumen mit dem Ansatz in [19]

Weitere Bildverarbeitungsalgorithmen finden sich in den in [19] enthaltenen Referenzen

[1] Die Berechnung dieser Werte erfolgt meist auf Pixel- oder Feature-Level

2.7 Fusion auf Signal-Level-Ebene

Methoden:

Die Fusion von Sensordaten mit der Methode des gewichteten Mittelwerts [62] ist nur für Sensoren gleicher Art geeignet. Die mathematische Betrachtung erfolgt in 3.4.1.

Die meiste Anwendung findet die Beobachterstruktur, hier vor allem der Kalman-Filter (siehe 3.3.2.3) mit all seinen Varianten und Abarten. In [75] werden 4 verschiedene Strukturen vorgestellt, in der der Kalman-Filter zur Fusion verwendet werden kann. Bei der Feedforward-Struktur befindet sich der Kalman-Filter im Vorwärtszweig, d.h. es werden nur gefilterte Werte ausgegeben. Bei der Feedback-Struktur wird der Kalman-Filter dagegen im Rückführzweig platziert. Der Vorteil ist hier, dass bei dieser Methode das System weiter funktioniert, wenn auch meist schlechter, wenn die Einheit ausfällt, die den Kalman-Filter berechnet. Desweiteren wird zwischen der Fehlerzustandsform und der Vollzustandsform unterschieden. Die letztere entspricht dem gewöhnlichen Kalman-Filter mit den Systemzuständen, aber die erste Form wird mit der Differenz zwischen realen und idealen Zuständen, den sogenannten Fehlerzuständen definiert. Der Kalman-Filter ergibt sich dann aus den Fehlerzustandsgleichungen und dient zur Korrektur der realen Systemzustände (siehe Bild 3.2.16).

Ein 3-stufiges auf Statistik basierendes Verfahren wird in [85] vorgestellt, die Vorgehensweise ist dabei folgende:

1. Test, ob die Daten der einzelnen Sensoren zusammenpassen. Dazu wird die „Pitman's Closeness" Technik verwendet.
2. Zusammenfassen der Daten von Sensoren, die am besten zusammenpassen mit dem „Complete Linkage" Algorithmus. Die anderen Daten werden verworfen.
3. Optimierung der noch verbleibenden Sensordaten mit der „Maximum-Likelyhood-Estimation".

Dieses Verfahren kann nur für gleichwertige Sensoren verwendet werden und eignet sich gut zum Erkennen von Ausreissern und defekten Sensoren.

In [61] wird mit dem „Bayesian Estimation Using Consensus Sensors" Verfahren eine ähnliche Methode vorgestellt, die auch wieder nur für redundante Sensoren in Frage kommt.

Zeytinoglu und Mintz [119] haben ein Verfahren entwickelt, das auf der „Statistical Decision Theory" basiert und für redundante Sensoren geeignet ist.

Anwendungen:

Eine wichtige Anwendung der Sensorfusion ist die Erstellung von 2D- bzw. 3D-Karten zur Umgebungsmodellierung. Bei diesen Anwendungen ist die zeitlich richtige Synchronisation nicht wichtig, es handelt sich um eine geometrische Sensorfusion. In [78] werden CCD-Kameradaten und Laser-Range-Daten fusioniert, um eine Karte zu erstellen, anhand der sich der autonome mobile Roboter lokalisieren kann. Zur Lokalisation und Exploration werden in [19] die Daten zweier CCD-Kameras für ein 3D-Umgebungsmodell fusioniert. In [17], [18] geht man noch einen Schritt weiter und

fusioniert Stereobilddaten mit Laser-Range Daten, um mit den weit verbesserten Daten eine Objekterkennung durchführen zu können. Eine andere Alternative, ein 3D-Modell zu generieren, wäre die Fusion von Sonar- und CCD-Kameradaten [2]. In [37] werden zur besseren Mensch-Maschine-Interaktion Sprachdaten mit CCD-Daten auf Pixel-Level-Ebene fusioniert um in einer 2D-Karte die Position des menschlichen Bedieners zu bestimmen.

Ein weiteres großes Anwendungsgebiet stellt die Navigation von autonomen mobilen Systemen (z. B. mobile Roboter, Fahrzeuge und Flugzeuge) dar. Die Fusion wird hier auf Signal-Level-Ebene durchgeführt, wobei eine zeitliche Synchronisation der Sensoren notwendig wird. In [112] werden Odometrie-Daten mit 3D-Positionsdaten, die mit Motion Stereo ermittelt wurden, mit einem Kalman-Filter fusioniert. Zur Navigation von Fahrzeugen auf unebenem Grund werden in [4] GPS-, Gyroskop-, Odometrie-, Rollwinkel- und Neigewinkeldaten ebenfalls mit einem Kalman-Filter fusioniert. Das besondere an diesem System ist, dass man vier verschiedene physikalische Einheiten hat. Für denselben Zweck werden in [13] GPS-, Radar- und Odometriedaten mit einem linearen bzw. nicht linearen Informationsfilter fusioniert. Ebenfalls zur genauen Navigation werden in [55] DGPS-, Gyroskop- und Geschwindigkeitsdaten mit einem komplexen Kalman-Filter fusioniert. Um mit dem Beschleunigungssensor basierten initialen Navigationssystem eines Minenfahrzeugs ohne Referenzierung bei der Doppelintegration (z. B. mit Landmarken, da GPS unterirdisch nicht funktioniert) auszukommen, werden in [63] verschiedene Sensorkombinationen untersucht. Die Fusion wird mit einem Kalman-Filter durchgeführt. Man kommt zu dem Ergebnis, dass man durch die Fusion zwar den Referenzierungstakt erniedrigen, jedoch nie ganz auf eine Referenzierung verzichten kann.

Im Rahmen des europaweiten Projekts PROMETHEUS (1994) wurde von Dickmanns [23] ein autonomer Pkw (VaMP und sein Vorläufer, der Lieferwagen VaMoRs) realisiert, bei dem der 4D-Ansatz eingesetzt wird und Daten von konventionellen inertialen Sensoren (Tacho, Beschleunigungssensoren, Gyroskope und Winkelgeber) mit Bilddaten (bifokale oder trifokale Anordnung) und GPS-Daten fusioniert. Der hier verwendete Ansatz kann auch auf Luftfahrzeuge, bei denen man 6 räumliche Freiheitsgrade hat, angewendet werden. Zum bordautonomen automatischen Landeanflug wird das Inertialsystem des Flugzeugs und Bilddaten mit einem Kalman-Filter kombiniert [96].

Zur Bestimmung der Orientierung eines autonomen mobilen Roboters fusioniert man in [104] Neigungssensoren und Kompass mit einem erweiterten Kalman-Filter. Die Neigung eines Fahrzeugs wird in [110] durch die Fusion von Beschleunigungssensoren, Gyroskop und Odometrie ermittelt. Die Fusion dient zur Schätzung der dynamischen Beschleunigung, die bei Bewegungen von der Erdbeschleunigung abgezogen werden muss.

Im Bereich der Automatisierung mit Industrierobotern wird heutzutage eher noch die Multisensorintegration eingesetzt, wie z. B. in [120]. Hier werden zur Annäherung an das Objekt CCD-Daten verwendet, doch zum Greifen wird auf Kraftrückkopplung mit einem Kraftsensor umgeschaltet. Es gibt jedoch auch auf diesem Gebiet Systeme mit Multisensorfusion, wie in [90], bei dem eine Mikropositioniereinrichtung mit drei verschiedenen Sensoren geregelt wird (siehe Kapitel 4) oder ein System zum Schweissen von unterschiedlich dicken Schweissnähten mittels Fusion von Vision- und Laser-Range-Sensordaten [62].

Zur 3D-Oberflächenerkennung werden in [109] Entfernungs- und Intensitätsbilddaten fusioniert und in [57] werden zur Gestenerkennung zwei 3D-Beschleunigungssensorsysteme verwendet, wobei durch deren geometrische Fusion die Gravitation zuverlässig extrahiert werden kann.

Im europäischen ESPRIT-Projekt AIMBURN werden Daten von physikalischen (z. B. Temperatur), optischen (z. B. Vision) und chemischen Sensoren (z. B. Sauerstoffkonzentration) fusioniert, um eine effiziente schadstoffarme Verbrennung bei Grossfeurungsanlagen zu erreichen.

In der Militärtechnik gibt es wie in [57] beschrieben ebenfalls Ansätze zur Sensordatenfusion. So soll durch eine Fusion auf Symbol-Level-Ebene von Radar-, Funk- und Videodaten ein Tracken der Truppenbewegungen auf einem Kampfschauplatz möglich werden.

In [42] werden zum Greifen von Objekten mit einem 6-Achs-Manipulator CCD-Kamera-Daten mit Modelldaten bzw. mit Gelenkwinkel-Daten fusioniert. Dabei wird eine Beobachter ähnliche Struktur verwendet. Eine Synchronisation der Sensordaten wird noch nicht durchgeführt.

3 Modellierung, Synchronisation und Fusion der Sensordaten

In Bild 2.7.1 ist das Blockschaltbild der Struktur zu sehen, auf der die entwickelte Toolbox basiert. Es handelt sich hierbei um die Erweiterung des klassischen 1-Sensor-Regelkreis mit Regler und Streckenmodell im Vorwärtszweig und den Sensor im Rückführungszweig. Zu den Sensoren befinden sich jetzt die funktionellen Blöcke zur Synchronisation (Abschnitt 3.3), Signalvorverarbeitung (Abschnitt 3.2) und Fusion (Abschnitt 3.4) der Sensoren im Rückwärtszweig. Würde man die Sensoren und die funktionalen Blöcke zu einem Superblock gruppieren, hätte man wieder den klassischen 1-Sensor-Regelkreis, jedoch jetzt mit einem komplexeren Sensor. Dies ist auch das Ziel der Sensorfusion auf Signal-Level, mehrere Sensoren so zu fusionieren, dass sie sich wie ein Sensor mit verbesserten Daten verhalten.

Der Regler kann in zeitdiskreter Zustandsdarstellung, zeitdiskreter Pol-Nullstellendarstellung oder mit seiner Übertragungsfunktion angegeben werden. Wichtig ist, dass die Abtastrate f_a des Reglers angegeben wird, da sie die Abtastrate des gesamten Regelsystems darstellt und so die funktionalen Blöcke beeinflusst. Die Strecke kann als beliebiges LTI-Objekt (linear time invariant) angegeben werden, dabei ist es unerheblich, ob es sich um eine zeitkontinuierliche oder um eine äquivalente zeitdiskrete Darstellungsform handelt. Zudem muss die Kovarianz des Prozessrauschens angegeben werden.

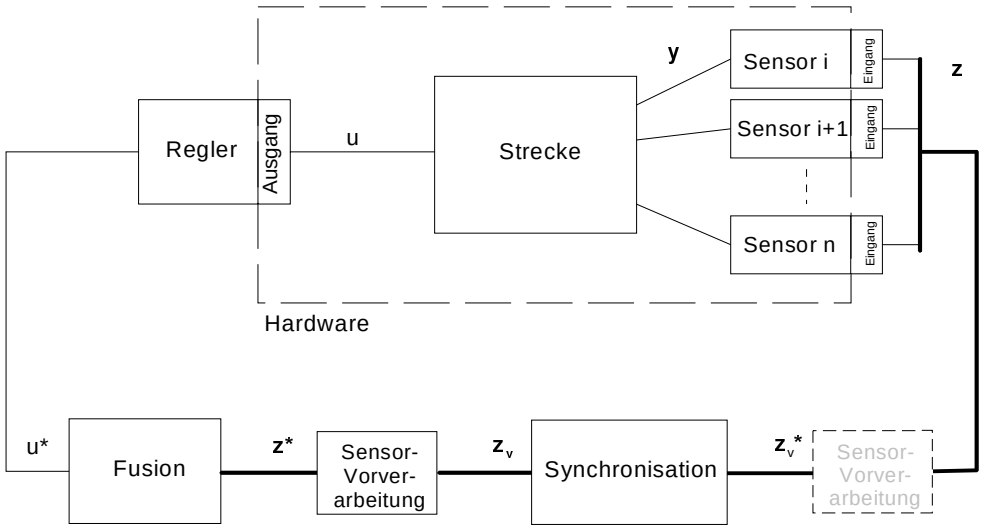

Bild 2.7.1 : Systemarchitektur

Für eine Software-Simulation (Abschnitt 3.5) wird ein Modell der Regelstrecke verwendet, wobei die Eigenschaften der Sensoren mitmodelliert werden. Es werden in den Sensormodellen (Abschnitt 3.1) nicht nur die dynamischen Eigenschaften, sondern auch das Abtastverhalten, die eingeschränkte Genauigkeit durch Rauschen und Stö-

rungen, die Auflösung und evtl. systematische Fehler berücksichtigt. Zur Strecke wird ebenfalls die Ausgabeeinheit des Reglers gezählt. Dies ist üblicherweise ein DA-Wandler, dessen Quantisierung und Zeitverzögerung mitmodelliert werden.

Mit dieser Architektur ist auch eine Hardware-in-the-Loop-(HIL)-Simulation (Abschnitt 3.6), bei der die reale Strecke und/oder die realen Sensoren verwendet werden, mit dem gleichen Systemmodell möglich. Hier verbergen sich dann hinter den betreffenden Blöcken nur noch die Hardware-Interfaces der einzelnen Komponenten. Der Wechsel zwischen HIL- und Software-Simulation kann somit automatisch erfolgen.

Die Realisierung dieser Architektur erfolgt auf der Basis von MatLab/SimuLink und stellt somit eine Toolbox bzw. einen Blocksatz zur Sensorfusion auf Signal-Level dar. In SimuLink-Bibliotheken können die verschiedenen Verfahren der funktionalen Blöcke aus Bild 2.7.1 abgelegt werden. Die Toolbox kann somit leicht mit neuen Verfahren erweitert werden. Der hierarchische Aufbau der Toolbox ist in Bild 2.7.2 zu sehen, wobei die einzelnen funktionalen Blöcke deutlich sichtbar werden.

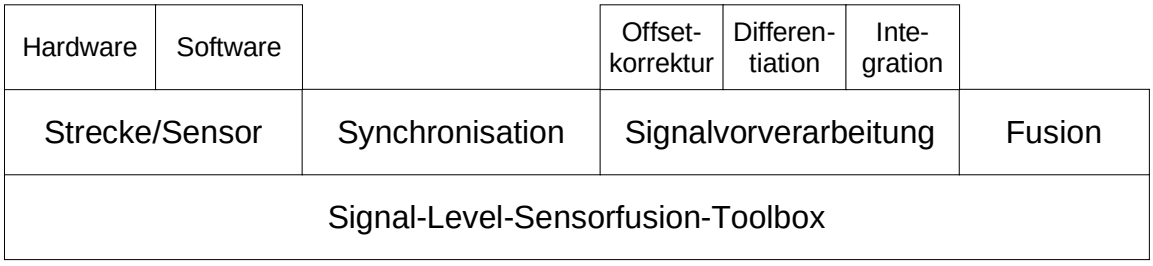

Bild 2.7.2 : hierarchischer Aufbau der Toolbox

Durch die Realisierung auf Basis von MatLab/SimuLink können die damit entwickelten Systeme auf verschiedene Zielsysteme implementiert werden. Es kann dabei der C-Code-Generator von MatLab (Real-Time Workshop) verwendet werden.

Zusammenfassend kann man mit der Toolbox, die auf dieser Architektur basiert, ein Multisensor-Regelsystem komplett, zuerst in Software, dann mit realer Hardware simulieren und abschließend die endgültige Implementierung auf dem gewünschten Zielsystem durchführen.

In diesem Kapitel gelten folgende Bezeichnungen:

t_a	Abtastperiode des Regelsystems
t_{a1}	Abtastperiode des zyklischen Sensorsystems
$t_s(n)$	n-ter Zeitstempel des Sensorsystems
t_k	k-ter Abtastschritt $t_k = k \cdot t_a$;
σ	Rauschamplitude
$\hat{}$	synchronisierter Schätzwert
$\sim$	nicht synchronisierter Schätzwert

3.1 Klassifikation von Sensoren

Die Klassifikation der Sensoren erfolgt in dieser Arbeit an Hand des von den Sensoren verwendeten Abtastverhaltens. Dies ist die wichtigste Eigenschaft hinsichtlich der zeitlich richtigen Kopplung von Sensoren.

Das Abtastverhalten wird zum einen charakterisiert durch den Abstand der einzelnen Messwerte und zum anderen durch die Häufigkeit der Messwerte. Liegt ein äquidistanter Abstand vor, so spricht man von einem zyklischen Sensor, andernfalls von einem asynchronen. Sind die Messwerte nur in einem bestimmten Zeitraster verfügbar, spricht man von einem diskreten Sensor; können dagegen Messwerte zu jedem beliebigen Zeitpunkt aufgezeichnet werden, hat man einen kontinuierlichen Sensor. In Bild 3.1.1 ist dieser Sachverhalt veranschaulicht.

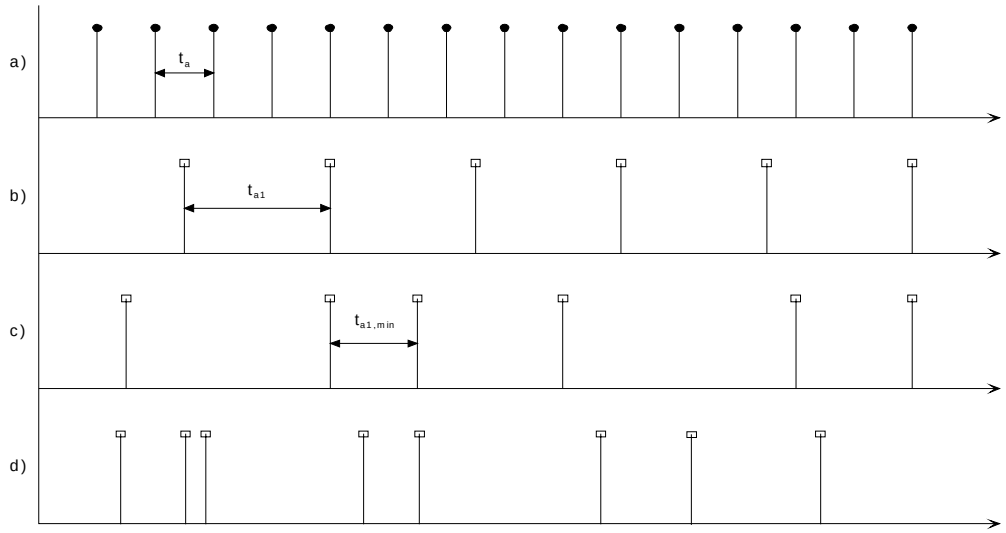

Bild 3.1.1 : Sensorarten: a) kontinuierlich zyklisch b) diskret zyklisch c) diskret asynchron d) kontinuierlich asynchron

Kombiniert man die Möglichkeiten der beiden Eigenschaften, so kann man 4 Sensorgruppen unterscheiden. Tabelle 3.1.1 gibt einen Überblick über die Einteilung und qualitative Klassifikation dieser Sensorarten. In den folgenden Unterpunkten werden Simulationsmodelle und Systembeschreibungen für die einzelnen Sensorklassen entwickelt. Es wird dabei besonders darauf geachtet, dass eine anschließende Synchronisation und Fusion der Messwerte einfach möglich wird. Es wird deshalb von jedem Sensor gefordert, dass der Zeitpunkt der Messwertaufnahme mit einem Zeitstempel festgehalten wird. Sollte dies beim realen Sensor nicht möglich sein, so muss zumindest gesichert sein, dass man auf den Zeitpunkt der Messwertaufnahme zurückschließen kann. Zum Beispiel kann man auch den Zeitpunkt des Eintreffens der Messwerte festhalten, wenn man die konstante Verzögerung gegenüber der Messwertaufnahme (Wandlungszeit des A/D-Wandler, Umschaltzeit des Multiplexers, ...) vorab bestimmen kann (siehe 3.1.5). In den folgenden vorgestellten Simulationsmodellen wird jedenfalls immer ein Zeitstempel mit ausgegeben.

Sensorart	Qualität der Daten	Abtastfrequenz	Totzeit	Datenmenge
kontinuierlich zyklisch	Rohdaten	hoch	gering (0,1 μs)	hoch
diskret zyklisch	aufbereitete Rohdaten	mittel bis hoch	mittel (0,1 ms)	mittel bis hoch
diskret asynchron	intelligent vorverarbeitete Daten	niedrig	groß 0,1s	niedrig
kontinuierlich asynchron	binäre Signale	-	Interrupt-Latenzzeit	-

Tabelle 3.1.1 : Sensorklassifikation

3.1.1 Kontinuierlicher zyklischer Sensor

Dieser Sensortyp spiegelt die Gruppe von analogen Sensoren wieder. Sie besitzen nur wenig Intelligenz und liefern üblicherweise reine Rohdaten. Diese Sensoren sind physikalisch direkt mit dem A/D-Wandler des Regelsystems verbunden und werden deshalb mit der höchsten Abtastrate, der Abtastrate des Regelsystems, zyklisch abgetastet. Typische Vertreter dieses Sensortyps sind z. B. positionsempfindliche Dioden (PSD), Thermoelemente, Beschleunigungssensoren usw..

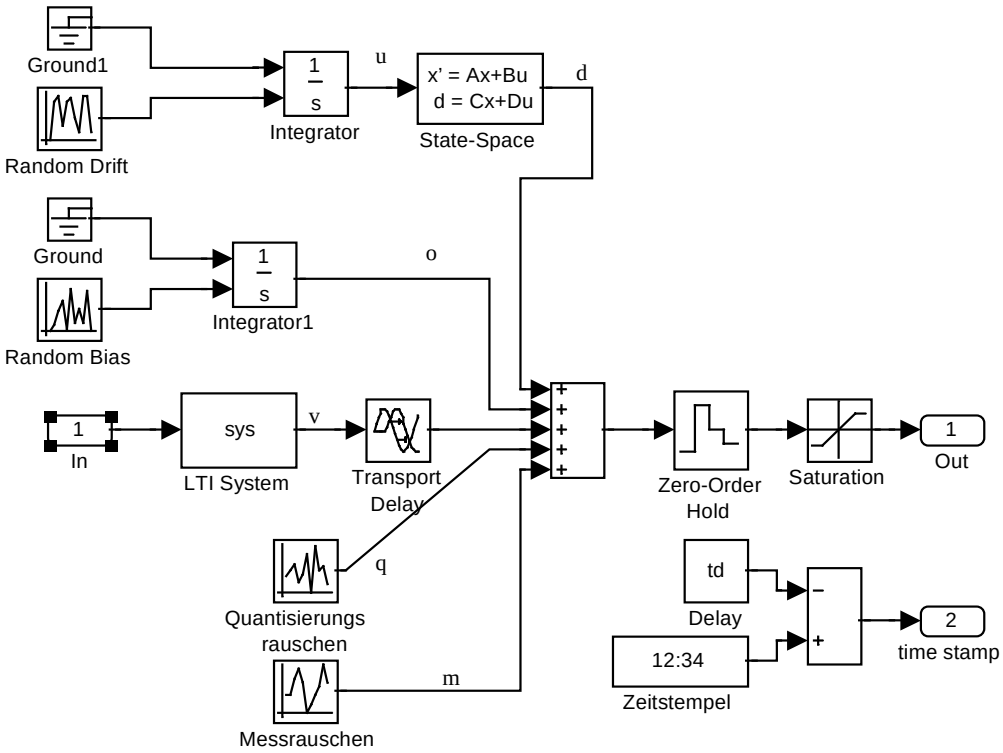

Bild 3.1.2 : kontinuierlicher zyklischer Sensor

In Bild 3.1.2 ist das Simulationsmodell für diesen Sensortyp zu sehen. Es besteht aus einem LTI-System (lineares zeitinvariantes System), das die ideale Übertragungsfunkti-

on des Sensors in zeitkontinuierlicher Zustandsdarstellung beschreibt. Hinzu kommen Blöcke, die die Störungen und Messfehler simulieren. Da wäre zum einen das Messrauschen, das als ein normalverteiltes Rauschen mit Mittelwert $\overline{m} = 0$ und Varianz σ_m angenommen wird. Das Quantisierungsrauschen, hervorgerufen durch die Bit-Auflösung des A/D-Wandlers, wird mit folgender Gleichung nach [31] spezifiziert.

$$\sigma_q = \frac{2^{-bits}}{\sqrt{12}}; \overline{q} = 0; \tag{3.1}$$

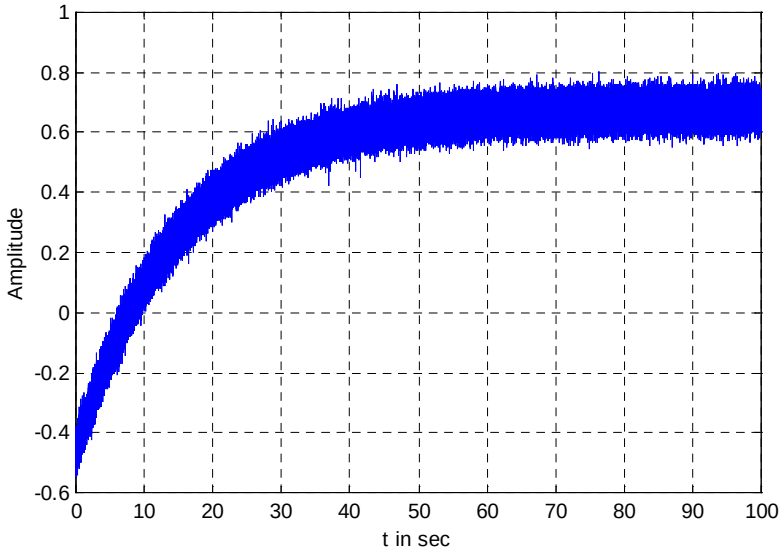

Bild 3.1.3 : Sensorausgang bei konstantem Eingang

Sensoroffsets werden mit offenen Integratoren und normalverteilten Anfangsbedingungen ($\overline{o}, \sigma_o$) angegeben um eine Beschreibung in Zustandsdarstellung zu ermöglichen [75]. Sensoroffsets bei dieser Klasse von Sensoren spiegeln entweder systematische Messfehler wie z. B. Erdbeschleunigung bei Beschleunigungssensoren oder systematische Sensorfehler wie z. B. Spannungsoffset des Ausgangsoperationsverstärkers wieder. Die Sensordrift wird nach [75], [7] mit folgender Differentialgleichung 1.Ordnung beschrieben:

$$\dot{d} = -\frac{1}{T} \cdot d + \frac{C1}{T}; \tag{3.2}$$

Oder in Zustandsdarstellung als

$$\begin{aligned} \dot{x} &= -\frac{1}{T} \cdot x + 1 \cdot u; \\ d &= 1 \cdot x; \end{aligned} \tag{3.3}$$

wobei hier als Erweiterung der konstante Driftendwert $u = \frac{C1}{T}$ mit $\overline{u}, \sigma_u$ normalverteilt ist. Die Drift, die mit (3.2)(3.3) beschrieben wird, ist in erster Linie die Warm-Up-Phase des Sensors, die sich über mehrere Stunden hinziehen kann. Der Transport-Delay-Block spiegelt die Verzögerungszeit durch die A/D-Wandlung und den Datentransport

wieder. Der Zero-Order-Hold-Block simuliert das Abtasten durch den A/D-Wandler mit der Abtastfrequenz des Regelsystems. Der Sättigungsblock simuliert den beschränkten Messbereich des Sensors, der entweder durch die Sensorbauart oder durch den Wertebereich des A/D-Wandlers bestimmt wird. Mit dem Zeitstempel wird der Zeitpunkt der Messwertaufnahme festgehalten; dies ist für eine zeitgenaue Synchronisation mit anderen Sensoren notwendig.

Die gesamte Systembeschreibung ergibt sich mit den Bezeichnern aus Bild 3.1.2 und (3.3) zu:

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_{sys} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 1 & -1/T & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & A_{sys} \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_{sys} \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ B_{sys} \end{bmatrix} \cdot in;$$

$$out = \begin{bmatrix} 0 & 1 & 1 & C_{sys}^T \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_{sys} \end{bmatrix} + D_{sys} \cdot in + q + m; \qquad \textbf{(3.4)}$$

wobei in A_{sys} bereits der Delay berücksichtigt wird.

In Bild 3.1.3 ist das simulierte Ausgangssignal eines Sensors bei konstanter Anregung zu sehen. Die Zeitkonstante der Sensordrift ist 15 Sekunden.

3.1.2 Diskreter zyklischer Sensor

Diese Art von Sensoren zeichnet sich dadurch aus, dass sie eine eigene Abtastrate vorgeben und mit dieser zyklisch Messwerte zur Verfügung stellen. Diese Eigenschaft haben entweder Sensoren mit digitalem Interface wie z. B. Sigma-Delta-Wandlerausgänge oder Sensoren, die einen Mikrocontroller „on-board" haben. Die Messwerte können wieder Rohdaten, aber auch bereits mit einfachen Algorithmen vorverarbeitete Daten sein (z. B. Skalierung, Kalibrierung). Als Beispiel für einen solchen Sensor kann der triaxiale Beschleunigungssensor des Forschungszentrums Karlsruhe [115] mit seinen 3 on-board Sigma-Delta-Wandlern genannt werden. Zur Dezimierung der hohen Abtastrate (512-fache Überabtastung) und zur Erhöhung der Genauigkeit wird ein in einem FPGA implementiertes digitales Filter eingesetzt, das dann die endgültige Abtastrate vorgibt. Eine Erweiterung dieses digitalen Filters mit Vorverarbeitungs-Algorithmen wie Eichung und Offset-Reduktion wären möglich.

Bild 3.1.2 stellt auch für diesen Sensortyp das Simulationsmodell dar. Der einzige Unterschied zum analogen Sensor ist, dass das LTI-System jetzt in einer zeitdiskreten Zustandsdarstellung vorliegen muss und das Halteglied mit der Abtastrate des Sensors und nicht mit der des Regelsystems betrieben wird. Je nach Grad der Messwert-Vorverarbeitung können Störquellen wie Drift oder Offset weggelassen werden.

Die gesamte Systembeschreibung ergibt sich dann ebenfalls zu (3.4), wobei die genannten Unterschiede zum analogen Sensor zu beachten sind.

3.1.3 Diskreter asynchroner Sensor

Diese Art von Sensoren ist dadurch charakterisiert, dass der zeitliche Abstand der Messwerte nicht konstant ist, jedoch ein minimaler Abstand existiert. Es handelt sich hierbei um komplexe Sensoren mit einer intelligenten Sensordatenvorverarbeitung, wie z. B. CCD-Kamera mit Bildverarbeitung. Aufgrund der aufwendigen Datenvorverarbeitung, die auf einem eigenen Mikroprozessorsystem durchgeführt wird, treten große Totzeiten auf, jedoch werden Störungen wie Drift- und Offsetfehler ausgeglichen. Die Asynchronizität der Messdaten ergibt sich durch die Abhängigkeit der Verarbeitungszeit der Vorverarbeitung von den Sensorwerten. Zudem ist dadurch dann das Messrauschen abhängig von der Verarbeitungszeit.

Nimmt man das Beispiel CCD-Kamera und Bildverarbeitung, werden diese Eigenschaften deutlicher. Die Bildverarbeitung, wie die Kantenextraktion nach [91] bzw. das Template-Matching nach [10], ist sehr stark abhängig vom Bildinhalt und es existiert eine Mindestverarbeitungszeit (z.B. Bildeinzug, Operatoren angewandt auf das ganze Bild). Der variable Beitrag ergibt sich durch die Segmentierung und Zusammenfassung der Kanten [89].

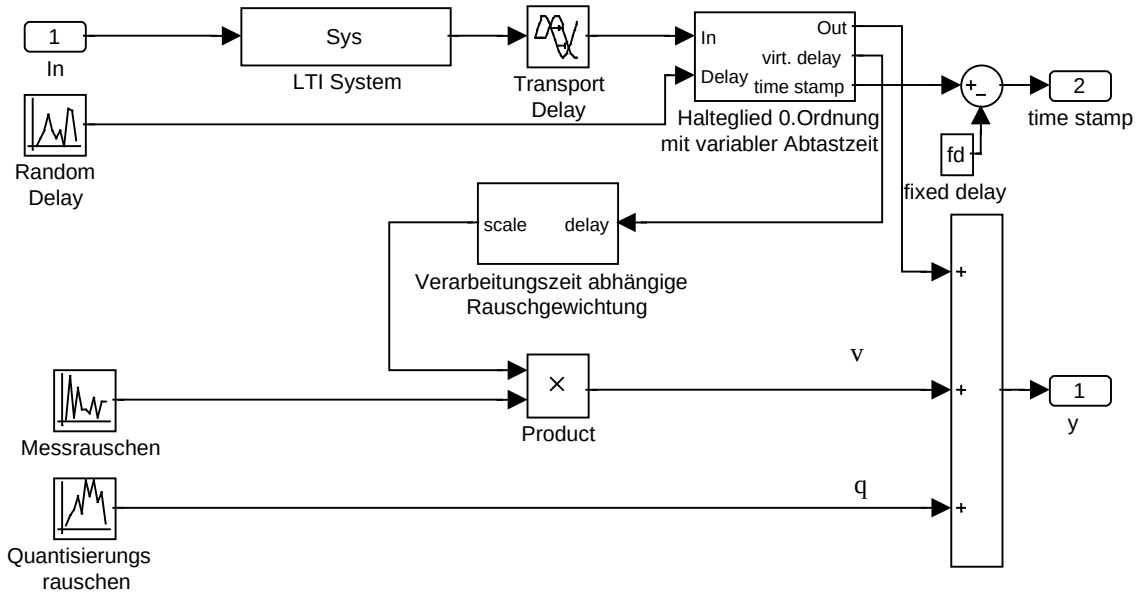

Bild 3.1.4 : asynchroner diskreter Sensor

Erhält man die Ergebnisse schon kurz nach der Mindestverarbeitungszeit, so ist der Bildinhalt nicht sehr aussagekräftig und die berechneten Ergebnisse sind ungenau. Wird die benötigte Verarbeitungszeit größer, so steigt auch die Qualität der Ergebnisse. Führt man aber wie in [64] wegen Echtzeiteigenschaften eine Obergrenze der Verarbeitungszeit ein, bei deren Überschreitung das Verfahren abgebrochen wird, sinkt die Qualität der Ergebnisse wieder.

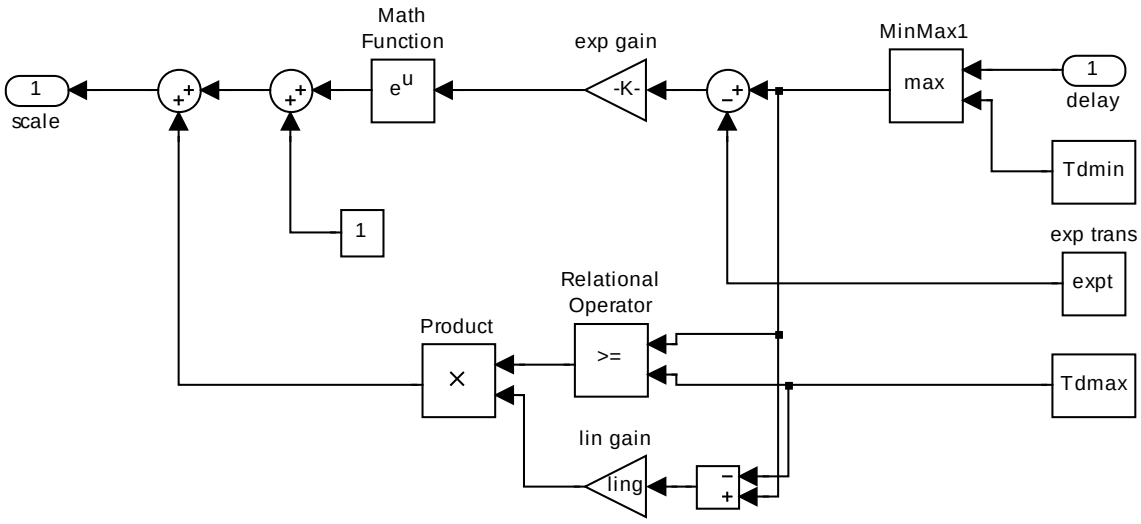

Bild 3.1.5 : Verarbeitungszeit abhängige Rauschgewichtung

Im Simulationsmodell dieses Sensortyps (Bild 3.1.4) werden diese Ungenauigkeiten durch Skalierung des Messrauschens in Abhängigkeit von der benötigten Verarbeitungszeit berücksichtigt. Die Skalierung des Messrauschens übernimmt dabei der „Verarbeitungszeit abhängige Rauschgewichtungs"-Block, dessen Struktur in Bild 3.1.5 zu sehen ist. Eine mögliche Rauschgewichtung, die mit diesem Block generiert wird, ist in Bild 3.1.6 dargestellt.

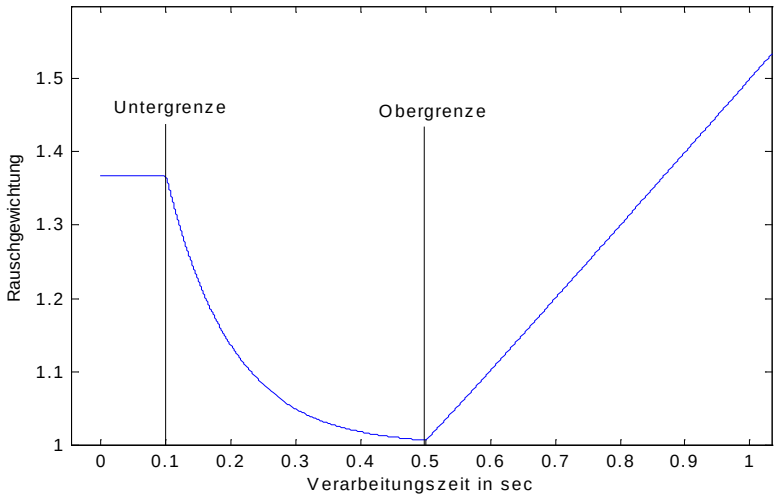

Bild 3.1.6 : Verarbeitungszeit abhängige Rauschgewichtung (Beispiel)

Die Sensorübertragungsfunktion kann wieder mit einem zeitdiskreten LTI-System in Zustandsbeschreibung angegeben werden, aber im Hinblick auf die intelligente Sensorvorverarbeitung kann diese meist gleich „1" zu setzen sein. Die konstante Mindestverarbeitungszeit wird durch den Delay-Block nachgebildet. Die additive, variable normalverteilte Verarbeitungszeit (für Bildverarbeitung siehe z.B. [18]) wird mit dem „Random Delay"-Block erzeugt und die Messwerte (z.B. Bildverarbeitungsergebnisse) dementsprechend im „Halteglied 0.Ordnung mit variabler Abtastzeit"-Block (Bild 3.1.7) verzögert abgetastet und gehalten. Dieser Block generiert auch den nach außen geführten Zeitstempel und erlaubt die Angabe einer Verarbeitungszeit-Obergrenze.

Wird eine Verarbeitungszeit größer dem maximalen Wert eingespeist, so wird der Messwert mit der maximalen Verarbeitungszeit abgetastet, jedoch der eingespeiste Wert für die Skalierung des Messrauschens verwendet. Ungenauigkeiten, die durch die Zahlendarstellung verursacht werden, können wieder mit (3.1) berücksichtigt werden.

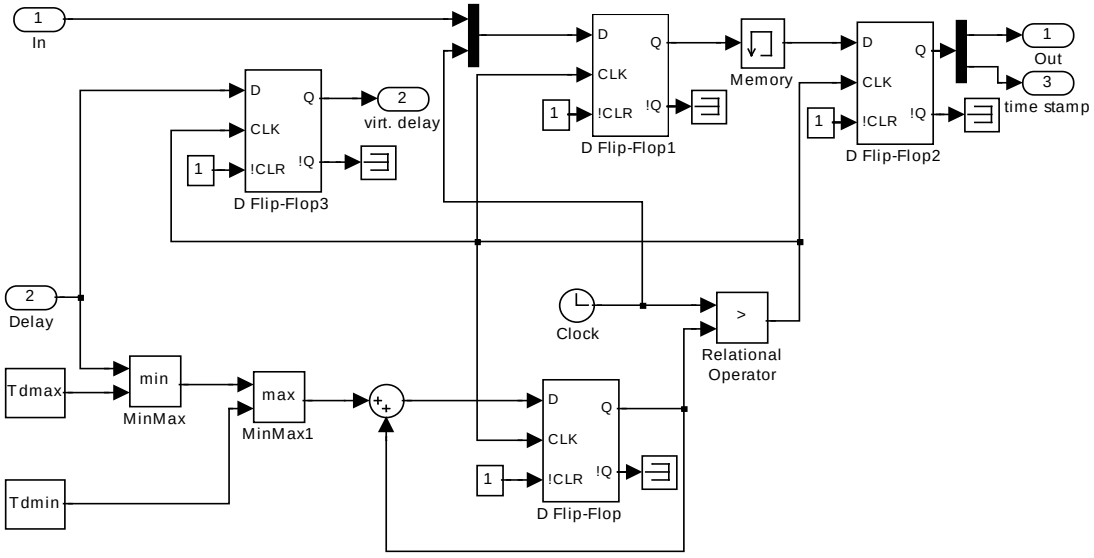

Bild 3.1.7 : Halteglied 0.Ordnung mit variabler Abtastzeit

In Bild 3.1.8 ist beispielhaft das Simulationsergebnis für die asynchrone Abtastung eines 0.1 Hz Signals mit 0.1s minimaler und 0.5s maximaler Verarbeitungszeit und einem Mittelwert von 0.3s dargestellt.

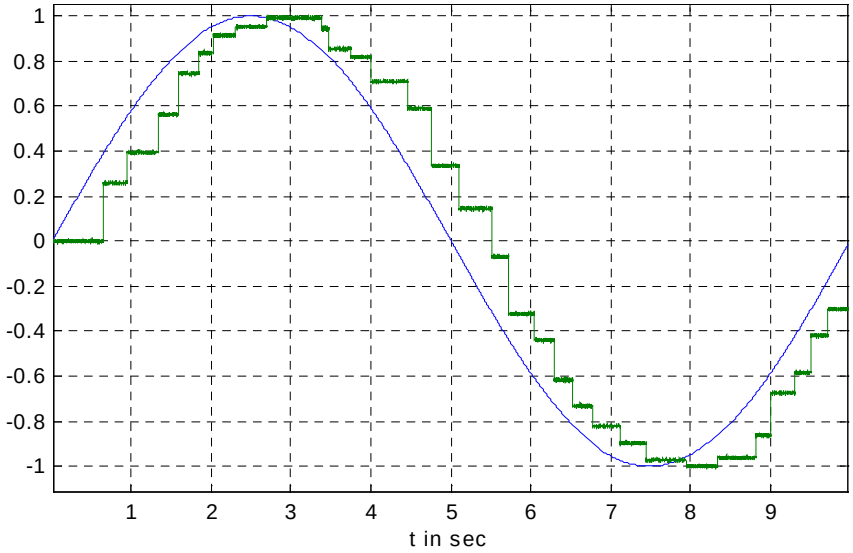

Bild 3.1.8 : asynchrone Abtastung mit variabler Totzeit

Die Zustandsbeschreibung des gesamten Sensormodells ergibt sich mit den Bezeichnern aus Bild 3.1.4 und eigenen Systemmodellen für die Totzeit (delay) und der Gewichtung des Messrauschens (noise) zu

$$\begin{bmatrix} \dot{x}_{sys} \\ \dot{x}_{delay} \\ \dot{x}_{noise} \end{bmatrix} = \begin{bmatrix} A_{sys} & 0 & 0 \\ B_{delay} \cdot C_{delay}^T & A_{delay} & 0 \\ 0 & 0 & A_{noise} \end{bmatrix} \cdot \begin{bmatrix} x_{sys} \\ x_{delay} \\ x_{noise} \end{bmatrix} + \begin{bmatrix} B_{sys} \\ B_{delay} \cdot D_{sys} \\ 0 \end{bmatrix} \cdot in + \begin{bmatrix} 0 \\ F_{delay} \\ B_{noise} \end{bmatrix};$$
$$y = \begin{bmatrix} 0 & C_{delay}^T & C_{noise}^T \end{bmatrix} \cdot \begin{bmatrix} x_{sys} \\ x_{delay} \\ x_{noise} \end{bmatrix} + \begin{bmatrix} G_{delay} + D_{noise} \end{bmatrix} \cdot v + q; \quad \textbf{(3.5)}$$

3.1.4 Kontinuierlicher asynchroner Sensor

Bei diesem Typ von Sensoren handelt es sich nicht um Sensoren im herkömmlichen Sinne, sondern um binäre Signale, die asynchron vom Regelsystem wahrgenommen werden können und sofort eine Änderung im Kontrollfluss bewirken. Diese Signale müssen somit nicht synchronisiert werden und erfordern deshalb auch keinen Zeitstempel.

Es werden im folgenden zwei verschiedene Arten dieser Signale unterschieden.

3.1.4.1 Triggerimpuls

Triggersignale werden verwendet, um Modi im Regelungssystem umzuschalten. Der Zeitpunkt dieser Signale ist spezifiziert, kann aber aufgrund von äußeren Einflüssen streuen. Der Triggerzeitpunkt *tt* wird mit $\bar{t} = tt$ und mit σ_t normalverteilt spezifiziert und der Triggerimpuls mit dem Modell von Bild 3.1.9 simuliert.

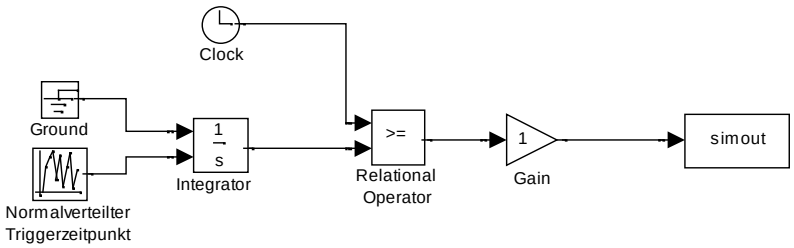

Bild 3.1.9 : Sensormodell Triggerimpuls

3.1.4.2 Events

Events sind Ereignisse, die zu zufälligen nicht vorhersehbaren Zeiten stattfinden und zu einer Ausnahmebehandlung führen. Als Beispiel wären hier Grenzwertüberschreitung oder Komponentenausfälle zu nennen.

Ereignisse, die zu zufälligen Zeiten eintreten können, unterliegen der Poisson-Verteilung. Die Wahrscheinlichkeitsdichte-Funktion der Poisson-Verteilung ist in [99] definiert als

$$y = f(x|\lambda) = \frac{\lambda^x}{x!} e^{-\lambda}; \quad \textbf{(3.6)}$$

wobei λ die Anzahl der Events pro Zeiteinheit bezogen auf die Abtastrate zu

$$\lambda = \frac{events \cdot Ta}{T}; \qquad \textbf{(3.7)}$$

angibt.

Im Simulationmodell (Bild 3.1.10) wird in jedem Abtastschritt des Regelsystems eine Zufallszahl, die der Poisson-Verteilung unterliegt, mit dem gegebenen Lambda generiert. Ist diese Zahl größer als Null, so tritt in diesem Abtastschritt ein Ereignis ein.

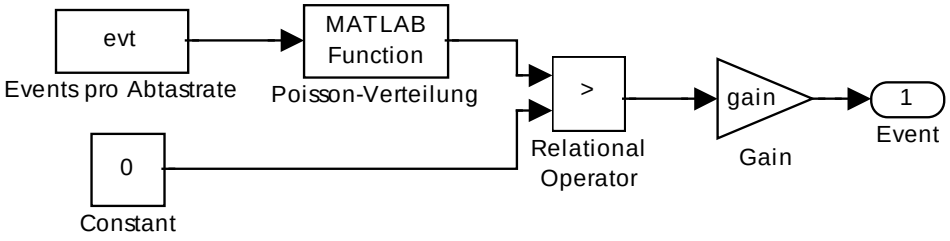

Bild 3.1.10 : Sensormodell Event

In Bild 3.1.11 ist eine Folge von simulierten zufälligen Poisson verteilten Ereignissen mit spezifizierten zwei Ereignissen pro Sekunde zu sehen.

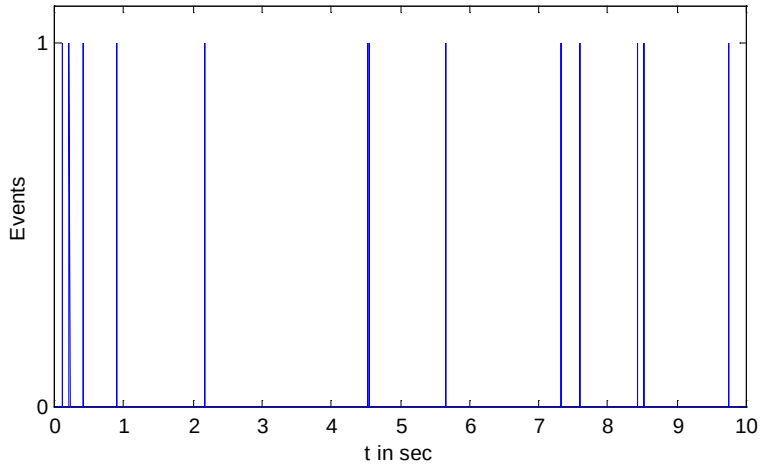

Bild 3.1.11 : zufällige Events

3.1.5 Modellidentifikation

Die Bestimmung der Modellparameter kann entweder aus der Sensorspezifikation z.B. [115] oder aus experimentellen Messdaten geschehen. Am schwierigsten gestaltet sich bei letzterer Methode die Identifikation der Übertragungsfunktion des Sensors. Hier müssen dann System-Identifikationsverfahren z.B. nach [59] und Tools wie die System Identification Toolbox von MatLab [60] eingesetzt werden. Die Drift kann nach [66] einfach (auch grafisch) aus den experimentellen Daten ermittelt werden, indem man die Sensorwerte bei nicht angeregtem Sensor über einen längeren Zeitraum aufzeichnet. Der Offset kann, nachdem die Drift abgeklungen ist (meist nach dem Erreichen der Betriebstemperatur), ebenso bestimmt werden. Das Messrauschen ergibt sich dann aus den Abweichungen zum bestimmten Offset.

Die Totzeit kann entweder mit der Kreuzkorrelation zwischen Referenzdaten und Messdaten oder mittels einer separaten hochgenauen Kalibriermessung mit Logicanalyzer und Oszilloskop bestimmt werden.

3.2 Vorverarbeitung

In diesem Abschnitt wird näher auf die Verarbeitung der Sensordaten vor der Fusion eingegangen. Hierzu zählt jedoch nicht die on-chip oder on-sensor Vorverarbeitung der Sensortypen 2 und 3, auch wenn es sich hierbei um ähnliche (evtl. auch ausgelagerte) Algorithmen handelt, da diese bereits im Sensormodell (siehe 3.1.2 und 3.1.3) berücksichtigt werden. Die Vorverarbeitung auf der Regelungseinheit ist notwendig zur

- Kalibrierung des DA-Wandlers mit $z^* = (z - z_{Nullpunkt}) \cdot \frac{Wertebereich}{2^{DA\_Bits}}$
- Koordinatentransformation der Sensorwerte in das Regelungskoordinatensystem
- Korrektur von nicht dynamischen systematischen Fehlern (Kalibrierung des Sensors), wie z. B. Linsenfehlerkorrektur nach [58] oder Montagefehlerkorrektur nach [114]
- Korrektur von veränderlichen oder unbekannten systematischen Fehlern wie Alterungseffekte und unbekannten Anfangszuständen
- Anpassung der physikalischen Einheiten durch Integration oder Differentiation. In manchen Fällen kann es vorkommen, dass das gewünschte Signal ohne größeren Aufwand nicht direkt oder nicht mit ausreichender Güte gemessen werden kann, das integrierte oder differenzierte Signal jedoch schon. Als Beispiel kann die Messung der Beschleunigung anstatt der Position für ein inertiales Navigationssystem oder die Messung der Robotergelenkposition anstatt der Gelenkgeschwindigkeit angeführt werden.

Die ersten drei Punkte stellen keine Veränderung der dynamischen Eigenschaften der Strecke dar, so dass eine genauere Betrachtung im Rahmen dieser Arbeit unterbleibt. Zudem können für diese Punkte Standardverfahren angewandt werden. Im folgenden werden Verfahren vorgestellt, die zur Bewältigung der letzten beiden Punkte verwendet werden können.

3.2.1 Offsetunterdrückung

Gegeben sei der Sensorwert z, der mit einem additiven, normal verteilten Rauschen $n(t)$ und einem Offset off überlagert ist. Ziel dieser Verfahren ist es, den ungestörten Sensorwert $\hat{z}$ zu bestimmen. In (3.8) ist dieser Zusammenhang dargestellt.

$$z = \hat{z} + off + n(t); \qquad \textbf{(3.8)}$$

Um auch über lange Zeit sich verändernde Offsets ausgleichen zu können, wird $off = off(t)\ mit\ T >> 1s$ angenommen.

3.2.1.1 Hochpassfilterung

Wie bereits unter 2.3 erwähnt, wird in [83] und [6] zur Unterdrückung des Offsets ein digitaler Hochpass vorgeschlagen. In der Realität stellt die Wahl des Filtertyps und der Filterparameter ein Optimierungsproblem mit zwei hypothetischen gegensätzlichen Forderungen dar.

1. unendliche Dämpfung des Offsets bei 0Hz
2. keine Dämpfung und Phasenverschiebung des Nutzsignals

Die gefundenen Filterparameter und Filtertypen werden immer einen schlechten Kompromiss der beiden Forderungen darstellen, denn keine der Forderungen kann je zu 100% erfüllt werden.

In [98] wird für das Anwendungsbeispiel aus Kapitel 4 ein Butterworth-Filter 4. Ordnung als geeignetster Filtertyp vorgeschlagen, so dass für den Vergleich der Verfahren dieser Filtertyp mit einer Eckfrequenz von $\frac{f_0}{f_a} = \frac{1Hz}{2000Hz}$ verwendet wird. In Bild 3.2.1 ist das Bode-Diagramm dieses Filters dargestellt. Man kann die endliche Dämpfung bei 0Hz sowie die enorme Phasenverschiebung im Nutzsignalbereich sehr gut erkennen. Eine Phasenkorrektur durch einen Allpassfilter, wie ebenfalls in [83] vorgeschlagen, lässt sich immer nur für eine Frequenz erreichen.

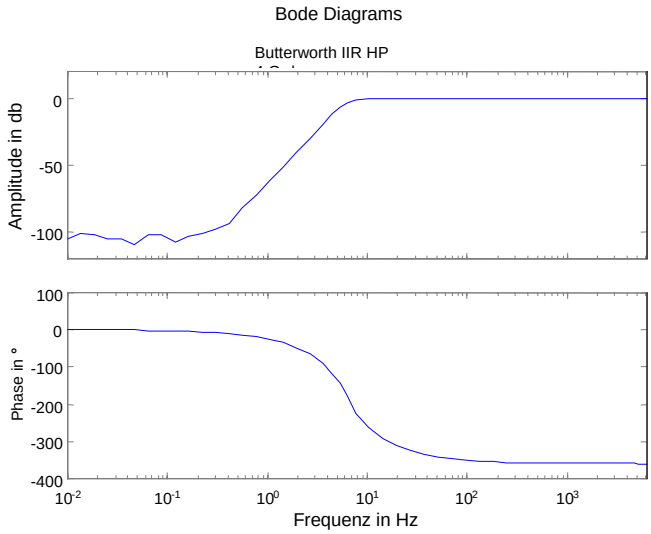

Bild 3.2.1 : Bode Diagramm Butterworth IIR Hochpass 4.Ordnung

3.2.1.2 Gefensterter Mittelwert

Die Offsetkorrektur bei diesem Verfahren erfolgt durch Subtrahieren des Wertes der diskreten Fourier-Transformation (DFT) bei 0Hz, der dem Mittelwert über das DFT-Fenster entspricht. Die Länge des DFT-Fensters, d. h. die Anzahl der vergangenen Messwerte *N*, die zur Mittelwertbildung herangezogen werden, bestimmt die spektrale Auflösung $\Delta f = f_a / N$ der DFT.

Der Offset ergibt sich im k-ten Schritt dann zu

$$off_k = \frac{1}{\min(k,N)} \sum_{k}^{k-1-\min(k,N)} z_i \qquad \textbf{(3.9)}$$

und kann nach *N* Messwerten vereinfacht wie folgt berechnet werden.

$$off_k = off_{k-1} + \frac{z_k - z_{k-N}}{N}; \qquad \textbf{(3.10)}$$

Für den Vergleich der Verfahren wurde eine spektrale Auflösung von 1Hz verwendet, was bei einer Abtastrate von f_a=2000Hz einer Datenpufferlänge von 2000 Werten entspricht.

Bei realen Anwendungen (siehe Kapitel 4) bedingt die begrenzte Datenpufferlänge Schwankungen des bestimmten Offsets. Um dann auch gute Ergebnisse zu erhalten, kann das Verfahren in einer erweiterten Variante verwendet werden (siehe [92]). Dabei wird eine Mittelung der maximalen Abweichungen der resultierenden Schwankungen des Offsets wie folgt durchgeführt:

$$off_k = \frac{\min(DFT_1(0Hz),...,DFT_k(0Hz)) + \max(DFT_1(0Hz),...,DFT_k(0Hz))}{2}; \qquad \textbf{(3.11)}$$

Die jeweilige Anwendung der Offsetunterdrückung kann es erfordern, dass der Offset während der Initialisierungsphase ($>\frac{N}{f_a}$) bestimmt und dann festgehalten wird, da z. B. bei Führungssprüngen der bestimmte Offset durch den gewünschten Offset verfälscht würde.

3.2.1.3 Sukzessiver Mittelwert (gleitender Mittelwert)

Die einfachste Bestimmung eines Offsets kann durch die Berechnung des Mittelwerts über alle momemtan zur Verfügung stehenden Messwerte erfolgen. In Daten getriebenen Anwendungen erfolgt die Berechnung des Mittelwerts sukzessive mit

$$m_k = m_{k-1} \cdot (k-1) + z_k \cdot k; \qquad \textbf{(3.12)}$$

Dies entspricht einem Tiefpass 1.Ordnung.

3.2.1.4 Offset-Beobachter

Der Offset-Beobachter schätzt mit dem Streckenmodell und der Steuergröße den offset- und störungsfreien Sensorwert. Der integrierte und mit der Beobachterverstärkung gewichtete Schätzfehler stellt den Schätzwert des gesuchten Offsets dar. Das Blockschaltbild des Offset-Beobachters ist in Bild 3.2.2 zu sehen.

Gegeben sei eine Strecke in Zustandsdarstellung zu

$$\begin{aligned} \dot{\hat{x}} &= A \cdot \hat{x} + B \cdot u; \\ \hat{z} &= C^T \cdot \hat{x}; \end{aligned} \qquad \textbf{(3.13)}$$

Die Beobachter-Gleichung ergibt sich mit der Beobachter Verstärkung k zu

$$\dot{\varepsilon} = \hat{z} - z - k \cdot \varepsilon; \qquad \textbf{(3.14)}$$

und hat für alle $k > 0$ einen stabilen Pol bei $-k$ im zeitkontinuierlichen bzw. $\left|\frac{1-k}{1+k}\right|$ im zeitdiskreten. Die Wahl von *k* wirkt sich nur auf die Dauer des Einschwingvorgangs aus. Für den Vergleich wurde ein Wert von *k=100* gewählt.

Der beobachtete Offset ergibt sich zu $\hat{off} = k \cdot \varepsilon$. Mit (3.13) und (3.14) ergeben sich die Systemgleichungen zu

$$\begin{aligned} \begin{bmatrix} \dot{\hat{x}} \\ \dot{\varepsilon} \end{bmatrix} &= \begin{bmatrix} A & 0 \\ C^T & -k \end{bmatrix} \cdot \begin{bmatrix} \hat{x} \\ \varepsilon \end{bmatrix} + \begin{bmatrix} B \\ -1 \end{bmatrix} \cdot \begin{bmatrix} u \\ z \end{bmatrix}; \\ \hat{off} &= \begin{bmatrix} 0 \\ k \end{bmatrix}^T \cdot \begin{bmatrix} \hat{x} \\ \varepsilon \end{bmatrix}; \end{aligned} \qquad \textbf{(3.15)}$$

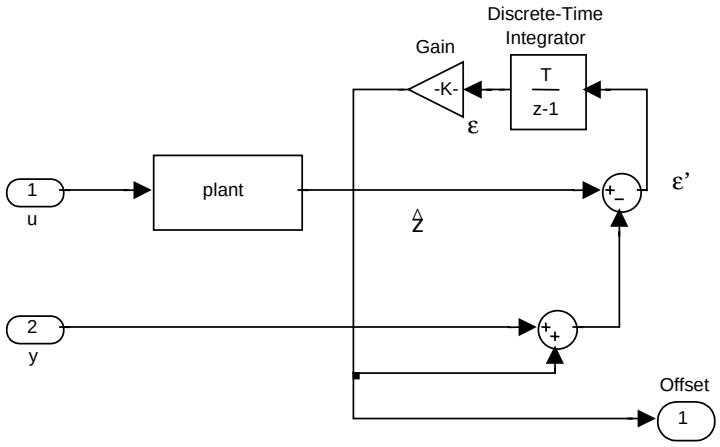

Bild 3.2.2 : Offset-Beobachter

Im statischen Fall ($\dot{\varepsilon} = 0$ bei $t \to \infty$) ergibt sich mit (3.8) zu

$$\hat{off} = z - \hat{z} = \hat{z} - \hat{z} + off + n(t); \qquad \textbf{(3.16)}$$

3.2.1.5 Ergebnisse und Vergleich

Zum Vergleich der Verfahren wurde ein verrauschtes offsetbehaftetes Sinussignal mit verschiedenen Frequenzen den einzelnen Blöcken zugeführt. Das verarbeitete Signal wird mit dem ungestörten offsetfreien Referenzsignal verglichen. In Bild 3.2.3 links ist die maximale relative Abweichung vom Referenzsignal in Abhängigkeit von der Signalfrequenz zu sehen. Auf der rechten Seite ist der im Ausgangssignal noch enthaltene Offset in Abhängigkeit von der Frequenz zu sehen. Der eingespeiste Offset betrug 10^6. Die hohe Abweichung der Hochpassvariante ist, wie schon erwähnt, durch den Phasengang und Amplitudenverlauf des Filters zu erklären. Am besten schneiden die beiden neu vorgestellten Verfahren, Offset-Beobachter und gefensterter Mittelwert, ab. In Bild 3.2.4 ist die Einschwingzeit der Verfahren dargestellt und auch hier schneiden die neuen Verfahren am besten ab. Die Einschwingzeit des Offset-Beobachters kann durch einen höheren Verstärkungsfaktor noch verkürzt werden. Der Rechenaufwand ist jedoch beim Offset-Beobachter am größten, beim gefensterten Mittelwert ist wegen der

Speicherung von N zurückliegenden Werten der Speicherbedarf am höchsten. Die Ergebnisse sind in Tabelle 3.2.1 zusammengefasst.

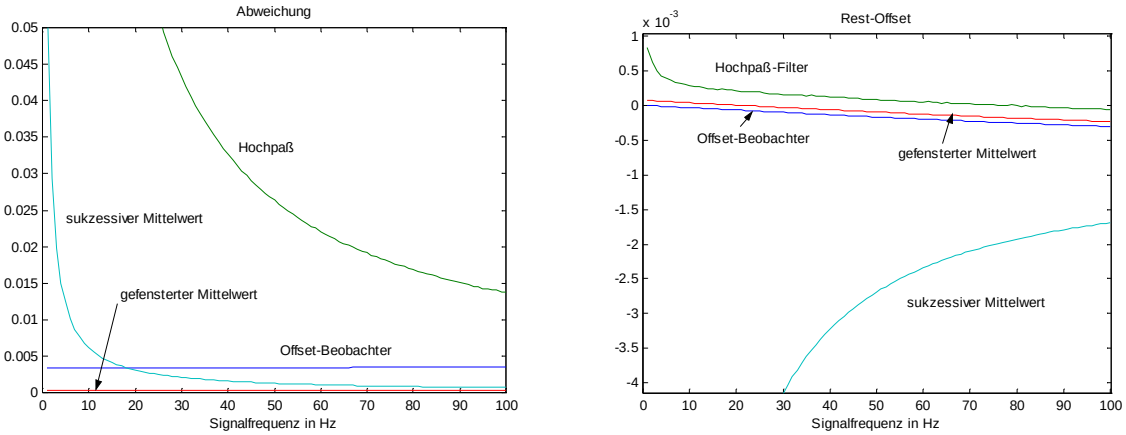

Bild 3.2.3 : maximale Abweichung (links) und Rest-Offset (rechts) der vorgestellten Verfahren

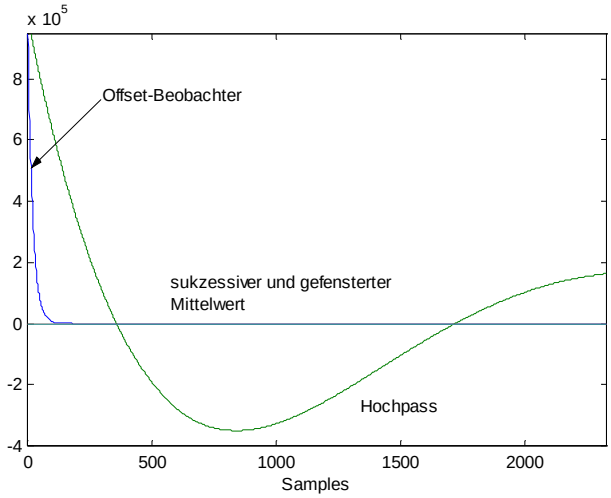

Bild 3.2.4 : Einschwingzeit der Verfahren

	Rechenaufwand	Einschwingzeit	Phasenlage	Offsetunterdrückung
IIR-Butterworth HP n=4	9 Multiplikationen 8 Additionen	>3000 Samples	stark voreilend	mittel, gering bei niedrigen Frequenzen
Gefensterter Mittelwert	1 Multiplikation 2 Additionen	1 Sample	sehr gering nacheilend	sehr groß
Sukzessiver Mittel-wert	2 Multiplikationen 2 Additionen	1 Sample	gering voreilend	groß
Offset-Beobachter	n^2+2n+2 Multipl. 4n-1 Additionen	ca. 200 Samples	sehr gering voreilend	sehr groß

Tabelle 3.2.1 : Vergleich der Offsetunterdrückungs-Verfahren

3.2.2 Differentiation

Die erste Ableitung einer Funktion ist im Zeitbereich definiert zu $\dot{g}(t) = \frac{dg(t)}{dt}$ bzw. im Frequenzbereich zu $s \cdot G(f)$.

Hier wird bereits eine Eigenschaft deutlich, die in der Signalverarbeitung Probleme bereitet, die lineare Abhängigkeit von der Frequenz. Dies hat zur Folge, dass niederfrequente Nutzsignale nach der Differentiation in den verstärkten höherfrequenten Störsignalen (Rauschen) untergehen. Aus diesem Grund wird es in der Regelungstechnik vermieden, Sensordaten zu differenzieren, oder es wird nach Auswegen gesucht trotz Differentiation verwertbare Sensordaten zu bekommen.

3.2.2.1 Differenzfunktion

In zeitdiskreten Systemen kann die 1. Ableitung nicht analytisch berechnet werden, sondern muss numerisch angenähert werden. Die einfachste Form stellt die Differenzfunktion

$$z^* = \frac{z_i - z_{i-1}}{T_a}; \qquad \textbf{(3.17)}$$

dar. Sie ist ebenso wie das analytische Gegenstück sehr rauschanfällig.

3.2.2.2 IIR- und FIR-Filter

In [83] und [6], wie schon in 2.3 erwähnt, werden einige Verfahren vorgestellt, wie man den idealen Differentiator durch IIR- bzw. FIR-Filter annähern kann. In dieser Arbeit wird hier nicht weiter darauf eingegangen. Es werden lediglich zwei IIR-Filterrealisierungen, eine 5.Ordnung und eine 8.Ordnung (siehe Tabelle 3.1.1), ausgewählt, die anhand der in [83] erzielten Ergebnisse erfolgsversprechend erschienen.

Filter 5. Ordnung (diff5)								
Pole	-0.7698	-0.1928	0.0119 + 0.0850i	0.0119 - 0.0850i	0.0726			
Nullst.	-0.6772	-2.9563 +17.0715i	-2.9563 - 17.0715i	20.7688	1.0000			
Filter 8. Ordnung (diff8)								
Pole	-0.8477	-0.1738 + 0.2434i	-0.1738 - 0.2434i	0.0329 + 0.2208i	0.0329 - 0.2208i	0.1489 + 0.1272i	0.1489 - 0.1272i	0.1886
Nullst.	-1.7026	-4.4083 - 3.8873	-4.4083 + 3.8873i	-1.7471 + 3.8165i	-1.7471 - 3.8165i	2.1001 + 4.9209i	2.1001 - 4.9209i	1.0000

Tabelle 3.2.2 : Pole und Nullstellen der IIR-Differentiator Approximation

3.2.2.3 Adaptiver FIR-Filter

In [111] wird eine interessante zeitkontinuierliche Variante vorgestellt, wie man mit einem adaptiven FIR-Filter den idealen Differentiator nachbilden kann. In Bild 3.2.5 ist das Blockschaltbild von diesem Verfahren zu sehen.

Das Eingangssignal wird zuerst integriert und soll dann nach der Filterung wieder dem Eingangssignal entsprechen. Der Fehler zwischen Eingangssignal und gefiltertem Signal wird dazu verwendet den Filter anzupassen; ein Algorithmus hierfür wird nicht vorgestellt. Die so zu jedem Abtastschritt ermittelten Filterkoeffizienten (Taps) werden dazu verwendet das Eingangssignal zu differenzieren.

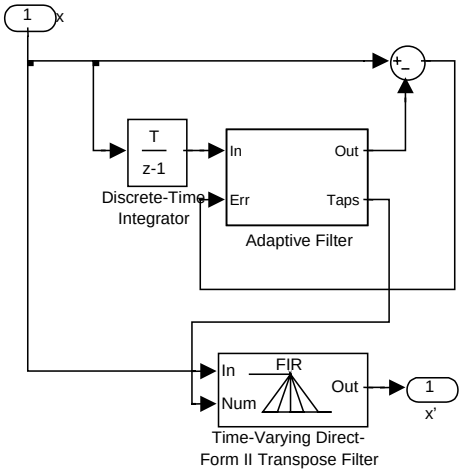

Bild 3.2.5 : Blockschaltbild adaptiver FIR-Filter

Als Integrator wird in dieser zeitdiskreten Realisierung ein Forward-Euler-Integrator verwendet und als Adaptionsverfahren werden die drei Standardverfahren des DSP-Blocksatzes von MatLab [69] eingesetzt, weitere Verfahren finden sich in [107].

- Least Mean-Square (LMS) adaptiver Filter mit Adaptionsalgorithmus nach [43]:

$$out(n) = taps^T(n-1)\cdot in(n);$$
$$taps(n) = taps(n-1) + \frac{in(n)}{in^T(n)\cdot in(n)}\cdot \mu \cdot err(n); \qquad \textbf{(3.18)}$$

  mit Adaptionskonstante μ.

- Kalman adaptiver Filter mit Adaptionsalgorithmus nach [81]:

$$g(n) = \frac{K(n-1)\cdot in(n)}{in^T(n)\cdot K(n-1)\cdot in(n) + Q_M};$$
$$out(n) = in^T(n)\cdot taps(n); \qquad \textbf{(3.19)}$$
$$taps(n+1) = taps(n) + err(n)\cdot g(n);$$
$$K(n) = K(n-1) - g(n)\cdot in^T(n)\cdot K(n-1) + Q_{P;}$$

  mit Eingangs-Kovarianzmatrix K, Kalmankoeffizient g und Mess- und Prozessrausch-Kovarianzmatrizen Q_M und Q_P.

- Rekursiver Least-Square (RLS) adaptiver Filter mit Adaptionsalgorithmus nach [81]:

$$k(n) = \frac{\lambda^{-1}P(n-1)\cdot in(n)}{1+\lambda^{-1}\cdot in^T(n)\cdot P(n-1)\cdot in(n)};$$
$$out(n) = taps^T(n-1)\cdot in(n); \qquad \textbf{(3.20)}$$
$$taps(n) = taps(n-1) + k(n)\cdot err(n);$$
$$P(n) = \lambda^{-1}\cdot(1-\lambda^{-1}\cdot k(n)\cdot in^T(n))\cdot P(n-1);$$

mit Rückführungskoeffizient k, inverser Korrelationsmatrix P und Alterungsfaktor λ.

3.2.2.4 Ergebnisse und Vergleich

In Bild 3.2.6 links ist der Amplitudengang relativ zum Referenzsignal der vorgestellten Verfahren für ein offsetfreies, verrauschtes, sinusförmiges Eingangssignal und rechts für ein offsetbehaftetes, verrauschtes, sinusförmiges Eingangssignal dargestellt. Die erhöhte Abweichung im unteren Frequenzbereich ist in der Verstärkung des Rauschens durch das Differenzieren begründet, das die einzelnen Verfahren unterschiedlich gut unterdrücken können. Es ist auch zu sehen, dass der Offset einen starken Einfluss auf die adaptiven Filter mit Kalman- und LMS-Adaption hat, so dass sie in diesem Fall nur noch eingeschränkt verwendet werden können. Die adaptiven Filter reagieren dafür aber, im Gegensatz zu den anderen Verfahren, weniger empfindlich auf Rauscheinflüsse (siehe Bild 3.2.7). Das RLS-Verfahren schneidet insgesamt am besten ab, da es weder vom Offset noch vom Rauschen beeinflusst wird.

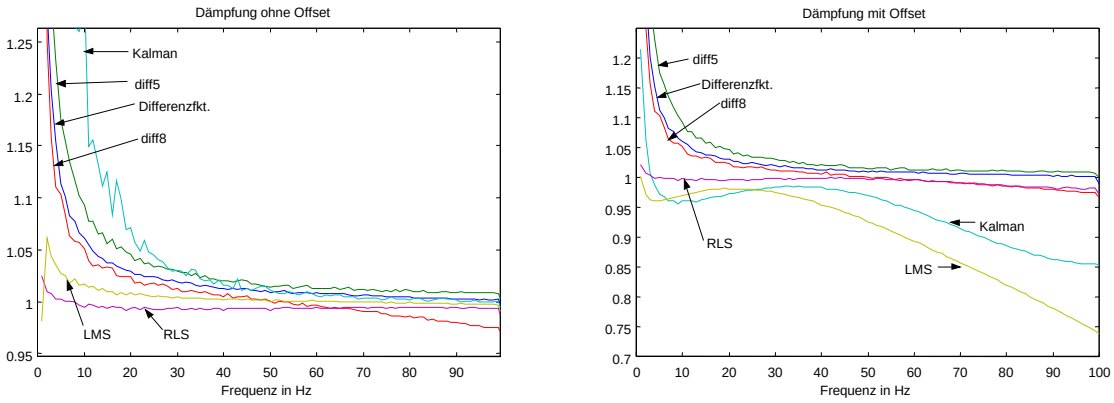

Bild 3.2.6 : Vergleich von Differenzierungsverfahren

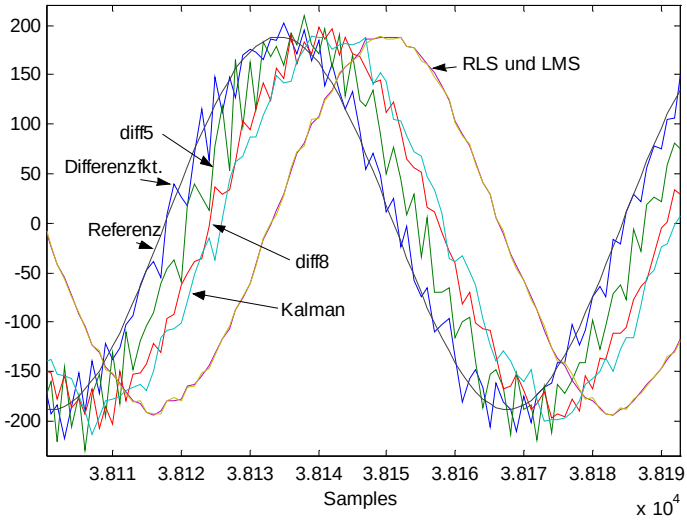

Bild 3.2.7 : Rauscheinfluß

Die Phasenverschiebung der einzelnen Verfahren resultiert aus der verwendeten Filterordnung. Die LMS- und RLS-Filter sind von 16. Ordnung, der Kalman-Filter ist von 8. Ordnung. Zusammenfassend kann man sagen, dass es sehr stark von der Anwendung

abhängt, welches Verfahren das geeignetste ist. Die adaptiven Filter können auch noch durch die Freiheitsgrade in den Adaptionsalgorithmen getunt werden, erzeugen aber gegenüber den anderen Verfahren auch eine höhere Rechenlast. Eine Zusammenfassung der Ergebnisse ist in .Tabelle 3.2.3. dargestellt.

	Phase	Dämpfung	Rauschempfindlichkeit	Rechenaufwand
Differenzfunktion	1 Sample	gering, ausser bei niedrigen Frequenzen	hoch	1 Multiplikation 1 Addition
IIR-Filter 5.Ordnung	5 Samples	gering, ausser bei niedrigen Frequenzen	hoch	11 Multiplikationen 10 Additionen
IIR-Filter 8.Ordnung	8 Samples	gering, ausser bei niedrigen Frequenzen	hoch	17 Multiplikationen 16 Additionen
Kalman adaptiver FIR-Filter	n Samples (hier n=8)	hoch, vor allem bei Offset	gering	3n+7 Multiplikationen 3n+3 Additionen
RLS adaptiver FIR-Filter	n Samples (hier n=16)	sehr niedrig	sehr gering	3n+11 Multiplikationen 3n+1 Additionen
LMS adaptiver FIR-Filter	n Samples (hier n=16)	hoch, vor allem bei Offset	sehr gering	6n+1 Multiplikationen 3n-1 Additionen

Tabelle 3.2.3 : Vergleich der Differentiations-Verfahren

3.2.3 Integration

Die Integration wirkt im Gegensatz zur Differentiation dämpfend auf höherfrequente Stör- und Rauschsignale. Das Problem, das die Integration aufwirft, ist der meist unbekannte Anfangszustand und das instabile Verhalten bei konstanten Anteilen des Eingangsignals. Die Integration eines offsetbehafteten Signals ergibt

$$\int_0^t g(\tau) + off \; d\tau = G(\tau)\Big|_0^t + off \cdot t + C; \qquad \textbf{(3.21)}$$

Der 1. Term ist das gewünschte integrierte Signal, der 2. Term die durch den Offset off hervorgerufene lineare Drift und C ist der unbekannte Anfangszustand. Im Zeitdiskreten kann die Integration ebenfalls wieder nur numerisch angenähert werden. Die Standardverfahren, die Rechteck-Regel (Forward oder Backward Euler, je nachdem, ob der linksseitige oder rechtsseitige Wert des Rechtecks verwendet wird), die Trapezregel und die Keppler'sche Fassregel, sowie nach [83] realisierte IIR-Filter, zeigen das instabile Verhalten bei konstanten Anteilen im Eingangssignal. Bei entsprechend ausreichend hoher Abtastrate kann jedoch immer die einfache Rechteckregel verwendet werden. Ihre Übertragungsfunktion in der z-Ebene ist definiert zu

$$Tf_{\mathrm{Re}chteck}(z) = \frac{T_a}{z-1} = \frac{z^{-1} \cdot T_a}{1-z^{-1}}; \qquad \textbf{(3.22)}$$

Sie hat eine stabile Nullstelle bei $z^{-1}=0$ und eine grenzstabile Polstelle bei $z^{-1}=1$, die das instabile Verhalten bei konstanten Eingangswerten bewirkt.

Im folgenden werden Integrationsverfahren vorgestellt, die trotz offsetbehafteten Eingangssignalen nicht instabil werden. Diese Verfahren können auch zu der noch empfindlicheren Doppelintegration herangezogen werden, bei der die Standardverfahren

eine quadratische Drift aufweisen. Ein Teil dieser Verfahren benötigt ein Referenzsignal eines anderen Sensors. Diese Referenzwerte dürfen eine wesentlich niedrigere Abtastrate aufweisen, sollten aber exakte Werte liefern (siehe diskreter asynchroner Sensor in 3.1.3). Integrationsverfahren mit Referenzierung durch einen anderen Sensor stellen somit einen Spezialfall der Sensorfusion auf Signal-Level-Ebene dar.

3.2.3.1 Integration ohne Referenz

3.2.3.1.1 Polverschiebung

Bei diesem einfachen Verfahren, das auch in [83] vorgeschlagen wird, wird der grenzstabile Pol von (3.22) einfach auf der reellen Achse ein wenig nach links verschoben. Die Auswirkungen, die sich daraus ergeben, sind im Bode-Diagramm von Bild 3.2.8 für drei verschiedene Pollagen dargestellt. Je näher man an die „1" herankommt, desto weiter schiebt sich die Grenzfrequenz des „Tiefpasses" nach unten. Es erhöht sich damit jedoch auch die Einschwingzeit.

Mit dem Optimierungsverfahren der Non-Linear Control Design Toolbox von MatLab [70] wurde die optimale Pollage zu k=0.99974997995650 bestimmt. Die verwendeten Optimierungskriterien waren die Abweichung zum entsprechenden integrierten offsetfreien Eingangssignal und die Einschwingzeit.

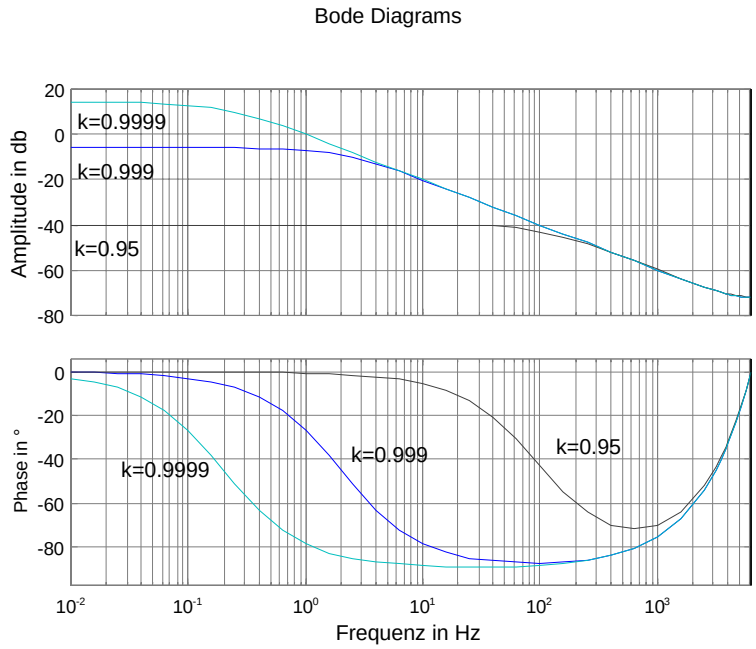

Bild 3.2.8 : Variation des Integrator Pols

3.2.3.1.2 PID-Feedback

Bei diesem Verfahren wird, wie in Bild 3.2.9 zu sehen, der Integratorausgang über einen PID-Regler wieder zum Integratoreingang zurückgekoppelt.

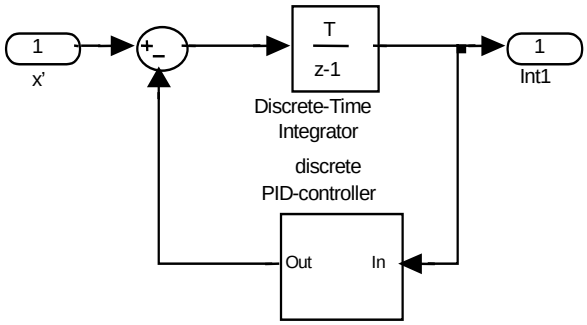

Bild 3.2.9 : PID-Feedback Integrator

Der PID-Regler hat folgende zeitdiskrete Übertragungsfunktion:

$$Tf_{PID}(z^{-1}) = k_P + k_I \cdot T_a \cdot \frac{z^{-1}}{1-z^{-1}} + \frac{k_D}{T_a}(1-z^{-1}) \qquad \textbf{(3.23)}$$

mit den Verstärkungfaktoren für den proportionalen (k_P), integralen (k_I) und differentiellen (k_D) Anteil. Mit (3.22) ergibt sich daraus die Übertragungsfunktion des kompletten Integrationsverfahren zu

$$Tf_{PID-Feedback}(z) = \frac{T_a \cdot z \cdot (z-1)}{z^3 + z^2 \cdot (-2 - T_a \cdot k_P - k_D) + z \cdot (1 + T_a \cdot k_P - T_A^2 \cdot k_I + 2k_D) - k_D}; \qquad \textbf{(3.24)}$$

Die Auswahl der Verstärkungsfaktoren gestaltet sich schwierig, da sich die Pol- und Nullstellen von (3.24) nicht mehr ohne größeren Aufwand analytisch in Abhängigkeit von k_P, k_I und k_D ausdrücken lassen. Die verwendeten Parameterwerte (k_P=1.65948335766823, k_I=9.52796879062057, k_D=0.01177439075405) wurden ebenfalls mit dem Optimierungsverfahren der Non-Linear Design Control Toolbox bestimmt. Es wurden dabei die gleichen Kriterien wie unter 3.2.3.1.1 verwendet. Eine weitere Variante zum Tunen der PID-Parameter stellt in [52] der Unfalsified PID-Regler-Ansatz dar.

3.2.3.1.3 I+Feedback

Eine analytisch einfachere Alternative zum PID-Feedback Integrator stellt das I+Feedback Verfahren dar, dessen Struktur in Bild 3.2.10 zu sehen ist.

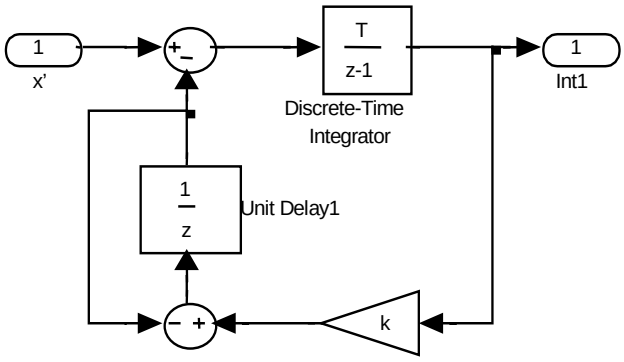

Bild 3.2.10 : I+Feedback Integrator

Der Integratorausgang wird hierbei über einen gewichteten, negativ rückgeführten zusätzlichen Integrator wieder am Integratoreingang eingespeist. Die zeitdiskrete Übertragungsfunktion des Rückführungszweiges ergibt sich zu

$$Tf_{I+}(z^{-1}) = \frac{-k_I z^{-1}}{1+z^{-1}}; \qquad \textbf{(3.25)}$$

In Verbindung mit (3.22) ergibt sich daraus für den gesamten Integrator

$$Tf_{I+Feedback}(z) = \frac{T_a \cdot (z+1)}{z^2 - 1 + k_I \cdot T_a}; \qquad \textbf{(3.26)}$$

dessen Polstellen bei

$$z_{p1,2} = \pm\sqrt{1 - k_I \cdot T_a} \qquad \textbf{(3.27)}$$

liegen. Für k_I=0 erhält man wieder einen einfachen zeitdiskreten Integrator, für $0 < k_I \leq \frac{1}{T_a}$ stabile reelle Pole und für $\frac{1}{T_a} < k_I < \frac{2}{T_a}$ stabile imaginäre Pole. Die imaginären Pole kommen nicht in Frage, da es sich dann um ein schwingungsfähiges System handeln würde. In Bild 3.2.11 auf der linken Seite ist die Wurzel-Ortskurve der beiden Pole zu sehen. Auf der rechten Seite ist das Bode-Diagramm für vier verschiedene k_I-Werte abgebildet.

Der optimale Rückführungskoeffizient wurde wieder mit zuvor genannten Optimierungsverfahren mit den gleichen Optimierungskriterien zu k_I=6.40877100925282 bestimmt. Wie schon bei der Polverschiebung bewirkt ein Verschieben des Poles in Richtung des ursprünglichen Pols bei k_I=1 eine Erniedrigung der Grenzfrequenz des „Tiefpasses“, wobei sich auch hier die Einschwingzeit wieder verlängert.

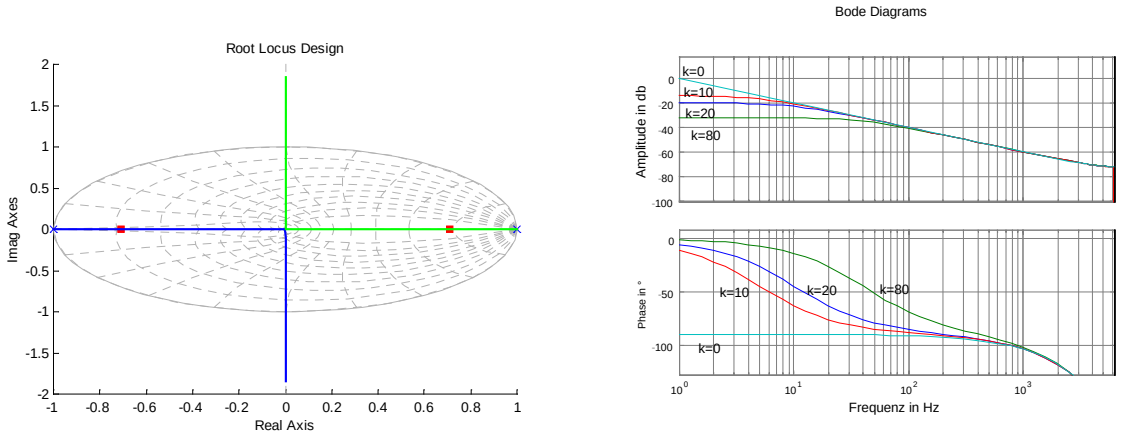

Bild 3.2.11 : Einfluss des Rückführungskoeffizienten

3.2.3.2 Doppel-Integration mit Referenz

Diese Kategorie von Verfahren versucht mit Hilfe von Referenzwerten eine stabile Integration zu erreichen. Das einfachste aller Verfahren hierzu ist, zyklisch den Integrator-Zustand auf den aktuellen Referenzwert zu setzen. Dies führt allerdings nur zum Erfolg, wenn gewährleistet ist, dass die Drift des Integratorausgangs bis zum Eintreffen des nächsten Referenzwertes innerhalb der gewünschten Toleranz bleibt. In Bild 3.2.12 ist das prinzipielle Verhalten für die Integration eines offsetbehafteten Sinussignals, das alle 2000 Werte referenziert wird, dargestellt.

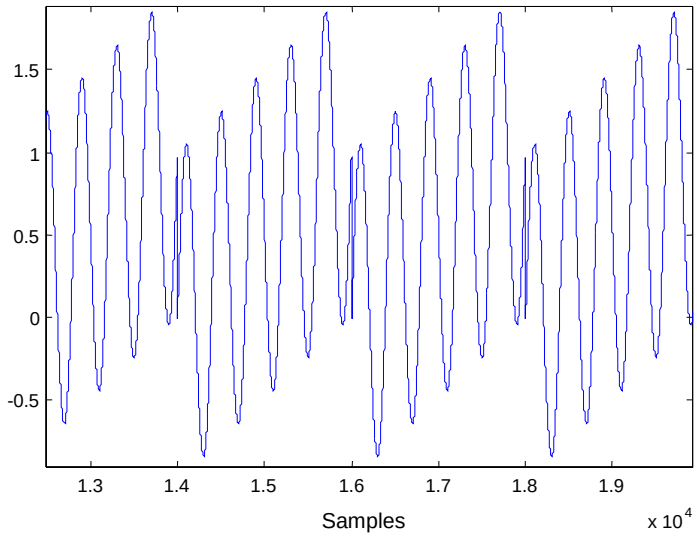

Bild 3.2.12 : "Harte Referenz"

Die folgenden Verfahren werden gleich für die aufwendigere und empfindlichere Doppelintegration hergeleitet und können dann bei Bedarf leicht für die einfache Integration angepasst werden.

3.2.3.2.1 Korrektur der initialen Zustände

Hier wird versucht, den Fehler zwischen Referenzwert und integriertem Wert dafür zu verwenden, die unbekannten initialen Zustände iterativ rückwärts zu bestimmen. Geht man von einem offsetfreien Signal aus, das zweimal integriert werden soll, so erhält man zum Zeitpunkt i mit den beiden unbekannten Anfangszuständen C und K

$$z_i = \iint in_i \cdot dt^2 + C \cdot \Delta t_i + K \qquad \textbf{(3.28)}$$

und zum Zeitpunkt $i+1$

$$z_{i+1} = \iint in_{i+1} \cdot dt^2 + C \cdot \Delta t_{i+1} + K; \qquad \textbf{(3.29)}$$

Mit den Zeiträumen

$$\Delta t_i = t_i - t_{i-1} \text{ und } \Delta t_i + 1 = t_{i+1} - t_{i-1}; \qquad \textbf{(3.30)}$$

Nimmt man nun an, dass der Referenzwert dem gewünschten zweimal integrierten Wert entspricht, so gilt

$$ref_i = \iint in_i \cdot dt^2 \text{ und } ref_{i+1} = \iint in_{i+1} \cdot dt^2 \qquad \textbf{(3.31)}$$

Aus (3.28), (3.29), (3.30) und (3.31) ergibt sich dann für den Anfangszustand C

$$C = \frac{z_i - ref_i - (z_{i+1} - ref_{i+1})}{\Delta t_i - \Delta t_{i+1}} = \frac{err_i - err_{i+1}}{\Delta t_i - \Delta t_{i+1}}; \qquad \textbf{(3.32)}$$

und (3.32) eingesetzt in (3.29) und (3.30) ergibt den Anfangszustand K zu

$$K = z_{i+1} - ref_{i+1} - C \cdot \Delta t_{i+1} = err_{i+1} - C \cdot (t_{i+1} - t_{i-1}) \qquad \textbf{(3.33)}$$

Mit (3.32) und (3.33) können somit die Anfangsbedingungen mit der halben Referenzierungsfrequenz ermittelt und korrigiert werden, wie es auch beispielhaft in Bild 3.2.14 für die Realisierung von Bild 3.2.13 zu sehen ist.

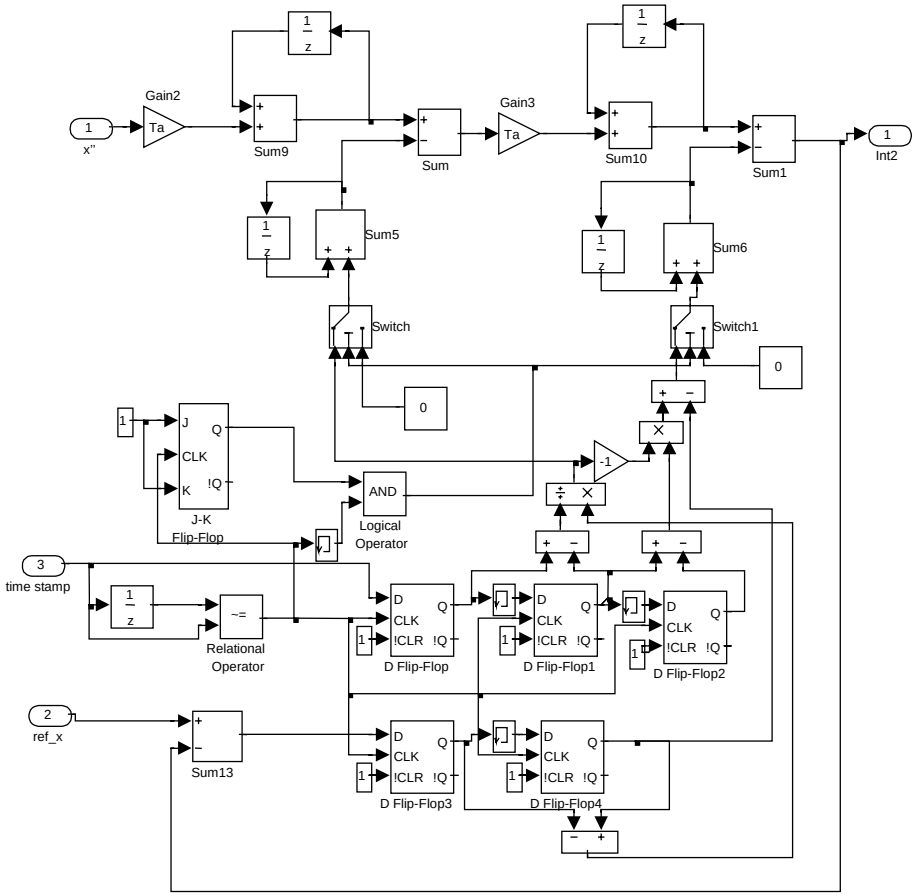

Bild 3.2.13 : Bestimmung der Anfangszustände

Wird dieses Verfahren zweimal verzahnt angewendet, so ist die Korrektur der Anfangsbedingungen auch im Referenzierungstakt möglich. Da für dieses Verfahren vorausgesetzt wird, dass das zu integrierende Signal offsetfrei ist, muss ein Offsetunterdrückungsalgorithmus vorgeschaltet werden oder es müssen Integratoren nach 3.2.3.1 verwendet werden. Im Vergleich der verschiedenen Verfahren wurde eine Realisierung mit vorgeschaltetem gefensterten Mittelwert und eine Realisierung mit Integratoren verwendet, deren Pole nach 3.2.3.1.1 verschoben wurden.

Im Gegensatz zu [92], bei dem die Auswirkungen der Anfangsbedingungen durch eine driftende, dadurch bis zum Zahlenüberlauf zeitlich begrenzte Offsetkorrektur kompensiert werden, ist dieses Verfahren dazu geeignet, den Wert der Anfangsbedingungen rückwärts zu bestimmen. Damit können die Anfangsbedingungen stabil ausgeglichen werden.

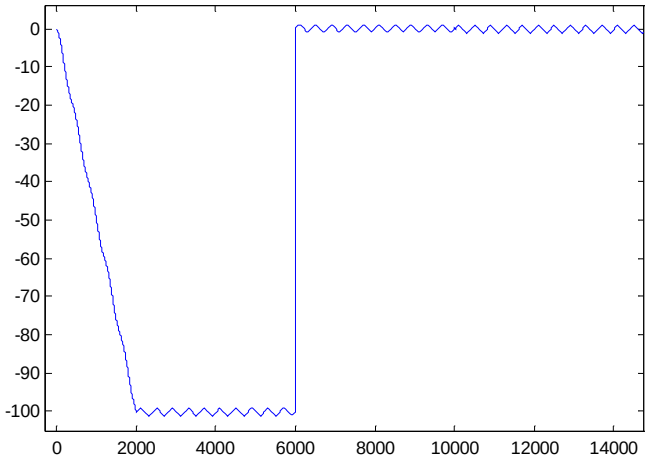

Bild 3.2.14 : Korrektur der Anfangsbedingungen

3.2.3.2.2 Feedforward-Kalman-Filter

Der Kalman-Filter (Gleichungen siehe 3.3.2.3) ist das Allzweckverfahren zum fusionieren von Sensordaten (siehe 2.3). Er ist dadurch natürlich auch dazu geeignet, das Referenzsignal mit dem zweifach integrierten Signal zu fusionieren. Als Streckenmodell wird der diskrete zweifache Integrator verwendet. In zeitdiskreter Zustandsdarstellung ergibt sich dafür:

$$x(k+1) = \begin{bmatrix} 2 & -1 \\ 1 & 0 \end{bmatrix} x(k) + \begin{bmatrix} T_a^2 \\ 0 \end{bmatrix} in(k); \qquad \textbf{(3.34)}$$

$$z(k) = \begin{bmatrix} 0 & 1 \end{bmatrix} x(k);$$

Das Eingangssignal ist das zu integrierende Signal und als Messwert wird die Referenz verwendet (siehe Bild 3.2.15). Die Messwertgleichungen müssen somit nur aktualisiert werden, wenn ein neuer Referenzwert eingetroffen ist. Der gefilterte Kalman-Filter-Ausgang stellt dann eine Schätzung für das zweifach integrierte Signal dar.

Das Einbringen eines zusätzlichen Zustands zum Modellieren des Offset mittels $\dot{x} = 0$ bzw. $x(k+1) = x(k)$ bringt keine Verbesserung mit sich, da sich der Rang der Systemmatrix A damit nicht erhöht. Diese Modellierung stellt einen offenen Integrator dar, dessen unbekannte Anfangsbedingung den Offset nachbildet.

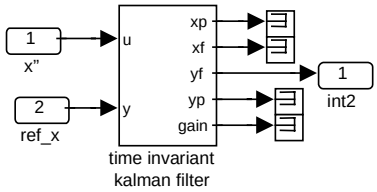

Bild 3.2.15 : Feedforward-Kalman-Filter als Doppelintegrator

3.2.3.2.3 Fehler-Feedback-Kalman-Filter

In [73] wird der Fehler-Feedback Kalman-Filterstruktur aus Robustheitsgründen der Vorrang vor der oben beschriebenen Feedforward Kalman-Filterstruktur gegeben. In dieser Struktur (siehe Bild 3.2.16) befindet sich der Kalman-Filter im Rückführungszweig und korrigiert die Systemzustände der Strecke. Somit ist das Gesamtsystem robuster gegenüber Modellierungsfehlern im Kalman-Filter. Auch wenn der Kalman-Filter (z. B. als Hardwareeinheit) ausfällt, kann das verbleibende System noch vernünftige, wenn auch ungenauere Ergebnisse liefern.

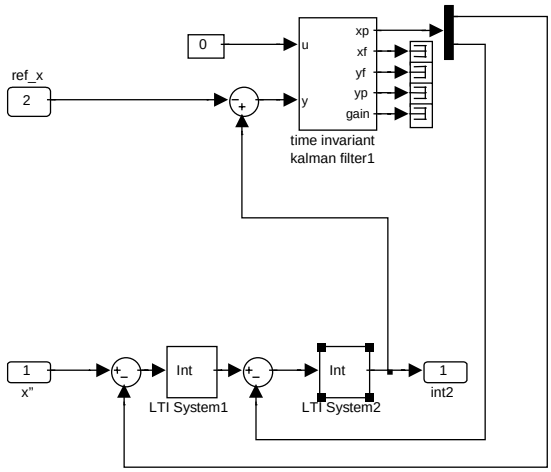

Bild 3.2.16 : Fehler-Feedback Kalman-Filterstruktur

Als Zustandsvariablen werden die Fehler zwischen den realen und geschätzten Zuständen verwendet, also in diesem Fall die Zustände des Referenzsignals und die Zustände des Eingangssignals. Die Fehlerzustände ergeben sich zu:

$$\Delta p = p_{\text{int}} - p_{ref};$$
$$\Delta v = v_{\text{int}} - v_{ref};$$ **(3.35)**

Die Messwertgleichung schließt sogar noch das Messrauschen w der Referenz mit ein:

$$z = p_{\text{int}} - (p_{ref} - w);$$ **(3.36)**

Mit (3.35) ergibt sich

$$z = \Delta p + w;$$ **(3.37)**

Das Streckenmodell in zeitkontinuierlicher Darstellung ergibt mit dem Prozessrauschen n zu

$$\begin{bmatrix} \dot{p}_{\text{int}} \\ \dot{v}_{\text{int}} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} p_{\text{int}} \\ v_{\text{int}} \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} \cdot (a_{ref} + n);$$ **(3.38)**

und das ideale Doppelintegrator-Modell zu

$$\begin{bmatrix} \dot{p}_{ref} \\ \dot{v}_{ref} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} p_{ref} \\ v_{ref} \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} \cdot a_{ref};$$ **(3.39)**

(3.38)-(3.39) ergibt dann die Zustandsbeschreibung mit den Fehlerzuständen für den Kalman-Filter in der Rückführung zu

$$\begin{bmatrix}\Delta\dot{p}\\ \Delta\dot{v}\end{bmatrix}=\underbrace{\begin{bmatrix}0 & 1\\ 0 & 0\end{bmatrix}}_{A}\cdot\begin{bmatrix}\Delta p\\ \Delta v\end{bmatrix}+\underbrace{\begin{bmatrix}0\\ 1\end{bmatrix}}_{B}\cdot n;$$
$$z=\underbrace{\begin{bmatrix}1 & 0\end{bmatrix}}_{C}\cdot\begin{bmatrix}\Delta p\\ \Delta v\end{bmatrix}+w; \qquad \textbf{(3.40)}$$

Die zeitdiskrete Form ergibt sich aus (3.40) mit der Tustin-Transformation zu

$$\begin{bmatrix}\Delta p(k+1)\\ \Delta v(k+1)\end{bmatrix}=\begin{bmatrix}0 & 1\\ 1 & -2\end{bmatrix}\cdot\begin{bmatrix}\Delta p(k)\\ \Delta v(k)\end{bmatrix}+\begin{bmatrix}0\\ T_a^2\end{bmatrix}\cdot n(k);$$
$$z(k)=\begin{bmatrix}1 & 0\end{bmatrix}\cdot\begin{bmatrix}\Delta p(k)\\ \Delta v(k)\end{bmatrix}+w(k); \qquad \textbf{(3.41)}$$

3.2.3.2.4 Ergebnisse und Vergleich

In Bild 3.2.17 werden die Integratoren ohne Referenz in Doppelintegrator-Anwendung miteinander verglichen. Die PID-Feedback Verfahren, egal ob in symmetrischer oder asymmetrischer Konfiguration, haben die höchste Offsetunterdrückung, jedoch sind aufgrund der Überhöhung des Amplitudenganges im Bereich der Grenzfrequenz sowie des großen Phasensprungs die beiden Verfahren nur bei offsetbehafteten höherfrequenten Signalen anwendbar. Das Polverschiebungsverfahren hat die niedrigste Offsetdämpfung und die längste Einschwingzeit, verfügt aber über den besten Phasengang. Das I+Feedback Verfahren hat eine kurze Einschwingzeit aber nur eine schwache Offsetdämpfung und einen schlechteren Phasengang wie die anderen Verfahren. Trotzdem ist es die beste Alternative für Breitbandanwendungen. Es sei darauf hingewiesen, dass alle Verfahren unterhalb ihrer Grenzfrequenz aufgrund der gewünschten Offsetunterdrückung ein differenzierendes Verhalten zeigen. Ihr sinnvoller Einsatzbereich begrenzt sich damit auf Nutzsignale die überhalb der Grenzfrequenz liegen. Dies wird auch am Beispiel in 3.6.3 deutlich, bei dem das Polverschiebungverfahren im Bereich der Grenzfrequenz betrieben wird.

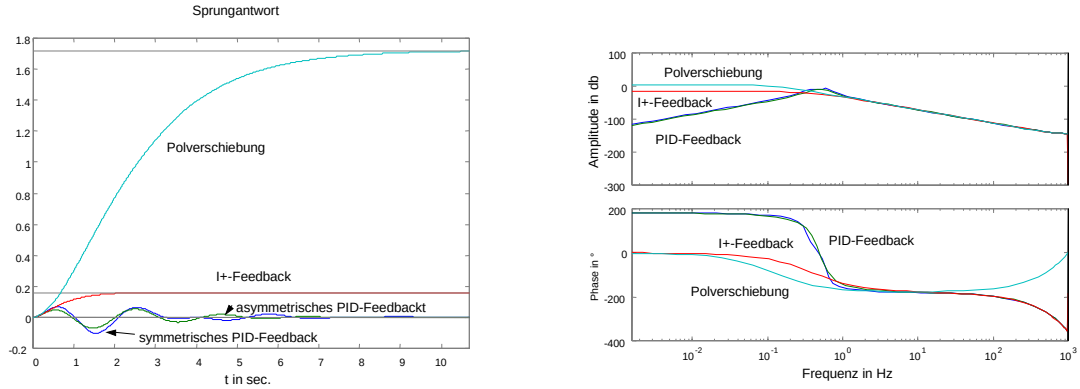

Bild 3.2.17 : Sprungantwort und Bodediagramm der Doppelintegratoren ohne Referenz

In Bild 3.2.18 werden nun alle vorgestellten Integrationsverfahren in Doppelintegrationsanordnung miteinander verglichen. Hierzu werden die Integratoren mit einem offsetbehafteten, verrauschten Sinussignal unterschiedlicher Frequenzen beaufschlagt. Die Abtastfrequenz beträgt 2000Hz und der Referenzierungstakt 1Hz. Das Verfahren

zur Korrektur der Anfangszustände kombiniert mit dem gefensterten Mittelwert zur Offsetunterdrückung ist das einzige Verfahren ohne Phasenverschiebung über den gesamten Frequenzbereich. Alle anderen weisen einen deutlichen Phasengang bei niedrigen Frequenzen auf, der sich jedoch schnell auf einen nahezu konstanten Wert einschwingt. Bei der Dämpfung der Amplitude verhält es sich ähnlich, nach einer größeren Abweichung bei niedrigen Frequenzen erreichen alle Verfahren bei höheren Frequenzen mehr oder weniger schnell einen konstanten Wert der einer geringeren

	Phase	Grenz-frequenz	Dämpfung	Rechenaufwand	Refe-renz	Einschwingzeit
Polverschiebung	0 Samples	~5Hz	gering	2 Multiplikationen 2 Additionen	nein	lang
I+Feedback	-1 Sample	~9Hz	hoch	6 Multiplikationen 4 Additionen	nein	kurz
PID-Feedback	0-2 Sam-ples	4-8Hz	hoch	10 Multiplikationen 8 Additionen	nein	mittel
Kalman-Filter Feedforward	-1 Sample	~5Hz	gering	12 Multiplikationen 8 Additionen	ja	Abhängig vom Referenztakt
Fehler-feedback Kalman-Filter	+1 Sample	~5Hz	gering	12 Multiplikationen 8 Additionen	ja	Abhängig vom Referenztakt
Korrektur der initialen Zu-stände	0 Samples	-	gering	2 Multiplikationen 5 Additionen + 3Multiplikationen, 6 Additionen pro Referenz	ja	Abhängig vom Referenztakt

Tabelle 3.2.4 : Vergleich der Doppelintegrations-Verfahren

Abweichung entspricht. Bild 3.2.18 rechts zeigt auch, dass die Verfahren ohne Referenz mit Ausnahme der Polverschiebung bei Rauschen ungenauere Werte liefern. Eine Erhöhung des Offsets bewirkt bei allen Verfahren eine Verlängerung der Einschwingzeit, wobei eine Erhöhung des Rauschens kaum einen Einfluss auf die Genauigkeit der Verfahren hat. Dies begründet sich durch das Tiefpassverhalten der Integration. Eine kleinere Referenzierungsfrequenz bewirkt beim Feedforward-Kalman-Filter kaum eine Verschlechterung der Ergebnisse, während bei den anderen Verfahren doch die Genauigkeit abnimmt. Alle Verfahren liefern ein mehr oder weniger gutes aber offsetbehaftetes zweifachintegriertes Signal, das gegebenenfalls anschließend noch einer Offsetkorrektur unterzogen werden muss. Alles in allem entscheidet die Anwendung, welches Verfahren am geeignetsten ist. Hat man ein Referenzsignal zur Verfügung, sollte man es unbedingt benutzen, da man dann unempfindlicher gegenüber Störungen ist. Der Rechenaufwand ist bei den Kalman-Filterverfahren natürlich höher als bei den anderen Verfahren, so dass bei vorhandenem Referenzsignal mit ausreichender Abtastung auf jeden Fall das Verfahren zur Korrektur der Anfangszustände auch wegen des guten Phasengangs zu favorisieren ist. In Tabelle 3.2.4 sind die Eigenschaften der hier vorgestellten Verfahren nochmals zusammengefasst.

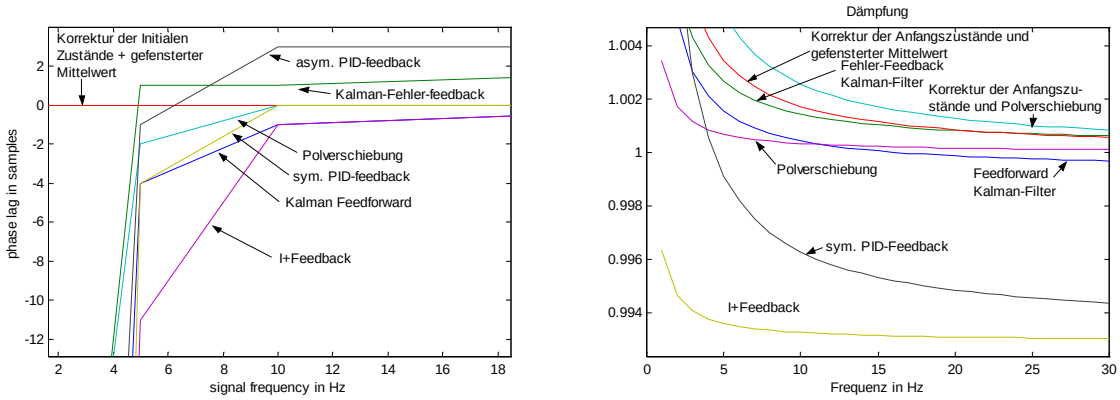

Bild 3.2.18 : Phasengang und Dämpfung bei der Doppelintegration mit den einzelnen Verfahren

3.3 Synchronisation

Im folgenden Abschnitt werden verschiedene Verfahren zur Synchronisation der Messdaten verschiedener Sensoren mit dem Regelungssystem vorgestellt. Im allgemeinsten Fall müssen hierbei verschiedene Abtastfrequenzen der Abtastrate des Reglers angepasst werden und Totzeiten der Sensoren oder der Sensordaten-Vorverarbeitung ausgeglichen werden. Der genaue Zeitpunkt der Messung wird wie unter 3.2 vorgestellt mit einem Zeitstempel festgehalten.

Es wird davon ausgegangen, dass das Regelungssystem die höchste Abtastfrequenz besitzt. Wie in Bild 3.3.1 zu sehen, müssen bei der Synchronisation die abgetasteten Sensormesswerte zum zeitlich nächsten Abtastzeitpunkt des Systems propagiert werden. Als Sensoren kommen dabei die unter 3.1.1 bis 3.1.3 vorgestellten Sensortypen in Frage.

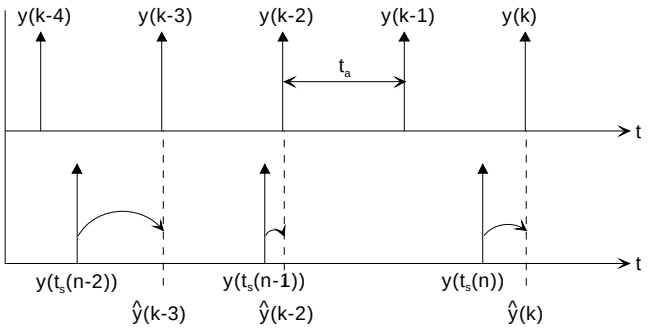

Bild 3.3.1 : Messwert-Vorwärts-Propagation

Die Verfahren können in zwei Gruppen unterteilt werden: Datenbasierte Verfahren, die zur Synchronisation nur auf aktuelle und zurückliegende Messwerte zurückgreifen und modellbasierte Verfahren, bei denen das Modell der Regelstrecke und der Sensoren sowie die Regelgröße in die Verarbeitung mit eingehen.

3.3.1 Datenbasierte Verfahren

3.3.1.1 Halteglied 0.Ordnung (Zero Order Hold ZOH)

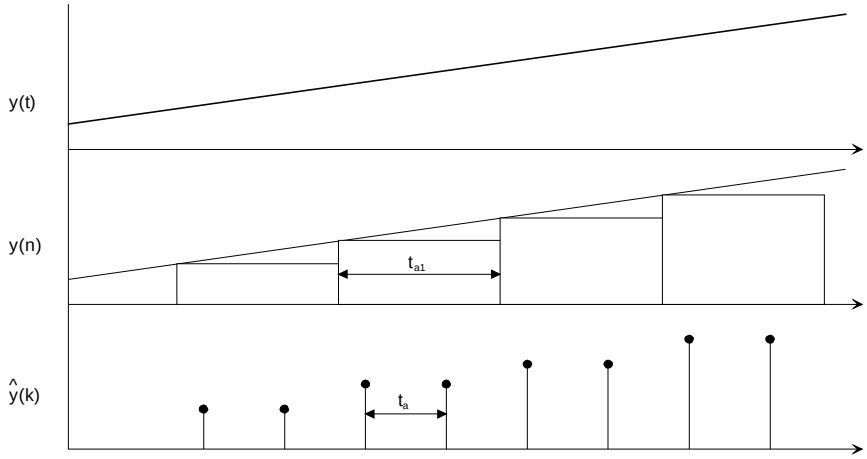

Bild 3.3.2 : Prinzip eines Halteglieds 0.Ordnung

Dieses „Verfahren“ stellt den Fall dar, wenn nicht synchronisiert wird. Es wird der zuletzt bekannte Wert des Sensors solange gehalten, bis ein neuer Messwert eintrifft (siehe Bild 3.3.2)

Der daraus resultierende Synchronisationsfehler für periodische Systeme mit verschiedenen Abtastfrequenzen ist beispielhaft für ein Sinussignal mit der Amplitude 1 und Frequenz 30 Hz in Bild 3.3.3 zu sehen und kann mit dem Modell in Bild 3.3.4 simuliert werden. Die MatLab-Function beinhaltet dabei die Modulo-Funktion.

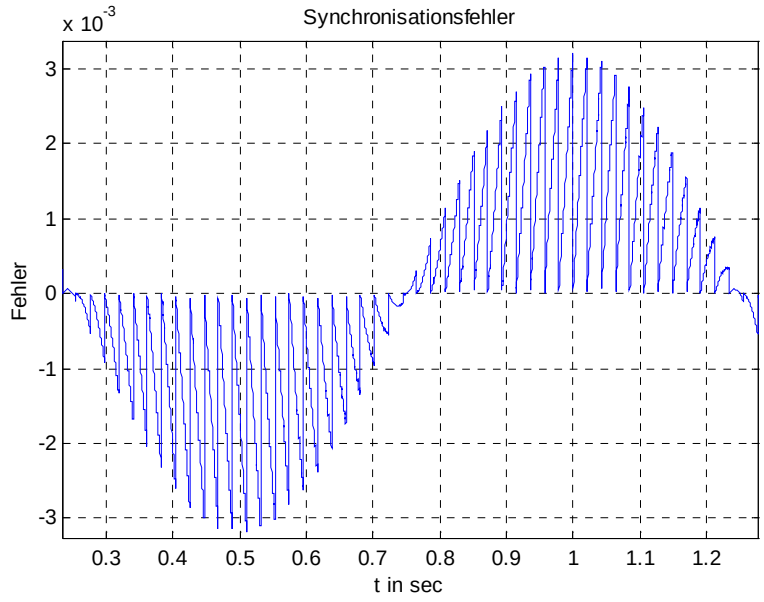

Bild 3.3.3 : Synchronisationsfehler bei T_a=1/2000Hz, T_{a1}=1/1953Hz

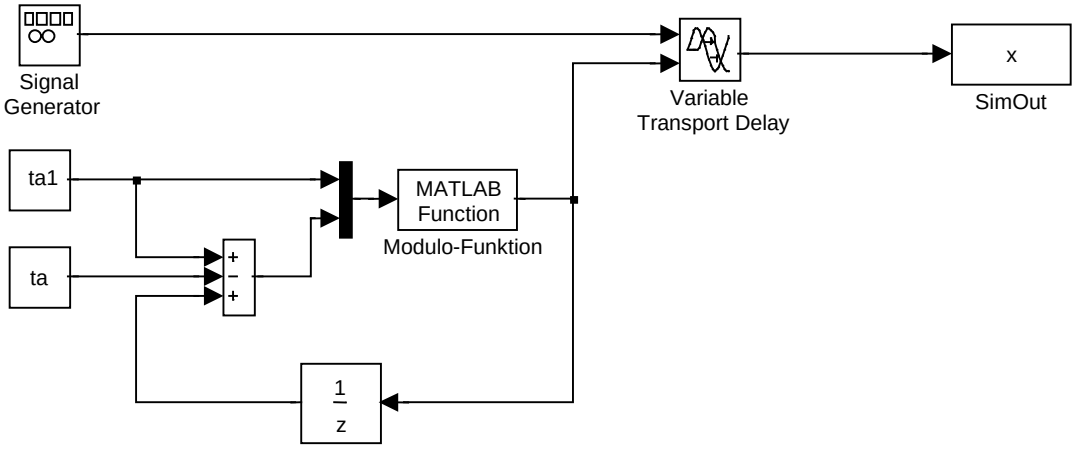

Bild 3.3.4 : Simulationsmodell zur Bestimmung des Synchronisations-fehlers

Der Versatz der beiden Abtastsysteme ergibt sich zu

$$\Delta t_i = \left\{\Delta t_{i-1} + (t_{a1} - t_a)\right\} mod\ t_{a1}; \qquad \textbf{(3.42)}$$

mit $\Delta t_0 = t_0$.

Der maximale Fehler lässt sich zu

$$\varepsilon_{\max} = \max\left\{f'(t)\right\} \cdot t_{a1}; \qquad \textbf{(3.43)}$$

bestimmen.

Das Messrauschen wird unverändert weitergegeben. Daraus folgt

$$\hat{\sigma}_{ZOH} = \sigma; \qquad \textbf{(3.44)}$$

Eine Kompensation der Totzeit findet bei dieser Methode nicht statt.

3.3.1.2 Halteglied 1.Ordnung (First Order Hold FOH)

Der geschätzte Wert bei diesem Verfahren ergibt sich durch lineare Extrapolation durch die letzten beiden Messwerte (siehe Bild 3.3.5). Es ist

$$\hat{y}_k = m_n \cdot (t - t_s(n)) + b_n; \qquad \textbf{(3.45)}$$

mit

$$b_n = y_n;\ m_n = \frac{y_n - y_{n-1}}{t_{a1}}; \qquad \textbf{(3.46)}$$

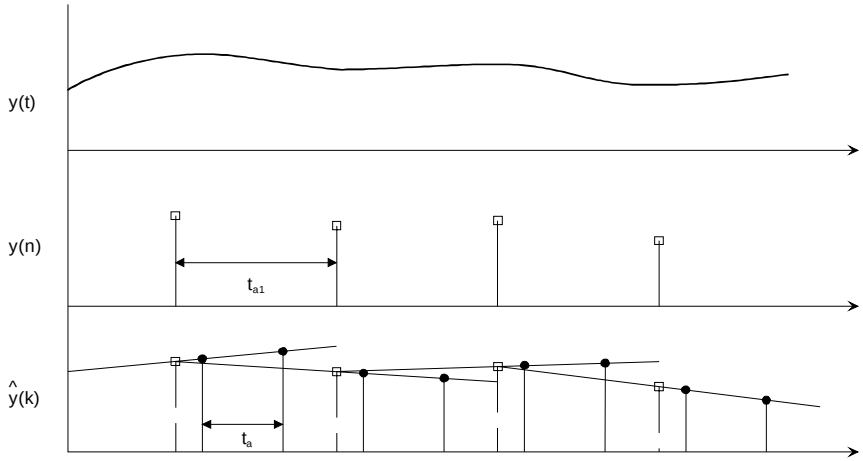

Bild 3.3.5 : Prinzip eines Halteglieds 1.Ordnung

Das Simulationsmodell für dieses Verfahren ist in Bild 3.3.6 zu sehen und entspricht dem „First Order Hold"-Block, der standardmäßig im Lieferumfang von SimuLink enthalten ist.

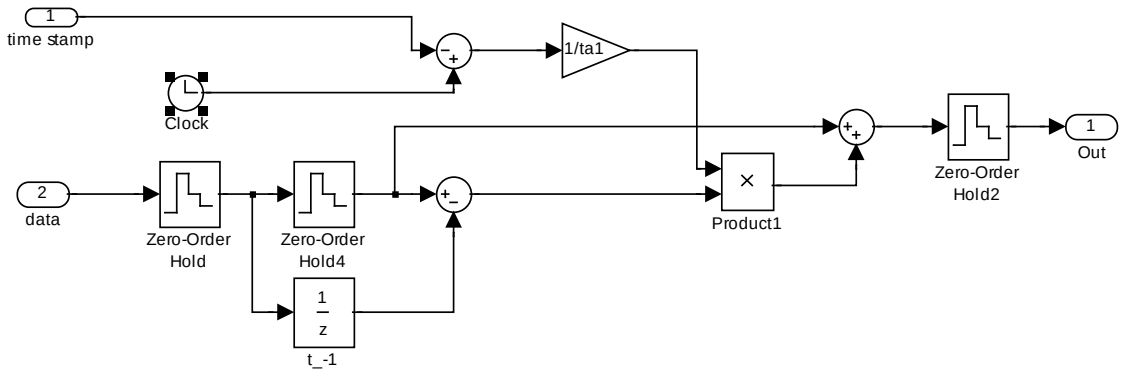

Bild 3.3.6 : Simulationsmodell Halteglied 1.Ordnung

Der Rechenaufwand pro Regelungszyklus liegt bei 3 Additionen und 2 Multiplikationen. Im worst-case muss der Messwert um t_a in die Zukunft extrapoliert werden. Somit folgt bei verrauschten Messwerten:

$$\hat{y}_k + \hat{\sigma}_{FOH} = y_n \pm \sigma + \frac{y_n \pm \sigma - y_{n-1} \mp \sigma}{t_{a1}} \cdot t_{a1;} \qquad \textbf{(3.47)}$$

$$\Rightarrow \hat{\sigma}_{FOH} = 3\sigma; \qquad \textbf{(3.48)}$$

Wie in Bild 3.3.7 und in den Gleichungen (3.47) (3.48) zu sehen, verstärkt sich bei diesem Verfahren das Messrauschen maximal um den Faktor 3.

Eine Totzeit t_t kann ebenfalls ausgeglichen werden, solange der resultierende relative maximale Fehler $e_{rel,max}$ unter der gewünschten Schranke bleibt. Er ergibt sich zu:

$$e_{rel,\max} = \sqrt{\left(\cos(\omega \cdot t_t) + \omega \cdot t_t \cdot \sin(\omega \cdot t_t) - 1\right)^2 + \left(-\sin(\omega \cdot t_t) + \omega \cdot t_t \cdot \cos(\omega \cdot t_t)\right)^2}; \qquad \textbf{(3.49)}$$

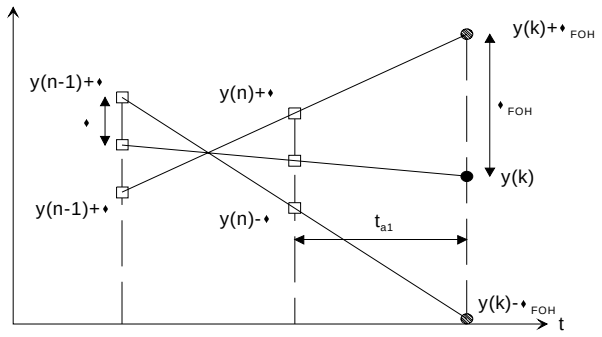

Bild 3.3.7 : Rauscheinfluß bei einem FOH-Glied

3.3.1.3 Halteglied 3.Ordnung (Third Order Hold TOH)

Bei diesem Verfahren ergibt sich der geschätzte Wert durch eine Extrapolation mit einem Polynom 3.Ordnung durch die letzten 4 Messwerte (siehe Bild 3.3.8). Ein Halteglied 2.Ordnung macht keinen Sinn, weil damit keine Wendepunkte realisiert werden können.

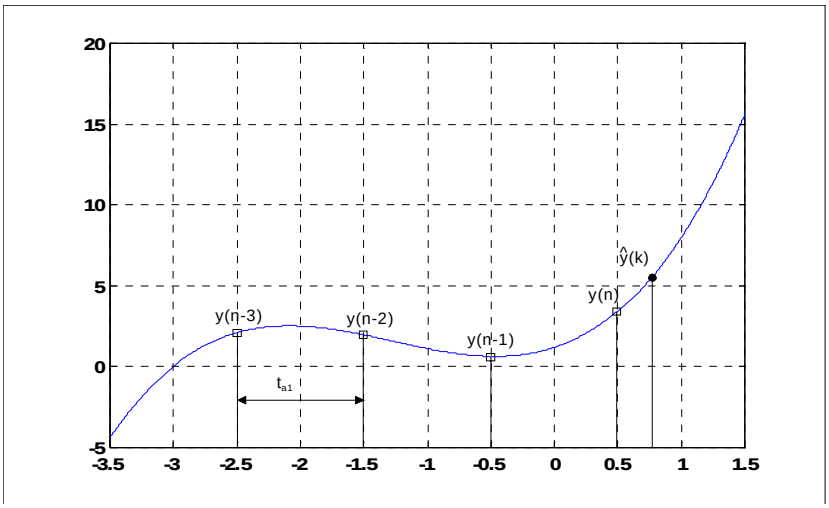

Bild 3.3.8 : Prinzip eines Haltegliedes 3.Ordnung

Es ist

$$\hat{y}_k = a_n \cdot (t - t_s(n))^3 + b_n \cdot (t - t_s(n))^2 + c_n \cdot (t - t_s(n)) + d_n; \qquad \textbf{(3.50)}$$

mit

$$a_n = \frac{1}{6 \cdot t_{a1}^{\ 3}} (y_n - 3y_{n-1} + 3y_{n-2} - y_{n-3}); \qquad \textbf{(3.51)}$$

$$b_b = \frac{1}{4 \cdot t_{a1}^{\ 2}} (4y_n - 10y_{n-1} + 8y_{n-2} - 2y_{n-3}); \qquad \textbf{(3.52)}$$

$$c_n = \frac{1}{6 \cdot t_{a1}}(11y_n - 18y_{n-1} + 9y_{n-2} - 2y_{n-3}); \qquad \textbf{(3.53)}$$

$$d_n = y_n; \qquad \textbf{(3.54)}$$

Es folgt daraus ein Rechenaufwand im optimierten Fall pro Regelzyklus von 19 Multiplikationen und 13 Additionen. Das dazugehörige Simulationsmodell ist in Bild 3.3.9 zu sehen.

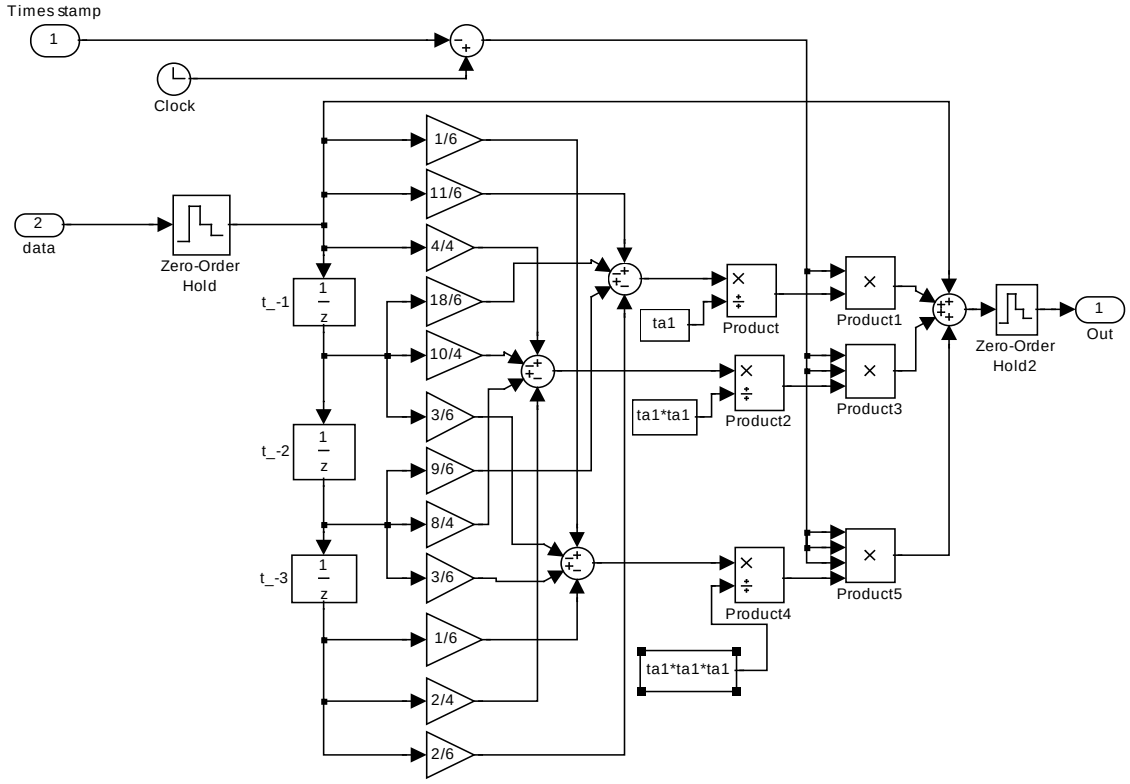

Bild 3.3.9 : Simulationsmodell Halteglied 3.Ordnung

Zur Bestimmung des extrapolierten Schätzwertes muss im ungünstigsten Fall wieder um t_a in die Zukunft geschaut werden. Daraus folgt für verrauschte Messwerte

$$\hat{\sigma}_{TOH} = \frac{8}{6}\sigma + \frac{26}{4}\sigma + \frac{40}{6}\sigma + \sigma = 15.5\sigma; \qquad \textbf{(3.55)}$$

Das bedeutet, dass im ungünstigsten Fall das Messrauschen um den Faktor 15.5 verstärkt wird. Dieser Effekt wird im Bild 3.3.10 an einer Serie konstanter Messwerte verdeutlicht. Sehr kleine Totzeiten können ebenfalls ausgeglichen werden.

Die robustere, aus der Ausgleichsrechnung bekannte Spline-Interpolation [100] kann hier zur Synchronisation nicht verwendet werden, weil dazu die 2. Ableitung am aktuellen Messwert erforderlich wäre. Die 2. Ableitung kann im Zeitdiskreten aber nur um einem Abtastschritt verzögert berechnet werden.

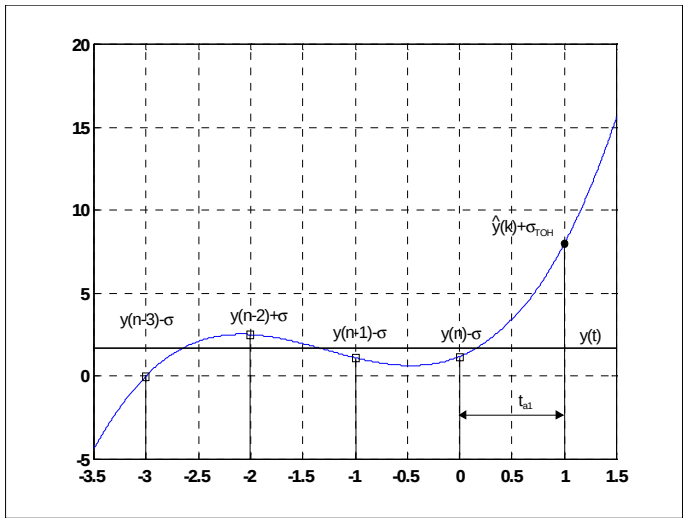

Bild 3.3.10 : Rauscheinfluß bei einem TOH-Glied

3.3.1.4 Halteglied 1.Ordnung mit Polynom 3.Grades (FTOH)

Eine Alternative zu Splines und zum Halteglied 3.Ordnung bildet dieses Verfahren. Es ist eine lineare Extrapolation, wie bei einem Halteglied 1.Ordnung. Die 1.Ableitung bestimmt man jedoch aus dem Polynom 3.Grades durch die letzten 4 Messwerte, wie bei einem Halteglied 3.Ordnung. In Bild 3.3.11 ist das Prinzip dieses Verfahrens und in Bild 3.3.12 das dazugehörige Simulationsmodell dargestellt.

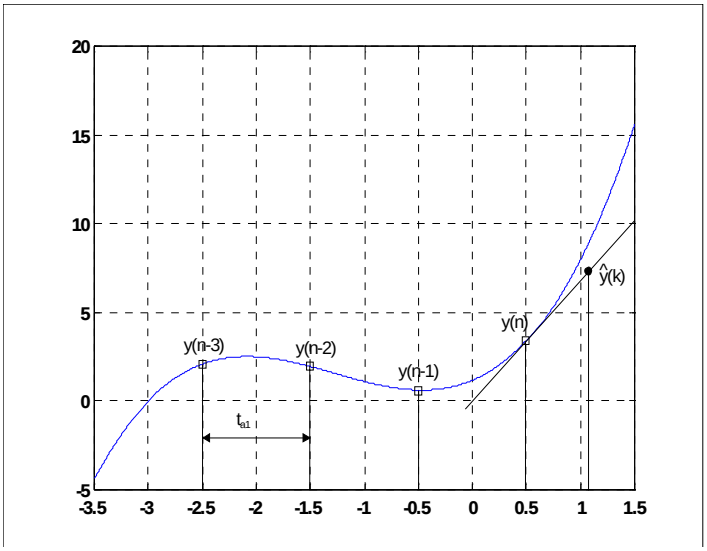

Bild 3.3.11 : Prinzip des FTOH-Blocks

Der Schätzwert ergibt sich zu

$$\hat{y}_k = m_n \cdot (t - t_s(n)) + b_n; \qquad \textbf{(3.56)}$$

mit

$$b_n = y_n; \qquad \textbf{(3.57)}$$

und mit (3.50) bis (3.53)

$$m_n = \hat{y}_{k,TOH}(t = t_s(n))' = c_n = \frac{1}{6 \cdot t_{a1}}(11y_n - 18y_{n-1} + 9y_{n-2} - 2y_{n-3}); \qquad \textbf{(3.58)}$$

Dieses Verfahren ist gegenüber dem Messrauschen robuster als ein Halteglied 3.Ordnung, erreicht jedoch nicht die Genauigkeit aufgrund der linearen Extrapolation. Es stellt gewissermaßen einen Kompromiss zwischen diesen beiden Verfahren dar.

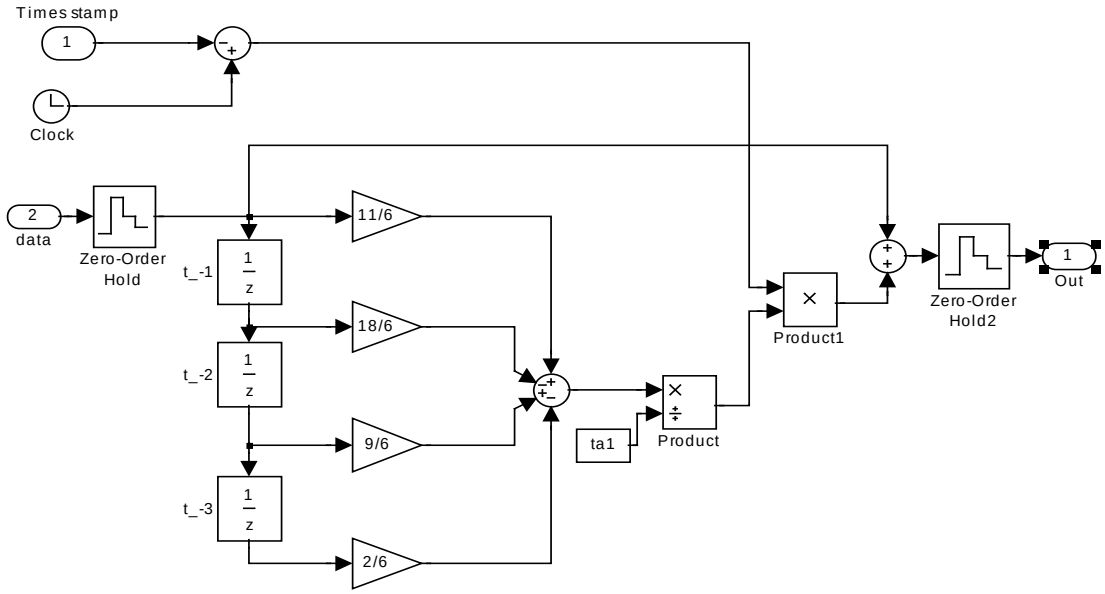

Bild 3.3.12 : Simulationsmodell FTOH

Der Rauscheinfluss ergibt sich zu

$$\hat{\sigma}_{FTOH} = \frac{40}{6}\sigma + \sigma = \frac{23}{3}\sigma \approx 7.67\sigma; \qquad \textbf{(3.59)}$$

Der Rechenaufwand pro Regelzyklus beträgt 6 Multiplikationen und 5 Additionen. Totzeiten der Messwerte können kompensiert werden solange der daraus resultierende Fehler (ähnlich zu (3.49)) unter der gewünschten Schranke bleibt.

3.3.1.5 Linearer 1-Schritt Prädiktor (LPC)

Der im DSP-Blockset von SimuLink [69] enthaltene LPC-Block bestimmt die Koeffizienten eines linearen 1-Schritt-Prädiktors. Der Prädiktionsfehler wird dabei durch die Methode der kleinsten Quadrate minimiert. Die berechneten Koeffizienten bestimmen dann den FIR-Filter, der dazu verwendet wird, den nächsten Messwert zu schätzen. Die hier implementierte Variante berechnet Filterkoeffizienten für einen FIR-Mittelwertsfilter 2.Ordnung.

Der geschätzte Messwert ergibt sich somit zu

$$\tilde{y}_{n+1} = -y_n - a_2 \cdot y_{n-1}; \qquad \textbf{(3.60)}$$

wobei sich der Filterkoeffizient a_2 durch die Berechnung der Lösung von

$$\min \| X \cdot a - b \|_2 \qquad \textbf{(3.61)}$$

ergibt [43]. Mit

$$X = \begin{bmatrix} y_{n-1} & 0 \\ y_n & y_{n-1} \\ 0 & y_n \end{bmatrix}; \quad b = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}; \quad a = \begin{bmatrix} 1 \\ a_2 \end{bmatrix}; \tag{3.62}$$

Da der gewünschte Schätzwert zwischen den Werten y_n *und* $\tilde{y}_{n+1}$ liegt, erfolgt eine lineare Gewichtung innerhalb dieser Zeitspanne (siehe Bild 3.3.13).

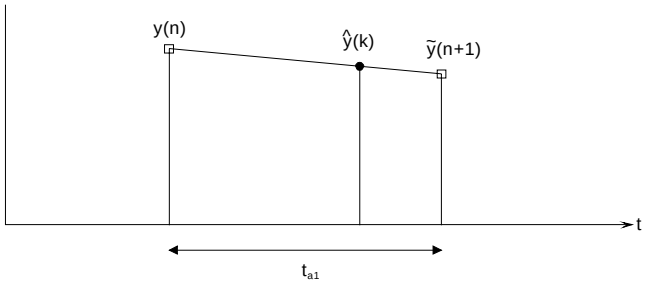

Bild 3.3.13 : Synchronisation mit einem 1-Schritt-Prädiktor

Der gewünschte Schätzwert ergibt somit zu

$$\hat{y}_k = \frac{k \cdot t_a - n \cdot t_{a1}}{t_{a1}} \cdot \tilde{y}_{n+1} + \frac{t_{a1} - k \cdot t_a + n \cdot t_{a1}}{t_{a1}} \cdot y_n; \tag{3.63}$$

Das gesamte Simulationsmodell ist in Bild 3.3.14 zu sehen.

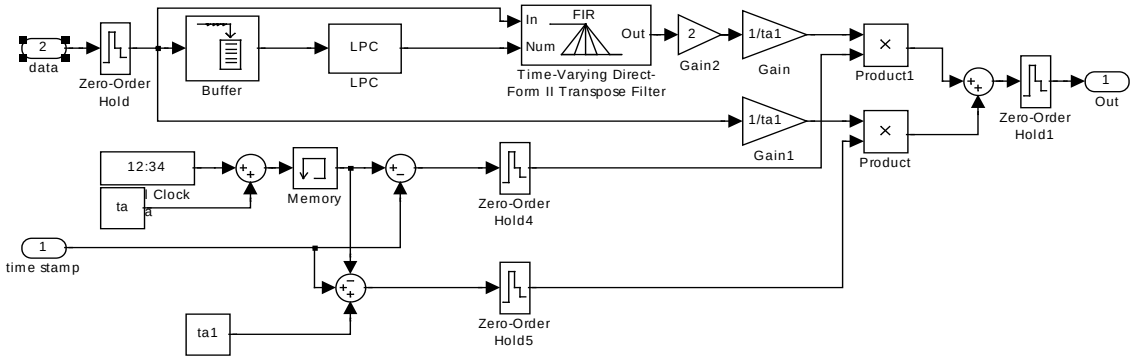

Bild 3.3.14 : Simulationsmodell linearer 1-Schritt-Prädiktor

Die hier implementierte Variante ermöglicht eine Totzeitkompensation nur, wenn gilt:

$$totzeit < t_s + t_{a1} - k \cdot t_a; \tag{3.64}$$

Um größere Totzeiten verarbeiten zu können, müsste eine Fallunterscheidung implementiert werden. Da aber die Genauigkeit dieses Verfahrens hinter den anderen vorgestellten Verfahren zurückbleibt (siehe 3.3.3), wurde darauf verzichtet.

Pro Regelzyklus müssen 6 Multiplikationen, 1 Division und 3 Additionen durchgeführt werden.

Der Einfluss des Messrauschens ergibt sich mit (3.60) und (3.63) im ungünstigsten Fall (nur der Schätzwert wird verwendet) zu

$$\hat{y}_{n+1} + \hat{\sigma}_{LPC} = -y_n \mp \sigma - a_2 \cdot y_{n-1} \mp a_2 \cdot \sigma; \qquad \textbf{(3.65)}$$

Es wird dabei angenommen, dass der Filterkoeffizient aufgrund der Fehlerminimierung über die Methode der kleinsten Quadrate nicht vom Messrauschen beeinflusst wird. Mit (3.65) ergibt sich dann mit

$$\hat{\sigma}_{LPC} = (1 + a_2) \cdot \sigma; \qquad \textbf{(3.66)}$$

auf alle Fälle eine Verstärkung des Messrauschens.

3.3.2 Modellbasierte Verfahren

3.3.2.1 Parallelmodell

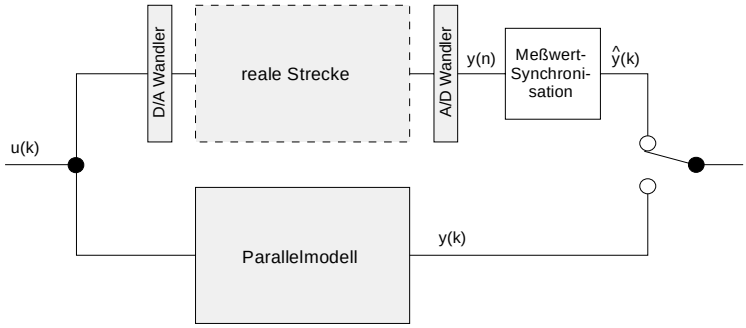

Bild 3.3.15 : Synchronisation mit Parallelmodell

In Bild 3.3.15 ist die Synchronisation über ein Parallelmodell im Prinzip abgebildet. Es stellt das einfachste modellbasierte Verfahren dar. Die auch asynchron anfallenden Messwerte könnten mit den unter 3.3.2.4 vorgestellten Messwertsynchronisationsblock zum nächsten Abtastzeitpunkt des Regelsystems propagiert werden. Liegt zu einem Abtastzeitpunkt kein neuer propagierter Messwert vor, so wird der Ausgang des Parallelmodells als Schätzung für den Messwert verwendet. Dieser Wert muss auch nicht erst synchronisiert werden, da das Parallelmodell mit der Abtastrate des Regelsystems betrieben wird. Es erfolgt keine Korrektur zwischen realem Messwert und vom Parallelmodell geschätztem Messwert, so dass dieses Verfahren eine rein gesteuerte Anordnung darstellt.

Eine Simulation dieses Verfahrens ist nicht sinnvoll, da das verwendete Simulationsmodell dem Parallelmodell entspricht. Deshalb und aufgrund der Fehlerfortpflanzung durch die rein gesteuerte Anordnung wird dieses Verfahren nicht weiter verfolgt.

3.3.2.2 Kontinuierliche Beschreibung

Hat man neben dem diskreten Streckenmodell auch das kontinuierliche zur Verfügung, so können die exakten Werte zwischen den Abtastzeitpunkten nach [1] bestimmt werden. Damit erreicht man eine korrekte Interpolation zwischen zwei Messwerten. Man

kann somit den Messwert zu dem um einen Abtastschritt zurückliegenden Wert des Regelsystems synchronisieren (siehe Bild 3.3.16). Dies führt jedoch zu einer Totzeit von einem Abtastschritt, so dass dieses Verfahren in den meisten Fällen nicht zur schritthaltenden Synchronisation verwendet werden kann.

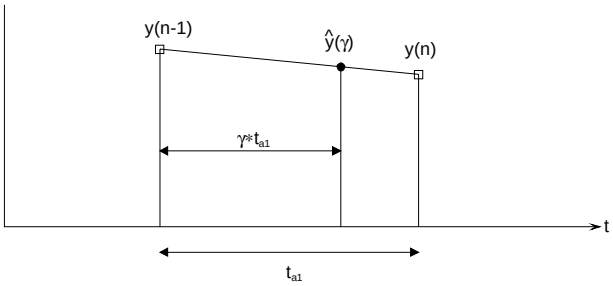

Bild 3.3.16 : kontinuierliche Zwischenwerte bei diskreten Systemen

Das Verfahren soll am Beispiel der Systemgleichungen für eine Halteglied 1.Ordnung erläutert werden. Die kontinuierliche Beschreibung erhält man nach (3.45) und (3.46) zu

$$\hat{y}(t) = \frac{u_n - u_{n-1}}{t_{a1}}(t - t_s(n)) + u_n; \qquad \textbf{(3.67)}$$

Setzt man die kontinuierliche Zeit t in Abhängigkeit der Abtastrate zu

$$t = t_s(n) + \gamma \cdot t_{a1;} \qquad \textbf{(3.68)}$$

so erhält man die kontinuierliche Beschreibung zwischen den einzelnen Abtastschritten.

$$\hat{y}(\gamma) = \frac{u_n - u_{n-1}}{t_{a1}}(t_s(n) + \gamma \cdot t_{a1} - t_s(n)) + u_n = u_n \cdot (\gamma + 1) - \gamma \cdot u_{n-1}; \qquad \textbf{(3.69)}$$

Mit $\gamma \in [0;1]$, wobei sich für $\gamma = 1$ die normale diskrete Beschreibung ergibt.

(3.69) lässt sich in die Zustandsdarstellung umformen und man erhält

$$\begin{bmatrix} x(t_s(n) + \gamma \cdot t_{a1}) \\ u(t_s(n) + \gamma \cdot t_{a1}) \end{bmatrix} = \begin{bmatrix} A_\gamma & B_\gamma \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x(t_s(n)) \\ u(t_s(n)) \end{bmatrix};$$
$$y(t_s(n) + \gamma \cdot t_{a1}) = C_\gamma^T \cdot x(t_n) + D_\gamma \cdot u(t_s(n)); \qquad \textbf{(3.70)}$$

Mit den diskreten Systemmatrizen

$$A = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}; B = \begin{bmatrix} 0 \\ 1 \end{bmatrix}; C = \begin{bmatrix} -1 & 0 \end{bmatrix}; D = 2; \qquad \textbf{(3.71)}$$

und den Beziehungen der diskreten und kontinuierlichen Systemmatrizen nach [1]

$$C_\gamma^T = C^T \cdot A_\gamma; \quad D_\gamma = C^T \cdot B_\gamma + D; \qquad \textbf{(3.72)}$$

ergeben sich die Matrizen der kontinuierlichen Beschreibung der Zwischenwerte zu

$$A_\gamma = \begin{bmatrix} \gamma & 0 \\ 0 & 0 \end{bmatrix}; B_\gamma = \begin{bmatrix} -\gamma + 1 \\ 0 \end{bmatrix}; C_\gamma = \begin{bmatrix} -\gamma & 0 \end{bmatrix}; D_\gamma = \gamma + 1; \quad \textbf{(3.73)}$$

Wie unter 2.5 schon beschrieben, stellt die Lifting-Technik nach [117] ein ähnliches Verfahren dar. Es werden jedoch die Strecke in kontinuierlicher Zustandsdarstellung und der Regler in diskreter Zustandsdarstellung angegeben. Die gesamte Systembeschreibung enthält damit sowohl kontinuierliche als auch diskrete Zustände. Die resultierenden Systemmatrizen ergeben sich aus Anteilen beider Darstellungen. Die Werte zwischen den Abtastwerten erhält man ebenfalls mit $\gamma \in [0;1]$.

Lässt man bei beiden Verfahren Werte $\gamma > 1$ zu, so würde man eine Extrapolation erreichen. Die Herleitung der jeweiligen Systemmatrizen (der gemischt kontinuierlich diskrete Teil der Systemmatrix A ergibt sich z. B. bei der Lifting-Technik zu $A_{cd} = \int_0^{t_{a1}} e^{A_c \cdot (t_{a1} - \tau)} \cdot B_c \cdot H(\tau) \cdot C_d \, d\tau$) gestaltet sich jedoch relativ aufwendig, so dass eine automatische Bestimmung nicht möglich ist. Dies aber wäre die Voraussetzung für einen Synchronisationsblock innerhalb der vorgestellten Systemarchitektur. Dieser Ansatz wird somit nicht mehr weiter verfolgt.

3.3.2.3 Kalman-Filter

Zum Entwurf eines Kalman-Filters ist die Beschreibung der Regelstrecke in folgender Zustandsform notwendig, dies entspricht der Struktur in Bild 3.3.17.

$$\begin{aligned} x(k+1) &= A(k) \cdot x(k) + B(k) \cdot u(k) + D(k) \cdot z(k), \\ y(k) &= C(k) \cdot x(k) + w(k). \end{aligned} \quad \textbf{(3.74)}$$

wobei z(k) das normalverteilte Prozessrauschen mit der Kovarianz Z und w(k) das normalverteilte Messrauschen mit der Kovarianz W darstellt. Korrelationen zwischen Prozessrauschen und Messrauschen könnten im Kalman-Filter berücksichtigt werden [82], es wird hier aber nicht weiter darauf eingegangen.

Die Kalman-Filter-Gleichungen in der allgemeinen zeitvarianten Form lauten:

$$\begin{aligned} \tilde{x}_F(k) &= \tilde{x}_P(k) + H(k) \cdot [y(k) - C(k)\tilde{x}_P(k)], \\ \tilde{x}_P(k+1) &= A(k) \cdot \tilde{x}_F(k) + B(k) \cdot u(k), \\ H(k) &= \Pi_P(k) \cdot C(k)^T \cdot [C(k) \cdot \Pi_P(k) \cdot C(k)^T + W(k)]^{-1}, \\ \Pi_P(k+1) &= A(k) \cdot \Pi_F(k) \cdot A(k)^T + D(k) \cdot Z(k) \cdot D(k)^T, \\ \Pi_F(k) &= [\tilde{1} - H(k) \cdot C(k)] \cdot \Pi_P(k). \end{aligned} \quad \textbf{(3.75)}$$

mit

$\tilde{x}_F$ gefilterter Zustand

$\tilde{x}_P$ geschätzter Zustand

Π_F gefilterte Kovarianz

Π_P geschätzte Kovarianz

H Kalman Rückführungskoeffizient

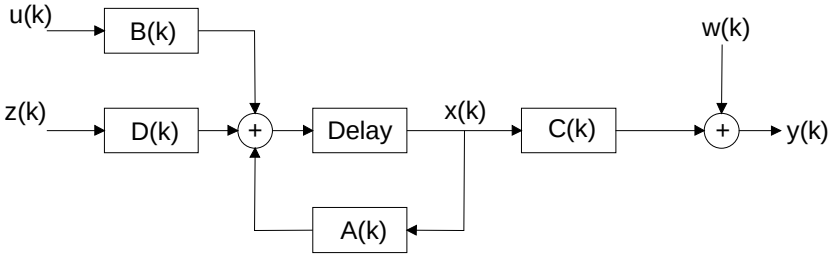

Bild 3.3.17 : Struktur der Strecke in Zustandsdarstellung

Um einen Kalman-Filter auch realisieren zu können, müssen nach [82], [68] folgende Voraussetzungen erfüllt sein.

- Das Matrizenpaar [A,C] ist beobachtbar.
- Das Matrizenpaar [A,F] ist steuerbar, wobei F eine beliebige Matrix ist, die $F \cdot F^T = D \cdot Z \cdot D^T$ erfüllt.
- $W > 0 \; und \; F \cdot F^T - D \cdot W^{-1} \cdot D^T \geq 0.$

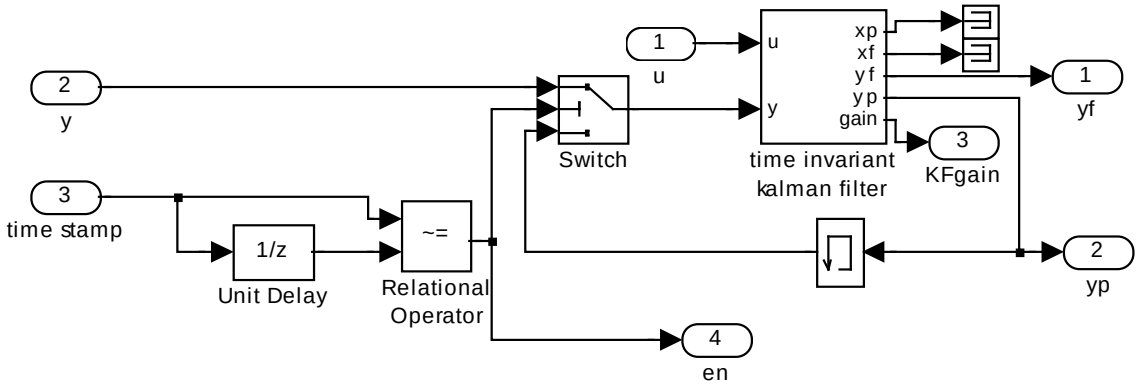

Bild 3.3.18 : Synchronisation mit Kalman-Filter

Zur Synchronisation mit den asynchronen Messwerten y, wird, wenn kein neuer Messwert verfügbar ist, der Schätzwert $C(k) \cdot \hat{x}_P(k)$ als Messwert verwendet. Der gefilterte Messwert $C(k) \cdot \tilde{x}_F(k)$ dient als synchronisierter Messwert (siehe Bild 3.3.18). Der Kalman-Filter verhält sich in diesem Fall wie wenn der Rückführungskoeffizient H=0 wäre.

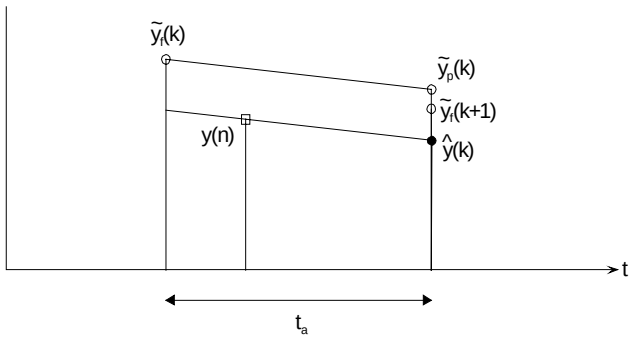

Bild 3.3.19 : Messwertsynchronisation

Die Abtastzeitpunkte von Sensor und Regelstrecke sind normalerweise nicht synchron, so dass die Sensorwerte zum nächsten Abtastzeitpunkt des Regelsystems linear geschätzt werden (siehe Bild 3.3.19). Es wird dabei als 1.Ableitung an der Stelle des Messwertes, die im diskreten ohne zusätzliche Messwerte nicht zu berechnen ist, die Steigung der Verbindungsgeraden zwischen $\tilde{y}_F(k)$ und $\tilde{y}_P(k)$ verwendet.

Der synchronisierte Messwert ergibt sich zu

$$\hat{y}(k) = y(t_s(n)) + \frac{\tilde{y}_P(k) - \tilde{y}_F(k)}{t_a} \cdot (k \cdot t_a - t_s(n)); \qquad \textbf{(3.76)}$$

In Bild 3.3.20 ist das Simulationsmodell des Synchronisationsblocks für den Kalman-Filter zu sehen.

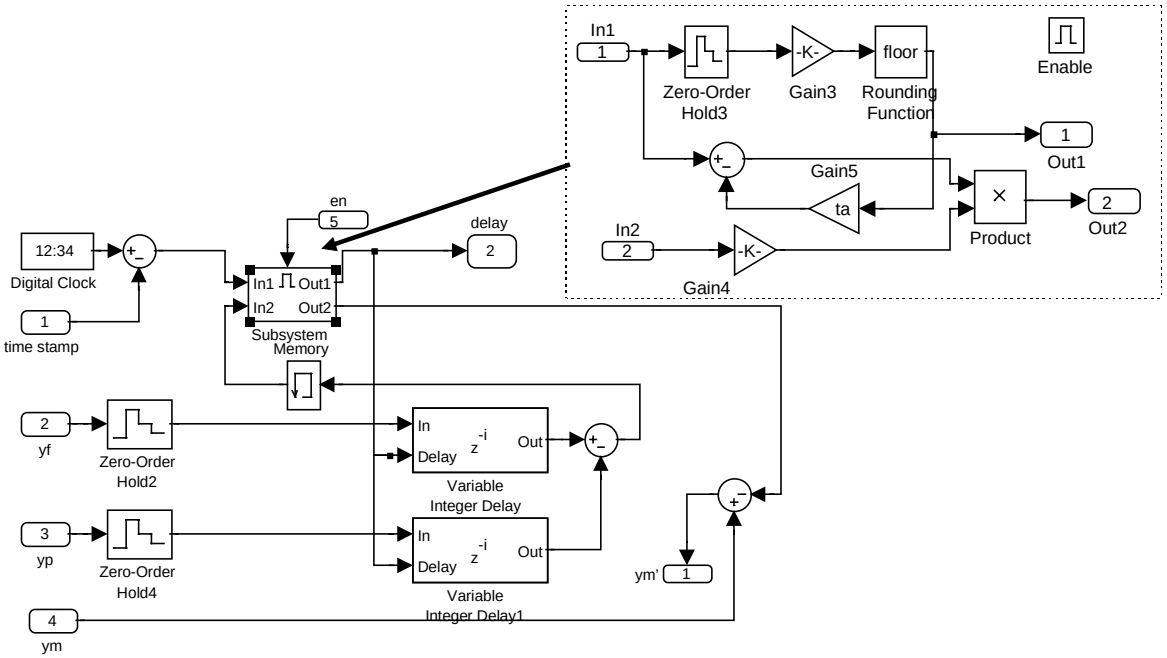

Bild 3.3.20 : Simulationsmodell Messwertsynchronisation

Um auch Totzeit behaftete Messwerte verarbeiten zu können, wird Gleichung (3.76) erweitert zu

$$\hat{y}(k-N) = y(t_s(n)) + \frac{\tilde{y}_P(k-N) - \tilde{y}_F(k-N)}{t_a} \cdot ((k-N) \cdot t_a - t_s(n)); \qquad \textbf{(3.77)}$$

wobei N die Anzahl der Abtastwerte des Regelsystems darstellt, die der Totzeit entspricht und wie folgt zu bestimmen ist:

$$N = floor(\frac{k \cdot t_a - t_s(n)}{t_a}); \qquad \textbf{(3.78)}$$

Bei Totzeit behafteten Messwerten ist der Messwert zum eigentlichen Zeitpunkt nicht verfügbar, so dass der Schätzwert $\tilde{y}_P(k)$ statt dessen verwendet wird. Da der Schätzwert in der Regel nicht mit dem realen Messwert übereinstimmt und je nach System der

daraus resultierende Fehler nicht zu vernachlässigen ist, müssen Messwert und Schätzwert abgeglichen werden sobald der verzögerte Messwert vorliegt.

Der Schätzfehler ergibt sich zu

$$\varepsilon(k-N) = \tilde{y}_P(k-N) - \hat{y}(k-N); \qquad \textbf{(3.79)}$$

Der Einfluss dieser Differenz auf den aktuellen Kalman-Filterausgang ($\tilde{y}\ (k)$) kann folgender Maßen bestimmt werden:

$$\Delta\hat{y}(k) = C(k) \cdot \prod_{i=1}^{N} A(i) \cdot H(k-N) \cdot \varepsilon(k-N); \qquad \textbf{(3.80)}$$

Der kompensierte und synchronisierte Messwert ergibt zu

$$\hat{y}(k) = \tilde{y}_F(k) - \Delta\hat{y}(k); \qquad \textbf{(3.81)}$$

Das dazugehörende Simulationsmodell ist in Bild 3.3.21 zu sehen.

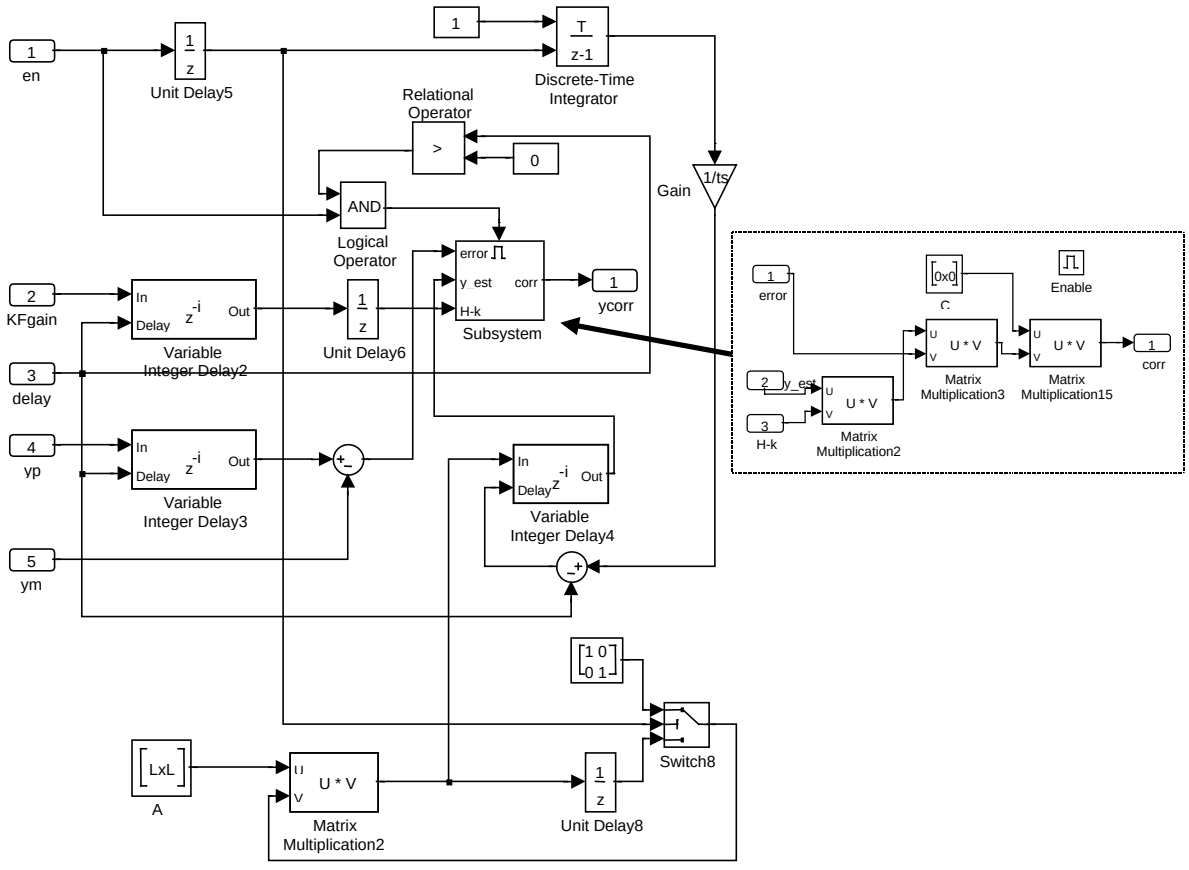

Bild 3.3.21 : Simulationsmodell Totzeitkorrektur

Aus (3.80) ist zu erkennen, dass der Einfluss des Schätzfehlers hauptsächlich von der Systemmatrix A abhängt. Je weiter die Eigenwerte im Zentrum des Einheitskreises liegen, desto schneller wird der Einfluss des Schätzfehlers gegen 0 streben. Hat A dagegen einen grenzstabilen Eigenwert (liegt genau auf dem Einheitskreis), so bleibt der Einfluss des Schätzfehlers bestehen.

Der Rechenaufwand hängt von der Ordnung, dem Zeitverhalten und der Stabilität der Regelstrecke ab. Die Eigenschaften der Regelstrecke, genauer der Systemmatrix A, bestimmen also, welcher der in Tabelle 3.3.1 zusammengestellten Kalman-Filter-Synchronisationsvarianten zum Einsatz kommt.

Verfahren	Multiplikationen	Additionen
Kalman-Filter zeitinvariant	l^2+4l	$l(l-1)+3(l-1)+l+1$
+ Messwertsynchronisation	$+7$	$+6$
+ Totzeitkorrektur	$+l^2+2l+2$	l^2+l+1
Kalman-Filter zeitvariant	$4l^2+7l+4$	$4l(l-1)+7(l-1)+2l+3$
+ Messwertsynchronisation	$+7$	$+6$
+ Totzeitkorrektur	$N*l^3+2l+2$	$N*(l^3-l^2)+2l+1$

Tabelle 3.3.1 : Rechenaufwand der Kalman-Filter-Varianten

Der Entwurfprozess des Kalman-Filters berücksichtigt sowohl das Messrauschen als auch das Prozessrauschen, wie am Systemmodell von (3.74) auch zu erkennen ist. Der Kalman-Filter stellt die Lösung des Problems der optimalen Filterung im Sinne stochastischer Optimierung dar, so dass der Einfluss des Rauschens minimiert wird.

Das Rauschen wird als normalverteilt vorausgesetzt. Dies ist jedoch keine Einschränkung, da durch geeignete Formfilter nahezu jedes beliebige Rauschen generiert werden kann [75]. Der Formfilter wird dann Bestandteil der Systembeschreibung von (3.74) und erhöht somit die Systemordnung.

Dieses vorgestellte Verfahren zur Synchronisierung und Totzeitkorrektur von Messwerten kann auch mit dem Informationsfilter realisiert werden. Der Informationsfilter stellt die inverse Kovarianzform des Kalman-Filters dar.

Der Informationszustandsvektor ergibt sich zu

$$\begin{aligned} i_P &= \Pi_P^{-1} \cdot x_P; \\ i_F &= \Pi_F^{-1} \cdot x_F; \end{aligned} \qquad \textbf{(3.82)}$$

Die inversen Kovarianzmatrizen Π_F^{-1} und Π_P^{-1} werden Informationsmatrizen genannt.

Mit (3.82) kann der Informationsfilter auch berechnet werden, wenn die Kovarianzmatrizen des Kalman-Filter singulär sind. Außerdem ergibt sich im zeitinvarianten Fall nach [66] ein geringerer Rechenaufwand gegenüber dem Kalman-Filter. Die vollständigen Informationsfiltergleichungen finden sich in [75].

3.3.2.4 Beobachter

In Bild 3.3.22 ist das Simulationsmodell für einen Beobachter basierten Synchronisationsblock ohne Messwertsynchronisation und Totzeitkorrektur zu sehen. Wie schon beim Parallel-Modell und Kalman-Filter wird bei den Abtastzeitpunkten, bei denen kein neuer Messwert zur Verfügung steht, der vom Beobachter geschätzte Messwert verwendet.

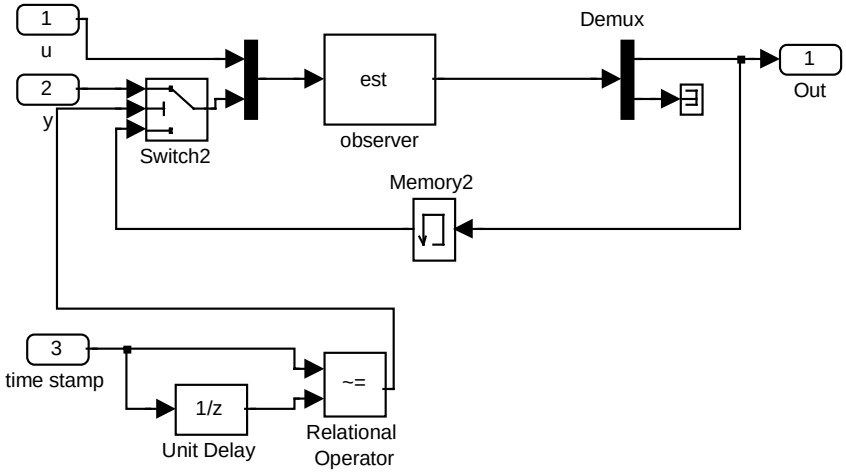

Bild 3.3.22 : Simulationsmodell Beobachter

Zur Propagation der auch asynchronen Sensor-Messwerte zum nächsten Abtastzeitpunkt des Regelsystems kann der in Bild 3.3.20 dargestellte und im vorigen Unterpunkt vorgestellten Synchronisationsblock verwendet werden.

Die Beobachtergleichungen ergeben sich zu

$$\begin{aligned}\tilde{x}(k+1) &= \left[A(k) - L(k)\cdot C(k)\right]\cdot \tilde{x}(k) + L(k)\cdot y(k) + B(k)\cdot u(k);\\ \tilde{y}(k) &= C(k)\cdot \tilde{x}(k);\end{aligned} \qquad \textbf{(3.83)}$$

Die Beobachterverstärkung L kann mittels Polvorgabe als Luenberger-Beobachter nach [1] oder mit $L(k) = A(k)\cdot H(k)$ als Kalman-Filter bestimmt werden.

Bei Totzeit behafteten Sensor-Messwerten kann der Schätzfehler durch Abgleich mit dem Messwert, wie im vorigen Abschnitt beschrieben, verkleinert werden.

Der Schätzfehler ergibt sich zu

$$\varepsilon(k-N) = \tilde{y}(k-N) - \hat{y}(k-N); \qquad \textbf{(3.84)}$$

und der Einfluss auf den aktuellen Beobachterwert zu

$$\Delta\hat{y}(k) = C(k)\cdot\prod_{i=1}^{N-1} A(i)\cdot L(k-N)\cdot\varepsilon(k-N) \qquad \textbf{(3.85)}$$

Der kompensierte Messwert ergibt sich damit zu

$$\hat{y}(k) = \tilde{y}(k) - \Delta\hat{y}(k); \qquad \textbf{(3.86)}$$

Der Beobachter wird mit der Abtastrate des Regelsystems betrieben. Der Rechenaufwand pro Regelzyklus bei einem System l-ter Ordnung beträgt:

l^2+3l Multiplikationen und l^2+2l-3 Additionen ohne Totzeitkorrektur und Messwertsynchronisation. Bei der Totzeitkorrektur kommen noch $(N-1)l^3+2l+1$ Multiplikationen sowie $(N-1)(l^3-l^2)+2l$ Additionen dazu. Die angesprochene Messwertsynchronisation schlägt dann mit noch einmal 7 Multiplikationen und 6 Additionen zu Buche.

Der Einfluss von Mess- und Prozessrauschen ist abhängig von der Wahl der Beobachterpole. Bei der Kalman-Filter-Variante wird der Rauscheinfluss, wie im vorigen Abschnitt beschrieben, minimiert und auch bei stabilen Luenberger-Beobachtern ist ein gutes Rauschverhalten zu erreichen.

3.3.3 Vergleich der Verfahren:

Zum Vergleich und zur Abschätzung der Leistungsfähigkeit der Synchronisations-Verfahren wurde exemplarisch eine Strecke mit PT_2-Verhalten mit verschieden frequenten Sinussignalen beaufschlagt. Es wurde sowohl das Messrauschen als auch das Prozessrauschen variiert, die Messwertfrequenz und die Totzeit verändert.

Soweit nicht anders angegeben, wurden folgende Parametereinstellungen verwendet:

- Abtastfrequenz des Regelsystems 2000Hz
- Abtastfrequenz des Sensors 1953Hz
- Dämpfung der PT_2-Strecke 0,5
- Resonanzfrequenz der PT_2-Strecke 100Hz
- kein Mess- oder Prozessrauschen
- keine Totzeit der Messwerte
- Signalfrequenz 30Hz
- Kovarianzmatrizen: Z=1 und W=10

3.3.3.1 Abhängigkeit von der Messwertfrequenz

Bei den Modell basierten Methoden hat das Verhältnis Regelfrequenz zu Messwertfrequenz keinen Einfluss, da diese Verfahren synchron mit der Regelsystemfrequenz betrieben werden und nur mit den mehr oder weniger synchronisierten Messwerten abgeglichen werden.

Aus (3.43) folgt für ein sinusförmiges Signal

$$\varepsilon_{\max} = 2\pi f_s \cdot t_{a1}; \qquad \textbf{(3.87)}$$

der lineare Zusammenhang zwischen Synchronisationsfehler und Abtastrate t_{a1} bzw. reziproken Zusammenhang mit der Messwertfrequenz f_{a1} bei einem Halteglied 0.Ordnung.

Erweitert man (3.43) für die anderen Daten basierten Verfahren so erhält man

$$\varepsilon_{\max} = \max\{f'(t)\} \cdot t_{a1} - g(t_{a1}); \qquad \textbf{(3.88)}$$

wobei g(t) das jeweilige Verfahren beschreibt. Bei den beiden Haltegliedern 1. Ordnung (FOH und FTOH) ergibt g(t) zu (3.45) bzw. (3.56). Da die Steigung der Extrapolations-Geraden m_n in beiden Fällen ebenfalls linear von t_{a1} abhängt folgt der quadratische Zusammenhang zwischen Synchronisationsfehler und t_{a1}. Beim Halteglied 3.Ordnung ergibt sich durch ähnliche Überlegungen, dass hier der Synchronisationsfehler proportional t_{a1}^4 ist.

In Bild 3.3.23 sind diese Abhängigkeiten zur Messwertfrequenz $f_{a1} = 1/t_{a1}$ für die Signalfrequenzen 30Hz und 10Hz abgebildet.

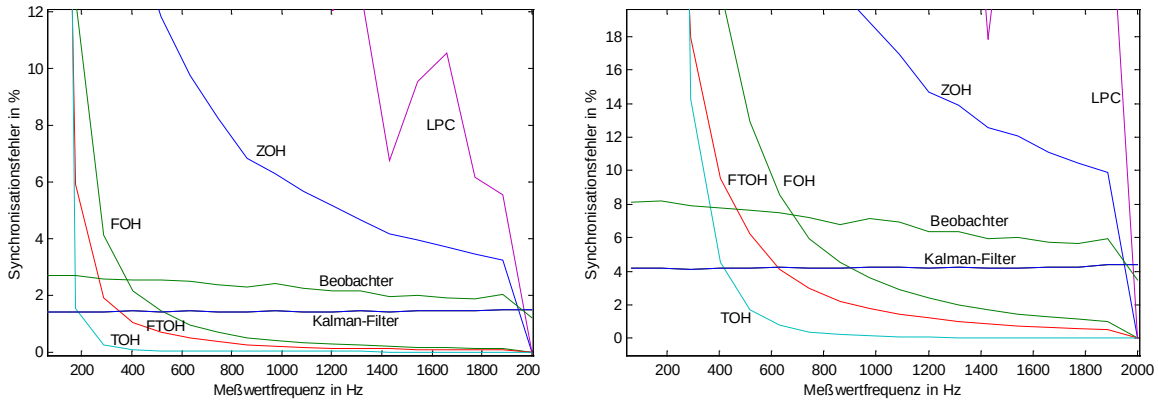

Bild 3.3.23 : Abhängigkeit von der Messfrequenz: links f_s=30Hz, rechts f_s=10Hz

3.3.3.2 Abhängigkeit von der Signalfrequenz

Bild 3.3.24 zeigt bei den Modell basierten Verfahren einen linearen Zusammenhang zwischen Synchronisationsfehler und Signalfrequenz. Dies lässt sich aus der resultierenden Phasenlage zwischen geschätztem und realem Messwert erklären.

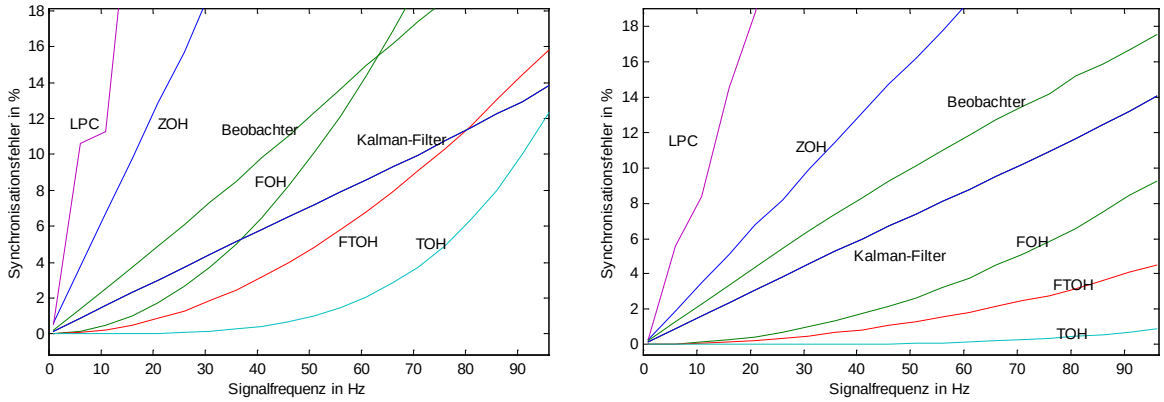

Bild 3.3.24 : Abhängigkeit von der Signalfrequenz: links f_{a1}=1001Hz, rechts f_{a1}=1953Hz

Aus (3.87) folgt für ein Halteglied 0.Ordnung ebenfalls ein linearer, jedoch mit stärkerer Steigung versehener Zusammenhang. Die gleichen Überlegungen zur Gleichung (3.88) wie im vorigen Punkt ergeben für die Halteglieder 1.Ordnung wieder eine Proportionalität zu f_s^2 und für das Halteglied 3.Ordnung zu f_s^4.

In Bild 3.3.24 sind diese Zusammenhänge für f_{a1}=1953Hz und f_{a1}=1001Hz wiedergegeben.

3.3.3.3 Rauschabhängigkeiten

Das Messrauschen führt bei den Daten basierten Verfahren zu einem linearen Ansteigen des Synchronisationsfehlers mit den Steigungen, die sich mit den Gleichungen

(3.44),(3.48), (3.55) und (3.59) ergeben. Die Daten basierten Verfahren werden vom Prozessrauschen nicht beeinflusst, da nur der Sensorausgang verwendet wird und es so nicht ins Verfahren mit eingeht.

Die beiden Modell basierten Varianten Kalman-Filter und Beobachter werden weder vom Messrauschen noch vom Prozessrauschen beeinflusst, da es im Streckenmodell mitspezifiziert wird.

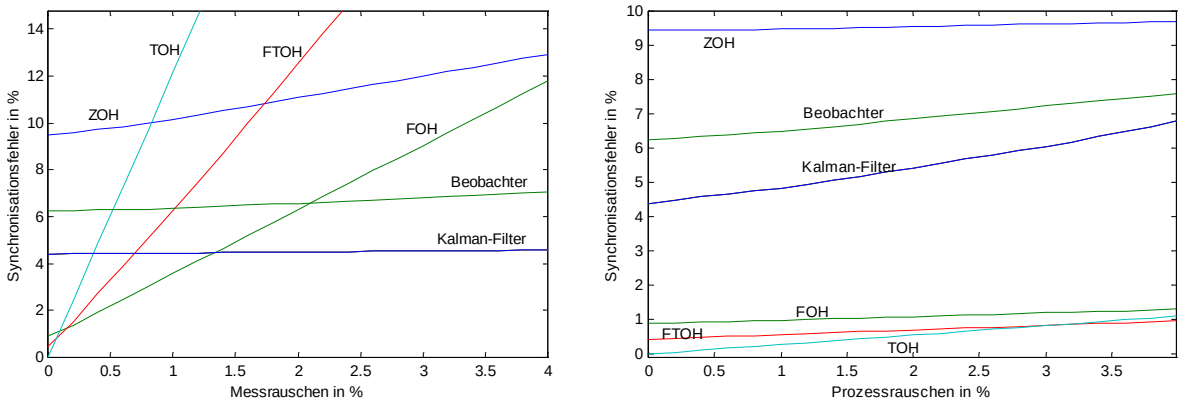

Bild 3.3.25 Rauschabhängigkeit: links Messrauschen, rechts Prozessrauschen

3.3.3.4 Abhängigkeit von der Totzeit

Der Einfluss der Totzeit auf den Synchronisationsfehler verhält sich in erster Linie wie der Einfluss der Messwertfrequenz, weil eine Totzeit nur die Zeit, die man in die Zukunft extrapolieren muss, vergrößert, was einer Verkleinerung der Messwertfrequenz gleichkommt. In Bild 3.3.26 links ist der Einfluss bei kleinen Totzeiten zu sehen. Bei großen Totzeiten (rechtes Bild) ist der Fehler der Daten basierten Verfahren so groß, dass sie nicht mehr angezeigt werden. Beim Kalman-Filter kann man hier zum ersten Mal einen Unterschied der drei vorgestellten Varianten erkennen. Wie auch nicht anders zu erwarten war, liefert die Variante mit Messwertsynchronisation und Totzeitkorrektur das beste Ergebnis, auch wenn es nur ca. 1% ausmacht.

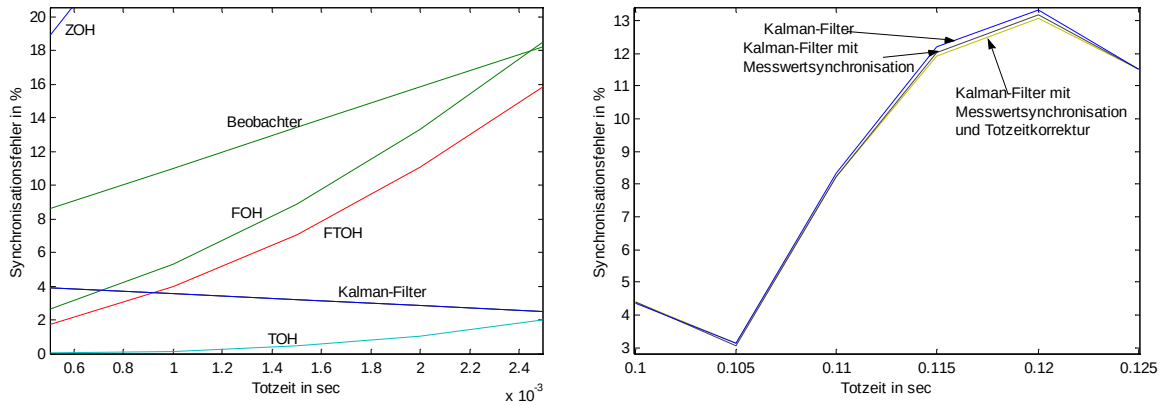

Bild 3.3.26 : Abhängigkeit von der Totzeit

3.3.3.5 Weitere Einflüsse auf den Kalman-Filter

Die Modell basierten Verfahren im allgemeinen, der Kalman-Filter hier im speziellen, werden von weiteren Parametern beeinflusst. Die Ergebnisse hängen in erster Linie von der Genauigkeit bzw. vom Übereinstimmen der verwendeten Streckenmodelle ab. Die Struktur der Streckenmodelle beeinflusst die Genauigkeit des Kalman-Filters nicht, solange die Voraussetzungen zum Entwurf eines Kalman-Filters (siehe 3.3.2.3) eingehalten werden. Die Modellierung des Rauschens geht dabei wesentlich in den zu erwartenden Fehler mit ein. Der Einfluss des Rauschens wird mit den beiden Kovarianzmatrizen bei der Berechnung im Kalman-Filter bestimmt. Wie in Bild 3.3.27 zu sehen, gibt es für jede Kovarianzmatrix einen optimalen Wert, der dann den Synchronisationsfehler minimiert. Durch Veränderung der Kovarianzmatrizen kann man somit den Kalman-Filter optimal an die Gegebenheiten anpassen. Durch die Wahl der optimalen Werte würde sich der Fehler in den vorigen Graphen um 1-2% reduzieren.

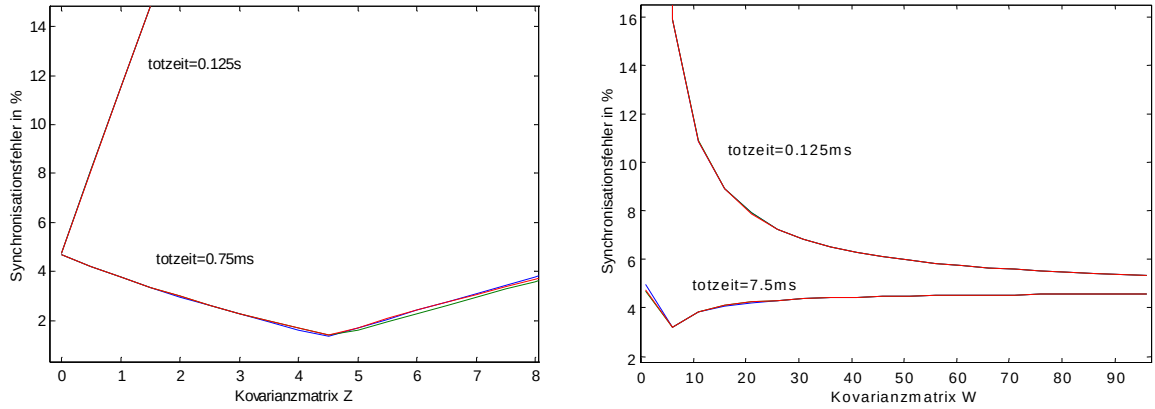

Bild 3.3.27 : Einflüsse der Kovarianzmatrizen Z und W

3.3.3.6 Zusammenfassung

In Tabelle 3.3.2 sind die Abhängigkeiten der einzelnen Methoden nochmals zusammengefasst. Man kann erkennen, dass die Daten basierten Verfahren nur für kleine oder keine Totzeiten verwendet werden können, zudem darf das Verhältnis Messwertfrequenz zu Regelfrequenz nicht zu klein werden. Ergeben sich für die Daten basierten Verfahren ähnlich große Fehler wie für die Modell basierten, so sind sie aufgrund des geringeren Rechenaufwands vorzuziehen. In [16] werden Kalman-Filter und Beobachter hinsichtlich des Rechenzeitbedarfs näher untersucht. Dabei schneidet, wie zu erwarten war, der erweiterte Kalman-Filter am schlechtesten ab.

Für große Totzeiten und sehr niedrige Abtastraten hingegen kommt nur der Kalman-Filter in Frage, ebenso bei sehr verrauschten oder gestörten Messwerten. Für nicht zyklisch anfallende Messwerte können auch nur die Modell basierten Varianten verwendet werden, die Daten basierten Verfahren benötigen äquidistante Abtastzeitpunkte.

Verfahren	Synchronisationsfehler abhängig von				
	Messwertrate	Signal-frequenz	Totzeit	Mess-rauschen	Prozess-rauschen
Daten basiert					
ZOH	Linear	linear	linear	konstant	Konstant
FOH	quadratisch	quadratisch	quadratisch	Linear	Konstant
FTOH	quadratisch	quadratisch	quadratisch	Linear	Konstant
TOH	4.Ordnung	4.Ordnung	4.Ordnung	Linear	Konstant
Modell basiert					
Kalman-Filter	konstant	linear	konstant	konstant	Konstant
Beobachter	konstant	linear	linear	konstant	Konstant

Tabelle 3.3.2 : Abhängigkeiten der Synchronisationsfehler der Verfahren

3.4 Sensorfusion

3.4.1 Gewichteter Mittelwert

Ein einfaches Verfahren zur Sensorfusion auf Signal-Level-Ebene, das meist zur Fusion von redundanten Sensoren verwendet wird, ist die Bildung des gewichteten Mittelwerts. Er ist für n-Sensoren definiert als

$$u_k^* = \frac{1}{n}\sum_{i=1}^{n} z_{k,i} \cdot g_i; \qquad \textbf{(3.89)}$$

Die einzelnen Sensorwerte müssen daher dieselbe physikalische Größe messen und auf das gleiche Referenzsystem kalibriert sein. Die konstanten Gewichtungsfaktoren g_i bestimmen den Einfluss der einzelnen Sensoren zum fusionierten Wert. Die Wahl der Gewichtungsfaktoren erfolgt anhand der Qualität der Sensordaten, dabei gilt

$$\sum_{i=1}^{n} g_i = 1; \qquad \textbf{(3.90)}$$

Bei Verwendung eines intelligenten Sensorauswahlprinzips wie der Perception Controller in [20] werden die Gewichtungsfaktoren anhand der von den Sensor gelieferten Daten angepasst. Ein Ausblenden von Ausreißern oder von defekten Sensoren kann dadurch erreicht werden.

3.4.2 Kalman-Filter

Im Gegensatz zu der starren Fusion mit konstanten Gewichtungsfaktoren von 3.4.1 stellt die Sensorfusion mit dem Kalman-Filter einen gewichteten Mittelwert mit adaptiven Gewichtungsfaktoren dar. Die Gewichtung erfolgt anhand der für den Kalman-

Filter notwendigen Kovarianzmatrizen und passt sich somit an die aktuellen Gegebenheiten an. Beim Kalman-Filter können auch die physikalischen Messgrößen der einzelnen Sensoren unterschiedlich sein, sie müssen lediglich eine beliebige, aber bekannte, Linearkombination der Systemzustände darstellen. Es kann sowohl eine Feedforward als auch eine Feedback-Struktur verwendet werden.

Wie man an den vorhergehenden Kapiteln sieht, ist der Kalman-Filter das Allzweckmittel schlechthin und kann in jedem der drei zur Sensorfusion notwendigen funktionellen Blöcke eingesetzt werden. Doch durch den hohen Rechenaufwand, der mit dem Kalman-Filter verbunden ist, wäre es wünschenswert, wenn man mehrere Kalman-Filter zu einem zusammenführen könnte, wie es in Bild 3.4.1 zu sehen ist.

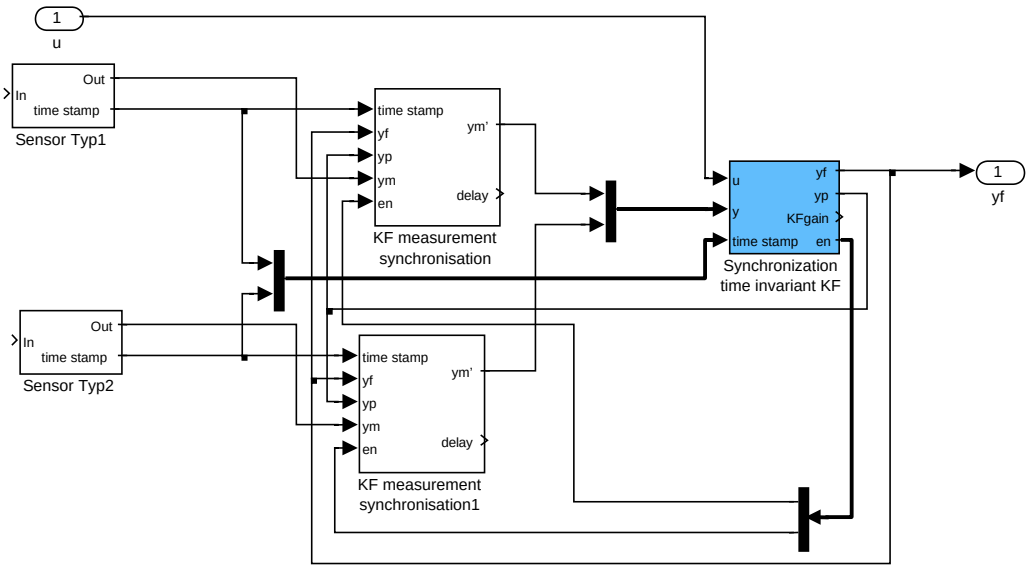

Bild 3.4.1 : zentraler Kalman-Filter

Die Messwertsynchronisation (siehe Bild 3.3.19) und die Totzeitkorrektur (siehe Bild 3.3.21) müssen bei Bedarf für jeden Sensor separat durchgeführt werden. Die Sensorfusion sowie eine mögliche Signalvorverarbeitung können dann für alle Sensoren in einem Kalman-Filter (dunkel hinterlegter Block in Bild 3.4.1) zusammengefasst werden. Für das Systemmodell eines 1-Sensor-Regelkreises gilt nach (3.74)

$$\begin{aligned} x(k+1) &= A(k)\cdot x(k) + B(k)\cdot u(k) + D(k)\cdot z(k), \\ y_1(k) &= c_1^T(k)\cdot x(k) + w_1(k). \end{aligned} \qquad \textbf{(3.91)}$$

Wenn man (3.91) auf n Sensoren erweitert ergibt sich

$$y(k) = \begin{bmatrix} y_1(k) \\ y_2(k) \\ \cdots \\ y_n(k) \end{bmatrix} = \underbrace{\begin{bmatrix} c_1^T(k) \\ c_2^T(k) \\ \cdots \\ c_n^T(k) \end{bmatrix}}_{C} \cdot x(k) + \underbrace{\begin{bmatrix} w_1(k) \\ w_2(k) \\ \cdots \\ w_n(k) \end{bmatrix}}_{w}; \qquad \textbf{(3.92)}$$

mit $c_i^T = c_j^T$ für identische Sensoren.

In der Kovarianzmatrix W wird beim Kalman-Filterentwurf das Messrauschen $w_i(k)$ der einzelnen Sensoren berücksichtigt. Eine Signalvorverarbeitung kann allgemein wie folgt angegeben werden:

$$\begin{aligned} z(k+1) &= A_{SVV} \cdot z(k) + B_{SVV} \cdot y(k), \\ z^*(k) &= C_{SVV} \cdot z(k) \end{aligned} \qquad \textbf{(3.93)}$$

Mit (3.91) und (3.92) ergibt sich dann das Systemmodell des zusammengefassten Kalman-Filters zu

$$\begin{aligned} \begin{bmatrix} x(k+1) \\ z(k+1) \end{bmatrix} &= \begin{bmatrix} A(k) & 0 \\ B_{SVV} \cdot C(k) & A_{SVV} \end{bmatrix} \cdot \begin{bmatrix} x(k) \\ z(k) \end{bmatrix} + \begin{bmatrix} B(k) \\ 0 \end{bmatrix} \cdot u(k) + \begin{bmatrix} D(k) \\ B_{SVV} \end{bmatrix} \cdot \begin{bmatrix} z(k) \\ w(k) \end{bmatrix}, \\ u^*(k) &= \begin{bmatrix} C^*(k) \\ C_{SVV} \end{bmatrix} \cdot \begin{bmatrix} x(k) \\ z(k) \end{bmatrix} + \begin{bmatrix} E^* \\ 0 \end{bmatrix} \cdot w(k); \end{aligned} \qquad \textbf{(3.94)}$$

wobei $C^*(k)$ durch Nullsetzen der Zeilen von vorverarbeiteten Signalen aus $C(k)$ hervorgeht. Die Matrix E^* besitzt eine „1" für ausgekoppelte und eine „0" für vorverarbeitete Sensorsignale und dient dazu das Messrauschen zu berücksichtigen. (3.94) zeigt, dass durch Erweiterung des Systemmodells ein Kalman-Filter ausreicht, um die Aufgaben der drei funktionalen Blöcke Synchronisation, Signalvorverarbeitung und Fusion zu erfüllen.

3.5 Simulation

Zur Simulation des dynamischen Verhaltens des Regelsystems wird das Simulationswerkzeug SimuLink [72], das auf MatLab basiert, eingesetzt. Jeder SimuLink-Block wird durch einen Eingangsvektor u, einen Ausgangsvektor y und einen Zustandsvektor x charakterisiert. Der Zustandsvektor kann aus kontinuierlichen, diskreten oder einer Kombination aus beiden bestehen. Der mathematische Zusammenhang ergibt sich zu

$$\begin{aligned} y &= f_{output}(t,x,u) \\ x_{d_{k+1}} &= f_{update}(t,x,u) \\ x_c' &= f_{derivative}(t,x,u) \quad mit \quad x = \begin{bmatrix} x_c \\ x_{d_k} \end{bmatrix} \end{aligned} \qquad \textbf{(3.95)}$$

Die Simulation besteht aus zwei Phasen: Initialisierung und Simulation. In der Initialisierungsphase geschieht folgendes:

1. Die symbolischen Blockparameter werden von MatLab auf ihre Gültigkeit geprüft und durch die numerischen Werte ersetzt
2. Die Modellhierarchie wird aufgelöst. Der Inhalt der Teilsysteme, die nicht bedingt ausführbare Blöcke sind, wird statt der Blöcke eingefügt.
3. Die resultierenden Basisblöcke werden in der Reihenfolge sortiert, in der ihre Zustände aktualisiert werden müssen. Blöcke mit direktem Durchgriff werden zu dem speisenden Block gerechnet.

4. Die Verbindungen zwischen den Blöcken wird auf ihre Konsistenz geprüft.

Die Simulation eines Modells wird mit numerischer Integration – es gibt davon verschiedene Verfahren – durchgeführt. Zuerst werden die Ausgänge der Blöcke anhand der zuvor genannten Sortierung bestimmt, danach in einem zweiten Schritt werden für jeden Block die Ableitungen zur momentanen Simulationszeit berechnet. Das Ergebnis wird der numerischen Integrationsmethode zurückgespeist, um die Zustandsvektoren zum nächsten Simulationszeitpunkt zu bestimmen. Um diskontinuierliche Signale und zustandsverändernde Ereignisse genauer behandeln zu können, wird ein Nulldurchgangs-Detektions-Verfahren verwendet und die Schrittweite der numerischen Integration (außer bei Verfahren mit konstanter Schrittweite) dementsprechend angepasst.

Beispielhaft wird nun eine diskrete PID-Folgeregelung für ein PT_2-Streckenglied mit einer Resonanzfrequenz von 100Hz simuliert. Es stehen 3 verschiedene Sensoren, zwei Beschleunigungssensoren und eine CCD-Kamera zur Verfügung. Die Daten der Beschleunigungssensoren sind Offset behaftet (Gravitation), verrauscht (± 100mm/s²) und weisen eine Drift auf. Sie arbeiten mit einer externen Abtastrate von 1953Hz und haben eine Totzeit von zwei Abtastschritten. Die CCD-Kamera hat eine variable Totzeit im Bereich von 0.5 und 1.0 Sekunden. Die gelieferten Daten sind nur leicht verrauscht (± 2μm). Die Führungsgröße sei ein Sinussignal mit einer Amplitude von 0.25mm und einer Frequenz von 10Hz. Das Prozessrauschen wird mit ± 5μm angegeben. Unter Verwendung der Algorithmen dieses Kapitels soll der Einfluss der Sensorwahl simulativ und der Unterschied zum idealen Entwurf gezeigt werden. In Bild 3.5.1 ist das Simu-Link-Blockdiagramm für die erste Realisierung aus Tabelle 3.5.1 abgebildet. Die mit Schatten hinterlegten Blöcke sind die zur Sensorfusion implementierten Verfahren und werden für die verschiedenen Konfigurationen einfach nur ausgetauscht.

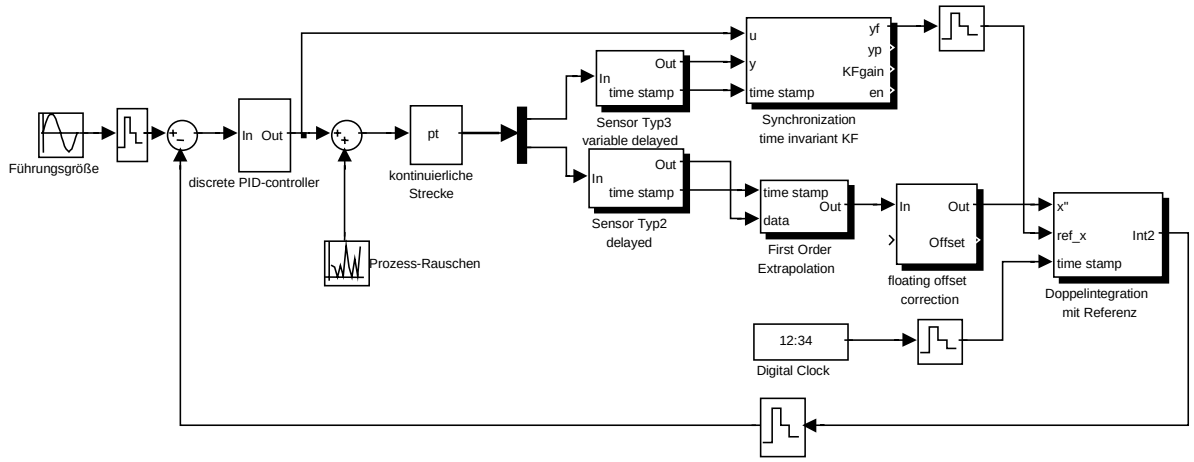

Bild 3.5.1 : Simulations Blockdiagramm

In Tabelle 3.5.1 sind die Simulationsergebnisse für verschiedene Sensorkombinationen und Verarbeitungsalgorithmen zusammengefasst. In Bild 3.5.2 sind für die Realisierung aus Bild 3.5.1 die Simulationsergebnisse zu sehen. Auf der linken Seite sind exemplarisch die Sensordaten (oben CCD-Sensor, unten Beschleunigungssensor) abgebildet. Rechts unten sind die Führungsgrößen zusammen mit dem simulierten Ergebnis der PID-Regelung mit den Daten der beiden Sensoren dargestellt. Es ergibt sich ein maximaler Regelfehler, der kleiner 3.2% ist.

Vergleicht man die prozentualen Regelfehler der verschiedenen Realisierungsvarianten aus Tabelle 3.5.1, so erkennt man, dass man nur durch die Wahl der Sensoren und Fusionsalgorithmen Regelsysteme mit Fehlern zwischen 3% und 200% entwerfen kann. Es ergibt sich sogar eine instabile Variante. Durch Tunen der Entwurfsparameter könnten die Ergebnisse noch verbessert werden. Der Fehler, der sich mit der idealen Simulation (ohne Störsignale und Rauschen) ergibt, ist um den Faktor 10 geringer als der der besten realen Variante. Bei der vierten Alternative wird der Einfluss der Signalvorverarbeitung deutlich. Synchronisiert man die Sensoren vor der Integration, so erhält man einen um den Faktor 5 größeren Fehler als wenn man die beiden Blöcke vertauscht. Dies lässt sich durch die Rauschempfindlichkeit der FOH-Synchronisation erklären, denn nach der Integration ist das Gewicht des Prozess- und Messrauschens gegenüber dem Nutzsignal wesentlich geringer.

Sensoren	Synchronisation	Sensorvorverarbeitung	Fusion	Fehler [%]
CCD B-Sensor	Kalman-Filter FOH	- Doppelintegration mit Referenz	- -	3.125
CCD B-Sensor	Kalman-Filter FTOH	- Doppelintegration mit PID-Feedback	gewichteter Mittelwert	instabil
CCD B-Sensor	Kalman-Filter Kalman-Filter	- Doppelintegration mit mit Kalman-Filter	Kalman-Filter	~20
B-Sensor	FOH	Doppelintegration mit I+-Feedback	-	~200 bzw. ~40
B-Sensor B-Sensor	FOH FOH	Doppelintegration mit I+-Feedback Doppelintegration mit I+-Feedback	gewichteter Mittelwert	~40
B-Sensor B-Sensor	FOH FOH	Doppelintegration mit I+-Feedback Doppelintegration mit PID-Feedback	gewichteter Mittelwert	~30
CCD	Kalman-Filter	-	-	~20 (Phase)
Ideal	-	-	-	0.3125

Tabelle 3.5.1 : verschiedene Sensorkombinationen

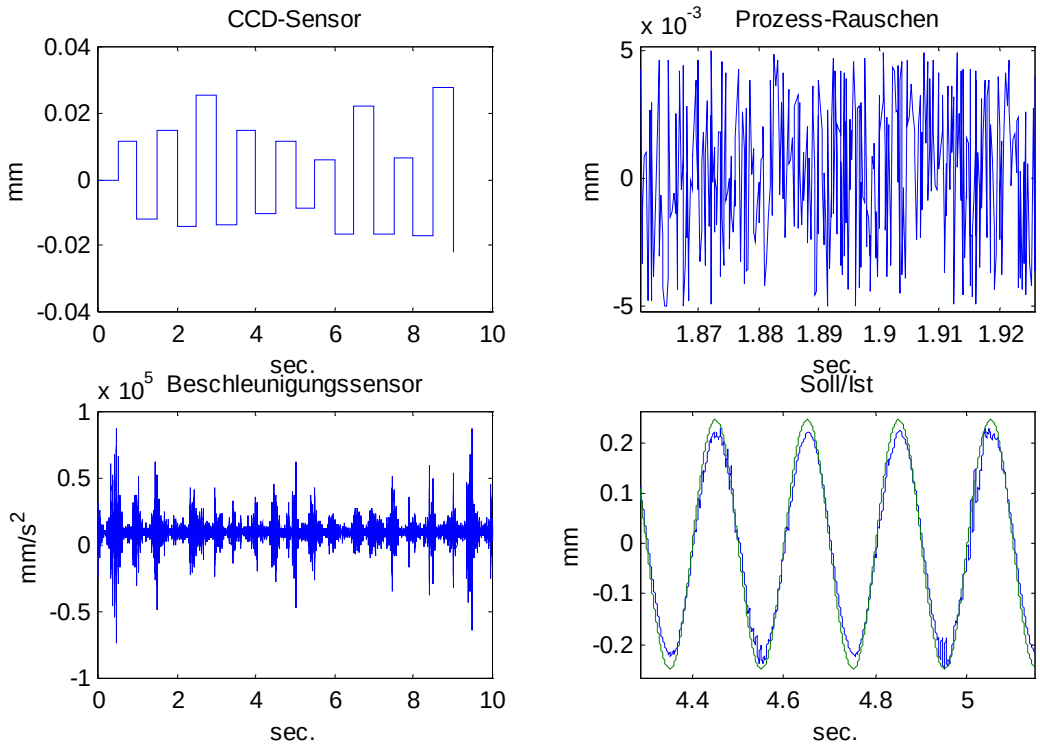

Bild 3.5.2 : Simulation realer Regelsysteme

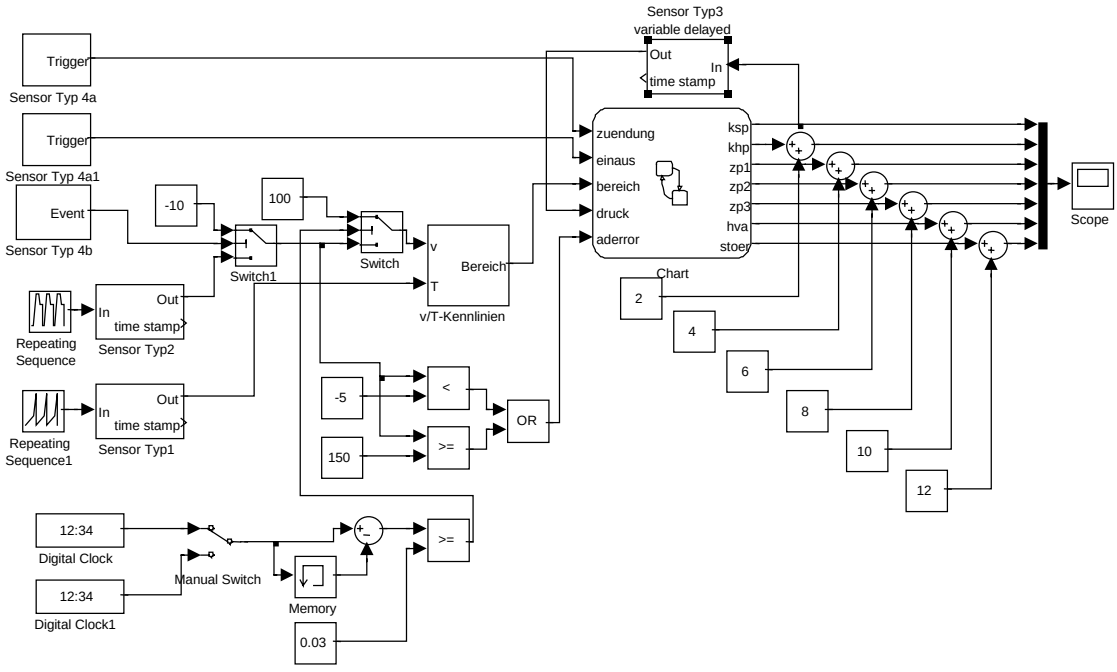

Bild 3.5.3 : Blockdiagramm der Beispiel-Steuerung

Die Stabilität der simulierten Systeme kann mit SimuLink durch Bestimmung der Polstellen des am Arbeitspunkt linearisierten Simulationsmodells ermittelt werden. Ebenso kann das Frequenz- und Sprungverhalten berechnet werden.

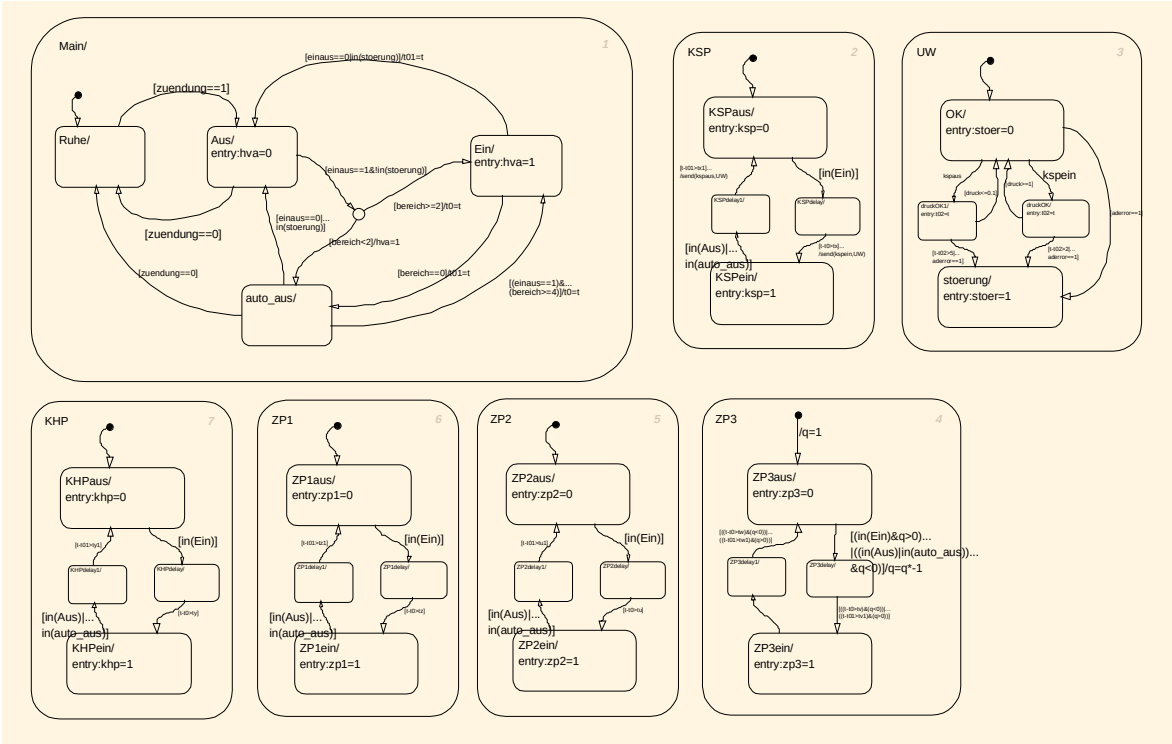

Bild 3.5.4 : Zustandsautomat der Beispiel-Steuerung

Ein weiterer Anwendungsbereich der entwickelten Toolbox sind diskrete Steuerungen, wie in Bild 3.5.3 zu sehen. Triggersignale, die Zustandsübergänge des Zustandsautomaten aus Bild 3.5.4 bewirken, können mit dem Sensortyp 4a (siehe 3.1.4.1) simuliert werden. Nicht vorhersehbare Ereignisse, wie Ausfälle von Bauteilen, können mit dem Sensortyp 4b (siehe 3.1.4.2) modelliert werden.

Die Beispiel-Steuerung aus Bild 3.5.3 und Bild 3.5.4 soll es ermöglichen, bei einem Nutzfahrzeug bei Stillstand und auch während der Fahrt einen hydrostatischen Vorderachsantrieb ein- und auszuschalten. Zum Ein- und Ausschalten muss die Steuerung mehrere Magnetschaltventile (ZP1-ZP3) und die Kupplungen der Haupt- und Speisepumpe in einem zeitlich festen Ablauf ansteuern. Anhand eines Kennlinienfelds zwischen Geschwindigkeit und Öltemperatur wird entschieden, ob der Antrieb ein-, aus-, automatisch ein- oder automatisch ausgeschaltet wird. Die Steuerung muss auch Fehler (AD-Wandlerausfall oder Bruch der Hydraulikleitung) erkennen und den Antrieb daraufhin abschalten. Die Sensoren für Geschwindigkeit, Öltemperatur und Hydraulikdruck werden mit den schon bekannten Sensormodellen des Typs 1-3 modelliert.

In Bild 6.0.1 sieht man im oberen Diagramm die Ergebnisse, die sich bei der Simulation des Geschwindigkeitsprofils, das im unteren Diagramm dargestellt ist, ergeben. Nach dem Einschalten der Zündung und des Schalters für den Antrieb wird der Antrieb aktiviert. Beim Überschreiten der Grenzgeschwindigkeit nach ca. 9 Sekunden erfolgt das automatische Abschalten des Antriebs, der dann nach abbremsen unter die Grenzgeschwindigkeit wieder automatisch eingeschaltet wird. Nach ca. 22 Sekunden tritt eine Störung, der Ausfall des Geschwindigkeitssensors, ein und es erfolgt die endgültige Abschaltung des Antriebs.

Bei der Simulation von Systemen mit verschiedenen Abtastraten in SimuLink muss man darauf achten, dass die eingesetzten Blöcke mit der richtigen Abtastrate simuliert wer-

den. Standardmäßig, wenn nicht explizit anders angegeben, wird die Abtastrate vom speisenden Block geerbt. Man kann dies einfach durch Aktivierung der verschiedenfarbigen Darstellung der Abtastraten im SimuLink-Blockdiagramm überprüfen. Ansonsten ist die Simulation in SimuLink auch von hybriden Systemen unproblematisch, vor allem wenn man sich an die in [72] angegeben Vorschläge zur Wahl des Integrationsverfahren hält.

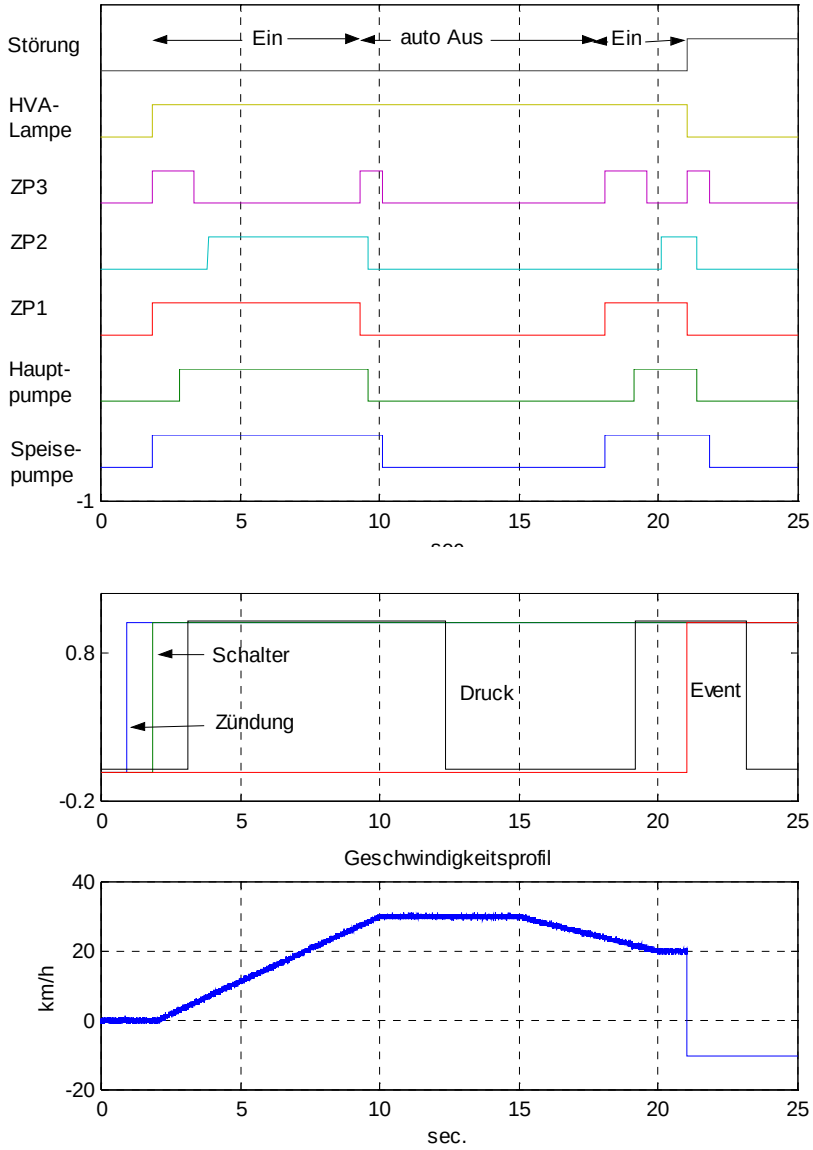

Bild 6.0.1 : Simulationsergebnisse der Beispielsteuerung

3.6 HIL-Simulation

3.6.1 Automatische C-Codegenerierung

Aus einem SimuLink-Blockdiagramm kann automatisch mit dem Real-Time Workshop [73] der ANSI-C-Code für eine Reihe von Zielsystemen generiert werden. Man unterscheidet hierbei zwischen Echtzeitanwendungen (VxWorks, dSPACE, DOS) und Nicht-Echtzeitanwendungen (UNIX, Windows).

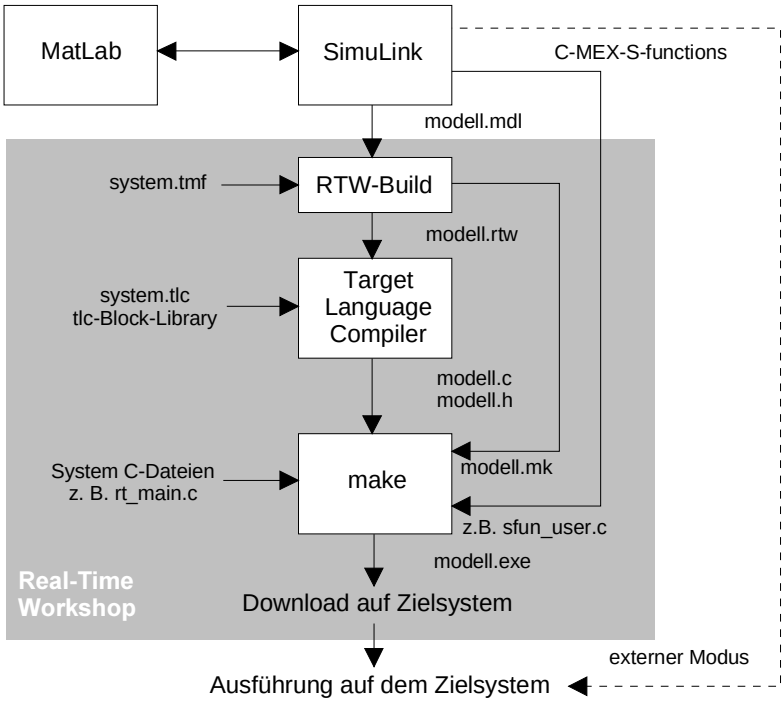

Bild 3.6.1 : automatische C-Codegenerierung mit Real-Time Workshop

Die offene Architektur des Real-Time Workshops ist in Bild 3.6.1 dargestellt und ermöglicht, einfach die Auto-Codegeneration zu modifizieren oder zu erweitern. Das SimuLink-Modell wird mit zielspezifischen Regeln aufbereitet (z. B. Festlegung der Ausführungsreihenfolge der Blöcke, Überprüfung des Integrationsverfahrens und Vergabe von Taskprioritäten bei Multitasking) bevor es dem Codegenerator zugeführt wird. Der Target Language Compiler erzeugt anhand der für jeden Block in Target-Language-Compiler-(TLC)-Dateien abgelegten Anweisungen den C-Code für das entsprechende Blockdiagramm. Die Regeln in den TLC-Dateien sind in von einer Programmiersprache unabhängigen Metasprache verfasst. Somit kann durch Änderung der TLC-Dateien der erzeugte C-Code beeinflusst werden, bzw. kann auch der Code in einer anderen Programmiersprache erzeugt werden (z. B. VHDL). Aus dem Zielsystem spezifischen Template-Makefile wird das Makefile für den generierten C-Code generiert, das die notwendigen Abhängigkeiten und Vorgehensweisen für die Compilierung beinhaltet. Es werden neben modellunabhängigen System-C-Dateien auch C-Dateien für ausgewählte erweiterte Funktionalitäten wie Daten-Logging und „externer Modus" angegeben. Außerdem stellt es die Möglichkeit dar, nach erfolgreicher Compilierung das Executable auf dem Zielsystem zu laden und zu starten.

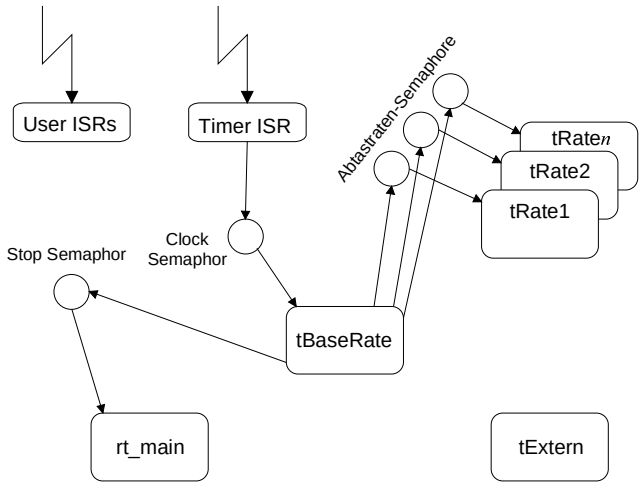

Bild 3.6.2 : Task-Struktur

Benutzereigene C-Dateien können in SimuLink in Form einer „S-Function" eingebunden werden. Zur Software-Simulation werden der darin enthaltene C-Code zu sogenannten plattformabhängigen MEX-Dateien (MatLab Executable) übersetzt und bei der Codegenerierung entweder der C-Code dazugelinkt oder die Anweisungen in einer TLC-Datei, falls vorhanden, eingebunden. Durch Definition der Compilerkonstanten „MATLAB-MEXFILE" bei der MEX-Datei-Compilierung kann über die Compilerdirektive „#ifdef" für die Software-Simulation ein anderes Verhalten als für die Hardware-Simulation für den betreffenden Block spezifiziert werden. Alle für die Simulation notwendigen Informationen werden in einer „SimStruct"-Variablen abgelegt. Die S-Functions des Level 1 besitzen folgende festgelegte Struktur:

- Header, der den S-Function-Namen definiert
- mdlInitializeSizes-Funktion: initialisiert die SimStruct-Variable (z. B. Anzahl der Ein- und Ausgänge, Anzahl der kontinuierlichen und diskreten Zustände)
- mdlInitializeSampleTimes-Funktion: legt die Anzahl und Offsets der Abtastraten in der SimStruct-Variable fest
- mdlInitializeConditions-Funktion: definiert die Anfangszustände
- mdlOutputs-Funktion: berechnet anhand des Eingangs- und Zustandsvektors den Ausgangsvektor
- mdlUpdate-Funktion: aktualisiert die Zustände nach jedem Simulationsschritt
- mdlDerivative-Funktion: berechnet bei variabler Schrittweite die 1. Ableitung
- mdlTerminate-Funktion: bestimmt das Verhalten bei Simulationsende
- festgelegter Trailer

Mit diesen S-Functions kann auch die Hardwareanbindung (DA- bzw. AD-Wandler) realisiert werden. Weitere Details und die erweiterte S-Function Struktur (Level 2) finden sich in [74].

Ein besonderer Simulationsmodus stellt der Aufruf von SimuLink im externen Modus dar. Hier wird nach erfolgreicher Compilierung das Modell auf dem Zielsystem gestartet und mittels „Remote-Procedure-Calls“ können von SimuLink auf dem Entwicklungshost über Ethernet Blockparameter verändert werden.

Das bei Echtzeitanwendungen auf den Zielsystemen ablaufende Modell erhält die in Bild 3.6.2 dargestellte Task-Struktur. Die Initialisierungstask (rt_main) erstellt alle anderen Tasks, Synchronisationselemente (Semaphore) und gibt die erforderlichen Interrupts frei. Anschließend legt sie sich bis zum Simulationsende an einem Synchronisationselement schlafen. Die Task mit der höchsten (=Basis-) Abtastrate (tBaseRate) erhält die höchste Priorität und koordiniert die Tasks der anderen Abtastraten (tRate1 – tRate*n*) über Synchronisationselemente. Zusätzlich gibt es noch die Task (tExtern), die zum Datenempfang beim externen Modus dient. Die tBaseRate-Task wird zyklisch ihrer Abtastrate entsprechend vom Interrupt des Timers über ein Semaphor angestoßen. Die dazugehörige Interrupt Service Routine überprüft zuvor, ob das Semaphor schon wieder zurückgegeben wurde. Ist das nicht der Fall, wurde die Echtzeitbedingung verletzt und die Simulation wird abgebrochen. Bei Verwendung eines PC’s als Zielsystem ist darauf zu achten, dass das Betriebssystem mit der Echtzeituhr (nur 8 einstellbare Frequenzen) und der Timer mit dem Programmable Interrupt Timer (2^{16} Einstellungsmöglichkeiten) betrieben werden, standardmäßig ist es genau umgekehrt.

3.6.2 Hardware-Interfaces

Für die hier vorgestellte Toolbox wurden mittels S-Functions folgende Hardware-Interfaces realisiert:

- ISA-Bus DA- und AD-Wandler
- PCI-Bus DA- und AD-Wandler
- Interrupt getriebene Anbindung des Beschleunigungssensorsystems aus [115] über eine DSP-Erweiterungskarte [3]
- generische Interrupt getriebene RS232-Anbindung
- generische PCI-Interrupt-Anbindung

Um den Zielsystem spezifischen C-Code für die Interruptbehandlung einfacher handhaben zu können, werden für die Interrupt getriebenen Interfaces TLC-Dateien entworfen. Die Abarbeitung der Interrupts erfolgt in Teilsystemen, die mit einem Funktionsaufruf (=function-call) getriggert werden. Pro Interrupt ist ein solches Teilsystem notwendig. Der C-Code aller Blöcke, die sich in diesem Teilsystem befinden, werden in die Interrupt-Service-Routine dieses Interrupts gepackt. Es dürfen sich jedoch keine Blöcke mit kontinuierlichen Zuständen oder festen Abtastraten darunter befinden. Die Trennung von Interrupt-Service-Routine und zyklischem Systemmodell erfolgt mit einem sogenannten „asynchronous rate transition“-Blocks. Die unter 3.3.1 vorgestellten Synchronisations-Blöcke können somit direkt nicht für Interrupt getriebene Eingänge benutzt werden, sondern müssen in zwei Teile aufgespalten werden, in den Teil, der mit der „Abtastrate“ des Interrupts betrieben und ins Interrupt-Service-Routinen-Teilsystem verschoben wird und in den Teil, der mit der Basisabtastrate betrieben werden muss. Die beiden Teile sind dann mit einem „asynchronous rate transistion“-Block zu verbinden. Ein Beispiel hierfür stellt das Bild 3.6.5 dar.

Die Interfaces wurden so realisiert, dass sie bei der Software-Simulation ein neutrales Verhalten aufweisen, d. h. sie werden mit direktem Durchgriff spezifiziert, erben die Abtastrate des speisenden Blocks und geben ihren Eingangsvektor unverändert am Ausgang weiter. Erst bei der Codegenerierung werden die notwendigen Hardwarezugriffe verwendet.

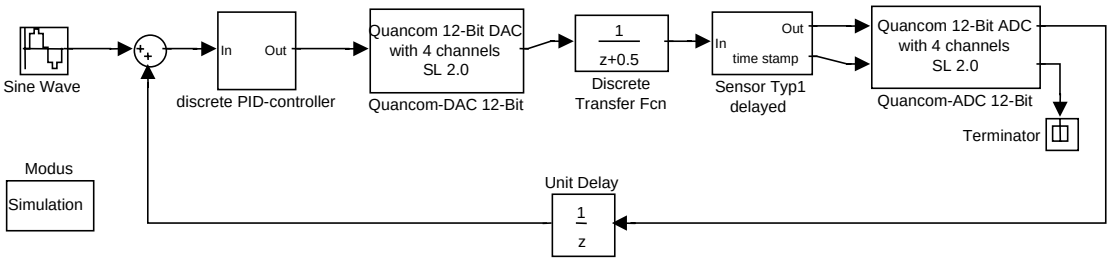

Bild 3.6.3 : Simulation eines Regelkreises mit Hardware-Interfaces

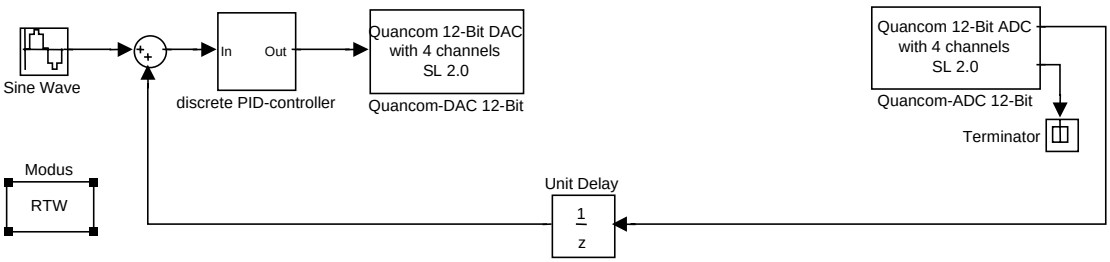

Bild 3.6.4 : Regelkreis mit Hardware-Interface zur Codegenerierung

Die Zuordnung der hier entworfenen Hardware-Interfaces zu dem Simulationsblock erfolgt durch Anhängen des jeweiligen Hardwareblocks. Wie in Bild 3.6.3 zu sehen, folgt auf den Block des Sensormodells direkt der Interfaceblock zu einem AD-Wandler und auf den Block des Reglers folgt der DA-Wandler Interface-Block. Durch das neutrale 1:1 Verhalten der Interfaces kann so ohne weiteres die Software-Simulation für den Regelkreis durchgeführt werden. Um unnötig generierten Code bei der Hardware-in-the-Loop-Simulation zu vermeiden, wurde ein Block zum Umschalten zwischen Software-Simulation und Codegenerierung entwickelt. Die Blöcke zwischen der Hardwareausgabe (z. B. DA-Wandler) und Hardwareeingang (z. B. AD-Wandler) werden bei der Hardware-Simulation nicht benötigt. Es handelt sich dabei um die Spezifikation der Regelstrecke und der Sensoren. Durch das Umschalten des Simulationsmodus werden automatisch diese unnötigen Blöcke sowie die Pseudoausgänge der Hardwareausgänge und die Pseudoeingänge der Hardwareeingänge entfernt. Außerdem wird das Integrationsverfahren von einer variablen Schrittweite auf eine feste Schrittweite mit der Basisabtastrate automatisch mitumgestellt. Die entfernten Blöcke werden in einem temporären Blockdiagramm gespeichert, um den Wechsel zurück ebenfalls zu ermöglichen. In Bild 3.6.4 ist das resultierende Blockdiagramm zur Codegenerierung des ursprünglichen Simulations-Blockdiagramms aus Bild 3.6.3 zu sehen.

3.6.3 Hardware-Simulationsbeispiel

Als einfaches Beispiel sollen die von einem 3D-Beschleunigungssensor [115], [3] gelieferten Daten zur Positionsbestimmung verwendet werden können. Das resultierende zur Codegenerierung verwendete Blockdiagramm ist in Bild 3.6.5 zu sehen. Der Beschleunigungssensor wurde mit dem Sensormodell Typ2 modelliert, die Identifikation der dazu notwendigen Größen wird in 4.3.1.2 näher beschrieben. Der C-Code wurde für ein PC-basiertes VxWorks-System mit dem oben beschriebenen automatischen Verfahren generiert, die Task tBaserate wird mit 2000Hz zyklisch angestoßen, der Beschleunigungssensor löst zyklisch mit einer Frequenz von $\frac{500000}{256} kHz$ einen Interrupt aus.

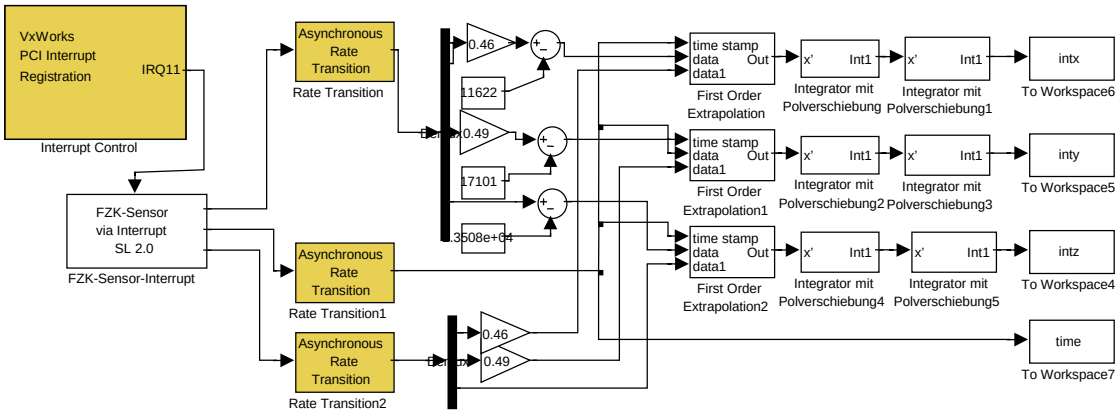

Bild 3.6.5 : Blockdiagramm zur Codegenerierung für das Beispiel

Die 2-fach integrierten Daten, die sich bei einer definierten Bewegung entlang den Seiten eines Quadrats des realen Beschleunigungssensors in der x-y-Ebene ergeben, sind in Bild 3.6.6 links und die vorab simulierten Daten rechts dargestellt.

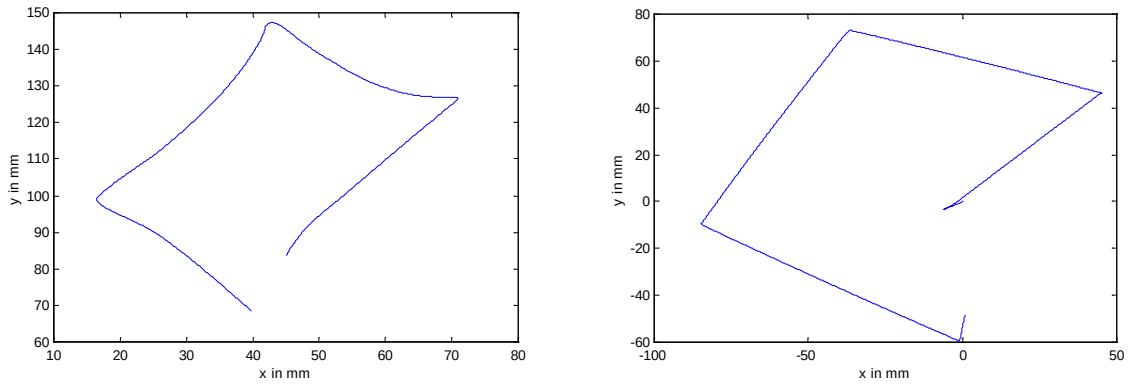

Bild 3.6.6 : HIL-Simulation (links) und Software-Simulation (rechts)

Die gekrümmte Bahn bei den reellen Daten kommt aufgrund der nicht wie in [15] kompensierten Querempfindlichkeiten der Beschleunigungssensorelemente zustande. Man kann bereits in der Simulation erkennen, dass das verwendete Verfahren aufgrund der fehlenden Referenz und niedriger Signalfrequenz zur genauen Positionsbestimmung ungeeignet ist. Die HIL-Simulation mit dem realen Sensor bestätigt dieses Verhalten, was an der guten Übereinstimmung der beiden Kurven gut zu erkennen ist.

4 Anwendungsbeispiel MPE

Von September 95 bis August 98 wurde von 5 Lehrstühlen im Rahmen des Forschungsverbunds Mikrosystemtechnik (FORMIKROSYS) der Bayerischen Forschungsstiftung im Teilprojekt 5 eine Mikropositioniereinrichtung (MPE) für die Präzisionsmontage entwickelt [45]. Die Regelung dieses Multisensor-Werkzeugs wurde im Projektzeitraum konventionell entworfen (Regelungsentwurf A). Das Redesign der Regelung mit den Methoden dieser Arbeit führt zu einer Vereinfachung der Regelstruktur (Regelungsentwurf B). Im folgenden wird die MPE samt Regelung vorgestellt und es werden die beiden verwendeten Regelungsansätze miteinander verglichen.

4.1 Motivation

Die Anwendung von Mikrotechniken ermöglicht nicht nur die Miniaturisierung von Sensorelementen, sondern zunehmend auch neue Produkte mit bisher nicht erreichter Leistungsfähigkeit in anderen Bereichen, wie z. B. in der Kommunikationstechnik. Ein Spektrum solcher Produkte zeigt Bild 4.1.1: ein mikrooptisches Übertragungsmodul (oben), ein LIGA-Planetengetriebe für Mikromotoren (rechts), eine Funkarmbanduhr mit mikro-mechanischen Funktionseinheiten (links) und ein 3D-Beschleunigungssensorsystem mit mikromechanischen Sensorelementen und mikroelektronischen Komponenten (unten)[115].

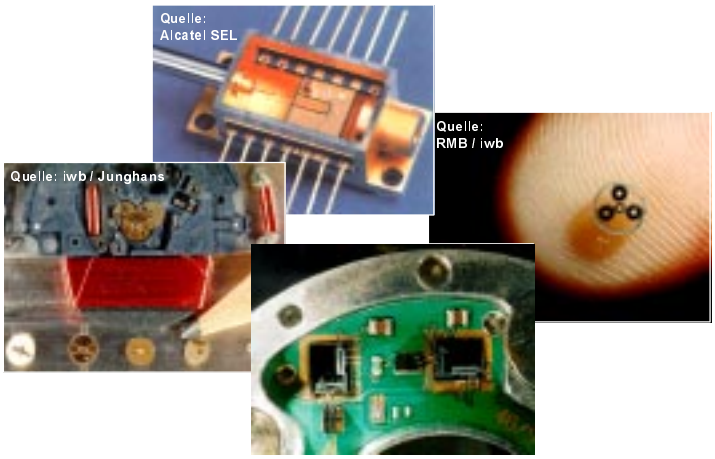

Bild 4.1.1 : Beispiele für Produktminiaturisierung

Damit diese neuen Produkte auch im Bereich der Konsumgüter (z.B. Funkarmbanduhr) an Bedeutung gewinnen und sich gegenüber den alten Produkten behaupten können, ist eine automatisierte Produktion notwendig. Die geforderten Montagegenauigkeiten in der Präzisions- und Mikromontage liegen heute bei 25μm (in naher Zukunft bei 5μm). In einem Montagesystem addieren sich die Einzeltoleranzen der Peripheriekomponenten und Bauteile zu einem resultierenden Positionsfehler (siehe Bild 4.1.2). Die geforderte hohe Montagegenauigkeit lässt sich daher nur durch Erfassung und Kom-

pensation der auftretenden Handhabungs- und Bauteiltoleranzen erreichen. Diese Aufgabe erfordert heute den Einsatz von teueren, zum Teil statischen Präzisionsrobotern und –werkzeugen in Kombination mit aufwendigen Messsystemen oder die Entwicklung spezialisierter, nur für ein Produkt anwendbarer Montagesysteme. Auch die manuelle Montage von Mikrosystemen mit Mikroskopen findet heute noch in der Produktion ihre Anwendung. Diese Tatsachen stehen der gewünschten preiswerten Mikromontage im Weg. Eine Lösung dieses Problems stellt die MPE als preiswertes, universelles, sensorgeführtes Montagewerkzeug für automatisierte Handhabungssysteme dar.

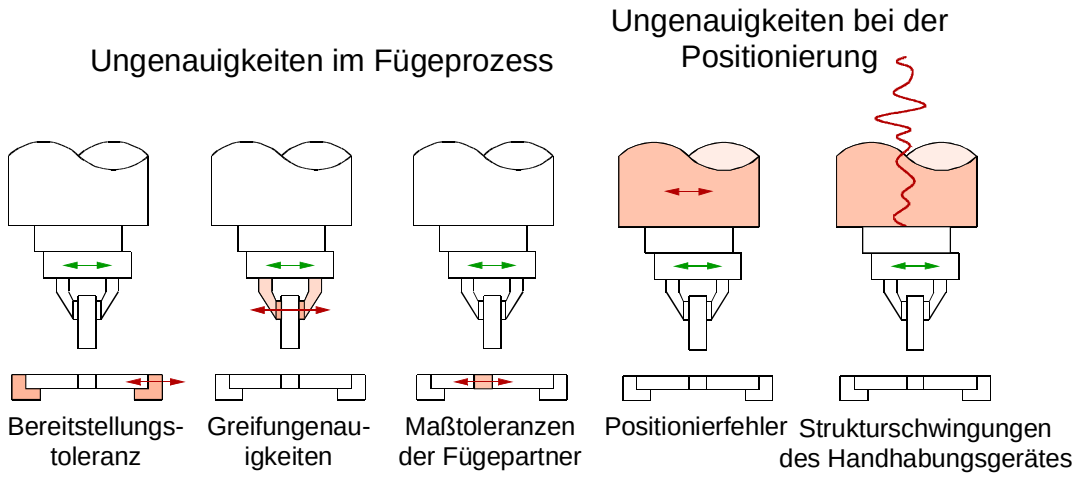

Bild 4.1.2 : Fehlereinflüsse bei der Präzisionsmontage

4.2 Systembeschreibung

4.2.1 Positionierstrategie

Die MPE basiert auf einem sogenannten Mikro-Makro-Konzept, bei dem ein Standardindustrieroboter (mit 3 oder 6 Freiheitsgraden) die Grobfügebewegungen übernimmt und die MPE die Ungenauigkeiten des Fügeprozesses und des Handhabungsgerätes mit ihrer Präzisionsaktorik kompensiert. Es werden somit die Vorteile des Industrieroboters – geringer Preis, hohe Flexibilität und großer Arbeitsraum – mit den Vorteilen der MPE – hohe Dynamik und hohe Genauigkeit – kombiniert. Die Nachteile beider Systeme, geringe Dynamik und unzureichende Genauigkeit (Wiederholgenauigkeit ca. 50μm) beim Industrieroboter sowie geringer Stellweg bei der MPE, kommen nicht zum tragen.

Bei konventionellen Positioniersystemen werden die auftretenden Ungenauigkeiten mittels Bildverarbeitung in mehreren Messschritten erfasst und bei der Positionierung durch eine Korrektur der Ablagekoordinaten berücksichtigt. Dieses Verfahren benötigt einen hohen Zeitbedarf zur Durchführung aller notwendigen Messschritte. Aufgrund der langen Verfahrwege, wie sie zur separaten Positionsmessung an verschiedenen Orten erforderlich sind, ist eine hohe Montagegenauigkeit nur durch eine sehr aufwendige und teure Präzisionsmechanik mit hoher Positioniergenauigkeit in einem

verhältnismäßig großen Arbeitsraum erreichbar. Die Montagegenauigkeit wird hierbei nach [46] durch die Positioniergenauigkeit des Handhabungsgeräts, Kalibrierfehler und die Auflösung der Sensorik begrenzt. Zudem führt das Auftreten selbst erregter Schwingungen aufgrund von Trägheitskräften dazu, dass sich der Positionierbewegung eine Fehlbewegung überlagert, die die Montagepräzision einschränkt. Zur Vermeidung schwingungsinduzierter relativer Bewegungen zwischen dem Greifer und dem Zielobjekt sind daher meist aufwendige konstruktive Maßnahmen und eine Begrenzung der maximalen Verfahrgeschwindigkeit nötig.

Das Prinzip der fehlerkompensierenden Montage hingegen erlaubt eine Reduktion der Anforderungen hinsichtlich Genauigkeit sowie statischer und dynamischer Steifigkeit des Handhabungsgerätes. Mittels einer optischen Sensorik wird direkt der Positionsversatz zwischen dem gegriffenen Bauteil und dem Fügeort für eine nachfolgende Lagekorrektur gemessen. Diese Strategie der relativen Positionsmessung bewirkt, dass nach [46] der Positionsversatz trotz Kalibrierfehler kompensiert werden kann. Die Montagegenauigkeit wird nur durch die Sensorauflösung begrenzt. Durch aktive Schwingungsbeeinflussung werden Strukturschwingungen des Handhabungsgerätes und Vibrationen aus der Umgebung unterdrückt. Die Störschwingungen werden direkt am Endeffektor durch ein in das Werkzeug integriertes Beschleunigungssensorsystem erfasst und durch eine dynamische Ausgleichsbewegung bis auf eine Restamplitude kompensiert. Die Sensorsignale der Positions- und Schwingungsmessung werden in einer Reglereinheit zu Stellgrößen für die Feinpositionierung und Schwingungskompensation durch eine integrierte Piezo basierte Aktorik verarbeitet.

Aus den ausführlichen Genauigkeitsmessungen an Industrierobotern in [98], [90] und aus montagetechnischen Gesichtspunkten [45] wurden die Anforderungen an die MPE abgeleitet:

- Zur Kompensation statischer, dynamischer und temperaturabhängiger Positionierfehler des Handhabungsgerätes ist ein Mindeststellbereich der Aktorik von $\pm 200\mu$m erforderlich. Toleranzen von Montageperipherieeinrichtungen, die außerhalb dieses Stellbereiches liegen, können durch eine Nachführung des Roboters kompensiert werden.
- Durch eine dynamische Ausgleichsbewegung der Aktorik sollen die auf etwa 70Hz bandbegrenzten Strukturschwingungen des Handhabungsgerätes und Vibrationen der Umgebung kompensiert werden.
- Die bei Präzisions- und Mikromontageaufgaben erforderliche Positioniergenauigkeit liegt unterhalb 25 μm. Die Summe der bei der Positionsmessung und Feinpositionierung auftretenden Fehler darf diesen Wert nicht überschreiten. Bei vergleichbarer Genauigkeit soll ein Kostenvorteil gegenüber einem Präzisionshandhabungssystem erzielt werden.
- Die Mikropositioniereinrichtung soll hinsichtlich Masse und Baugröße minimiert werden, da sie vom Handhabungsgerät mitgeführt wird. Eine große Bauweise und das zusätzliche Handhabungsgewicht schränken den Einsatz des Werkzeuges bei einer beengten Zugänglichkeit des Fügeortes und die Genauigkeit der Feinpositionierung ein und reduzieren die für die Schwingungskompensation erreichbare Dynamik.

- Durch die Verwendung verschiedener Greifer soll eine flexible Anpassung der Mikropositioniereinrichtung an unterschiedliche Handhabungs- und Montageaufgaben möglich sein.
- Durch eine autonome Steuerung soll ein flexibler Einsatz der Mikropositioniereinrichtung in automatisierten Handhabungssystemen (z. B. Roboter, Achssystem), unabhängig von der Ablauf-/Maschinensteuerung des Handhabungsgerätes, ermöglicht werden.

4.2.2 Aufbau der Aktorik

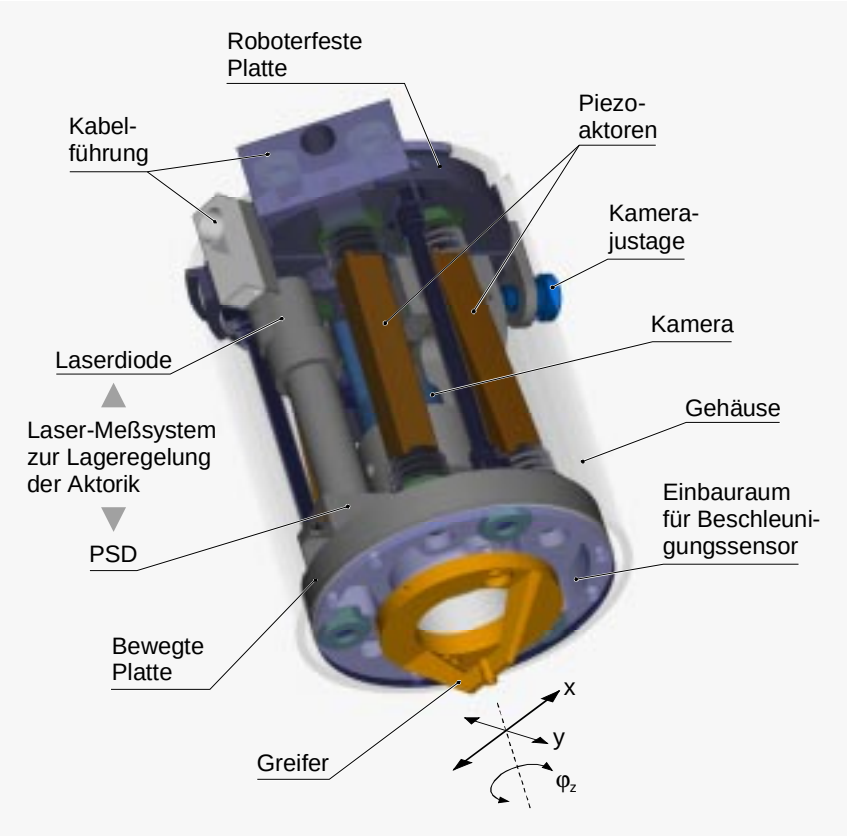

Bild 4.2.1 : Aufbau des Aktors

Das im Teilprojekt 5 realisierte Montagewerkzeug ist in Bild 4.2.1 zu sehen. Die Korrekturbewegung wird mit 4 Piezoaktoren, die so angeordnet sind, dass eine Translation der unteren Plattform in x- und y-Richtung sowie eine Rotation um die z-Achse möglich ist, durchgeführt. Weitere Details zur Entwicklung der Aktorik finden sich in [44], [87] und [90]. Zur Lageregelung der hysteresebehafteten, nichtlinearen Piezoaktoren ist ein Lasermesssystem, bestehend aus Laserdiode und positionsempfindlicher Diode (PSD), integriert. Die CCD-Kamera mit Objektiv zur Messung des Positionsversatzes ist mittig in das Werkzeug eingebaut und hat durch eine Öffnung in der unteren Plattform einen Blick auf den Greifer und das Fügeteil. Mit der Justagevorrichtung kann die Kamera auch außermittig (bei größeren Fügeteilen) befestigt werden. Die untere Plattform nimmt das 3D-Beschleunigungssensorsystem zur Messung der strukturellen Schwingungen des Handhabungsgerätes und Störschwingungen der Umgebung

auf, auch wird der anwendungsspezifische Greifer daran befestigt. Die obere Plattform wird an das Handhabungsgerät angeflanscht.

4.3 Systemidentifikation

4.3.1 Sensormodelle

4.3.1.1 Lasersensor

Das integrierte Messsystem zur Lageregelung kann aufgrund von EMV-Problemen bis jetzt nicht eingesetzt werden. Es wird zum Entwurf der Regelung deshalb durch externe Lasertriangulationssensoren vergleichbarer Auflösung und Dynamik ersetzt.

Die Lasertriangulationssensoren sind analoge Sensoren mit einer Empfindlichkeit von $10^{V}/_{mm}$ und einem Messbereich von $\pm 1\text{mm}$ bei einer Auflösung von $2\mu\text{m}$. Da die Sensoren an den Regelungsrechner mit einem 16 Bit A/D-Wandler angebunden werden, der mit der Basisabtastrate betrieben wird, eignet sich zur Simulation das kontinuierlich zyklische Sensormodell (3.1.1). Sie können Signale bis zu 10kHz verarbeiten, dadurch ist eine Berücksichtigung des dynamischen Übertragungsverhaltens im Sensormodell nicht notwendig und die Übertragungsfunktion kann zu „1“ gesetzt werden.

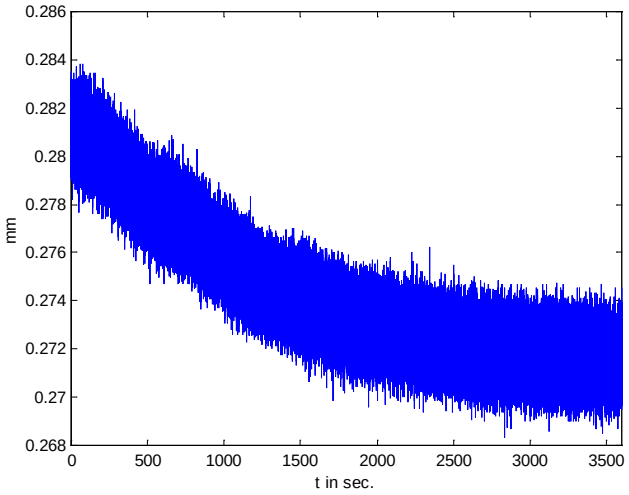

Bild 4.3.1 : Drift des Lasersensors

Die Drift des Sensors kann mit einer Langzeitmessung (z. B. 1 Stunde) bei nicht angeregtem Sensor ermittelt werden. Die Parameter der Gleichung (3.2) können mit den Messergebnissen grafisch ermittelt werden. Für den Sensor der x-Achse ergibt sich aus Bild 4.3.1 die Zeitkonstante T zu 1500s (Schnittpunkt der Tangente bei t=0s mit der Zeitachse) und die Driftamplitude $C1$ zu $0.281\text{mm}-0.2717\text{mm}=0.0093\text{mm}$. Den Mittelwert $\bar{o}$ bestimmt man nach abgeklungener Drift mit einer Messung bei nicht angeregtem Sensor zu 0.266mm. Die Varianz des offsetfreien Messrauschens kann aus der gleichen Messung mit dem MatLab-Befehl var zu $\sigma=6.8473\text{e-}08\text{mm}^2$ bestimmt werden. Zur Be-

stimmung der Totzeit wird der Aktor von der Regelungseinheit mit einem Führungssprung angeregt. Die Zeitdifferenz zwischen der Anregung und der ersten messbaren Antwort des Aktors ergibt die Totzeit, in der die komplette Verzögerung der Strecke (D/A-Wandlungszeit, Verstärkertotzeit und A/D-Wandlungszeit) mitberücksichtigt wird. Sie lässt sich bei einer Abtastfrequenz von 2000Hz zu zwei Abtastschritte bestimmen. Die Identifikation des Lasersensors für die y-Achse erfolgt analog, es ergeben sich dabei folgende Werte: T=800s, C1=0.009mm, $\bar{o}$ =0.0967mm, σ=1.8660e-08mm², Totzeit von einem Abtastschritt.

4.3.1.2 Beschleunigungssensor

Das verwendete 3D-Beschleunigungssensorsystem hat „on-board" pro Freiheitsgrad einen 1-Bit-Sigma-Delta-Wandler zur Digitalisierung der analogen Signale. Die pro Sensorelement seriell mit einer Frequenz von 500kHz anfallenden digitalen Signale werden zu einem DSP-Prozessinterface-Board [3] übertragen. Aufgrund der digitalisierten Signale ist hierbei die Störanfälligkeit bezüglich EMV gegenüber analogen Signalen deutlich reduziert. Auf dem Board erfolgt die nötige Dezimierung der Sensordaten um den Faktor 256, wobei sich die Bitbreite auf 24 Bit erhöht und einer Auflösung von $2\frac{\mu g}{\sqrt{Hz}}$ entspricht. Die Sensordaten liegen somit mit einer Abtastrate von $\frac{500kHz}{256} = 1953.125Hz \neq 2000Hz$ bereit. Dadurch muss dieser Sensor mit dem diskreten zyklischen Sensormodell (3.1.2) simuliert werden. Das Übertragungsverhalten der Sensorelemente wird in [115] mit einem PT_2-System mit der Eigenfrequenz von 700Hz und der Dämpfung von 0.7 spezifiziert. Die zeitdiskrete Übertragungsfunktion ergibt sich zu

$$Tf(z) = \frac{0.3298z^2 + 0.6596z + 0.398}{z^2 + 0.1393z + 0.1799}. \qquad \textbf{(4.1)}$$

Aufgrund des temperaturkompensierten Designs der Sensorelemente konnte keine Drift beobachtet werden. Der Messbereich beträgt ±6g. Die Empfindlichkeit kann bei kapazitiven Beschleunigungssensoren durch Ausnutzung der Erdbeschleunigung bestimmt werden. Dabei werden pro Sensorelement 2 Messungen durchgeführt. Bei der ersten Messung wird das Sensorelement mit der sensitiven Achse in Richtung der Erdbeschleunigung und bei der zweiten Messung entgegengesetzt der Erdbeschleunigung ausgerichtet. Die Empfindlichkeit ergibt sich damit zu

$$empf = \frac{mess_1 - mess_2}{2 \cdot g} = \frac{mess_1 - mess_2}{2 \cdot 9.81\frac{m}{s^2}}. \qquad \textbf{(4.2)}$$

Die Ergebnisse der Kalibriermessungen sind in Tabelle 4.3.1 wiedergegeben. Der Offset und die Varianz des Messrauschens werden wie beim Lasersensor bestimmt. Es ergeben sich für die drei Sensorelemente folgende Werte: $\bar{o}_x = -30113\frac{mm}{s^2}$, $\sigma_x = 1027.2\frac{mm^2}{s^4}$, $\bar{o}_y = 32354\frac{mm}{s^2}$, $\sigma_y = 3523.5\frac{mm^2}{s^4}$, $\bar{o}_z = 43787\frac{mm}{s^2}$ und

$\sigma_z = 188.363 \frac{mm^2}{s^4}$. Die Totzeit wird analog zum Lasersensor bestimmt, es ergibt sich eine Totzeit von 2 Abtastschritten, die hauptsächlich, wie die Abschätzung in [3] zeigt, durch die Bereitstellung der Sensorwerte im DSP zustande kommt.

Gravitation	+x	-x	+y	-y	+z	-z	*empf*
x-Element	2.5983^{E}6	5.1153^{E}6	3.8494^{E}6	3.8465^{E}6	3.8628^{E}6	3.8366^{E}6	128.28
y-Element	4.3391^{E}6	4.3609^{E}6	5.6607^{E}6	3.0244^{E}6	4.3480^{E}6	4.3420^{E}6	134.36
z-Element	4.6613^{E}6	4.5982^{E}6	4.6404^{E}6	4.6393^{E}6	5.9638^{E}6	3.2910^{E}6	136.22

Tabelle 4.3.1 : Kalibriermessung des Beschleunigungssensorsystems

4.3.1.3 CCD-Kamera

Die CCD-Kamera zur Bestimmung des Positionsversatzes könnte mit dem diskreten asynchronen Sensormodell (3.1.3) simuliert werden. Da aber durch die geforderte hohe Messauflösung im Bereich von einigen Mikrometern und einem Messbereich von ca. 500μm x 500μm mit einer einfachen Optik nur eine geringe Tiefenschärfe erreicht werden kann, stehen aufgrund der Verarbeitungszeit der Bildverarbeitung nur 1-2 Bilder zur Verfügung. In der Reglerstruktur (siehe 4.5) werden diese Messwerte deshalb nur als Führungsgrößen verwendet. Damit spielt für den Regelkreis das Zeitverhalten dieses Sensors keine Rolle, so dass dieses Sensormodell weggelassen werden kann. Will man der Vollständigkeit halber das Zeitverhalten trotzdem simulieren, so muss die anwendungsabhängige Bildverarbeitung bezüglich der Ausführungszeit und der davon abhängigen Genauigkeit statistisch näher untersucht werden. Für die im Teilprojekt 5 angestrebte Anwendung der Montage eines Zahnrads in das Uhrwerk einer Funkarmbanduhr wurde eine auf Template-Matching basierende Bildverarbeitung [10] entwickelt. Das Zeitverhalten dieses Verfahrens kann mit dem diskreten asynchronen Sensormodell gut nachgebildet werden, da bei diesem Verfahren die Ausführungszeit vom Bildinhalt und die Genauigkeit von der Ausführungszeit abhängt.

4.3.2 Streckenmodelle

Im Folgenden werden die einzelnen Komponenten der Regelstrecke sowie deren Verkopplung vorgestellt. Da die Positionskorrektur in zwei kartesischen Freiheitsgraden erfolgt, werden die einzelnen Streckenglieder ebenfalls in diesen zwei Freiheitsgraden beschrieben. Durch eine ausreichend genaue hard- bzw. softwaremäßige Kalibrierung [45] kann nahezu eine Entkopplung der beiden Freiheitsgrade, im folgenden x- bzw. y-Achse genannt, erreicht werden, so dass in den folgenden Unterpunkten immer nur ein Freiheitsgrad betrachtet wird. Die Beschreibung des zweiten Freiheitsgrades erfolgt dann analog. Eine Verdrehung der unteren Plattform, die Bauart bedingt im Bereich von $\pm 0.8°$ möglich wäre, wird nicht betrachtet, da es mit den verwendeten Sensoren nicht möglich ist, solch kleine Rotationen zu messen.

4.3.2.1 Piezoaktorik

Die Beschreibung eines nichtlinearen, hysteresebehafteten Systems gestaltet sich ziemlich aufwendig und wäre nach [35] und [76] prinzipiell möglich, lässt sich aber in der

geforderten Genauigkeit nicht realisieren. Da jedes mechanische System näherungsweise als PT_2-System angesehen werden kann, dient dies als Grundlage der Identifikation der Aktorik. Damit eine lineare Beschreibung dieses nichtlinearen hysteresebehafteten Systems gerechtfertigt ist, bedarf es eines gesonderten Regelkreises, der die geforderte Linearität und Zeitinvarianz (LTI-modell: linear timeinvariant model) herstellt (siehe 4.5.1).

Es soll nun der Aktor in einem Ersatzmodell, bestehend aus zwei Plattformen, die mittels einer Feder und eines Dämpfers miteinander verbunden sind, betrachtet werden. Die untere Plattform kann gezielt mit den Piezoaktoren proportional der angelegten Spannung angeregt werden. Die mechanischen Eigenschaften der vorgespannten Piezostacks, der Parallelführung und der Festkörpergelenke können näherungsweise einer Feder und einem Dämpfer gleichgesetzt werden.

Aus diesem Ersatzschaltbild der Aktorik kann somit die folgende lineare Differentialgleichung 2. Ordnung abgeleitet werden.

$$m \cdot \ddot{x} + k \cdot \dot{x} + c \cdot x = b \cdot u \qquad \textbf{(4.3)}$$

hierbei entsprechen

m:	effektive Masse des Aktors
k:	effektive Dämpfungskonstante des Aktors
c:	effektive Federkonstante des Aktors
b:	Einkoppelfaktor der Stellgröße
x:	Bewegung der beweglichen Aktorplattform
u:	Stellgröße

Die Übertragungsfunktion der Stellgröße *u* bezogen auf die Position der unteren Plattform *x* ist dementsprechend ein PT_2-Glied

$$\frac{x}{u} = \frac{b/m}{s^2 + k/m \cdot s + c/m} \quad . \qquad \textbf{(4.4)}$$

Ein PT_2-System kann auch durch seine Eigenfrequenz ω_0 und Dämpfung *D* wie folgt beschrieben werden:

$$\frac{x}{u} = \frac{p}{s^2 + 2 \cdot D \cdot \omega_0 \cdot s + {\omega_0}^2} \qquad \textbf{(4.5)}$$

oder in Zustandsdarstellung mit der Position der unteren Plattform bezogen auf die obere Plattform $y_{x,Aktor}$ und der daraus resultierenden Beschleunigung $y_{a,Aktor}$ der unteren Plattform als Messgrößen

$$\dot{x}_{Aktor} = \underbrace{\begin{bmatrix} 0 & 1 \\ (2\pi \cdot f_0)^2 & 4\pi \cdot D \cdot f_0 \end{bmatrix}}_{A_{Aktor}} \cdot x_{Aktor} + \underbrace{\begin{bmatrix} 0 \\ b \end{bmatrix}}_{B_{Aktor}} \cdot u$$

$$y_{Aktor} = \begin{bmatrix} y_{x,Aktor} \\ y_{a,Aktor} \end{bmatrix} = \begin{bmatrix} \overbrace{\begin{matrix} 1 & 0 \end{matrix}}^{C^T_{x,Aktor}} \\ \underbrace{\begin{matrix} (2\pi \cdot f_0)^2 & 4\pi \cdot D \cdot f_0 \end{matrix}}_{C^T_{a,Aktor}} \end{bmatrix} \cdot x_{Aktor} + \begin{bmatrix} 0 \\ \underbrace{b}_{D_{a,Aktor}} \end{bmatrix} \cdot u \qquad \textbf{(4.6)}$$

Zur Identifikation des linearen Anteils des Aktors ist somit nur die Bestimmung dieser beiden Kenngrößen und der Konstanten *p* notwendig. Diese können experimentell leicht durch die Anregung des Aktors mit einem Sprung bestimmt werden. In Bild 4.3.2 links ist die gemessene Sprungantwort und die Antwort des PT2-Modells für einen Führungssprung von 30 μm zu sehen.

Für den vorliegenden Aktor ergeben sich daraus folgende Werte:

f_{0x}=134.90Hz, D_x=0.03, f_{0y}=121.0Hz, D_y=0.04.

Mit der gleichen Vorgehensweise werden die Parameter für das Aktormodell der mit dem Dämpfungsregler (siehe 4.5.1) linearisierten und gedämpften Piezoaktorik ermittelt. In Bild 4.3.2 rechts ist die reale und die simulierte Sprungantwort der gedämpften x-Achse dargestellt. Es ergeben sich für den gedämpften Aktor folgende Parameterwerte:

$f_{0xd}=f_{0x}$, D_{xd}=2.5, $f_{0yd}=f_{0y}$, D_{yd}=2.5.

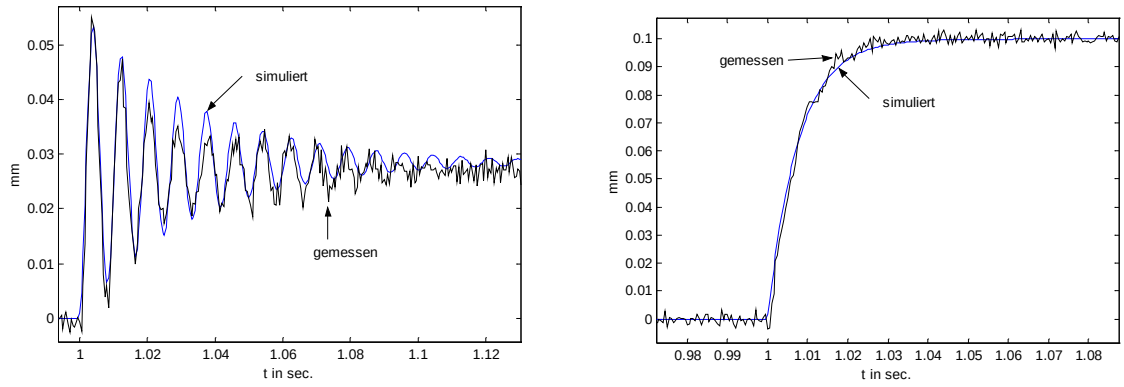

Bild 4.3.2 : Sprungantwort des Aktors (links ungedämpft, rechts gedämpft)

4.3.2.2 Störeinkopplung

Die strukturellen Schwingungen des Handhabungsgerätes sowie die Vibrationen aus der Umgebung werden über das Handhabungsgerät eingekoppelt. Es muss deshalb ebenfalls modelliert werden. Da jedes mechanische System in erster Näherung mit einem PT_2-Glied angenähert werden kann und die exakte Modellierung des Handhabungsgerätes nur mit großem Aufwand zu realisieren ist, wird ein PT_2-System für das Modell des Handhabungsgerätes angenommen. Es gelten dabei die selben Gleichungen wie bei der Modellierung der Piezoaktorik. Die beiden Parameter (Eigenfrequenz und Dämpfung) können über Messungen der Impuls- oder Sprungantwort bestimmt

werden. Bei dem betrachteten 3-Achs-Scara-Roboter ergab sich die Eigenfrequenz zu 16Hz und die Dämpfung zu 0.3.

Bei der Entwicklung des Regelsystems und bei der Hardware-in-the-Loop-Simulation wird die MPE auf einem Tisch montiert. Die Störgrößen werden in diesem Fall über dem Tisch eingekoppelt, so dass das Robotermodell durch das des Tisches ersetzt werden muss. Der Tisch wird wieder als PT_2-System angesehen und durch Messung der Impulsantwort ergeben sich für die Parameter folgende Werte:

f_{0x}=5.8Hz, D_x=0.03, f_{0y}=6.66Hz, D_y=0.1.

Die Zustandsgleichungen zweiter Ordnung ergeben sich analog zu Gleichung (4.6) mit der Störgröße w als Eingangsgröße zu

$$\begin{aligned}\dot{x}_{Rob} &= \underbrace{\begin{bmatrix} 0 & 1 \\ (2\pi\cdot f_0)^2 & 4\pi\cdot D\cdot f_0 \end{bmatrix}}_{A_{Rob}}\cdot x_{Rob} + \underbrace{\begin{bmatrix} 0 \\ q \end{bmatrix}}_{B_{Rob}}\cdot w \\ y_{Rob} &= \begin{matrix} y_{x,Rob} \\ y_{a,Rob} \end{matrix}\Bigg] = \left[\begin{matrix} \overbrace{\begin{matrix} 1 & 0 \end{matrix}}^{C^T_{x,Rob}} \\ \underbrace{\begin{matrix} (2\pi\cdot f_0)^2 & 4\pi\cdot D\cdot f_0 \end{matrix}}_{C^T_{a,Rob}} \end{matrix}\right]\cdot x_{Rob} + \left[\begin{matrix} 0 \\ \underbrace{q}_{D_{a,Rob}} \end{matrix}\right]\cdot w\end{aligned} \qquad \textbf{(4.7)}$$

Als Messgrößen werden wieder die Position $y_{x,Rob}$ und die Beschleunigung $y_{a,Rob}$ angenommen.

4.3.2.3 Piezoverstärker

Da Piezo-Stacks sich in erster Näherung wie Kapazitäten verhalten, kann das elektrische Ersatzschaltbild in Bild 4.3.3 angegeben werden. Daraus lässt sich folgende Differentialgleichung erster Ordnung ableiten.

$$\dot{U}_C = \frac{1}{R\cdot C}(U_0 - U_C) \qquad \textbf{(4.8)}$$

oder in der Zustandsform

$$\begin{aligned}\dot{x}_{PV} &= \underbrace{-\frac{1}{R\cdot C}}_{A_{PV}}\cdot x_{PV} + \underbrace{\frac{k}{R\cdot C}}_{B_{PV}}\cdot u_{in} \\ y_{PV} &= \underbrace{1}_{C^T_{PV}}\cdot x_{PV}\end{aligned} \qquad \textbf{(4.9)}$$

Für den verwendeten Piezo-Verstärker PCS 8.A3 von Marco S&E GmbH ergibt sich RC zu 3Ω*20μF. Der Piezo-Verstärker mit dem Verstärkungsfaktor k=20 wird direkt mit den D/A-Wandlersignalen beaufschlagt, was einer Folge von Sprunganregungen mit der Abtastfrequenz gleichkommt. Das PT_1-System trägt somit mit seinem Tiefpasscharakter einen Teil (350μs) zur Totzeit bei. Beim Regelungsentwurf A muss der Verstärker mitmodelliert werden, beim Regelungsentwurf B kann jedoch der Verstärker vernachlässigt werden, da die Totzeit in diesem Fall einfach den Sensoren angerechnet wird und somit kompensiert werden kann. Da die Grenzfrequenz des Tiefpasses bei ca.

1.6kHz liegt, kann im Hinblick auf die Systemordnung die Übertragungsfunktion vernachlässigt werden.

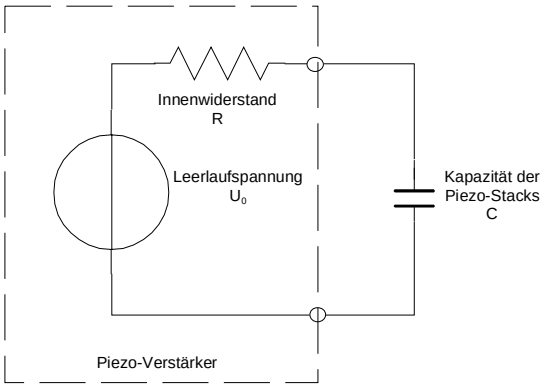

Bild 4.3.3 : Ersatzschaltbild des Piezoverstärkers

4.3.2.4 Gesamte Strecke

Für den H_2-Reglerentwurf (Regelungsentwurf A) wurde in [30] die Regelstrecke, wie in Bild 4.3.4 zu sehen, modelliert. Die Übertragungsfunktionen der Sensoren wurden dabei aufgrund des bandbegrenzten Störspektrums nicht berücksichtigt. Die Stellgröße wird über den Verstärker und die Störgröße über das Handhabungsgerät in die MPE eingekoppelt. Als Messgrößen stehen die Position der unteren Plattform bezogen auf die obere und die absolute Beschleunigung – die Summe aus den eingekoppelten Störbeschleunigungen und der Beschleunigung der oberen Plattform – zur Verfügung.

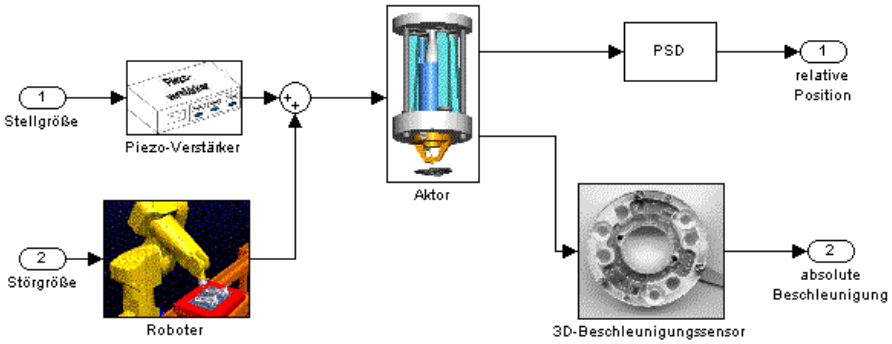

Bild 4.3.4 : Regelstrecke des ursprünglichen Reglerentwurfs

Die Zustandsgleichungen fünfter Ordnung lassen sich mit (4.6),(4.7) und (4.9) zu

$$\dot{x} = \begin{bmatrix} \dot{x}_{Aktor} \\ \dot{x}_{Rob} \\ \dot{x}_{PV} \end{bmatrix} = \begin{bmatrix} A_{Aktor} & B_{Aktor}C_{Rob}^T & B_{Aktor}C_{PV}^T \\ 0 & A_{Rob} & 0 \\ 0 & 0 & A_{PV} \end{bmatrix} \cdot x + \begin{bmatrix} 0 \\ 0 \\ B_{PV} \end{bmatrix} \cdot u + \begin{bmatrix} 0 \\ B_{Rob} \\ 0 \end{bmatrix} \cdot w$$

$$y = \begin{bmatrix} y_{x,Aktor} \\ y_{a,Aktor} + y_{a,Rob} \end{bmatrix} = \begin{bmatrix} C_{x,Aktor}^T & 0 & 0 \\ C_{a,Aktor}^T & C_{a,Rob}^T & D_{a,Aktor}C_{PV}^T 0 \end{bmatrix} \cdot x + \begin{bmatrix} 0 \\ D_{a,Rob} \end{bmatrix} \cdot w \qquad \textbf{(4.10)}$$

ableiten.

Das Streckenmodell, das dem Regelungsentwurf B zugrunde liegt, berücksichtigt im Gegensatz zum vorhergehenden Modell die feste Kopplung zwischen MPE und Handhabungsgerät. Die Sensoren werden ebenfalls nicht mit in das Streckenmodell aufgenommen. Wie in 4.3.2.3 beschrieben kann bei dieser Entwurfsmethode der Piezoverstärker weggelassen werden. Die Messgrößen sind wieder die Position der unteren Plattform bezogen auf die obere und die absolute Beschleunigung. Die Zustandsgleichungen vierter Ordnung ergeben sich aus den Gleichungen (4.6) und (4.7) zu

$$\begin{aligned} \dot{x} &= \begin{bmatrix} \dot{x}_{Aktor} \\ \dot{x}_{Rob} \end{bmatrix} = \begin{bmatrix} A_{Aktor} & 0 \\ 0 & A_{Rob} \end{bmatrix} \cdot x + \begin{bmatrix} B_{Aktor} & 0 \\ 0 & B_{Rob} \end{bmatrix} \cdot u + \begin{bmatrix} 0 \\ B_{Rob} \end{bmatrix} \cdot w \\ y &= \begin{bmatrix} y_{x,Aktor} \\ y_{a,Aktor} + y_{a,Rob} \end{bmatrix} = \begin{bmatrix} C_{x,Aktor}^T & 0 \\ C_{a,Aktor}^T & C_{a,Rob}^T \end{bmatrix} \cdot x + \begin{bmatrix} 0 \\ D_{a,Aktor} \end{bmatrix} \cdot u + \begin{bmatrix} 0 \\ D_{a,Rob} \end{bmatrix} \cdot v \end{aligned} \qquad \textbf{(4.11)}$$

4.4 Sensorkopplung

4.4.1 Regelungsentwurf A

Bei dem in [92] vorgestellten Regelungsentwurf wird keine Sensorkopplung im engeren Sinne durchgeführt. Die Daten des Beschleunigungssensorsystems werden nicht synchronisiert, d.h. die mit einer Abtastrate von 1953.125Hz eintreffenden Daten werden im Regelkreis mit der Basisabtastrate von 2000Hz weiterverarbeitet. Dies bewirkt im worst case nach Gleichung (3.42) eine zusätzliche Totzeit von 500μs und nach Gleichung (3.43) einen maximalen Fehler von ca. 2000 mm/s^2, wenn man das Amplituden-Störspektrum aus [45] zu $w(t) = \frac{1}{f}\sin 2 \cdot \pi \cdot f \cdot t,\ f \in]0,50Hz]$ annähert. Die Totzeiten der Sensoren werden nicht kompensiert. Die Beschleunigungsdaten werden einer Offsetkorrektur mit dem gefensterten Mittelwert nach 3.2.1.2 unterzogen. Zur Bestimmung der Stellgröße werden aufgrund der fehlenden Sensorkopplung nur die Daten des Beschleunigungssensorsystems herangezogen. Die dadurch notwendige zweifache Integration wird implizit im H_2-Regler durchgeführt, was dazu führt, dass das Ausgangssignal einer dreifachen Offsetkorrektur mit driftendem Offset unterworfen werden muss. Der Regler kann somit nur bis zum Zahlenüberlauf der Offsetkorrektur auf dem Regelungsrechner betrieben werden und muss dann wieder neu initialisiert werden.

4.4.2 Regelungsentwurf B

Bei diesem Entwurf wird eine Sensorkopplung mit den Methoden des Kapitels 3 durchgeführt. Die Lasersensoren müssen nicht synchronisiert werden, da sie synchron mit der Basisabtastrate betrieben werden. Es ist aber eine Kompensation der Totzeit, hier wird auch die Totzeit der A/D-Wandler, D/A-Wandler und des Piezoverstärkers miteingerechnet, notwendig. Die wird aufgrund des Rauschverhaltens mit einem FOH-Glied durchgeführt. Die hohe Varianz des Messrauschens der Beschleunigungssensoren, darunter fallen auch reelle hochfrequente Störschwingungen mit niedriger Amplitude,

bedingt zur Synchronisation und Totzeitkompensation ein rauschunempfindliches Verfahren. Nach Bild 3.3.25 kommen das FOH-Glied bei den Daten basierten Verfahren oder der Kalman-Filter bei den Modell basierten Verfahren in Betracht. Im Hinblick auf die Hardwareanbindung der Beschleunigungssensorik mittels Interrupt fällt die Wahl auf das FOH-Glied, da es weniger Rechenaufwand, wie in einer Interruptservice-Routine auch gefordert wird, benötigt.

Die Sensorfusion erfolgt mit einem zeitinvarianten Kalman-Filter, dem das Streckenmodell von Gleichung (4.11) zugrunde liegt. Als Eingangsgrößen werden die Stellgröße u, die absolute Beschleunigung y_a und die Position der unteren Plattform bezogen auf die obere $y_{x,Aktor}$ verwendet. Durch die Eigenschaft des Kalman-Filters als Zustandsbeobachter, erhält man Schätzwerte für die vier Systemzustände des Modells, die mit den Eingangsgrößen des Kalman-Filters abgeglichen werden. Mit den geschätzten Zuständen des Handhabungsgerätes bzw. des Tisches $\hat{x}_{Rob}$ kann ein Schätzwert der unbekannten Störgröße $\hat{w}$ berechnet werden. Die Zustandsgleichungen der Störgrößenschätzung mit Kalman-Filter in zeitdiskreter Darstellung ergeben sich zu

$$
\begin{aligned}
x_{Kalman}(k+1) &= A_{Kalman} \cdot x_{Kalman}(k) + B_{Kalman} \cdot \begin{bmatrix} u \\ y_a \\ y_{x,Aktor} \end{bmatrix} \\
\begin{bmatrix} \hat{y}_a \\ \hat{y}_{x,Aktor} \\ \hat{x}_{Aktor} \\ \hat{x}_{Rob} \end{bmatrix}(k) &= C_{Kalman} \cdot x_{Kalman}(k) + D_{Kalman} \cdot \begin{bmatrix} u \\ y_a \\ y_{x,Aktor} \end{bmatrix} \\
\hat{y}_{x,Rob}(k) &= C_{a,Rob}^T \cdot \hat{x}_{Rob}(k)
\end{aligned}
\qquad \textbf{(4.12)}
$$

mit

$$
\begin{aligned}
A_{Kalman} &= \begin{bmatrix} A_{Aktor} & 0 \\ 0 & A_{Rob} \end{bmatrix} - H \cdot \begin{bmatrix} C_{x,Aktor}^T & 0 \\ C_{a,Aktor}^T & C_{a,Rob}^T \end{bmatrix} \\
B_{Kalman} &= \begin{bmatrix} \begin{bmatrix} B_{Aktor} & 0 \\ 0 & B_{Rob} \end{bmatrix} - H \cdot \begin{bmatrix} 0 & 0 \\ D_{a,Aktor} & D_{a,Rob} \end{bmatrix} & 0 \\ 0 & H \end{bmatrix}
\end{aligned}
\qquad \textbf{(4.13)}
$$

Der Kalman-Filter Rückführungskoeffizient H wird mit Gleichung (3.75) bestimmt. Unter Angabe des Systemmodells und der Kovarianzmatrizen, die sich bei Messungen des Systems in Ruhe mit der MatLab-Funktion *cov* bestimmen lassen, kann man sich den Kalman-Filter einfach mit der MatLab-Funktion *kalman* berechnen lassen. Für die MPE ergeben sich für den Kalman-Filter folgende stabile Polstellen:

x-Achse:	-0.0076	y-Achse:	0.0571
	0.9153		0.9252
	0.9982 + 0.0180i		0.9982 + 0.0180i
	0.9982 - 0.0180i		0.9982 - 0.0180i

Man sieht an der Lage der Pole, dass keine implizite doppelte Integration durchgeführt wird, dadurch ist der Kalman-Filter auch bei Offset behafteten Eingangssignalen stabil und schätzt die Zustände mittelwertsfrei. Eine Offsetkorrektur der Beschleunigungsdaten mit den Methoden von 3.2.1 ist nicht notwendig, es erfolgt nur eine einfache Kalibrierung (Subtraktion einer Konstanten) um den Einschwingvorgang des Kalman-Filters zu verkürzen. Weitere Vorverarbeitungsschritte der Sensordaten sind nicht notwendig.

4.5 Reglerstruktur

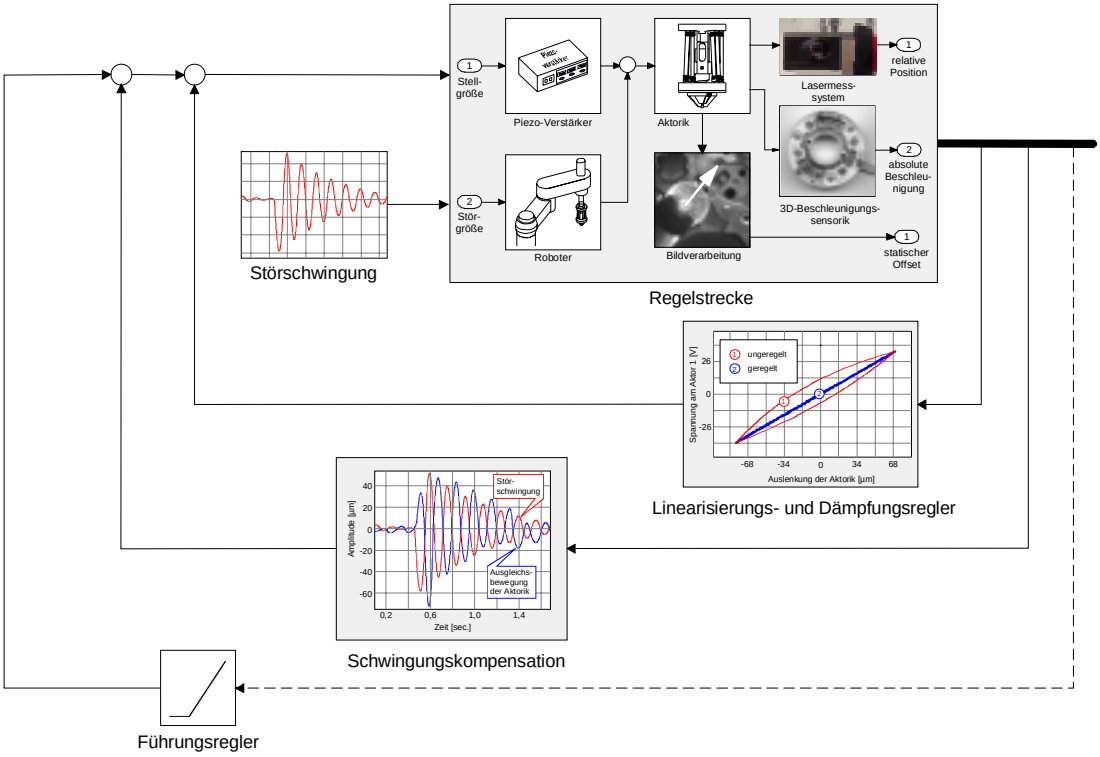

Bild 4.5.1 : Reglerstruktur

In Bild 4.5.1 ist die allgemeine Reglerstruktur der MPE ohne die Komponenten zur Sensorkopplung dargestellt. Die Piezoaktorik weist ein nichtlineares, Hysterese behaftetes Verhalten auf und zudem, wie die Systemidentifikation unter 4.3.2.1 zeigt, stellt sie näherungsweise ein sehr schwach gedämpftes PT_2-System dar. Mit Hilfe des Linearisierungs- und Dämpfungsreglers im inneren Regelkreis kann man eine lineare Aktorbewegung mit einstellbarem Dämpfungsverhalten erreichen [30]. Dies ist die Voraussetzung für alle weiteren Reglerentwürfe, da nur lineare Modelle in die Berechnung eingehen und die Problematik der nichtlinearen Regelung dadurch entfällt.

Die Aufgabe des Reglers zur Schwingungskompensation ist die Dämpfung der strukturellen Schwingungen des Handhabungsgerätes sowie der Vibrationen, die den Endeffektor beeinflussen. Dies geschieht durch eine der Störung entgegengesetzte Korrekturbewegung der oberen Plattform der MPE.

Durch die beiden ersten Regelkreise wird erreicht, dass die Relativbewegung des Endeffektors der MPE mit dem Fügeteil gegenüber dem Basisteil minimiert wird. Der

statische Versatz zwischen Fügeort und Fügepartner, der mit der Bildverarbeitung aus den Bildern der CCD-Kamera ermittelt wird, muss durch den Führungsregler als neue Führungsgröße bereitgestellt werden. Aufgrund der geringen Tiefenschärfe der CCD-Kamera und der großen Verarbeitungszeit der Bildverarbeitung im 1-Sekundenbereich werden während des Fügevorgangs nur 1-2 Bilder verarbeitet werden können. So wird die Einspeisung der Ergebnisse der Bildverarbeitung eher als Steuerung betrachtet.

4.5.1 Dämpfungs- und Linearisierungsregler

Für diesen Regler steht als einzig mögliche Eingangsgröße die Auslenkung der unteren Plattform bezogen auf die obere, die mit der Geschwindigkeit die Zustandsvariablen bilden, zur Verfügung. Die Ausgangsgröße ist, wie bei allen anderen Reglern auch, die Soll-Auslenkung der unteren Plattform.

Gegeben ist die Strecke $G_1(s)$ mit der Übertragungsfunktion

$$G_1(s) = \frac{k_1}{s^2 + 2 \cdot d_1 \cdot \omega_{01} \cdot s + \omega_{01}{}^2} \qquad \textbf{(4.14)}$$

wobei d_1 der Dämpfung, ω_{01} der Resonanzfrequenz und k_1 dem Proportionalbeiwert entsprechen. Ziel ist es, mit Hilfe einer einfachen Regelkreisstruktur eine Übertragungsfunktion $G_2(s)$ zu realisieren, die das gewünschte PT_2-Verhalten hat. Es gilt somit

$$G_2(s) = \frac{k_2}{s^2 + 2 \cdot d_2 \cdot \omega_{02} \cdot s + \omega_{02}{}^2} \qquad \textbf{(4.15)}$$

wobei d_2 der Dämpfung, ω_{02} der Resonanzfrequenz und k_2 dem Proportionalbeiwert des gewünschten PT_2-Systems entsprechen.

Die Führungsübertragungsfunktion $F_w(s)$ des einfachen Regelkreises ergibt sich zu

$$F_w(s) = \frac{y(s)}{w(s)} = \frac{R(s) \cdot G_1(s)}{1 + R(s) \cdot G_1(s)}. \qquad \textbf{(4.16)}$$

Setzt man für die Strecke $G_1(s)$ und den Regler $R(s)$ gebrochen rationale Funktionen an

$$G_1(s) = \frac{A_1(s)}{B_1(s)} \qquad \textbf{(4.17)}$$

$$R(s) = \frac{P(s)}{Q(s)} \qquad \textbf{(4.18)}$$

so folgt mit Gl. (4.17) und (4.18) eingesetzt in Gl.(4.16)

$$F_w(s) = \frac{\frac{P(s)}{Q(s)} \cdot \frac{A_1(s)}{B_1(s)}}{1 + \frac{P(s)}{Q(s)} \cdot \frac{A_1(s)}{B_1(s)}}. \qquad \textbf{(4.19)}$$

Besitzt $B_1(s)$ nur Wurzeln, die links der imaginären Achse liegen, d.h. die Strecke $G_1(s)$ besitzt nur stabile Pole, so dürfen diese kompensiert werden, da dies einer erlaubten

Serienkompensation entspricht [97]. Setzt man daher für das Zählerpolynom *P(s)* des Reglers *R(s)*

$$P(s) = V \cdot B(s) \tag{4.20}$$

mit einem konstanten Faktor *V*, so folgt mit Gl.(4.20) eingesetzt in Gl.(4.19)

$$\begin{aligned} F_w(s) &= \frac{\frac{V \cdot B_1(s)}{Q(s)} \cdot \frac{A_1(s)}{B_1(s)}}{1 + \frac{V \cdot B_1(s)}{Q(s)} \cdot \frac{A_1(s)}{B_1(s)}} \\ &= \frac{\frac{V \cdot A_1(s)}{Q(s)}}{1 + \frac{V \cdot A_1(s)}{Q(s)}} = \frac{V \cdot A_1(s)}{Q(s) + V \cdot A_1(s)} \end{aligned} \tag{4.21}$$

und mit den Gl.(4.14) und (4.17) eingesetzt in Gl. (4.21)

$$F_w(s) = \frac{V \cdot k_1}{Q(s) + V \cdot k_1}. \tag{4.22}$$

Da $F_w(s)$ gleich $G_2(s)$ sein soll, muss mit den Gl.(4.15) und (4.22) gelten

$$G_2(s) = \frac{k_2}{s^2 + 2 \cdot d_2 \cdot \omega_{02} \cdot s + \omega_{02}^2} \stackrel{!}{=} F_w(s) = \frac{V \cdot k_1}{Q(s) + V \cdot k_1}. \tag{4.23}$$

Durch den Vergleich der Zählerpolynome von Gl.(4.23) kann *V* durch

$$V = \frac{k_2}{k_1} \tag{4.24}$$

berechnet werden, somit wird Gl.(4.22) mit Gl.(4.24) zu

$$F_w(s) = \frac{k_2}{Q(s) + k_2}. \tag{4.25}$$

Da ein PT_2-Verhalten gefordert wird, muss Q(s) ein Polynom 2. Ordnung sein.

$$Q(s) = q_2 \cdot s^2 + q_1 \cdot s + q_0. \tag{4.26}$$

Das Nennerpolynom von $F_w(s)$, Gl. (4.25) wird daher mit Gl. (4.26) zu

$$q_2 \cdot s^2 + q_1 \cdot s + (q_0 + k_2). \tag{4.27}$$

Durch einen Koeffizientenvergleich des Nennerpolynoms von $G_2(s)$, Gl. (4.15) und $F_w(s)$, Gl. (4.25) können die Parameter von Q(s), Gl. (4.26) bestimmt werden.

Koeffizientenvergleich:

$$
\begin{aligned}
s^2 + 2 \cdot d_2 \cdot \omega_{02} \cdot s + {\omega_{02}}^2 &= q_2 \cdot s^2 + q_1 \cdot s + (q_0 + k_2) \\
\text{daraus folgt: } {\omega_{02}}^2 &= q_0 + k_2 \Leftrightarrow q_0 = {\omega_{02}}^2 - k_2 \\
2 \cdot d_2 \cdot \omega_{02} &= q_1 \\
1 &= q_2
\end{aligned}
\quad . \qquad \textbf{(4.28)}
$$

Die Reglerübertragungsfunktion *R(s)* nimmt mit den Gl.(4.14), (4.15), (4.18), (4.20), (4.24), (4.26) und (4.28) folgende Form an

$$
R(s) = \frac{P(s)}{Q(s)} = \frac{V \cdot B_1(s)}{Q(s)} = \frac{\frac{k_2}{k_1} \cdot \left(s^2 + 2 \cdot d_1 \cdot \omega_{01} \cdot s + {\omega_{01}}^2 \right)}{s^2 + 2 \cdot d_2 \cdot \omega_{02} \cdot s + {\omega_{02}}^2 - k_2} . \qquad \textbf{(4.29)}
$$

Die diskrete Realisierung erhält man durch Anwendung der bilinearen Tustin-Transformation $s = \frac{2}{T_a} \cdot \frac{z-1}{z+1}$, wobei zur Wahrung der Kausalität in der Rückführschleife des Reglers ein Verzögerungsglied (Unit-Delay) eingesetzt werden muss. Dadurch erhöht sich die Ordnung des Reglers von zwei auf drei.

Wählt man als Wunschdämpfung 0.7 (aperiodischer Grenzfall) und lässt die Resonanzfrequenz unverändert, so erhält man einen diskreten Regler, der folgende Polstellen aufweist:

x-Achse:	1.0000	y-Achse:	1.0000
	0.3658		0.4459
	0.0915		0.0622

Die grenzstabilen Pole ergeben sich durch das Verzögerungsglied, alle weiteren Pole sind stabil.

4.5.2 Schwingungskompensation

4.5.2.1 H_2-Regler (Regelungsentwurf A)

H_2-Regler gehören zu der Gruppe der robusten Regler und werden zum Beispiel in der Bedämpfung von Gebäudeschwingungen bei Erdbeben eingesetzt [105]. Diese Anwendung hat eine große Ähnlichkeit mit der Mikropositioniereinrichtung, da hier ebenfalls versucht wird, über eine Beschleunigungsrückführung eine wegproportionale Stellgröße zu erzeugen. Betrachtet man die Mikropositioniereinrichtung als ein einstöckiges Gebäude, die Roboterschwingung als Erdbeben sowie die Piezo-Stacks als aktives Dämpfungsglied, kann man das Regelungsziel direkt übertragen. Eine konventionelle Zustandsregelung mit Zustandsbeobachter ist in diesem Fall nicht so einfach zu entwerfen, da die Messgröße, die Beschleunigung, kein Zustand in einem PT_2-System darstellt.

Die Ein- und Ausgangsgrößen einer jeden Regelstrecke *P(s)* können in beeinflussbare und nicht beeinflussbare Eingangsgrößen sowie messbare und nicht messbare Ausgangsgrößen unterteilt werden. Die nicht beeinflussbaren Eingangsgrößen, wie z.B. Anregung und Störungen, werden mit z, die beeinflussbaren Eingangsgrößen bzw. Stellgrößen der Strecke mit u, die nicht messbaren Ausgangsgrößen mit v und die

messbaren Ausgangsgrößen der Strecke mit y bezeichnet. Die nicht messbaren Ausgangsgrößen können benutzerdefiniert sein, d.h. jede beliebige Kombination der Systemzustände und jede Komponente des Stellgrößenvektors kann eine solche Ausgangsgröße definieren.

Die Entwurfsaufgabe besteht nun darin, einen Regler $K(s)$ zu entwerfen, der die messbaren Ausgangsgrößen als Eingangsgrößen und die beeinflussbaren Eingangsgrößen als Ausgangsgrößen besitzt. Die Übertragungsfunktion $H_{vz}(s)$ beschreibt nun die Auswirkungen der Eingangsgrößen z auf die Ausgangsgrößen v. Gesucht wird der Regler $K(s)$, der das System stabilisiert und die Übertragungsfunktion $H_{vz}(s)$ bezüglich der H_2-Norm minimiert, d.h. die Auswirkungen von z auf v werden minimiert, somit stellt sich das Problem mathematisch folgendermaßen dar

$$\min_{K(s)\in Ks} \left\| H_{vz} \right\|_2 . \tag{4.30}$$

Um die Übertragungsfunktion $H_{vz}(s)$ zu erhalten, muss $P(s)$ in die einzelnen Übertragungsfunktionen $P_{yz}(s)$, $P_{vu}(s)$, $P_{vz}(s)$ und $P_{yu}(s)$ unterteilt werden, somit kann $P(s)$ auf die folgende Form gebracht werden

$$P(s) \overset{\text{def}}{=} \begin{bmatrix} P_{vz}(s) & P_{vu}(s) \\ P_{yz}(s) & P_{yu}(s) \end{bmatrix} \tag{4.31}$$

was einer kompakten Form der Darstellung

$$\begin{aligned} v(s) &= P_{vz}(s) \cdot z(s) + P_{vu}(s) \cdot u(s) \\ y(s) &= P_{yz}(s) \cdot z(s) + P_{yu}(s) \cdot u(s) \end{aligned} \tag{4.32}$$

entspricht. Aus $P(s)$ kann $H_{vz}(s)$ folgendermaßen berechnet werden

$$H_{vz}(s) = P_{vz}(s) + P_{vu}(s) \cdot K(s) \cdot \left(I - P_{yu}(s) \cdot K(s) \right)^{-1} \cdot P_{yz}(s) . \tag{4.33}$$

Die H_2-Norm einer stabilen Übertragungsmatrix $G(s)$ ist definiert zu

$$\left\| G(s) \right\|_2 = \sqrt{\frac{1}{2 \cdot \pi} \int_{-\infty}^{\infty} \text{Spur}\left[G(j\omega) \cdot G^*(j\omega) \right] d\omega} \tag{4.34}$$

Im Fall der MPE wird als Messgröße die absolute Beschleunigung und als Stellgröße die der Auslenkung proportionale Piezospannung verwendet. Die nicht beeinflussbaren Eingangsgrößen sind Rauschen und Störschwingungen, die nicht messbare Ausgangsgröße ist die Auslenkung der unteren Plattform in absoluten Koordinaten.

Mit Gewichtsfunktionen (a_i), vergleichbar zu Kosten- oder Straffunktionen, kann das Regelungsziel bezüglich einer guten Messrauschunterdrückung und einer guten Störgrößenunterdrückung festgelegt werden. Aus [30] ergibt sich jedoch, dass beide Forderungen über das komplette Frequenzband nicht erfüllt werden können, so wird vorgeschlagen, gute Störgrößenunterdrückung für niedrige Frequenzen (bandbegrenzte Störschwingungen) und gute Messrauschunterdrückung für hohe Frequenzen (starker Einfluss im Beschleunigungssignal) zu fordern. Dies kann mit frequenzabhängigen Gewichtungsfunktionen ($W(s)$) erreicht werden.

Bei der H_2-Minimierung wird für die nicht beeinflussbaren Eingangsgrößen von weißem Rauschen ausgegangen. Da dies für reale Systeme meist nicht zutrifft, kann man durch geeignete Formfilter *F(s)* das Eingangsspektrum dem realen annähern. Da die Amplituden der Störschwingungen des Handhabungsgerätes bandbegrenzt sind, fällt in diesem Fall die Wahl auf einen Tiefpass. In der gleichen Weise kann das Messrauschen mit einem Formfilter (*n(s)*) modelliert werden. In Bild 4.5.2 ist die resultierende, zum H_2-Regelerentwurf notwendige erweiterte Strecke („augmented plant") dargestellt.

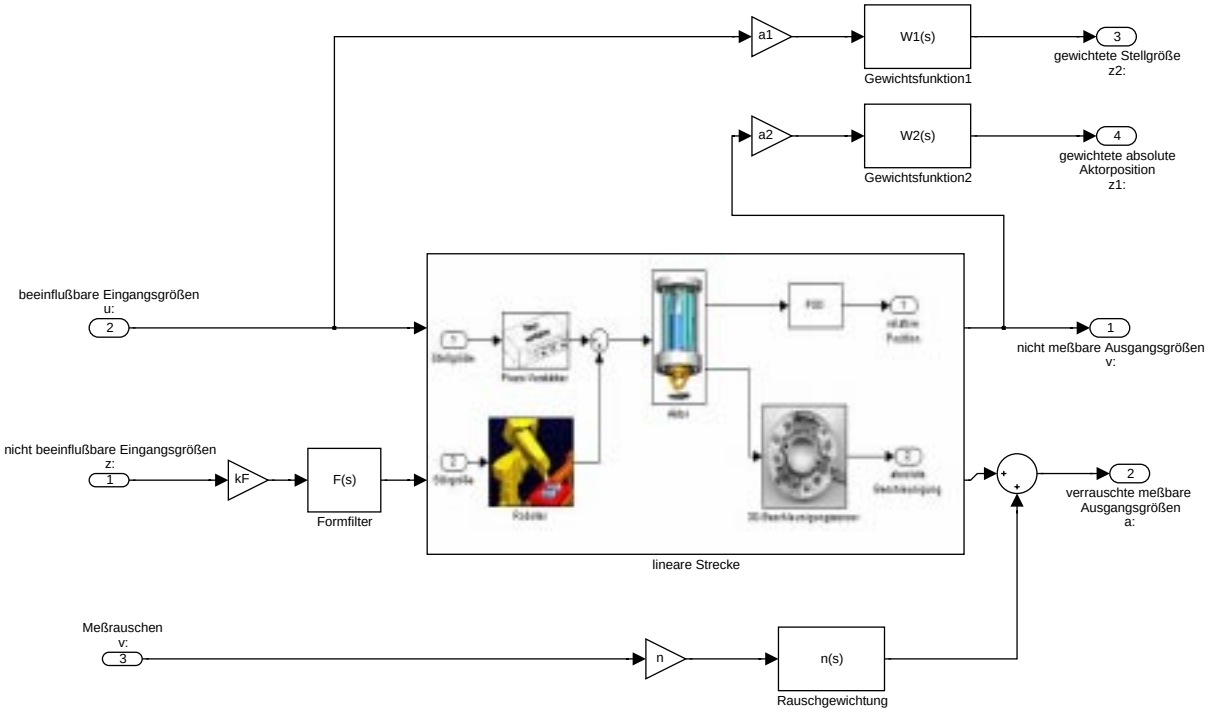

Bild 4.5.2 : erweiterte Strecke beim H_2-Reglerentwurf

Zur Lösung des H_2-Minimierungsproblems muss die erweiterte Strecke, wegen der einfacheren und stabileren numerischen Darstellung im Zustandsraum, in eine Zustandsdarstellung der Form

$$\begin{aligned}\dot{\vec{x}}(t) &= A\vec{x}(t) + B_1\vec{z}(t) + B_2\vec{u}(t)\\ \vec{v}(t) &= C_1\vec{x}(t) + D_{11}\vec{z}(t) + D_{12}\vec{u}(t)\\ \vec{y}(t) &= C_2\vec{x}(t) + D_{21}\vec{z}(t) + D_{22}\vec{u}(t)\end{aligned} \qquad \textbf{(4.35)}$$

gebracht werden, obwohl es sich eigentlich um eine Frequenzgangmethode handelt. Mit (4.35) ergibt sich (4.31) zu

$$P(s) = \left[\begin{array}{c|cc} A & B_1 & B_2 \\ \hline C_1 & D_{11} & D_{12} \\ C_2 & D_{21} & D_{22} \end{array}\right] \qquad \textbf{(4.36)}$$

Um eine H_2-Minimierung von (4.36) durchführen zu können, müssen die Voraussetzungen von [21] erfüllt sein, ehe die mathematische Lösung mit den Methoden aus [25] bzw. mit dem *dh2lqg*-Befehl der Robust Control Toolbox von MatLab bestimmt werden kann.

Im Fall der MPE ergibt sich nach Entfernen aller nicht minimaler Zustände ein Regler 9.Ordnung, dessen Pol- und Nullstellen in der z-Ebene in Bild 4.5.3 zu sehen sind. Dabei fällt die doppelte, grenzstabile Nullstelle bei 1 auf, die einen zweifachen Integrator darstellt. Dies ist nicht verwunderlich, da aus einem Beschleunigungsmesswert mit dem H_2-Regler eine wegproportionale Stellgröße erzeugt werden muss. Man kann den H_2-Regler somit in zwei Regler aufspalten. Der eine Regler von der Ordnung 7 erhält alle stabilen Pole, die beiden grenzstabilen Pole werden dem zweiten Regler zugewiesen. Das doppelt-integrierende Verhalten des zweiten Reglers bedarf bei der Regleraufschaltung, wie unter 4.4.1 beschrieben, ein besonderes Augenmerk, da die beiden Anfangsbedingungen der Integratoren (Anfangsgeschwindigkeit und Anfangsweg) nicht bekannt sind und dies zu einem instabilen Verhalten führen würde.

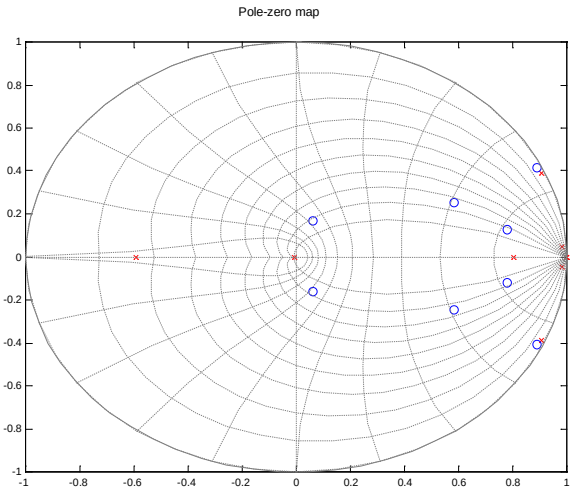

Bild 4.5.3 : Pol- und Nullstellen des H_2-Reglers

4.5.2.2 P-Regler (Regelungsentwurf B)

Das Regelungsziel der Störgrößenunterdrückung ist definiert zu

$$y_{x,Aktor} = -y_{x,Rob} \qquad \textbf{(4.37)}$$

Da durch die Sensorfusion und Zustandsschätzung mit dem Kalman-Filter ohne doppelte Integration eine optimale Schätzung für die Störgröße $y_{x,Rob}$ vorliegt, kann (4.37) näherungsweise auch als

$$y_{x,Aktor} = -\hat{y}_{x,Rob} \qquad \textbf{(4.38)}$$

geschrieben werden. Der Regelungsentwurf vereinfacht sich dadurch wesentlich, da $\hat{y}_{x,Rob}$ als Führungsgröße für den Dämpfungs- und Linearisierungsregler verwendet werden kann. Man könnte jetzt vermuten, dass dadurch dieser Regelkreis zu einer Steuerung verkommt, da die Zustände der Piezoaktorik und des Handhabungsgerätes bzw. Tisches in dem in diesem Reglerentwurf verwendeten Modell keine Verkopplung aufweisen. Im Kalman-Filter werden zur Schätzung der Störgröße jedoch auch die Zustände der Piezoaktorik herangezogen, wie aus Gl. (4.13) deutlich wird. Dadurch existiert eine Verkopplung der Zustände, so dass es sich schon um einen Regelkreis handelt. Durch den Einsatz eines einfachen P-Reglers wird die Regelkreisverstärkung

festgelegt und durch Vorschalten eines Butterworth-Filters 1.Ordnung wird erreicht, dass, ähnlich wie bei der Gewichtung der Messrauschunterdrückung im H_2-Regler, das durch den Beschleunigungssensor eingebrachte hochfrequente Messrauschen gedämpft wird ohne das Nutzsignal zu manipulieren.

4.5.3 Führungsregler

4.5.3.1 Regelungsentwurf A

Das Ergebnis der Bildverarbeitung wird dem Regelkreis asynchron als Führungsgröße zugeführt. Da aber der H_2-Regler so ausgelegt ist, dass er Störschwingungen bedämpft, würde er einen solchen Führungssprung ebenfalls bedämpfen und kompensieren. Mit Hilfe des Führungsreglers kann dieses Verhalten des H_2-Reglers umgangen werden.

Der Führungsregler besteht im wesentlichen aus einem Modell der Regelstrecke, einem sogenannten Parallelmodell. Bei einer neuen Führungsgröße von der Bildverarbeitung wird diese zur Regelgröße addiert und die entsprechende Sprungantwort des Parallelmodells am H_2-Reglereingang subtrahiert. Die durch den Führungssprung ausgelöste Beschleunigung wird so beim H_2-Regler nicht sichtbar und wird dadurch nicht bedämpft. Diese Kompensation gelingt allerdings nur so gut, wie das Parallelmodell mit der Realität übereinstimmt.

4.5.3.2 Regelungsentwurf B

Die realisierte Schwingungskompensation, die im wesentlichen aus einem P-Regler besteht, bewirkt ein gutes Führungsverhalten des zweiten Regelkreises. Mit dem Führungsregler muss somit nur noch der Amplitudengang des Schwingungskompensationsreglers ausgeglichen werden. Da es sich bei den Führungsgrößen um statische Größen handelt, werden diese durch den Tiefpass, der dem P-Regler folgt, nicht beeinflusst, so dass als Führungsregler auch ein P-Regler eingesetzt werden kann. Die Verstärkung dieses P-Reglers ergibt sich zur reziproken Verstärkung des Schwingungskompensationsreglers. Das Führungsverhalten der gesamten Regelung ist somit im Gegensatz zum ursprünglichen Regelungsentwurf unabhängig vom verwendeten Streckenmodell.

4.6 Simulation

Beim Regelungsentwurf A wurden bei der Simulation der MPE die Sensoren sowie die Hardwareanbindung nicht berücksichtigt. Es konnte daher mit der Simulation nur die Aussage getroffen werden, ob das Reglerprinzip verwendet werden kann oder nicht. Es konnten keine Aussagen über den reellen Regelfehler getroffen werden. Außerdem mussten zur Simulation und zur Codegenerierung je ein eigenes SimuLink-Blockdiagramm erstellt werden, was immer wieder zu Konsistenzproblemen führte.

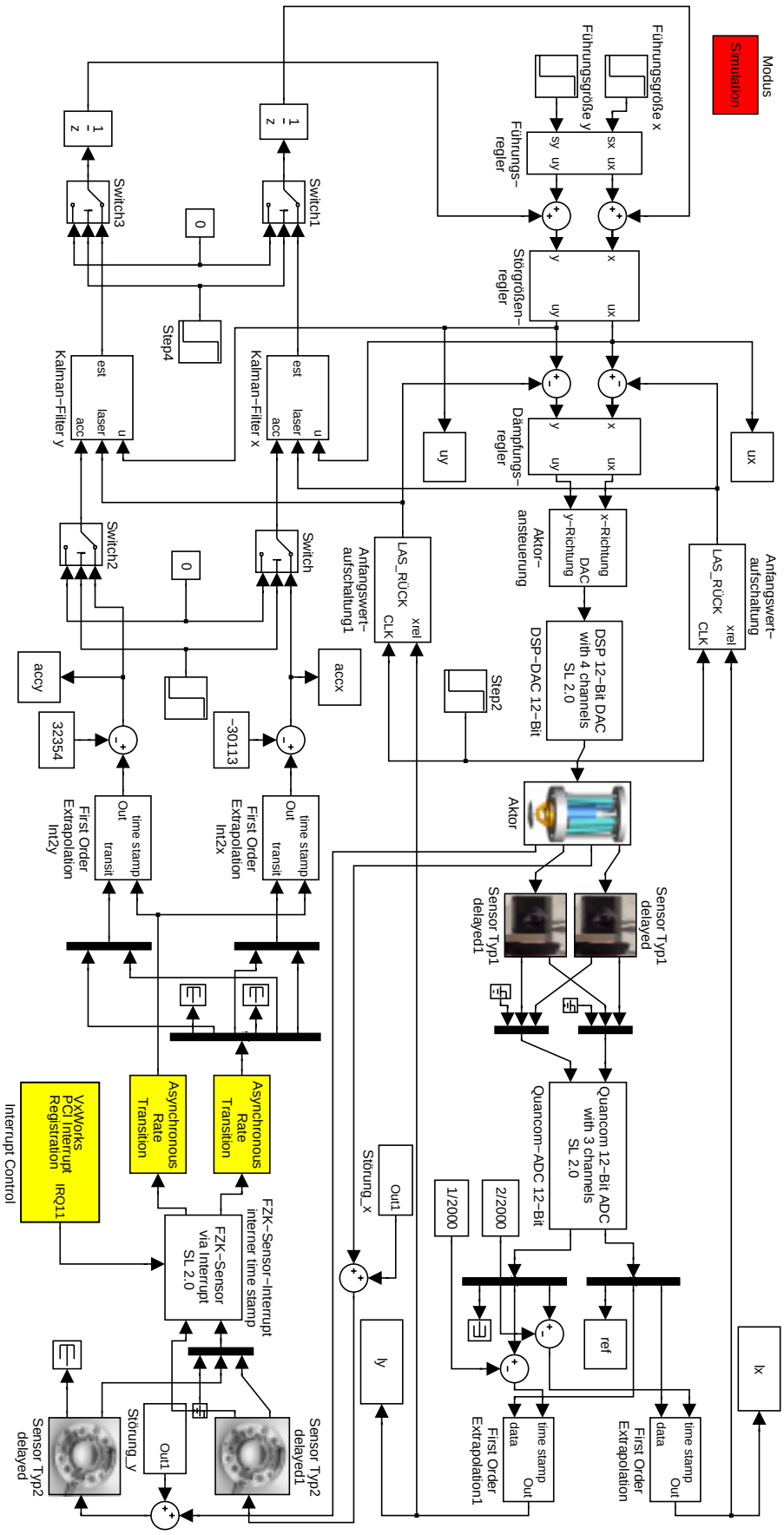

Bild 4.6.1 : Simulationsblockdiagramm

Mit den in dieser Arbeit entwickelten SimuLink-Erweiterungen ist es möglich, zur Codegenerierung und zur Simulation das gleiche Blockdiagramm zu verwenden. In Bild 4.6.1 ist das für den neuen Reglerentwurf verwendete Blockdiagramm für den Fall zu sehen, dass es zur Simulation verwendet wird. Bild 4.6.2 zeigt das aus Bild 4.6.1 automatisch zur Codegenerierung konvertierte Blockdiagramm. Wie hier zu sehen ist, wurden die Blöcke entfernt, die die Regelstrecke spezifiziert haben. Des weiteren wurden die Modifikationen durchgeführt, die unter 3.6.2 beschrieben werden. Es ist dazu notwendig, dass den verwendeten Sensoren und Aktoren Hardware-Interfaces zugeordnet werden.

Es werden folgende Zuordnungen getroffen:

- Die Lasersensordaten werden über einen 4-fach A/D-Wandler eingelesen.
- Die asynchron zur Basisabtastrate eintreffenden Beschleunigungsdaten werden Interrupt gesteuert eingelesen. Jedesmal, wenn ein kompletter Datensatz nach der Dezimierung und Übertragung im DSP zur Verfügung steht, wird ein PCI-Interrupt (IRQ 12) ausgelöst.
- Die Ansteuerdaten werden über einen 4-fach DA/Wandler direkt an den Piezoverstärker ausgegeben.

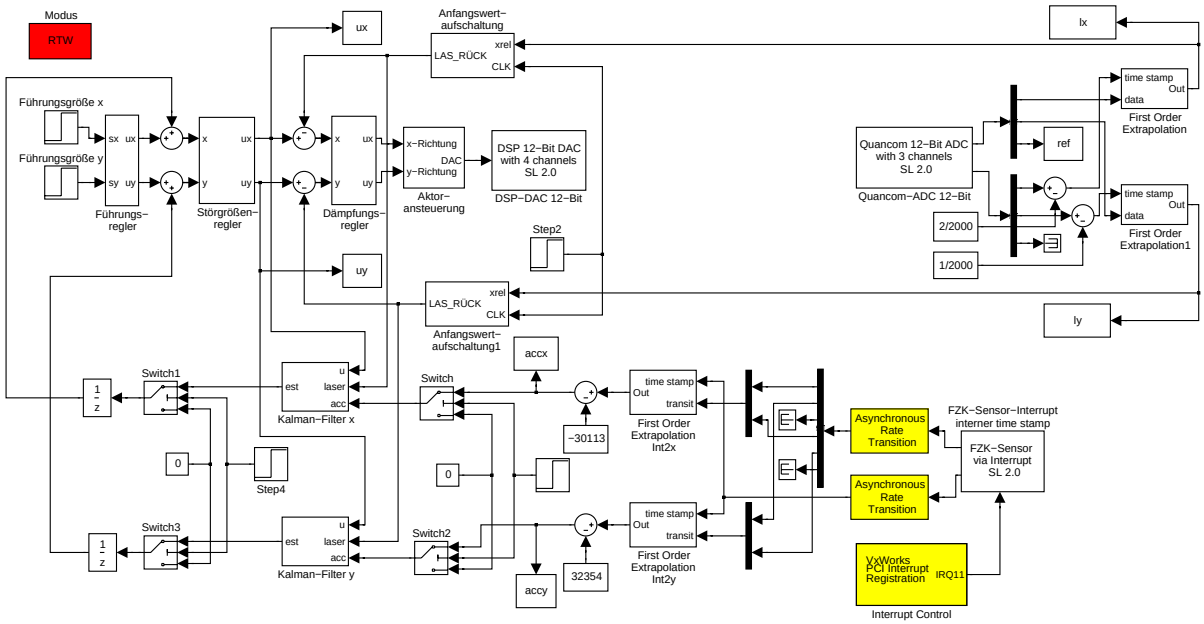

Bild 4.6.2 : Blockdiagramm zur Codegenerierung

Nach Hinzufügen von definierten Störungen und geeigneten Führungsgrößen kann das Gesamtsystem simuliert werden. Im Fall der MPE werden Impulsanregungen des Handhabungsgerätes bzw. des Tisches simuliert. Die Führungsgröße wird mit einer Sprungfunktion, die das Eintreffen eines Bildverarbeitungsergebnisses darstellen soll, definiert.

Nach erfolgreichen Software-Simulationsergebnissen erfolgt anhand des automatisch konvertierten Blockdiagramms die automatische C-Codegenerierung. Als Zielsystem für die MPE wurde ein PC mit einem Pentium II-233 Prozessor und mit dem Echtzeitbetriebssystem VxWorks, wegen der direkten MatLab-Unterstützung, eingesetzt. Zur Hardware-Simulation wird der Versuchsaufbau von Bild 4.6.3 verwendet, bei dem die

MPE auf dem Kopf stehend auf einen Tisch montiert ist und das interne Lasermesssystem wegen der schon erwähnten Probleme durch die externen Lasertriangulationssensoren ersetzt wird. Die Störgröße wird durch manuelle Impulsanregung des Tisches erzeugt, die Führungsgröße wird wie bei der Software-Simulation als Sprungfunktion definiert. Tests am Industrieroboter, bei denen dann auch die Bildverarbeitung, die über die serielle Schnittstelle an den Regelungs-PC angebunden wäre, eingesetzt hätte werden können, konnten ebenfalls aufgrund der Probleme mit dem internen Lasermesssystem nicht durchgeführt werden. Die translatorische x-y-Bewegung des Tisches wird mit zusätzlichen Lasertriangulationssensoren gemessen. Die Koordinatensysteme der Sensoren und Aktoren werden nur manuell kalibriert, eine softwaremäßige Feinkalibrierung, wie sie in [30] vorgeschlagen wird, wird aufgrund der zufriedenstellenden Ergebnisse nicht durchgeführt.

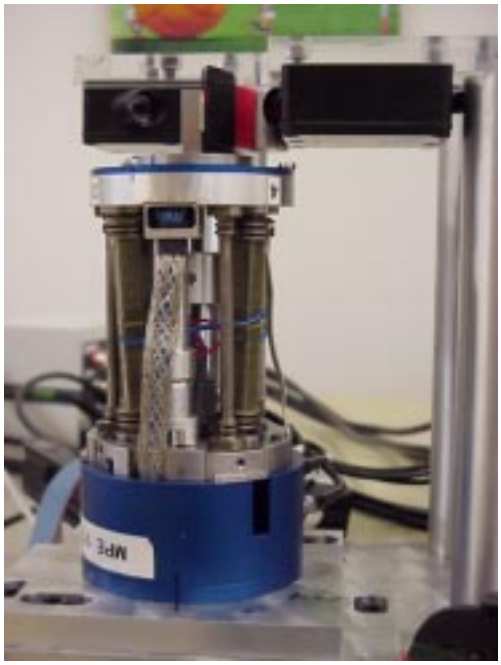

Bild 4.6.3 : Versuchsaufbau

4.7 Ergebnisse

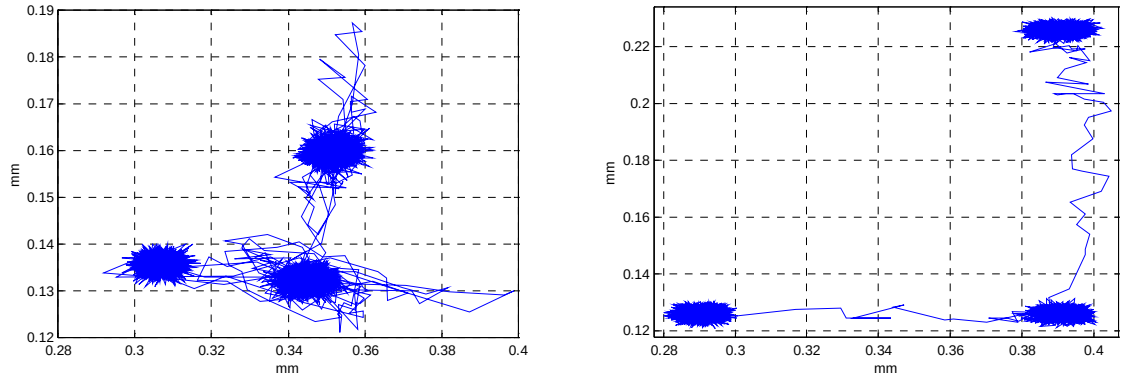

Bild 4.7.1 : x-y-Bewegung ohne (links) und mit (rechts) Linearisierungs- und Dämpfungsregler

Die Wirkungsweise des Linearisierungs- und Dämpfungsreglers wird in Bild 4.7.1 deutlich. Links ist die Bewegung der beweglichen Plattform in der x-y-Ebene für eine Führungsgröße von 100μm in x-Richtung gefolgt von einer Führungsgröße von 100μm in y-Richtung abgebildet. Es ist das ungedämpfte Hysterese behaftete Verhalten der Piezoaktorik sichtbar, zu dieser Bewegung passt auch die Sprungantwort aus Bild 4.3.2 links. Auf der rechten Seite von Bild 4.7.1 ist die Bewegung der Plattform für die gleiche Führungsgröße, jedoch mit aktivem Linearisierungs- und Dämpfungsregler, zu sehen. Die Hystereseeffekte treten nicht mehr auf – die Plattform bewegt sich die geforderten 100μm – und es wird eine höhere Dämpfung des Aktors erreicht, wie auch die Sprungantwort in Bild 4.3.2 rechts zeigt. Die Dämpfung wird von 0.03 auf 2.5 erhöht, die Funktion des inneren Regelkreises ist dadurch bestätigt worden.

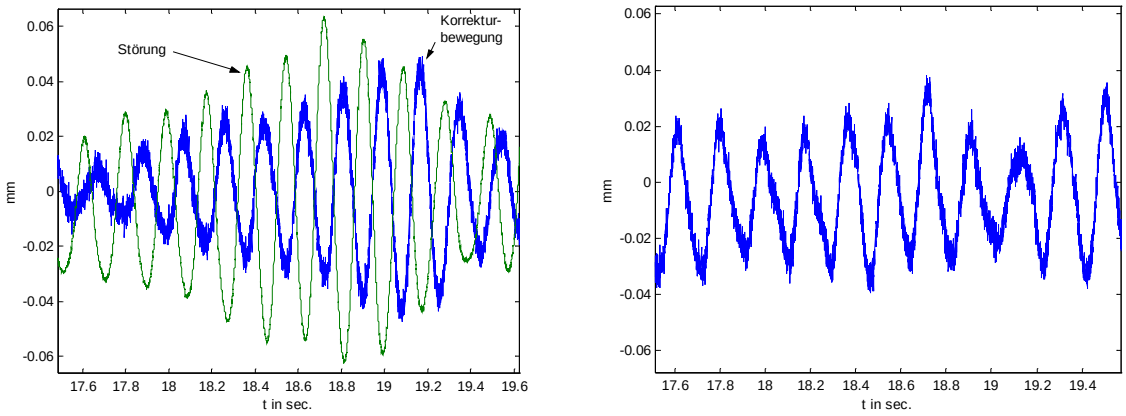

Bild 4.7.2 : Störung und Korrekturbewegung der Schwingungskompensation (links), resultierender Regelfehler (rechts)

In Bild 4.7.2 links ist die durch die Schwingungskompensation des Regelungsentwurfs B hervorgerufene Korrekturbewegung sowie die Störgröße abgebildet. Auf der rechten Seite ist der resultierende Regelfehler, der im Bereich von $\pm$30μm bei einer Störung von 60μm liegt, zu sehen. Bild 4.7.3 links zeigt eine Ausschnittsvergrößerung bei einer

Störung von ca. 150μm. Man sieht, dass die Störung effektiv im geforderten Bereich bedämpft werden kann, die Funktion der auf Sensorfusion und einem P-Regler basierenden Schwingungskompensation wird damit bestätigt.

Einem Führungssprung kann durch den Führungsregler mit P-Verhalten nahezu ideal gefolgt werden, wie Bild 4.7.3 auf der rechten Seite für einen 100μm Sprung in x-Richtung und einen 50μm Sprung in y-Richtung zeigt.

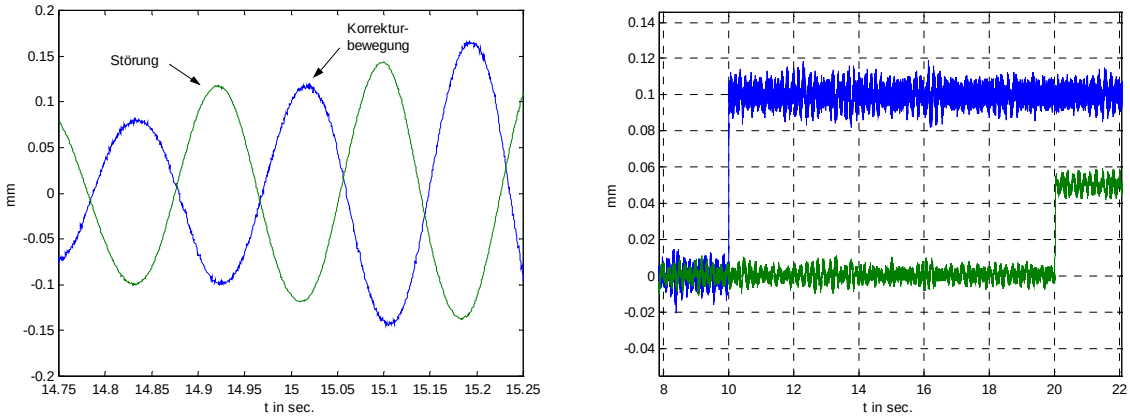

Bild 4.7.3 : Ausschnittsvergrößerung der Schwingungskompensation (links) und Führungsgrößensprung (rechts)

Die Grenzen der Schwingungskompensation werden in Bild 4.7.4 deutlich (links: Störung und Korrekturbewegung; rechts: resultierender Regelfehler). Bei einer zu großen Impulsanregung kann der Regler der Störung zuerst nicht folgen. Das liegt daran, dass im Kalman-Filter die unbekannte Störgröße als weißes Rauschen behandelt wird und eine Anpassung des Störspektrums mittels eines Formfilters nicht durchgeführt werden kann – das Störspektrum der manuell eingespeisten Störung ist schwierig zu bestimmen.

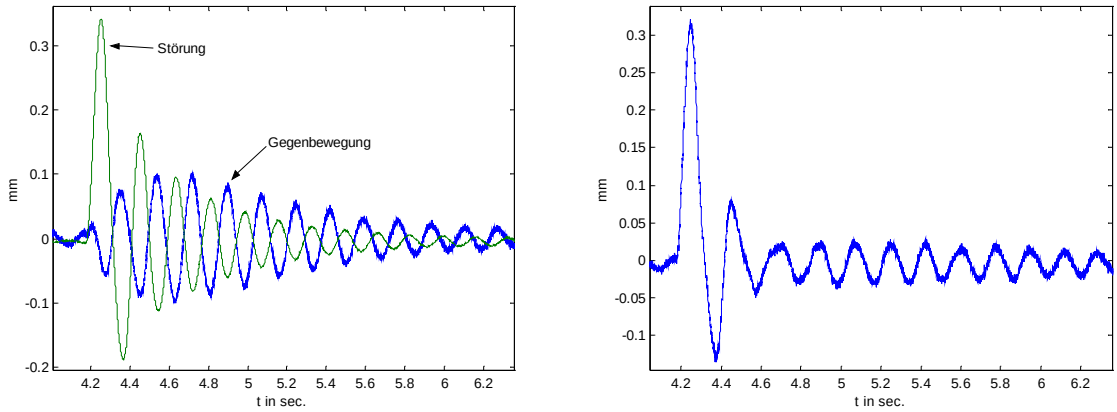

Bild 4.7.4 : Grenzen der Schwingungskompensation

Bild 4.7.5 zeigt die Auswirkung eines falschen Modells der Störeinkopplung auf die Korrekturbewegung. In diesem Fall wurde die Resonanzfrequenz des PT_2-Systems „Tisch“ falsch spezifiziert. Man sieht, dass deshalb die Schätzung der Störgröße im modellbasierten Kalman-Filter auch eine falsche Resonanzfrequenz enthält.

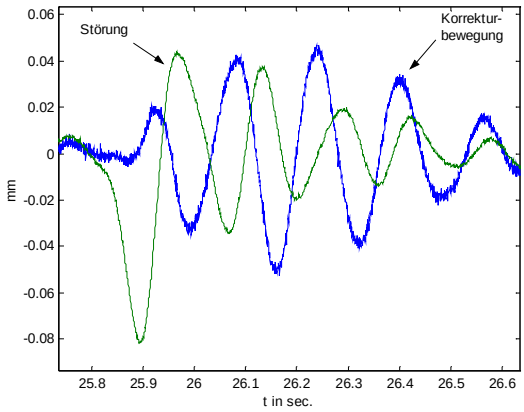

Bild 4.7.5 : Auswirkungen eines falschen Modells der Störeinkopplung bei der Schwingungskompensation

Vergleicht man die Ergebnisse der Schwingungskompensation des Regelungsentwurf B (Sensorfusion und P-Regler) von Bild 4.7.2 mit den Ergebnissen der H_2-Regler basierten Schwingungskompensation (Regelungsentwurf A) von Bild 4.7.6 links, so sieht man eine vergleichbare Leistungsfähigkeit beider Regelungssysteme. Der Entwurf eines H_2-Reglers ist, wie Abschnitt 4.5.2.1 zeigt, wesentlich aufwendiger als der Entwurf eines Kalman-Filters und eines P-Reglers. Zudem gestaltet sich beim H_2-Regler aufgrund der impliziten doppelten Integration die Regleraufschaltung sowie die driftende Offsetkorrektur problematisch. Diese Effekte treten beim Regelungsentwurf B nicht auf. Das Führungsverhalten beider Regler (Bild 4.7.3 rechts bzw. Bild 4.7.6 rechts) unterscheidet sich jedoch deutlich. So ist das Führungsverhalten des Reglers aus Regelungsentwurf B nahezu ideal, während beim Regelungsentwurf A durch Verwendung eines nicht rückgeführten Parallelmodells deutliche Überschwinger und Verkopplungen zwischen der x- und y-Achse auftreten, im eingeschwungenen Zustand zeigen beide Regelungssysteme dann wieder das gleiche Verhalten.

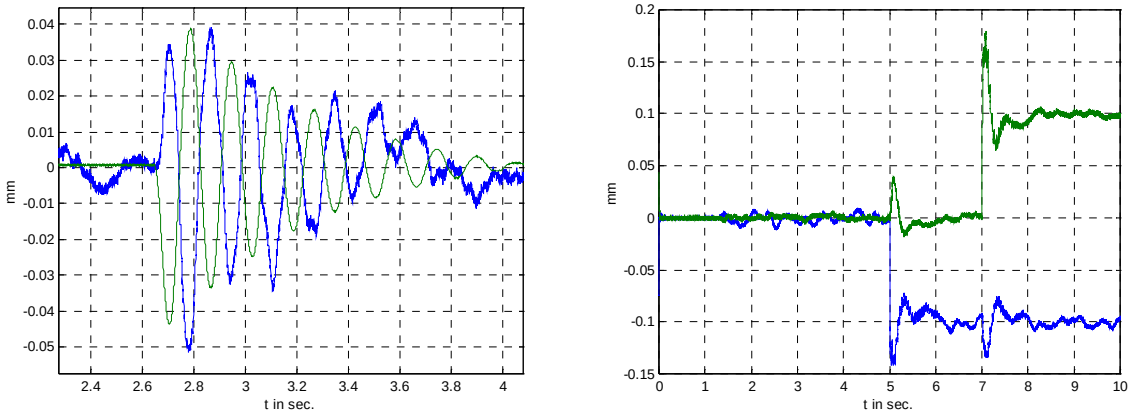

Bild 4.7.6 : Schwingungskompensation (links) und Führungsgrößensprung (rechts) des Regelungsentwurfs A

Fasst man die Ergebnisse zusammen (siehe Tabelle 4.7.1), ergibt sich, dass mit einem wesentlich einfacheren Regelungsentwurf durch die Sensorfusion und Synchronisation mit den Methoden dieser Arbeit bessere Ergebnisse erzielt werden als mit dem ursprünglichen ohne Sensorfusion und Synchronisation entworfenen Regelungssystem.

Zudem wird durch die entwickelte Toolbox die Simulation und C-Codegenerierung erleichtert. Vergleicht man die simulierten Ergebnisse mit den reellen Ergebnissen z.B. für die Auslenkung der beweglichen Plattform in Bild 4.7.7 (links: Korrekturbewegung; rechts: Führungsgrößensprung), so sieht man, dass jeweils beide Kurven fast deckungsgleich sind. Durch die Einbindung der Sensormodelle und der Hardwareeigenschaften in das Streckenmodell wird also erreicht, dass der Reglerentwurf zuerst rein simulativ ohne Hardware durchgeführt werden kann und dann die dort erzielten Ergebnisse fast 1:1 auf das reale System übertragen werden kann. Grundlage dazu ist die gewissenhafte Modellierung der Sensoren und der Strecke. Beim Regelungsentwurf A lag ein Faktor von ungefähr 100 zwischen simulierten und realem Regelungsfehler (siehe [30]), da bei der Simulationen ohne den Sensoreigenschaften der ideale Regelkreis vorliegt und die Störungen nahezu ganz kompensiert werden können (Restfehler $<0.4\mu$m).

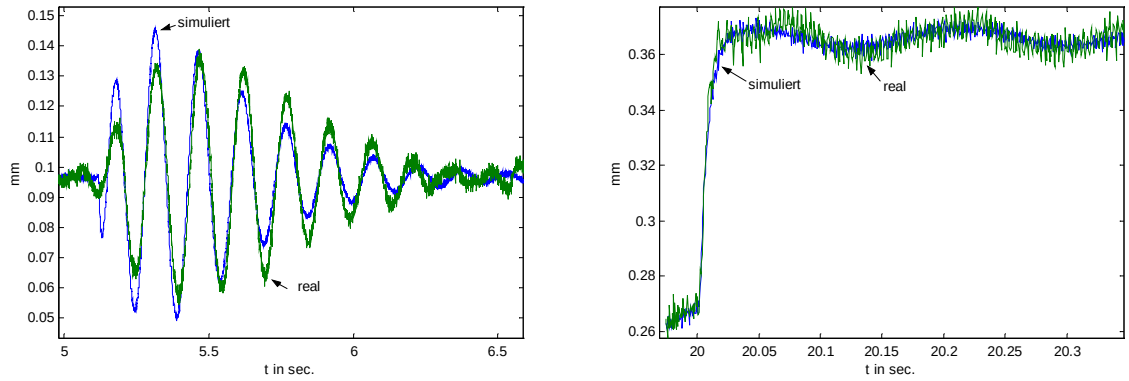

Bild 4.7.7 : Vergleich Simulation und Realität: links Schwingungskompensation, rechts Führungsgrößensprung

	Regelungsentwurf A		**Regelungsentwurf B**	
	Methode	Ergebnis		Ergebnis
Linearisierungsregler	Polverschiebung	$\pm 2\mu$m	Polverschiebung	$\pm 2\mu$m
Schwingungskompensation	H_2-Regler	$\pm 30\mu$m	P- bzw. LQG-Regler	$\pm 30\mu$m
Führungsregler	Parallelmodell basiert	T=500ms	P-Regler	T=1ms
Sensorsynchronisation	-		FOH	
Sensorfusion	-		Kalman-Filter	
Regelungsfehler (Simulation)	ohne Sensoren	$\pm 0.4\mu$m	mit Sensoren	$\pm 30\mu$m

Tabelle 4.7.1 : Vergleich der beiden Regelungsentwürfe

5 Ausblick

Die in dieser Arbeit entwickelten Methoden und Algorithmen wurden in einer Toolbox zusammengefasst, die jedoch nur die Basisstrukturen einer Standard-MatLab-Toolbox enthält. Hilfetexte und Anwendungsbeispiele zu den einzelnen SimuLink-Blöcken (siehe Anhang A: Blockbeschreibung) wurden bisher nicht erstellt. Eine Kurzbeschreibung wird dabei im „Help"-Teil der Blockmaske eingetragen, während der ausführliche Teil mit weiterführenden Links, auch zu bereits bestehenden MatLab-Hilfetexten, in einer html-Datei realisiert werden muss. Dies Hilfe-Datei kann dann durch Anklicken des „Help"-Buttons in der Eingabemaske des jeweiligen Blocks geöffnet werden. Durch die Integration von html-Hilfe-Dateien mit Links hat man das komplette MatLab-Hilfesystem zur Verfügung.

Des weiteren ist die grafische Überarbeitung der Toolbox von Nöten. Mit Icons wird, wie z.B. das Aktor-Icon in Bild 4.6.1, die Übersichtlichkeit von SimuLink-Blockdiagrammen erhöht. Es erscheint so sinnvoll für die erstellten Blöcke aussagekräftige Icons zu entwickeln. Die Einbindung der Icons erfolgt dann im „Display"-Teil der Blockmaske.

Die Toolbox kann jederzeit mit neuen Verfahren, sei es zur Sensorfusion, Synchronisation oder Sensorvorverarbeitung erweitert werden. Es wurden in die bestehende Toolbox in erster Linie allgemeine Standardverfahren implementiert, so dass in weiterführenden Arbeiten neue speziellere Verfahren eingebunden werden können, wie z.B. ein diskreter adaptiver Beobachter zur Offsetschätzung [106] oder „unfalsified PID-Regler" [52].

Die implementierten Hardware-Interfaces beziehen sich auf die Hardware, die zur Realisierung der MPE benötigt wurde. Bei der Anbindung neuer Hardware stehen die bereits implementierten Interfaces als Templates zur Verfügung, da die im wesentlichen vorkommenden Hardware Kategorien – passive I/O-Komponenten oder aktive Interrupt getriggerte Eingabe-Komponenten – abgedeckt wurden.

Für I/O-Komponenten, die eine eigene CPU besitzen, z.B. das DSP-Prozess-Interface-Board zur Anbindung des Beschleunigungssensorsystems [3], wäre es wünschenswert, wenn man die Algorithmen, welche dort ablaufen, auch im Simulationsdiagramm des Regelsystems spezifizieren und daraus C-Code generieren könnte. Dies hätte den Vorteil, dass man, zur Ausnutzung der auf diesen Komponenten zur Verfügung stehenden Rechenleistung, Blöcke auf diese Einheiten auslagern könnte. Der DSP des Prozess-Interface-Boards dient im Moment nur zur Byte-weisen Sortierung der Sensorwerte, könnte aber problemlos noch Teile der Sensorvorverarbeitung und der Synchronisation übernehmen. Es müsste dazu den einzelnen Blöcken die gewünschte CPU bzw. Ausführungseinheit im Eigenschaftsfeld eingetragen werden. Durch Parsen und Teilen des Simulationsblockdiagramms, ähnlich dem Umschalten zwischen Simulation und Codegenerierung, würde für jede Ausführungseinheit automatisch ein eigenes Blockdiagramm generiert werden und die Kommunikationsblöcke zwischen den Ausführungseinheiten eingefügt werden können. Durch Verwendung unterschiedlicher Makefile-Templates kann dann mit dem Real-Time Workshop C-Code für die ver-

schiedenen Ausführungseinheiten generiert werden. Diese Art des Blockdiagramm-Splittings könnte weiterführend auch in Rapid-Prototyping-Systemen verwendet werden.

Der Kommunikationsfluss im externen Simulationsmodus ist standardmäßig nur unidirektional, vom Entwicklungshost zum Zielsystem, möglich. Durch Erweiterung der Kommunikationsstrukturen ist auch eine bidirektionale Kommunikation denkbar, was bei den ersten Schritten der Hardware-in-the-Loop-Simulation sehr hilfreich wäre.

Das entwickelte Regelsystem für die MPE kann durch weitergehende Maßnahmen noch verbessert und robuster gestaltet werden. Die Definition von Formfiltern, die das als weißes Rauschen angenommene unbekannte Störsignal dem reellen Störspektrum besser anpassen, würden bessere Schätzungen im eingesetzten Kalman-Filter bewirken. Des weiteren könnte zur Schwingungskompensation der verwendete P-Regler, der nur zwei der vier geschätzten Zustände verwendet, durch einen vollständigen Zustandsregler, der in Kombination mit dem Kalman-Filter als LQG-Regler bezeichnet wird, ersetzt werden, um die Dynamik der Korrekturbewegungen zu verbessern. In Simulationen konnten damit Regelungsfehler kleiner 20μm sowie eine bessere Dynamik (erster Peek) erzielt werden. Die Korrektur der Querempfindlichkeiten der Beschleunigungssensoren nach [15] sowie die Auswertung des Temperatursensors des Beschleunigungssensorsystems und Anpassen der temperaturabhängigen Parameter der Sensoren und Aktoren würde das Regelsystem robuster gegenüber Parameterschwankungen machen. Messungen haben gezeigt, dass die Eigenfrequenz sowie die Steifigkeit der Piezoaktorik von der Temperatur abhängen. Eine Kalibrierung des Streckenmodells (Eigenfrequenz, Dämpfung) vor jedem Einsatz würde dadurch überflüssig werden und Temperaturveränderungen während des Betriebs hätten dann keinen Einfluss auf die Leistungsfähigkeit des Regelungsystems.

Der Test der MPE am Industrieroboter mit angeschlossener Bildverarbeitung würde letztendlich den Beweis erbringen, ob das Komplettsystem, bestehend aus Aktorik, Regler und Bildverarbeitung für die Mikromontage geeignet ist. Es scheiterte bisher an den schon geschilderten Problemen mit dem internen Lasermesssystem, das zusammen mit der Aktorik von einem Projektpartner entwickelt wurde. Durch den Einsatz einer schnelleren Bildverarbeitung oder einer besseren Optik, die es ermöglicht, pro Fügevorgang mehrere Bilder zu bearbeiten, könnten diese Messdaten zur Verbesserung der Schätzungen im Kalman-Filter herangezogen werden. Alles in allem erscheint mit diesem Optimierungen ein erzielbarer Regelungsfehler von 10μm realistisch.

6 Literatur

[1] J. Ackermann, „Abtastregelung“, ISBN 3-540-50112-6, „Sampled data control systems“, ISBN 0-387-50112-6, Springer Verlag, 1988.

[2] H. Akbarally and L. Kleeman, „3D Robot Sensing from Sonar and Vision“, in Proceeding of the IEEE International Conference on Robotics and Automation, Minneaplois, Minnesota, April 1996.

[3] T. Andreas, “Erweiterung einer DSP-PCI-Karte zur Anbindung externer Prozesse und Integration in eine Standard-PC-Umgebung”, Diplomarbeit am Lehrstuhl für Prozessrechner, TU München, Mai 1998.

[4] T. Aono, K. Fujii, S. Hatsumoto, T. Kamiya, „Positioning of vehicle on undulating ground using GPS and dead reckoning“, Proceedings of the IEEE International Conference on Robotics & Automation ICRA’98, Leuven, Belgium, 1998.

[5] M. Araki and K. Yamamoto, „Multivariable Multirate Sampled-Data Systems: State-Space Description, Transfer Characteristics, and Nyquist Criterion“, in IEEE Transaction on Automatic Control, Vol. AC-31, No. 2, Feb 1986.

[6] U. Bahr, „Messfehler bei der Wegmessung mit Beschleunigungssensoren“, in tm Technisches Messen 6/94, Oldenbourg Verlag, 1994.

[7] B. Barshan, H. F. Durrant-Whyte, „An Inertial Navigation System for a Mobile Robot“, in IFAC Intelligent Autonomous Vehicles, Southampton, UK, 1993.

[8] R. Behringer, V. v. Holt, D. Dickmanns, „Road and Relative Ego-State Recognition“, in Proceedings of the Conference on Intelligent Vehicles, Detroit, 1992.

[9] G. S. Bell, W. J. Wilson, „Coordinated Controller Design for Position Based Robot Visual Servoing in Cartesian Coordinates“, in the Proceedings of the IEEE International Conference on Robotics and Automation, Minneapolis, USA, 1996.

[10] Stefan Blum, Christian Robl, „Sensor data processing and control of a micro positioning system“, in Proc. of the Workshop on European Scientific and Industrial Collaboration on Promoting 'Advanced Technologies in Manufacturing' (WESIC'98), Girona, Spain, June 1998.

[11] S. Boyd, C. Barratt and S. Norman, „Linear Controller Design: Limits of Performance Via Convex Optimization“, in Proceedings of the IEEE, Vol. 78, No. 3, March 1990.

[12] S. Boyd, “Convex Matrix Optimization Problems, with Applications in Control, Signal Processing and Circuit Design”, In Proceedings of 1999 IEEE Symposium on Computer Aided Control System Design, Hawai’i, USA, August 1999.

[13] M. Bozorg, E. M. Nebot and H. F. Durrant-Whyte, „A Decentralised Navigation Architecture“, Proceedings of the IEEE International Conference on Robotics & Automation ICRA’98, Leuven, Belgium, 1998.

[14] L. J. Brown, S. P. Meyn, R. A. Weber, „Adaptive Dead-Time Compensation with Application to a Robotic Welding System“, in IEEE Transactions on Control Systems Technology, Vol. 6, No. 3, May 1998.

[15] R. Brunnbauer, “Einsatz eines LIGA-Beschleunigungssensorsystems als inertiales Messsystem für die Positionierkontrolle eines Industrieroboters”, Diplomarbeit am Lehrstuhl für Prozessrechner, TU München, 1997.

[16] J. J. Buchholz, „Sensorfehlererkennung in Flugzeugen mit Beobachter und Ploynomklassifikator“, Institut für Flugmechanik, Braunschweig, DLR-Forschungsbericht, DLR-FB 91-34, 1991.

[17] D. Burschka, C. Eberst, C. Robl, „Vision Based Model Generation for Indoor Environments“, in Proc. IEEE Int. Conf. on Robotics and Automation (ICRA'97), pages 1940-1945, Albuquerque, USA,1997.

[18] Darius Burschka, Christof Eberst, Christian Robl, Georg Färber, „Interaction of Perception and Control for Indoor Exploration“, in „Robust Vision for Vision-Based Control of Motion“, Workshop at the ICRA '98, Leuven, Belgium, May 1998.

[19] D. Burschka, „Videobasierte Umgebungsexploration am Beispiel eines binokularen Stereo-Kamerasystems“, Dissertation am Lehrstuhl für Prozessrechner, TU München, 1998, in Fortschritt-Berichte VDI, Reihe 10, Nr. 573.

[20] T. Celinski, B. McCarragher, „Achieving Efficient Data Fusion Through Integration of Sensory Perception Control and Sensor Fusion“, Proceedings of the IEEE International Conference on Robotics & Automation ICRA ‘99, Detroit, USA, Mai 1999.

[21] R. Y. Chiang, M. G. Safonov, “Robust Control Toolbox”, The Mathworks Inc., 24 Prime Park Way, Natick, MA 01760-1500, 1996.

[22] M. Dekhil and T. C. Henderson, „Instrumented Logical Sensor Systems - Practice“, Proceedings of the IEEE International Conference on Robotics & Automation ICRA’98, Leuven, Belgium, 1998.

[23] E.-D. Dickmanns, „Vehicles Capable of Dynamic Vision“, In 15^{th} International Joint Conference on Artificial Intelligence IJCAI’97, Nagoya, Japan, 1997.

[24] G. Dissanayake, S. Sukkarieh, E. Nebot, H. Durrant-Whyte, „A New Algorithm for the Alignment of Inertial Measurement Units without External Observation for Land Vehicle Applications“, Proceedings of the IEEE International Conference on Robotics & Automation ICRA ‘99, Detroit, USA, Mai 1999.

[25] J. C. Doyle, K. Glover, P. P. Khargonekar, B. A. Francis, „State-Space Solutions to Standard H_2 and H_∞ Control Problems“, in IEEE Transactions on Automatic Control, Vol. 34, No. 8, Aug. 1989.

[26] G. Dudek, P. Freedman, I. M. Rekleitis, „Just-in-time sensing: efficiently combining sonar and laser range data for exploring unknown worlds“, in Proceeding of the IEEE International Conference on Robotics and Automation, Minneaplois, Minnesota, April 1996.

[27] H. F. Durrant-Whyte, „Sensor Models and Multisensor Integration“, in The International Journal of Robotics Research Vol. 7, No. 6, Dec. 1988.

[28] H. Elmqvist, S. E. Mattsson, M. Otter, "Modelica – A Language for Physical System Modeling, Visualization, and Interaction", ", in Proceeding of the 1999 IEEE International Symposium on Computer Aided Control System Design, Hawai'i, USA, August 1999.

[29] S. Emura, S. Tachi, „On Design of Sequential Sensor Fusion System", Proceedings of the IEEE International Conference on Robotics & Automation ICRA'98, Leuven, Belgium, 1998.

[30] G. Englberger, "Systemmodellorientierter Entwurf einer Regelung für eine Mikropositioniereinrichtung", Diplomarbeit am Lehrstuhl für Prozessrechner, TU München, Juli 1998.

[31] W. Entenmann, „Digitale Filter", Vorlesungsskript WS93, Lehrstuhl für Netzwerktheorie und Schaltungstechnik, TU München, 1993.

[32] N. J. Fliege, "Multirate Digital Signal Processing: Multirate Systems, Filter Banks, Wavelets", West Sussex, England: John Wiley & Sons, 1994.

[33] H. Fujioka, Y Yamamoto, S. Hara, „Sampled-Data Control Toolbox: A Software Package via Object-Oriented Programming", in Proceeding of the 1999 IEEE International Symposium on Computer Aided Control System Design, Hawai'i, USA, August 1999.

[34] R. E. Gibson, D. L. Hall, J. A. Stover,"An Autonomous Fuzzy Logic Architecture for Multisensor Data Fusion", in Proceedings of the IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems MFI'94, Las Vegas, USA,1994.

[35] M. Goldfarb, N. Celanovic, „Behavioral Implications of Piezoelectric Stack Actuators for Control of Micromanipulation", IEEE International Conference on Robotics and Automation ICRA'96, Minneapolis, USA, 1996.

[36] G. H. Golub, C. F. van Loan, "Matrix Computations", 3rd ed. Baltimore, MD: Johns Hopkins University Press, 1996.

[37] S. S. Goodridge, „Multimedia Sensor Fusion for Intelligent Camera Control and Human-Computer Interaction", PhD-Thesis, 1998.

[38] U. Hanebeck, J. Horn, „A New Estimator for Mixed Stochastic and Set Theoretic Uncertainty Models Applied to Mobile Robot Localization!, Proceedings of the IEEE International Conference on Robotics & Automation ICRA '99, Detroit, USA, Mai 1999.

[39] H. Hanselmann et al, "Production Quality Code Generation from SimuLink Block Diagrams",dSPACE GmbH, www.dspace.de, in Proceeding of the 1999 IEEE International Symposium on Computer Aided Control System Design, Hawai'i, USA, August 1999.

[40] H. R. Hashemipour, S. Roy, A. J. Laub, „Decentralized Structures for Parallel Kalman-Filtering", in IEEE Transactions on Automatic Control, Vol. 33, No. 1, Jan 1988.

[41] K. Hashimoto, T. Noritsugu, "Visual Servoing with Linearized Observer", in Proceedings of the IEEE International Conference on Robotics and Automation ICRA'99, Detroit, Mai 1999.

[42] A. Hauck, M. Sorg, G. Färber, T. Schenk, "What Can Be Learned from Human Reach-To-Grasp Movements for the Design of Robotic Hand-Eye Systems?", in Proceedings of the IEEE International Conference on Robotics and Automation ICRA'99, Detroit, Mai 1999.

[43] S. Haykin, "Adaptive Filter Theory", 3rd ed. Englewood Cliffs, NJ: Prentice Hall, 1996.

[44] J. Heinzl, U. Reiländer, „Mikropositioniereinrichtung", Offenlegungsschrift DE 197 42 205 A1 des Deutschen Patentamtes, 1997.

[45] M. Höhn, C. Robl, U. Reiländer, S. Blum, B. Schäfer, "Mikropositioniereinrichtung für die Präzisionsmontage", Abschlußbericht des Teilprojekts 5 FORMIKROSYS, Bayerische Forschungsstiftung, 1998.

[46] M. Höhn and Christian Robl, „Qualification of standard industrial robots for micro-assembly", in Proc. of the IEEE International Conference on Robotics and Automation (ICRA'99), Detroit, USA, May 1999.

[47] G. E. Hovland and B. J. McCarragher, „Dynamic Sensor Selection for Robotic Systems", Proceedings of the IEEE International Conference on Robotics & Automation ICRA'97, Albuquerque, New Mexico, 1997.

[48] J. Hu, M. Queiroz, T. Burg, D. Dawson, „Adaptive Position/Force Control of Robot Manipulators Without Velocity Measurements", in Proceedings of th IEEE International Conference on Robotics and Automation, 1995.

[49] IEEE, "Programs for Digital Signal Processing", IEEE-Press, New York: John Wiley & Sons, 1979, Algorithm 8.1.

[50] R. V. Jategaonkar, E. Plaetschke, „Algorithms for Aircraft Parameter Estimation Accounting Process and Measurement Noise", in Journal of Aircraft, Vol. 26, No. 1, Jan 1989.

[51] R. Jamal, H. Illig, „Die Leichtigkeit des Echtzeitprogrammierens", in Design & Elektronik Heft 2/99, Februar, 1999.

[52] M. Jun, M. G. Safonov, "Automatic PID Tuning: An Application of Unfalsified Control", in Proceeding of the 1999 IEEE International Symposium on Computer Aided Control System Design, Hawai'i, USA, August 1999.

[53] R.E. Kalman and J. E. Bertram, „A unified approach to the theory of sampling systems", in J. Franklin Inst., vol. 267, pp. 405-436, 1959.

[54] F. Kobayashi, F. Arai, T. Fukuda, „Sensor Selection by Reliability Based on Possibility Measure", Proceedings of the IEEE International Conference on Robotics & Automation ICRA '99, Detroit, USA, Mai 1999.

[55] K. Kobayashi, F. Munekata, K. Watanabe, „Accurate Navigation via Differential GPS and Vehicle Local Sensors", in Proceedings of the IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems MFI'94, Las Vegas, USA,1994.

[56] Schwerpunkt-Programm KONDISK der Deutschen-Forschungs-Gemeinschaft (DFG), www.ifra.ing.tu-bs.de/kondisk/bewliste.html

[57] J. Lee, I. Ha, „Sensor Fusion and Calibration for Motion Captures using Accelerometers“, Proceedings of the IEEE International Conference on Robotics & Automation ICRA ‘99, Detroit, USA, Mai 1999.

[58] R. Lenz, “Linsenfehlerkorrigierte Eichung von Halbleiterkameras mit Standardobjektiven für hochgenaue 3D-Messungen in Echtzeit”, Lehrstuhl für Nachrichtentechnik, TU München, Mustererkennung 1987, Informatik Fachberichte 149, Springer Verlag.

[59] L. Ljung, T. Glad, “Modeling of Dynamic Systems”, Prentice Hall, Englewood Cliffs, N.J., 1994.

[60] L.Ljung, “System Identification Toolbox”, 24 Prime Park Way, Natick, MA 01760-1500, 1997.

[61] R. C. Luo, M. Lin, R. S. Scherp, “Dynamic Multi-Sensor Data Fusion System for Intelligent Robots” in IEEE Journal of Robotics & Automation RA-4(4), 386-396, 1988.

[62] R. C. Luo, M. G. Kay, „Data Fusion and Sensor Integration: State-of-the-Art 1990s“, in Data Fusion in Robotic and Machine Intelligence, M. A. Abidi and R. C. Gonzalez, eds., Academic Press Inc., ISBN 0-12-042120-8, 1992.

[63] R. Madhavan, E. Nettleton, E. Nebot, G. Dissanayake, J. Cunningham, H. Durrant-Whyte, P. Corke, J. Roberts, „Evaluation of Internal Navigation Sensor Suites for Underground Mining Vehicle Navigation“, Proceedings of the IEEE International Conference on Robotics & Automation ICRA ‘99, Detroit, USA, Mai 1999.

[64] G. Magin, C. Robl, „A Single Processor Real-Time Edge-Line Extraction System for Feature Tracking“, In IAPR Workshop on Machine Vision Applications (IAPR MVA'96), Tokyo, Japan, 1996.

[65] J. Manigel, „Autonome Fahrzeugführung durch Rechnersehen“, Dissertation, Fakultät für Maschinenbau und Elektrotechnik, Technische Universität Braunschweig, Jan. 1993.

[66] J. Manyika, H. Durrant-Whyte, „Data Fusion and Sensor Mangaement: A Decentralized Information-Theoretic Approach“, Ellis Horwood Series in Electrical and Electronic Engineering, ISBN 0-13-303132-2, 1994.

[67] T. A. Martin, “Software Architectures for OSEK/VDX Applications Using MATRIXx TM and AutoCode TM”, Integrated Systems Inc., in Proceeding of the 1999 IEEE International Symposium on Computer Aided Control System Design, Hawai’i, USA, August 1999.

[68] The Mathworks Inc., “Control System Toolbox User’s Guide”, 24 Prime Park Way, Natick, MA 01760-1500, 1996.

[69] The Mathworks Inc., “DSP Blockset User’s Guide”, 24 Prime Park Way, Natick, MA 01760-1500, 1999.

[70] The Mathworks Inc.,”Nonlinear Control Design Block Set User’s Guide”, 24 Prime Park Way, Natick, MA 01760-1500, 1997.

[71] The Mathworks Inc., "Signal Processing Toolbox User's Guide", 24 Prime Park Way, Natick, MA 01760-1500, 1999.

[72] The Mathworks Inc., "SimuLink: Dynamic System Simulation for MatLab", 24 Prime Park Way, Natick, MA 01760-1500, 1996.

[73] The Mathworks Inc.,"SimuLink: Real-Time Workshop User's Guide", 24 Prime Park Way, Natick, MA 01760-1500, 1997.

[74] The Mathworks Inc.,"SimuLink: Writing S-Functions", Version3, ", 24 Prime Park Way, Natick, MA 01760-1500, 1998.

[75] P. S. Maybeck, „Stochastic Models, Estimation, and Control Volume 1", Mathematics in Science and Engineering, Volume 141, ISBN 0-12-480701-1, Academic Press, 1979.

[76] I. Mayergoyz, "New Preisach-type models of hysteresis and their experimental testing", Journal of Applied Physics, Vol. 67(9), 1990.

[77] K. Müller, „Entwurf robuster Regelungen", B. G. Teubner Stuttgart, ISBN 3-519-06173-2, 1996.

[78] J. Neira, J. Horn, J. D. Tardos, G. Schmidt, „Multisensor Mobile Robot Localization", in Proceeding of the IEEE International Conference on Robotics and Automation, Minneaplois, Minnesota, April 1996.

[79] Oetken, Parks, Schüßler, "New Results in the Design of Digital Interpolators", IEEE Trans. Acoust., Speech, Signal Processing, Vol. ASSP-23, 1975.

[80] OPAL-RT Technologies, www.opal-rt.ca

[81] A. V. Oppenheim, R. W. Schafer, "Discrete-Time Signal Processing", Englewood Cliffs, NJ: Prentice Hall, 1989.

[82] M. Papageorgiou, „Optimierung", Oldenbourg Verlag München Wien, ISBN 3-486-21799-2, 1992.

[83] R. Pintelon, J. Schoukens, „Real-Time Integration and Differentiation of Analog Signals by Means of Digital Filtering", IEEE Transactions on Instrumentation and Measurement, Vol. 39, No. 6, Dec. 1990

[84] P. Quast, M. K. Sain, B. F. Spencer and S. J. Dyke, „Microcomputer Implementation of Digital Control Strategies for Structural Response Reduction", in Microcomputers in Civil Engineering, Vol. 10, pp 13-25, 1995.

[85] G. Raju, H. Wang, „Sensor Data Fusion Using Pitman's Closeness Technique and Complete Linkage Algorithm", in Proceedings of the IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems MFI'94, Las Vegas, USA,1994.

[86] D. Raviv, E. W. Djaja, „Technique for Enhancing the Performance of Discretized Controllers", in IEEE Control Systems, Vol. 19, No. 3, Juni 1999.

[87] U. Reiländer, "A New Design Of A Three Axis Manipulator", Actuator '98.

[88] J. M. Richardson, K. A. Marsh, „Fusion of Multisensor Data", in The International Journal of Robotics Research, Vol. 7, No. 6, Dec. 1988.

[89] C. Robl, "Echtzeitfähige Merkmalsextraktion aus Videobildfolgen für Online-Experimente mit einem autonomen mobilen Roboter", Diplomarbeit am Lehrstuhl für Prozessrechner, TU München, Juni 1995.

[90] C. Robl, S. Petters, B. Schäfer, U. Reiländer, A. Widl, „Micro Positioning System with 3dof for a Dynamic Compensation of Standard Robots", in Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS'97), pages 1105-1110, Grenoble, France, September 1997.

[91] Christian Robl, „Contour tracer for a fast and precise edge-line extraction", In Proc. of the IAPR Workshop on Machine Vision Applications (IAPR MVA'98), Makuhari, Chiba, Japan, November 1998.

[92] C. Robl, G. Englberger, G. Färber, „H_2-Control with acceleration feedback for a micro positioning system", In Proc. of the IEEE International Conference on Control Applications (CCA'99), Hawai'i, USA, August, 1999.

[93] C. Robl, G. Färber, "System Architecture for Synchronizing, Signal Level Fusing, Simulating and Implementing Sensors", In Proc. of the IEEE International Conference on Robotics And Automation ICRA2000, San Francisco, USA, April, 2000.

[94] F. Salam, N. Xi, „Integrated Photo and Acceleration Sensing Module for Robot Planning and Control", Proceedings of the IEEE International Conference on Robotics & Automation ICRA '99, Detroit, USA, Mai 1999.

[95] J. Z. Sasiadek, Q. Wang, „Sensor Fusion Based on Fuzzy Kalman-Filtering for Autonomous Robot Vehicle", Proceedings of the IEEE International Conference on Robotics & Automation ICRA '99, Detroit, USA, Mai 1999.

[96] F. R. Schell, „Bordautonomer automatischer Landeanflug aufgrund bildhafter und inertialer Messdatenauswertung", Dissertation, Universität der Bundeswehr München, Fakultät für Luft- und Raunfahrttechnik, Institut für Systemdynamik und Flugmechanik, 1992.

[97] G. Schmidt, „Grundlagen der Regelungstechnik", Springer Verlag, 1984.

[98] T. Schmidt, „Untersuchung der Einsatzmöglichkeiten eines mikrostrukturierten Beschleunigungssensorsystems zur Schwingungskompensation eines Industrieroboters", Diplomarbeit am Lehrstuhl für Prozessrechner, TU München, 1997.

[99] E. Schrüfer, "Zuverlässigkeit von Mess- und Automatisierungseinrichtungen", Carl Hanser Verlag München Wien, ISBN 3-446-14190-1, 1984.

[100] E. Schrüfer, „Signalverarbeitung", ISBN 3-446-16563-0, Carl Hanser Verlag München Wien, 1992.

[101] U. Shaked, C. E. de Souza, "H∞ Control of Linear Systems with Delayed Measurements", in Proceeding of the 1999 IEEE International Symposium on Computer Aided Control System Design, Hawai'i, USA, August 1999.

[102] V. P. Sizov, „Decomposition of an extended Kalman-Filter", in Radioelectronics and Communication Systems, Bd. 31, H. 12, 1988.

[103] M. H. Smith, M. Elbs, "Towards a More Efficient Approach to Automotive Embedded Control System Development", in Proceeding of the 1999 IEEE Interna-

tional Symposium on Computer Aided Control System Design, Hawai'i, USA, August 1999.

[104] R. Smith, A. Frost, P. Probert, „Aspects of Heading Determination via Fusion of Inclinometer and Magnetometer Data", Proceedings of the ICAR'97, Monterey, Canada, 1997.

[105] J. Suhardjo, B. F. Spencer Jr., A. Kareem, „Frequency Domain Optimal Control of Wind-Excited Buildings", in Journal of Engineering Mechanics, Vol. 118, No. 9, Sept. 1992.

[106] M.-W. L. Thein, T. Rendon, E. A. Misawa, "A Genetic Algorithm for the Tuning of a Discrete Adaptive Observer Implemented on an IBM Head/Disk Assembly" in Proceedings of the 1999 IEEE International Conference on Control Applications, Hawai'i, USA, August 1999.

[107] S. Theodoridis, M. G. Bellanger, "Adaptive Filters and Acoustic Control", in IEEE Signal Processing Magazine, Vol. 16, No. 4, Juli 1999.

[108] S. Toeppe, D. Bostic, S. Ranville, K. Rzemien, "Automatic Code Generation Requirements for Production Automotive Powertrain Applications", in Proceeding of the 1999 IEEE International Symposium on Computer Aided Control System Design, Hawai'i, USA, August 1999.

[109] K. Umeda, K. Ikushima, T. Arai, „Fusion of range image and intensity image for 3D shape recognition", in Proceeding of the IEEE International Conference on Robotics and Automation, Minneaplois, Minnesota, April 1996.

[110] J. Vaganay, M. J. Aldon, „Attitude Estimation for a Vehicle Using Inertial Sensors", in IFAC Intelligent Autonomous Vehicles, Southampton, UK, 1993.

[111] O. Vainio,"Adaptive Derivative Estimation for DSP-Based Acceleration Measurement", Proceedings of the IEEE International Conference on Robotics & Automation ICRA'98, Leuven, Belgium, 1998.

[112] M. Walter, "Videobasierte Lokalisation eines autonomen mobilen Roboters mit einem Kalman-Filter", Diplomarbeit am Lehrstuhl für Prozessrechner, TU München, Juli 1996.

[113] S. Werner, S. Fürst, D. Dickmanns, E.-D. Dickmanns, „A vision-based multisensor machine perception system for autonomous aircraft landing approach", In Enhanced and Synthetic Vision AeroSense'96, Orlando, Florida, 1996.

[114] O. Wollersheim, „Zur Bestimmung der statischen Kenngrößen von Beschleunigungssensoren im Winkelmesstisch", Bericht, Institut für Mikrostrukturtechnik IMT, Forschungszentrum Karlsruhe, 1996.

[115] S. Wüstling, "Hochintegriertes triaxiales Beschleunigungssensorsystem", Dissertation, Uni Karlsruhe, 1997, in Forschungszentrum Karlsruhe: Wissenschaftliche Berichte FZKA6003.

[116] L. Xu, M. Yamada, O. Saito, "Development of nD Control System Toolbox for Use with MatLab", ", in Proceeding of the 1999 IEEE International Conference on Control Applications, Hawai'i, USA, August 1999.

[117] Y. Yamamoto, P. P. Khargonekar, „Frequency Response of Sampled-Data Systems“, in IEEE Transactions on Automatic Control, Vol. 41, No.2, Februar 1996.

[118] Y. Yokokohji, Y. Sugawara, T. Yoshikawa, „Accurate Image Overlay on Head-Mounted Displays Using Vision and Accelerometers“, Proceedings of the IEEE International Conference on Robotics & Automation ICRA ‘99, Detroit, USA, Mai 1999.

[119] M. Zeytinoglu, M. Mintz, “Optimal Fixed Size Confidence Procedures for a Restricted Parameter Space”, Ann. Statistics 12(3), 945-957, 1984.

[120] Y. Zhou, B. J. Nelson, B. Vikramaditya, „Fusing Force and Vision Feedback for Micromanipulation“, Proceedings of the IEEE International Conference on Robotics & Automation ICRA’98, Leuven, Belgium, 1998.

Anhang A: Blockbeschreibung

Derivatives Library Blocks:

Blockname	Parameter	Beschreibung
Discrete time derivative		Diskrete 1.Ableitung
	Sample time	Abtastrate
Adaptive Derivative Estimation with Kalman		Adaptive Schätzung der 1. Ableitung mit einem adaptiven Kalman-Filter
	FIR filter Length	FIR-Filterordnung
	Measurement noise variance	Varianz des Messrauschens
	Process noise variance	Varianz des Prozessrauschens
	Initial value of filter taps	Initiale Filterkoeffizienten
	Initial error correlation matrix	Initiale Fehlerkorrelationsmatrix
	Sample time	Abtastrate
Adaptive Derivative Estimation with nLMS		Adaptive Schätzung der 1. Ableitung mittels der Methode der kleinsten Fehlerquadrate.
	FIR filter length	FIR-Filterordnung
	Step size	Schrittweite
	Initial value of filter taps	Initiale Filterkoeffizienten
	Sample time	Abtastrate
Adaptive Derivative Estimation with RLS		Adaptive Schätzung der 1. Ableitung mittels einer exponentiell gewichteten rekursiven Schätzung mit der Methode der kleinsten Fehlerquadrate
	FIR filter length	FIR-Filterordnung
	Memory weighting factor	Alterungsfaktor (zwischen 0 und 1)
	Initial value of filter taps	Initiale Filterkoeffizienten
	Initial input variance estimate	Initiale Schätzung der Varianz des Eingangs
	Sample time	Abtastrate

Hardware Library Block

Blockname	Parameter	Beschreibung
Quancom-DAC 12-Bit		Gerätetreiber für Quancom PCI DAC
	Base I/O Address	I/O Adresse der PCI-Karte
	Number of channels	Anzahl der DAC-Kanäle
	Gain	Verstärkungsfaktor

	Sample time	Abtastrate
Quancom-ADC 16-Bit		Gerätetreiber für Quancom PCI ADC
	Base I/O Address	I/O Adresse der PCI-Karte
	Bipolar	Bipolarer Betrieb (0=nein, 1=ja)
	Number of channels	Anzahl der ADC-Kanäle
	Gain	Vertärkungsfaktor
	Sample time	Abtastrate
DSP-DAC 12-Bit		Gerätetreiber für Sheldon DSP-PCI-DAC
	Base I/O Address	I/O Adresse der DSP-Karte
	Number of channels	Anzahl der DAC-Kanäle
	Gain	Verstärkungsfaktor
	Sample time	Abtatsrate
DSP-ADC 12-Bit		Gerätetreiber für Sheldon DSP-PCI-ADC
	Base Address M	I/O Adresse der Mailboxregister
	Base Address B	I/O Adresse der DSP-Karte
	Number of channels	Anzahl der ADC-Kanäle
	Gain	Verstärkungsfaktor
	Sample time	Abtastrate
DSP Sigma Delta Beschleunigungs-Sensor		Gerätetreiber zum Host initiierten Einlesen der Beschleunigungssensordaten
	Base Address M	I/O Adresse der Mailboxregister
	Base Address B	I/O Adresse der DSP-Karte
	Offset	Offset
	Gain	Verstärkungsfaktor
	Sample time	Abtastrate
Modus		Block zum Umschalten zwischen Simulation und Codegenerierung (RTW)
	Modus	Simulation bzw. RTW
Interrupt Control		Zuordung der PCI-Interrupts
	PCI-Interrupt number	PCI-Interrupt
	PCI-Interrupt vector offset	Vektoroffset
	Preemption flag	0=nicht preemptiv, 1=preemptiv
FZK-Sensor Interrupt		Interrupt getriggertes Einlesen der Beschleunigungssensordaten
	Base Address M	I/O Adresse der Mailboxregister
	Base Address B	I/O Adresse der DSP-Karte
	Offset	Offset
	Gain	Verstärkungsfaktor
	Sample time	Abtastrate muss –1 (Vererbung) sein

FZK-Sensor Interrupt Interner time stamp	 Base Address M Base Address B Offset Gain Sample time	Interrupt getriggertes Einlesen der Beschleunigungssensordaten, mit internem Zeitstempel I/O Adresse der Mailboxregister I/O Adresse der DSP-Karte Offset Verstärkungsfaktor Abtastrate muss –1 (Vererbung) sein
RS232 Interrupt	 Baudrate Sample time	RS232-Gerätetreiber (per Interrupt) Baudrate Abtastrate muss –1 (Vererbung) sein

Integrators Library Blocks

Blockname	Parameter	Beschreibung
Integrator mit Polverschiebung	 Inegrator pole Sample time	Integratornäherung mit Polverschiebungsverfahren Integratorpol (<1) Abtastrate
PID feedback integrator	 Proportional Integral Derivative Sample time	Integrationsnäherung mit PID-Feedback Proportionaler Anteil Integraler Anteil Ableitungsanteil Abtastrate
Integral feedback integrator	 Integral gain Sample time	Integrationsnäherung mit I+ Feedback Rückführungskoeffizient Abtastrate
Discrete time integrator	 Initial condition Sample time	Zeitdiskrete Interation mit Rechteckregel Anfangsbedingung Abtastrate
Integrator mit harter Referenz	 Sample time	Integration mit direkter Referenzierung Abtastrate
Doppelintegration mit Referenz	 Sample time	Zweifache Integration mit Korrektur der unbekannten Anfangszustände Abtastrate
Doppelintegration mit Referenz und error forward Kalman-Filter		Zweifache Integration mit einer niedriger abgetasteten Referenz in einer Fehler-Feedforward Kalman-Filterstruktur Kovarianz der Referenz

	Kovarianz der Referenz Kovarianz der Sensordaten Sample time	Kovarianz der Sensordaten Abtastrate
Doppelintegration mit Referenz und error feed-back Kalman-Filter	 Fehlerkovarianz Sample time	Zweifache Integration mit einer niedriger abgetasteten Referenz in einer Fehler-Feedback Kalman-Filterstruktur Kovarianz der Fehlerzustände Abtastrate
Doppelintegration mit Referenz und Polverschie-bung	 Integrator pole Sample time	Zweifache Integration mit polverschobenen Integratoren und mit Korrektur der unbekannten Anfangszustände Integratorpol Abtastrate

Misc Library Blocks

Blockname	Parameter	Beschreibung
Time invariant Kalman filter	 System matrix A Control matrix B Process noise matrix D Measurement matrix C Measurement noise covariance Z Process noise covariance W Sample time	Zeitinvarianter zeitdiskreter Kalman-Filter Systemmatrix A Steuermatrix B Störeinkopplungsmatrix D Messwertmatrix C Kovarianz des Messrauschens Z Kovarianz des Prozessrauschens W Abtastrate
Time variant Kalman filter	 Sample time	Zeitvarianter zeitdiskreter Kalman-Filter Abtastrate
KF measurement synchro-nization	 Sample time	Zusatzblock zum Kalman-Filter zur Synchroni-sation der Messwerte Abtastrate
Deadtime compensator for time invariant KF	 System matrix A Measurement matrix C Process noise covariance W Sample time	Korrektur des Kalman-Filter Schätzfehlers aufgrund der Totzeit der Messwerte Systemmatrix A Messwertmatrix C Kovarianz des Prozessrauschens W Abtastrate
Discrete PID-controller	 Proportional	Zeitdiskreter PID-Regler Proportionaler Anteil

	Integral Derivative Sample time	Integraler Anteil Ableitungsanteil Abtastrate
mm to Piezo-Voltage		Umwandlung der mm-Auslenkung in die vier Piezospannungen der MPE-Aktorik
Aktoransteuerung		Aktoransteuerung der MPE mit mm/U Wandlung und Stopp bei Überspannung (|U|>5V)
Anfangswertaufschaltung		Bestimmung der Anfangsposition der MPE als Anfangsbedingung
Kalman-Filter	 Kalman-Filter Ausgangsmatrix C	Speziell auf die MPE-Reglung zugeschnittener Beobachter auf Kalman-Filter Basis Kalman-Filter-LTI-Objekt Ausgangsmatrix C

Offset Library Blocks

Blockname	Parameter	Beschreibung
Floating offset correction	 Number of elements Sample time	Offsetkorrektur durch die DFT von 0Hz Fenstergröße Abtastrate
Sukzessive offset correction	 Sample time	Sukzessive Offsetbestimmung und –korrektur Abtastrate
Offsetbeobachter	 Plant Oberver gain Sample time	Offsetbeobachter LTI-Objekt der Strecke Beobachterverstärker Abtastrate

Sensors Library Blocks

Blockname	Parameter	Beschreibung
Sensor typ1	 Sensor-LTI-system Drift value Drift value variance Drift Time constant Measurement noise mean Measurement noise variance Lower limit Upper limit	Zyklischer zeitkontinuierlicher Sensor Übertragungsverhalten des Sensors Driftamplitude Varianz der Driftamplitude Driftzeitkonstante Mittelwert des Messrauschens Varianz des Messrauschens Untere Messbereichsgrenze Obere Messbereichsgrenze

	Bit quantization	Bitquantisierung
	Bias	Offset
	Bias variance	Varianz des Offsets
	Time delay	Totzeit
	Control sample time	Abtastrate des Regelsystems
Sensor typ2		Zyklischer zeitdiskreter Sensor
	Sensor-LTI-system	Übertragungsverhalten des Sensors
	Drift value	Driftamplitude
	Drift value variance	Varianz der Driftamplitude
	Drift Time constant	Driftzeitkonstante
	Measurement noise mean	Mittelwert des Messrauschens
	Measurement noise variance	Varianz des Messrauschens
	Lower limit	Untere Messbereichsgrenze
	Upper limit	Obere Messbereichsgrenze
	Bit quantization	Bitquantisierung
	Bias	Offset
	Bias variance	Varianz des Offsets
	Time delay	Totzeit
	Sample time	Abtastrate
Sensor typ3		Asynchroner zeitdiskreter Sensor
	LTI-System	Übertragungsverhalten des Sensors
	Fixed delay	Totzeit
	Random delay mean	Mittelwert der zufälligen Totzeit
	Random delay variance	Varianz der zufälligen Totzeit
	Minimal delay	Minimale zufällige Totzeit (>0)
	Maximal delay	Maximale zufällige Totzeit
	Measurement noise mean	Mittelwert des Messrauschens
	Measurement noise variance	Varianz des Messrauschens
	Bit quantization	Bitquantisierung
	Noise exponential translation	Verschiebung der exp. Rauschabhängigkeit
	Noise exponential gain	Verstärkung der exp. Rauschabhängigkeit
	Noise linear gain	Verstärkung der lin. Rauschabhängigkeit
	Control sample time	Abtastrate des Regelsystems
Sensor typ4a		Trigger (Asynchroner kontinuierlicher Sensor)
	Trigger time	Triggerzeitpunkt
	Trigger time variance	Varianz des Triggerzeitpunkts
	Control sample rate	Abtastrate des Regelsystems
Sensor typ4b		Event (Asynchroner kontinuierlicher Sensor)
	Events per second	Ereignisse pro Sekunde

	Gain Control sample rate	Verstärkungsfaktor Abtastrate des Regelsystems

Synchronization Library Blocks

Blockname	Parameter	Beschreibung
First order extrapolation		Lineare Extrapolation
	Sample rate out	Ausgehende Abtastrate
	Sample rate in	Eingehende Abtastrate
Third order extrapolation		Kubische Extrapolation
	Sample rate out	Ausgehende Abtastrate
	Sample rate in	Eingehende Abtastrate
Third order linear regression		Linear Extrapolation mit Polynom 3. Grades
	Sample rate out	Ausgehende Abtastrate
	Sample rate in	Eingehende Abtastrate
Linear prediction filter		Linearer Prädiktionsfilter
	Sample rate out	Ausgehende Abtastrate
	Sample rate in	Eingehende Abtastrate
Synchronization time invariant KF		Synchronisation mit zeitinvariantem Kalman-Filter
	System matrix A	Systemmatrix A
	Control matrix B	Steuermatrix B
	Process noise matrix D	Störeinkopplungsmatrix D
	Measurement matrix C	Messwertmatrix C
	Measurement noise covariance Z	Kovarianz des Messrauschens Z
	Process noise covariance W	Kovarianz des Prozessrauschens W
	Sample time out	Ausgehende Abtastrate
Synchronization time variant KF		Synchronisation mit zeitvariantem Kalman-Filter
	Sample time out	Ausgehende Abtastrate
Synchronization time invariant KF measurement synch		Synchronisation mit zeitinvariantem Kalman-Filter inclusive Messwerte-Synchronisation
	System matrix A	Systemmatrix A
	Control matrix B	Steuermatrix B
	Process noise matrix D	Störeinkopplungsmatrix D
	Measurement matrix C	Messwertmatrix C
	Measurement noise covariance Z	Kovarianz des Messrauschens Z
	Process noise covariance W	Kovarianz des Prozessrauschens W

	Sample time out	Ausgehende Abtastrate
Synchronization with measurement synch and dead time compensation with Kalman filter		Synchronisation mit zeitinvariantem Kalman-Filter inclusive Messwerte-Synchronisation und Korrektur des Schätzfehlers aufgrund der Totzeit der Messwerte
		Systemmatrix A
	System matrix A	Steuermatrix B
	Control matrix B	Störeinkopplungsmatrix D
	Process noise matrix D	Messwertmatrix C
	Measurement matrix C	Kovarianz des Messrauschens Z
	Measurement noise covariance Z	Kovarianz des Prozessrauschens W
	Process noise covariance W	Ausgehende Abtastrate
	Sample time out	
Observer based synchronization		Synchronisation mit einem Beobachter
	Plant	
	Gain	LTI-Objekt der Strecke
	Sample time out	Beobachterverstärkung
		Ausgehende Abtastrate
First order extrapolation Int1 / 2		Lineare Extrapolation, für Interrupt-Service Routinen gesplittet
	Number of ports	Anzahl der Verbindungen zwischen beiden Blöcken
		Ausgehende Abtastrate
	Sample rate out	Eingehende Abtastrate
	Sample rate in	

Anhang B: Modusumschaltung im Blockdiagramm

Die Implementierung des Modus-Blocks ist mit den beiden MatLab m-Files rtmodify.m und handle_block.m realisiert.

rtmodify.m

```
function rtmodify(mode)
%function rtmodify(mode)

sys=gcs
nsys=strcat(sys,'_Temp');
cr=sprintf('\n');

if mode==1
   new_system(nsys);
   open_system(nsys);
   hw=find_system(sys,'description','HWout');
   sb=char(get_param(hw,'name'));
   add_block(strcat(sys,'/',sb),strcat(nsys,'/',sb));
   handle_block(sb,sys,nsys,1);
   save_system(nsys);
   close_system(nsys);
   set_param(sys,'Solver','FixedStepDiscrete');
end

q=find_system(sys,'Description','HWin');
if ~isempty(q)
   for qq=1:length(q)
   qc=char(q(qq))
   switch qc
       case {[sys,'/Quancom-ADC 12-Bit']}
      disp('Quancom-ADC 12-Bit');
      rpb(qc,'Inport','Ground',mode);
       case {[sys,'/DSP-ADC 12-Bit']}
         disp('DSP-ADC 12-Bit');
      rpb(qc,'Inport','Ground',mode);
```

```
        case {[sys,'/DSP-Sigma-Delta',cr,'Beschleunigungs-Sensor']}
          disp(['DSP-Sigma-Delta',cr,'Beschleunigungs-Sensor']);
      rpb(qc,'Inport','Ground',mode);
        case {[sys,'/FZK-Sensor-Interrupt']}
          disp('FZK-Sensor-Interrupt');
      rpb(qc,'Inport','Ground',mode);
      if mode==1
         pp=e2t(sys,['FZK-Sensor-Interrupt']);
      end
      rpb(qc,'EnablePort','TriggerPort',mode);
      if mode==1
         set_param([qc,'/Enable'],'TriggerType','function-call');
         add_line(sys,pp);
      end
        case {[sys,'/FZK-Sensor-Interrupt',cr,'interner time stamp']}
          disp('FZK-Sensor-Interrupt');
      rpb(qc,'Inport','Ground',mode);
      if mode==1
         pp=e2t(sys,['FZK-Sensor-Interrupt',cr,'interner time stamp']);
      end
      rpb(qc,'EnablePort','TriggerPort',mode);
      if mode==1
         set_param([qc,'/Enable'],'TriggerType','function-call');
         add_line(sys,pp);
      end
            case {[sys,'/RS232-Interrupt']}
                disp('RS232-Interrupt');
                rpb(qc,'Inport','Ground',mode);
      if mode==1
         pp=e2t(sys,['RS232-Interrupt']);
      end
                rpb(qc,'EnablePort','TriggerPort',mode);
      if mode==1
         set_param([qc,'/Enable'],'TriggerType','function-call');
         add_line(sys,pp);
      end
        otherwise
          disp('Neuer HWin-Block gefunden');
   end
   end;
end;
```

```
q=find_system(sys,'Description','HWout');
if ~isempty(q)
   for qq=1:length(q)
        qc=char(q(qq));
        switch qc
            case {[sys,'/Quancom-DAC 12-Bit']}
                disp('Quancom-DAC 12-Bit');
      rpb(qc,'Outport','Terminator',mode);
            case {[sys,'/DSP-DAC 12-Bit']}
                disp('DSP-DAC 12-Bit');
       rpb(qc,'Outport','Terminator',mode);
            otherwise
                disp('Neuer HWout-Block gefunden');
        end;
   end;
end

q=find_system(sys,'MaskType','Rate Transition');
   if ~isempty(q)
      for k=1:length(q)
         if mode==1
            set_param([char(q(k)),'/In'],'Tag','AsyncRateTransition')
            set_param([char(q(k)),'/Out'],'Tag','AsyncRateTransition')
         else
            set_param([char(q(k)),'/In'],'Tag','')
            set_param([char(q(k)),'/Out'],'Tag','')
         end
      end
   end


q=find_system(sys,'FollowLinks','on','Name',['mm to ',cr,'Piezo Voltage']);
   if ~isempty(q)
      set_param(char(q),'Parameters',num2str(mode));
   end

if mode==0
   open_system(nsys);
   hw=find_system(nsys,'description','HWout');
   sb=char(get_param(hw,'name'));
   handle_block(sb,nsys,sys,0);
   close_system(nsys,0);
```

```
   set_param(sys,'Solver','ode45');
end

save_system(sys)
-----------------------------------------------------

function rpb(system,src,dst,mode)

if mode==1
   replace_block(system,src,dst,'noprompt')
else
   replace_block(system,dst,src,'noprompt')
end
-----------------------------------------------------

function points=e2t(system,block)

ll=get_param(system,'Lines');
for q=1:length(ll)
   if strcmp(get_param(ll(q).DstBlock,'Name'),block)==1
      if strcmp(ll(q).DstPort,'enable')==1
         delete_line(system,ll(q).Points(1,:));
         points=ll(q).Points;
         break
      end
   end
end
```

handle_block.m

```
function handle_block(block,sys,systemp,mode)
%function handle_block(block,sys,systemp,mode)

blockname=strcat(sys,'/',block);
ports=get_param(blockname,'ports');
for i=1:ports(2)
    lines=get_param(sys,'Lines');
    for j=1:length(lines)
        if strcmp(get_param(lines(j).SrcBlock,'name'),block)
            db=get_param(lines(j).DstBlock,'name');
            dbport=lines(j).DstPort;
            blockport=lines(j).SrcPort;
            ad=0;
       if isempty(find_system(systemp,'Name',db))
                add_block(strcat(sys,'/',db),strcat(systemp,'/',db));
      ad=1;
            end

            if mode==1

add_line(systemp,strcat(block,'/',num2str(blockport)),strcat(db,'/',num2str
(dbport)));

   delete_line(sys,strcat(block,'/',num2str(blockport)),strcat(db,'/',num2s
tr(dbport)));
            else
                tlines=get_param(systemp,'Lines');
                qq=0;
                for k=1:length(tlines)
                    if strcmp(get_param(tlines(k).SrcBlock,'name'),block)
                     if tlines(k).SrcPort==blockport
                        qq=1;
                        break;
                     end
                  end
                end
                if qq==0

add_line(systemp,strcat(block,'/',num2str(blockport)),strcat(db,'/',num2str
(dbport)));
```

```
delete_line(sys,strcat(block,'/',num2str(blockport)),strcat(db,'/',num2str(
dbport)));
                end
            end
            if
strcmp(get_param(strcat(sys,'/',db),'description'),'HWin')==0
                if ad==1
         handle_block(db,sys,systemp,mode);
         delete_block(strcat(sys,'/',db));
      end
            end
            break
        end
    end
end
if strcmp(get_param(blockname,'description'),'HWout')==0
for i=1:ports(1)
    lines=get_param(sys,'Lines');
    for j=1:length(lines)
        if strcmp(get_param(lines(j).DstBlock,'name'),block)
            db=get_param(lines(j).SrcBlock,'name');
            dbport=lines(j).SrcPort;
            blockport=lines(j).DstPort;
       ad=0;
            if isempty(find_system(systemp,'Name',db))
                add_block(strcat(sys,'/',db),strcat(systemp,'/',db));
      ad=1;
            end
            if mode==1

   add_line(systemp,strcat(db,'/',num2str(dbport)),strcat(block,'/',num2str
(blockport)));

   delete_line(sys,strcat(db,'/',num2str(dbport)),strcat(block,'/',num2str(
blockport)));
            else
                tlines=get_param(systemp,'Lines');
                qq=0;
                for k=1:length(tlines)
                    if strcmp(get_param(tlines(k).SrcBlock,'name'),db)
                     if tlines(k).SrcPort==dbport
                        qq=1
                        break
                     end
```

```
                  end
                end
                if qq==0

add_line(systemp,strcat(db,'/',num2str(dbport)),strcat(block,'/',num2str(bl
ockport)));

delete_line(sys,strcat(db,'/',num2str(dbport)),strcat(block,'/',num2str(blo
ckport)));
                end


            end
            if
strcmp(get_param(strcat(sys,'/',db),'description'),'HWout')==0
         if ad==1
                  handle_block(db,sys,systemp,mode);
                  delete_block(strcat(sys,'/',db));
                end
            end
            break
        end
    end
end
end
```

Anhang C: Gerätetreiber S-Function

Das folgende C-File in S-Function-Struktur stellt beispielhaft den Gerätetreiber für einen PCI-D/A Converter dar.

qu_da12.c

```
/* qu_da12 - Device driver for the Quancom 12-Bit analog output card.
*/

static char copyright_rti[] =
"(c) Copyright, C. Robl / LPR-1997.  All rights reserved.";

#define S_FUNCTION_NAME qu_da12

#include <stdio.h>   /* so mex.h doesn't screw up! */
#include <stdlib.h>  /* for strtoul() */

#include "simstruc.h" /* Where simulation structure, S, is defined */

#ifdef MATLAB_MEX_FILE
#include "mex.h"
#else
#include <sysLib.h>
#include <math.h>
#endif

/* Input Arguments */
#define BASE_ADDRESS_ARG   ssGetArg(S,0)
#define OUTPUT_RANGE_ARG   ssGetArg(S,1)
#define NUMBER_OF_CHANNELS_ARG  ssGetArg(S,2)
#define SAMPLE_TIME_ARG         ssGetArg(S,3)
#define NUMBER_OF_ARGS     (4)

#define NSAMPLE_TIMES      (1)

/* Storage Allocation */
#define INSTANCE_INFO      (0)
#define NUMBER_OF_PWORKS   (1)

/* hardware registers */
```

```
typedef int ByteRegister;

/* This abstract data type encapsulates one instance of this driver. */
typedef struct D2ADataTag {
    ByteRegister dataOutputRegister;
    int numChannels;
} D2ADataStorage, *D2AData;

#define DATAREGOFFSET (0x60)
#define minimum(x, y)   ((x) < (y) ? (x) : (y))
#define maximum(x, y)   ((x) > (y) ? (x) : (y))
#define DAC_MIN   (0)
#define DAC_MAX   (4095)
#define OFFSET 2048

static void mdlInitializeSizes(S)
    SimStruct *S;
{
    int noc;
    if (ssGetNumArgs(S) == NUMBER_OF_ARGS) {
   /* Set-up size information */
   ssSetNumContStates(S, 0);
   ssSetNumDiscStates(S, 0);
   noc=mxGetPr(NUMBER_OF_CHANNELS_ARG)[0];
   ssSetNumInputs(S, noc);
#ifdef MATLAB_MEX_FILE
   ssSetNumOutputs(S, noc);
#else
   ssSetNumOutputs(S, 0);
#endif
   ssSetDirectFeedThrough(S, 1);  /* Direct dependency on inputs */
   ssSetNumInputArgs(S, NUMBER_OF_ARGS);
   ssSetNumSampleTimes(S, NSAMPLE_TIMES);
   ssSetNumPWork(S, NUMBER_OF_PWORKS);  /* A/D board base address */
    } else {
#ifdef MATLAB_MEX_FILE
   mexErrMsgTxt("Wrong number of input arguments passed.\nFour arguments
are expected.\n");
#endif
    }
}

/* Function to initialize sample times */
```

```
static void mdlInitializeSampleTimes(S)
    SimStruct *S;
{
    ssSetSampleTimeEvent(S, 0, mxGetPr(SAMPLE_TIME_ARG)[0]);
    ssSetOffsetTimeEvent(S, 0, 0);
}

static void mdlInitializeConditions(x0, S)
    double *x0;
    SimStruct *S;
{
    int base_addr_str_len = 128;
    char base_addr_str[128];
    unsigned long base_addr;
    int output_range;
    D2AData d2a;
    int i;

    d2a = (D2AData) malloc(sizeof(D2ADataStorage));
    ssSetPWorkValue(S, INSTANCE_INFO, d2a);

    mxGetString(BASE_ADDRESS_ARG, base_addr_str, base_addr_str_len);
    sscanf(base_addr_str, "%lx", &base_addr);

    d2a->numChannels = minimum(4,
(int)mxGetPr(NUMBER_OF_CHANNELS_ARG)[0]);;

#ifndef MATLAB_MEX_FILE
    /* Initialize the board. */
    d2a->dataOutputRegister = (ByteRegister) (base_addr + DATAREGOFFSET);
    for (i = (d2a->numChannels-1); i >= 0 ; i--) {
   sysOutByte(d2a->dataOutputRegister+(i*0x08),OFFSET&255);
   sysOutByte(d2a->dataOutputRegister+(i*0x08)+0x04,OFFSET>>8);
    }
    sysInByte(d2a->dataOutputRegister);
#endif
}

/* Function to compute outputs */
static void mdlOutputs(y, x, u, S, tid)
    double * y;
    double * x;
    double * u;
```

```
    SimStruct * S;
    int tid;
{
    D2AData d2a = ssGetPWorkValue(S, INSTANCE_INFO);
    register int i;
    int data,data1;
    volatile int *baseadr=d2a->dataOutputRegister;

#ifdef MATLAB_MEX_FILE
    /* output=input for simulation */
    for (i = 0; i < d2a->numChannels; i++) {
       y[i]=u[i];
    }
#else
    for (i = 0; i < d2a->numChannels; i++) {
   /* Clip data before writing it to DAC */
   data=round(u[i]*204.8+OFFSET);
   data1=(minimum(maximum(data,DAC_MIN),DAC_MAX));
/* sysOutByte(d2a->dataOutputRegister+(i*0x08),data&255);
   sysOutByte(d2a->dataOutputRegister+(i*0x08)+0x04,data>>8);*/
   *(baseadr+i)=data;
    }
    sysInByte(d2a->dataOutputRegister);
#endif
}

/* Function to perform model update */
static void mdlUpdate(x, u, S, tid)
    double *x, *u;
    SimStruct *S;
    int tid;
{}

/* Function to compute derivatives */
static void mdlDerivatives(dx, x, u, S, tid)
    double *dx, *x, *u;
    SimStruct *S;
    int tid;
{}

/* Function to perform housekeeping at the end of a simulation run */
static void mdlTerminate(S)
```

```
    SimStruct *S;
{
register int i;
D2AData d2a = ssGetPWorkValue(S, INSTANCE_INFO);
#ifndef MATLAB_MEX_FILE

    for (i = (d2a->numChannels-1); i >= 0 ; i--) {
            sysOutByte(d2a->dataOutputRegister+(i*0x08),OFFSET&255);
       sysOutByte(d2a->dataOutputRegister+(i*0x08)+0x04,OFFSET>>8);
      }
   sysInByte(d2a->dataOutputRegister);
#endif
}

#ifdef MATLAB_MEX_FILE  /* Is this file being compiled as a MEX-file? */
#include "simulink.c"   /* MEX-File interface mechanism */
#else
#include "cg_sfun.h" /* Code generation registration function */
#endif
```

Anhang D: Gerätetreiber als TLC-File

Der Gerätetreiber für das Einlesen der Beschleunigungssensordaten ist zur Codegenerierung als Target Language Compiler File realisiert, da es in diesen Fall „inline“ und nicht als Funktionsaufruf in die Interruptservice Routine integriert wird.

dsp_int1.tlc

```
%% $RCSfile: dsp_int1.tlc,v $
%% $Revision: 1.3 $
%% $Date: 1999/10/27 13:08:04 $
%%
%% Christian Robl
%% Copyright (c) 1998 by LPR. All Rights Reserved.
%%
%% Abstract: Target file for inlining S-Function device driver dsp_int1.c
%%

%implements "dsp_int1" "C"

%openfile buffer
#define minimum(x, y)   ((x) < (y) ? (x) : (y))
#define REG_OFFSET      (0x280)
#define DSP_Filename    "dsp_matl.dsk"
#define fileident       ">>>> DSK3A C3x DSP Starter Kit Assembler Rev 1."
%closefile buffer
%<LibCacheDefine(buffer)>

%openfile buffer
/* hardware registers */
typedef int ByteRegister;
typedef unsigned int WordRegister;

/* Declaration of additional functions */

static int  PCI_in(WordRegister, WordRegister);
static void PCI_out(WordRegister, WordRegister, int);
static void DSP_loadprg(WordRegister);
static int  DSP_running(WordRegister);
```

```
static void DSP_omb(WordRegister, int, int);
static int  DSP_imb(WordRegister, int);
static int  DSP_mbstat(WordRegister);
static void DSP_int0(WordRegister);
static void DSP_startserial(WordRegister, WordRegister);
static void DSP_stopserial(WordRegister, WordRegister);
%closefile buffer
%<LibHeaderFileCustomCode(buffer,"trailer")>




%% Function: BlockInstanceSetup
%% ============================================
%%
%% Abstract:
%%      Setup parameters and error check.
%%
%function BlockInstanceSetup(block, system) void

%% Only allow 1 dsp_int1 block
%if EXISTS ("dsp_int1BlockSeen")
   %assign errTxt = "Only 1 dsp_int1 block is allowed in " ...
     "model: %<CopiledModel.Name>."
   %exit RTW Fatal: %<errTxt>
%else
   %assign ::dsp_int1BlockSeen = 1
%endif

%<LibRenameParameter(block,P1,"base_m")>
%<LibRenameParameter(block,P2,"base_b")>
%<LibRenameParameter(block,P3,"offset")>
%<LibRenameParameter(block,P4,"gain")>

%assign BaseRegisterMB = LibBlockParameterString(base_b)

%openfile buffer
int data;
static int *ptr=(int*)%<BaseRegisterMB>;
static int rcv,tmp;
%closefile buffer
%<LibSystemOutputCustomCode(system,buffer,"declaration")>
```

```
%endfunction %% BlockInstanceSetup

%% Function:
Start============================================================
%% Abstract:

%function Start(block, system) Output

%assign BaseRegisterMEM = LibBlockParameterString(base_m)
%assign BaseRegisterMB = LibBlockParameterString(base_b)

/* Initialize the board. */
fprintf(stdout,"DSP_serial: DSP-Init begins \n");
DSP_loadprg(%<BaseRegisterMEM>); /* initialise DSP */
DSP_startserial(%<BaseRegisterMB>, %<BaseRegisterMEM>);

PCI_out(%<BaseRegisterMEM>, 0xfe0001, 0xf);   /*LED an*/
DSP_omb(%<BaseRegisterMB>,0x0f,0x1c00);       /*enable PCI-IRQ*/
fprintf (stdout,"DSP-Init o.k.\n");

%endfunction %% Start

%% Function: Out-
puts=============================================================
%%
%% Abstract:
%%
%function Outputs(block, system) Output

%assign BaseRegisterMEM = LibBlockParameterString(base_m)
%assign BaseRegisterMB = LibBlockParameterString(base_b)
%assign Offset = CAST("Number",LibBlockParameterValue(offset,0))
%assign Gain = CAST("Number",LibBlockParameterValue(gain,0))



/* INTCSR Interrupt control & status register */
tmp=*(ptr+0x0e);
if (tmp & 0x800000) { /* interrupt assert */
   rcv=*(ptr+0x0d); /* MBEF*/
   rcv=*(ptr+0x07); /* Access Mailbox*/

   %%data=PCI_in(%<BaseRegisterMEM>,REG_OFFSET);
```

```
   %%<LibBlockOutputSignal(0,"","",0)> = data;

   %roll Idx=[0:3],lcv=2,block,"Roller",["Y"]
   data=PCI_in(%<BaseRegisterMEM>,REG_OFFSET+1+%<lcv>);
   %<LibBlockOutputSignal(0,"",lcv,Idx)> = ...
        (double)((float)(data-%<Offset>)*%<Gain> /65536.0);
   %endroll

   %%data=PCI_in(%<BaseRegisterMEM>,REG_OFFSET+5);
   %%<LibBlockOutputSignal(0,"","",0)> = data;

   DSP_imb(%<BaseRegisterMB>,4); /* empty mailbox 4*/

        /* this clears the interrupt (write on clear) */
        *(ptr+0x0e)=tmp;       /* try to clear it */
        } /* else an other device generated the interrupt, do nothing */

%endfunction %% Outputs

%% Function: Termi-
nate============================================================
%%
%% Abstract:

%function Terminate(block, system) Output

%assign BaseRegisterMEM =LibBlockParameterString(base_m)
%assign BaseRegisterMB = LibBlockParameterString(base_b)

DSP_stopserial(%<BaseRegisterMB>,%<BaseRegisterMEM>);
PCI_out(%<BaseRegisterMEM>,0xfe0001, 0x0);   /*LED aus*/

%endfunction %% Terminate

%openfile buffer
/* Function to perform the actual hw output of data  */
static void PCI_out(WordRegister baseaddr, WordRegister addr, int val)
{
        int *ptr;
        ptr=(WordRegister *)baseaddr;
        *(ptr + 1) = addr;              /* DAM write reg */
        *(ptr + 2) = val;               /* DAM data reg  */
}
```

```
/* Function to perform the actual hw input of data  */
static int PCI_in(WordRegister baseaddr, WordRegister addr)
{
        int *ptr;

        ptr= (WordRegister *)baseaddr;  /* DAM read reg */
        *(ptr+0)=addr;
        return (*(ptr + 2));            /* DAM data reg */

}

/* Function to output data on Mailbox Registers */
static void DSP_omb(WordRegister baseaddr, int num, int val)
{
        int *ptr;

        ptr= (WordRegister *)baseaddr;  /* Mailbox base addr */
        *(ptr + (num-1))=val;           /* write val to Mailbox number num
*/

}

/* Function to input data from Mailbox Registers */
static int  DSP_imb(WordRegister baseaddr, int num)
{
        int *ptr;

        ptr= (WordRegister *)baseaddr;  /* Mailbox base addr */
        return (*(ptr + (num+3)));      /* read Mailbox number num */

}

/* Function to determine Mailbox empty/full status */
static int  DSP_mbstat(WordRegister baseaddr)
{
        int *ptr;

        ptr= (WordRegister *)baseaddr;  /* Mailbox base addr */
        return (*(ptr + 0x0d));         /* read Mailbox status reg */

}
```

```
/* Function to perform DSP interrupt 0 */
static void DSP_int0(WordRegister baseaddr)
{
        int *ptr;

        ptr= (WordRegister *)baseaddr;  /* DAM base addr */
        *(ptr + 4)= 1;                  /* access int0 reg */
}

/* Function to load DSP Programm */
static void DSP_loadprg(WordRegister baseaddr)
{
    FILE *fdf;
    char inp[160];
    unsigned long val, addr;
    int *ptr;
        ptr=(WordRegister *)baseaddr;

    if (!DSP_running(baseaddr))  {
        *(ptr + 3) = 0;                 /* reset line reg  */

        fprintf(stdout,"Loading DSP-Program... \n");
        if ((fdf = fopen(DSP_Filename, "r")) == NULL)   /* Open file to
upload*/
                {
                fclose (fdf);
                perror("\n Can't find file to load ");
                exit (2);  /* exit with severe error */
                }
        fgets(inp, 160, fdf);
        inp[47] = 0;
        if (strcmp(inp, fileident))     /* test if file is valid */
                {
                fclose (fdf);
                perror("\n Invalid File Format ");
                exit (2);  /* exit with severe error */
                }
        while (fgets(inp, 160, fdf) != NULL) {
                if ((inp[0] != '>') && (inp[11] == '0')) {
                        addr = strtoul(inp   , NULL, 16);
                        val  = strtoul(inp+11, NULL, 16);
                        PCI_out (baseaddr, addr, val);
                }
```

```
        }
        fclose (fdf);
        *(ptr + 3) = 1;                 /* clear reset line*/
   }
   else fprintf(stdout,"DSP already running  \n");
}

/* Test if DSP already working */
static int DSP_running(WordRegister baseaddr)
{
        int t, tmp;
        if (PCI_in(baseaddr, 0x1f) != PCI_in(baseaddr, 0x1f))
         /*Heartbeat Reg. */
                return (0);
        else
                return (0);
}

/* Initialise serial port of DSP */
static void DSP_startserial(WordRegister baseaddrmb, WordRegister ba-
seaddrmem)
{
        int t;
        DSP_imb(baseaddrmb, 4);         /* empty mailbox 4*/

        DSP_omb(baseaddrmb, 1, 3);      /* Mailbox 1 (command) */
        DSP_int0(baseaddrmem);          /* int0,   init sensor */
        for (t=0; t<100; t++) sysDelay();   /* Warten bis Befehl ausgeführt
*/
        DSP_omb(baseaddrmb, 1, 2);      /* Mailbox 1 (command) */
        DSP_int0(baseaddrmem);          /* int0,  start serial */
}

/* stop serial port */
static void DSP_stopserial(WordRegister baseaddrmb, WordRegister ba-
seaddrmem)
{
        DSP_omb(baseaddrmb, 1, 4);      /* Mailbox 1 (command) */
        DSP_int0(baseaddrmem);          /* int0,   stop serial */
}
%closefile buffer
%<LibRegFileCustomCode(buffer,"header")>
%%END dsp_int1.tlc
```