Fortschritt-Berichte VDI

VDI

Reihe 8

Mess-,
Steuerungs- und
Regelungstechnik

Nr. 1271

Dipl.-Ing. Holger Zipper,
Magdeburg

Verfahren zur Synchronisation betriebsparalleler Simulationen durch Online-Parameterschätzung

OTTO VON GUERICKE UNIVERSITÄT MAGDEBURG

EIT FAKULTÄT FÜR ELEKTROTECHNIK UND INFORMATIONSTECHNIK

Verfahren zur Synchronisation betriebsparalleler Simulationen durch Online-Parameterschätzung

Dissertation
zur Erlangung des akademischen Grades

Doktoringenieur
(Dr.-Ing.)

von Dipl.-Ing. Holger Zipper
geboren am 17.10.1986 in Gera

genehmigt durch die Fakultät für Elektrotechnik und Informationstechnik
der Otto-von-Guericke Universität Magdeburg

Gutachter:	Prof. Dr.-Ing.	Christian Diedrich
	Prof. Dr.-Ing. Dr. h. c.	Michael Weyrich

Promotionskolloquium am 19.02.2021

Fortschritt-Berichte VDI

Reihe 8

Mess-, Steuerungs- und Regelungstechnik

Dipl.-Ing. Holger Zipper, Magdeburg

Nr. 1271

Verfahren zur Synchronisation betriebsparalleler Simulationen durch Online-Parameterschätzung

VDI verlag

Zipper, Holger
Verfahren zur Synchronisation betriebsparalleler Simulationen durch Online-Parameterschätzung
Fortschr.-Ber. VDI Reihe 08 Nr. 1271. Düsseldorf: VDI Verlag 2021.
104 Seiten, 40 Bilder, 22 Tabellen.
ISBN 978-3-18-527108-3 ISSN 0178-9546,
€ 43,00/VDI-Mitgliederpreis € 38,70.

Für die Dokumentation: Betriebsparallele Simulationen – Synchronisation – Optimierung – Co-Simulation – Digitaler Zwilling

Die vorliegende Arbeit beschäftigt sich mit der Entwicklung der Zustandssynchronisierung, welche eine betriebsparallele Simulation an eine industrielle Anlage kontinuierlich angleicht. Die betrachteten Simulationsmodelle liegen als Co-Simulation nach dem FMI Standard vor. Die Zustandssynchronisierung funktioniert nachdem Prinzip der Signal-Rückkopplung der Differenz zwischen betriebsparalleler Simulation und physischer Anlage. Die Synchronisation wird durch einen Optimierungsalgorithmus online erreicht, welche den Unterschied zwischen physischer Anlage und Simulation minimiert. Dabei wird der Einsatz von statischen und dynamischen Optimierungsalgorithmen untersucht. Es werden weiterhin Möglichkeiten zur Gewährleistung der Echtzeitfähigkeit sowie zur Zeitsynchronisation erarbeitet. Die entwickelten Methoden und Algorithmen werden anschließend an unterschiedlichen technischen Systemen erfolgreich validiert.

Bibliographische Information der Deutschen Bibliothek
Die Deutsche Bibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliographie; detaillierte bibliographische Daten sind im Internet unter www.dnb.de abrufbar.

Bibliographic information published by the Deutsche Bibliothek
(German National Library)
The Deutsche Bibliothek lists this publication in the Deutsche Nationalbibliographie (German National Bibliography); detailed bibliographic data is available via Internet at www.dnb.de.

Als Manuskript gedruckt. Printed in Germany.

ISSN 0178-9546

ISBN 978-3-18-527108-3

Danksagung

Ich danke Prof. Christian Diedrich für die gute Betreuung während der Promotion sowie Prof. Michael Weyrich für seinen Einsatz als Zweitgutachter.

Für Ihre Unterstützung während der Zeit meiner Promotion danke ich meiner Frau, Darina Schulze-Zipper, und meiner Familie.

Darüber hinaus gilt mein Dank den Kolleginnen und Kollegen sowie dem Institut ifak e.V. Magdeburg, welche mir das Umfeld zur Erstellung der Promotionsarbeit gegeben haben.

Inhaltsverzeichnis

Glossar

Co-Simulation	beschreibt die Simulation eines Gesamtsystems bestehend aus mehreren, gekoppelten Simulatoren. Im Rahmen dieser Arbeit wird ein einzelner Simulator als Simulationskomponente bezeichnet (Gomes u. a., 2017).
Distanz	beschreibt die minimale Anzahl von Iterationen der Co-Simulation für die Zustandssynchronisierung
Horizont	beschreibt die Anzahl von Iterationen der Co-Simulation, welche von der Zustandssynchronisierung betrachtet werden. Diese kann über die minimale Anzahl der Iterationen der Co-Simulation hinausgehen.
Regelungs- und Steuerungsfunktion	beschreibt die den Regelungs- oder Steuerungsalgorithmus umsetzende Komponente in der physischen Anlage.
Simulation	bezeichnet die Ausführung einer parametrierten Instanz Simulationsmodells.
Simulationskomponente	beschreibt einen einzelnen Simulator als Teil einer gekoppelten Co-Simulation. Jede Simulationskomponente enthält ein Simulationsmodell und einen ausführbaren Algorithmus, welcher das Simulationsmodell simulieren kann.
Simulationsmodell	bezeichnet ein simulierbares Modell einer mechatronischen Komponente.
Simulierte Anlage	bezeichnet das durch eine Co-Simulation ermöglichte digitale Abbild einer physischen Anlage.

Symbole

A	Index des Aktors
α	Verzögerung
C	Index des Kommunikationssystems
f	zeitvariantes, internes Verhalten einer Simulationskomponente
$\alpha_1 * \alpha_2$	Faltung von α_1 und α_2
h	Abbildung von internen Zuständen einer Simulationskomponente auf deren Ausgänge
$\hat{[\,]}$	von der physischen Anlage gemessene Variable
i	Index eines Simulationsschritt
k	Index einer Simulationskomponente
K	Gesamtzahl von Simulationskomponenten
N_{ges}	Gesamtzahl der Iterationen aller Simulationskomponenten
n	Gesamtzahl der Simulationsschritte
α^n	n-fache Faltung von α mit sich selbst
c	Weglänge
$\widetilde{c}$	kürzeste Weglänge
R	Index der Regelungs- und Steuerungsfunktion
S	Index der Simulation
$\tau_\bullet$	Totzeit von $\bullet$
$\theta_\bullet$	Zykluszeit von $\bullet$

T_S	Makroschrittweite
u	Eingang
$\mathcal{U}$	Gesamtheit der Eingänge aller Simulationskomponenten
$\alpha_{\bullet\circ}$	Verzögerung $\bullet \rightarrow \circ$
x	interner Zustand
y	Ausgang
$\mathcal{Y}$	Gesamtheit der Ausgänge aller Simulationskomponenten

Kurzfassung

Die Digitalisierung der Industrie schreitet durch Initiativen wie Industrie 4.0 voran. Dabei kommen zunehmend Methoden in den Fokus, welche auf Basis von Simulationsmodellen Online-Analysen und Online-Optimierungen industrieller Anlagen durchführen sollen.

Als Konsequenz nimmt die Bedeutung der betriebsparallelen Simulation zu. Die betriebsparallele Simulation muss jedoch stetig an die physische Anlage angepasst werden, da die physische Anlage Änderungen unterliegt. Hervorgerufen werden diese beispielsweise durch Alterung oder Umbauten aufgrund von Produktupdates. Um diese Anpassung zu erreichen, wird in der vorliegenden Arbeit die Methode der *Zustandssynchronisierung* entwickelt. Diese dient zur Synchronisation zwischen betriebsparalleler Simulation und der entsprechenden physischen Anlage.

Dazu wird zunächst der Stand der Wissenschaft untersucht und diese Arbeit eingeordnet. Als Grundlage für die weiteren Arbeiten wird anschließend ein formales Modell der Co-Simulation aufgestellt. Dieses richtet sich nach der Funktionsweise des *FMI*-Standards, wie er aktuell für Simulationen in der virtuellen Inbetriebnahme industrieller Anlagen eingesetzt wird.

Der Hauptteil der Arbeit beschäftigt sich mit der Entwicklung der Zustandssynchronisierung, welche ohne zusätzliche Steuerungskomponenten auskommt. Sie funktioniert nach dem Prinzip der Signal-Rückkopplung der Differenz zwischen betriebsparalleler Simulation und physischer Anlage. Die Synchronisation wird durch einen Optimierungsalgorithmus auf Basis dieses Unterschieds online durchgeführt. Dabei wird der Einsatz von statischen und dynamischen Optimierungsalgorithmen untersucht. Es werden weiterhin Möglichkeiten erarbeitet, mit denen sich die Anzahl der Ausführungen der Co-Simulation deutlich verringern lassen, ohne dass dies einen Einfluss auf die Funktionsweise der Zustandssynchronisierung hat. Zudem werden Aspekte der zeitlichen Synchronisation erläutert.

Die entwickelten Methoden und Algorithmen werden anschließend validiert. Dazu wird ein Co-Simulationsmasteralgorithmus nach dem *FMI*-Standard implementiert, welcher die Zustandssynchronisierung umsetzen kann. Zur Durchführung der Optimierung kommen unterschiedliche Optimierungsalgorithmen zum Einsatz. Zur Validierung wird die Zustandssynchronisierung im Hardware-in-the-Loop Betrieb an drei unterschiedliche Demonstratoren eingesetzt.

Mit Hilfe der Demonstratoren kann das erfolgreiche angleichen der betriebsparallelen Simulation an die physische Anlage durch die Zustandssynchronisierung nachgewiesen

werden. Weiterhin wird gezeigt, dass sich die Simulation dabei in Echtzeit durchführen lässt.

Abstract

The implementation of industry digitization is progressing through initiatives such as Industry 4.0. The focus is increasingly on methods that use simulation models to perform online analyses and online optimization of industrial plants.

As a consequence, the importance of online simulation in parallel to plant operation is increasing. However, the online simulation has to be constantly adapted to the physical plant, since the physical plant is subject to changes, caused for example by aging or changes due to product updates. To achieve this adaptation, the method of *state synchronization* is developed in this thesis. This method is used to synchronize the simulation with the corresponding physical plant.

For this purpose, the state of the art is first examined and this thesis is categorized. As a basis for the further work a formal model of the co-simulation is then established. This model is based on the functionality of the *FMI* standard, as it is currently used for simulation in the virtual commissioning of industrial plants.

The main part of the work deals with the development of the state synchronization, which does not require additional control components. It works according to the principle of signal feedback of the difference between online simulation and physical plant. The synchronization is performed online by an optimization algorithm based on this difference. The use of static and dynamic optimization algorithms is investigated. Furthermore, possibilities are being developed to significantly reduce the number of co-simulation runs without affecting the functionality of the state synchronization. In addition, aspects of time synchronization are explained.

The developed methods and algorithms are then validated. For this purpose, a co-simulation master algorithm according to the *FMI* standard is implemented, which can perform the state synchronization. Different optimization algorithms are used to perform the optimization. For validation purposes, state synchronization in hardware-in-the-loop operation is applied to three different demonstrators.

With the help of the demonstrators, the successful alignment of the online simulation with the physical plant by the state synchronization is proved. Furthermore, it is shown that the simulation can be performed in real time.

1. Einleitung

1.1. Motivation

In den letzten Jahrzehnten nahm der Grad der Digitalisierung im industriellen Umfeld stetig zu. Dieser Prozess wird sich in den kommenden Jahren weiter fortsetzen. Erkennen lässt sich dieser Umstand auch durch Trends wie *Cyber Physical Production Systems*, *Industrie 4.0* oder die digitale Repräsentation und digitale Nutzung aller Komponenten industrieller Anlagen über ihren gesamten Lebenszyklus hinweg. Die Basis dafür stellen die zahlreichen Engineeringmodelle dar, welche durchgehend weiter genutzt werden sollen. Der *Digitale Zwilling* beschreibt das digitale Abbild, welches ein Asset über den kompletten Lebenszyklus hinweg abbildet. Die Engineeringmodelle bilden eine Grundlage für diesen *Digitalen Zwilling*.

Ein Kernbestandteil dieser *Digitalen Zwillinge* sind die Simulationsmodelle, mit denen Produkte und industrielle Anlagen bereits während ihrer Planung getestet und optimiert werden. Zum Beispiel gibt es die virtuelle Inbetriebnahme, wo ein Simulationsmodell der gesamten Anlage aufgebaut wird, um die Korrektheit der Steuerungsprogramme zu testen. Diese Simulationsmodelle sind mindestens so präzise, dass sie gegen möglichst nicht modifizierte Versionen des Steuerungsprograms laufen (Drath, Weber und Mauser, 2008).

Soll ein *Digitaler Zwilling* auch während der Betriebsphase der industriellen Anlage genutzt werden, ist die Nachnutzung der Simulationsmodelle aus der virtuellen Inbetriebnahme zur betriebsparallelen Simulation eine mögliche Option. Diese Simulationsmodelle können die Grundlage für viele der *Self-X* (-Optimierung, -Organisation, -Konfiguration, …) Szenarien von Industrie 4.0 bilden (Schluse u. a., 2018).

Diese Simulationsmodelle sowie deren Parameter können jedoch nicht starr und unveränderlich sein, sondern müssen über die Zeit angepasst werden: Industrielle Anlagen unterliegen Änderungen. Ein Grund sind geplante Umbauten oder Anpassungen, zum Beispiel aufgrund von neuen Produkten oder Facelifts. Diese Art der Änderung wird durch Industrie 4.0 und das Bestreben nach der Losgröße eins in Zukunft zunehmen. Ein anderer Grund sind ungeplante Änderungen aufgrund Alterung und Verschleiß. In all diesen Fällen müssen diese Änderungen auch in der Simulation wieder gespiegelt werden. Darüber hinaus können Simulationen genutzt werden, um solche Änderungen überhaupt zu erkennen — durch einen Vergleich der physischen Anlage und des digitalen Zwillings, speziell

der Simulation.

Die subtilen Änderungen, welche bei der Alterung einer industriellen Anlage auftreten, können jedoch leicht dazu führen, dass das Verhalten der physischen Anlage und der simulierten Anlage nicht mehr übereinstimmen. Ein weiteres Problem ist die Parametrierung der Simulationsmodelle einer betriebsparallelen Simulation. Nicht alle Parameter einer mechatronischen Komponente einer industriellen Anlage lassen sich während der Inbetriebnahme exakt erfassen. Die Parametrierungen der physischen Anlage und der Simulationsmodelle weichen zwangsweise zu einem gewissen Grad voneinander ab. Wird die Simulation beispielsweise zur Überwachung der Anlage genutzt, kann sie dadurch sehr schnell ihre Aussagekraft verlieren und liefert falsche Aussagen über den Anlagenzustand.

Die in dieser Arbeit entwickelte Methode der *Zustandssynchronisierung* verhindert das Problem des Auseinanderlaufens von physischer Anlage und simulierter Anlage rückwirkungsfrei, daher ohne die physische Anlage zu beeinflussen. Die Zustandssynchronisierung funktioniert nach dem Prinzip der Signal-Rückkopplung der Differenz zwischen betriebsparalleler Simulation und physischer Anlage. Die Synchronisation wird durch einen Optimierungsalgorithmus auf Basis dieses Unterschieds online durchgeführt, wobei statische und dynamische Optimierungsalgorithmen zum Einsatz kommen. Die Optimierung soll in Echtzeit stattfinden, so dass weiterhin Ansätze untersucht werden, um die Echtzeitfähigkeit der Zustandssynchronisierung sicherzustellen. Trotz des Ausgleichs der Unterschiede zwischen physischer Anlage und simulierter Anlage bleibt es weiterhin möglich, den Unterschied zwischen physischer Anlage und simulierter Anlage zu ermitteln.

1.2. Erklärung an einem Beispiel

Um den Forschungsbedarf dieser Arbeit hervorzuheben wird folgendes Beispiel eingeführt: In Abbildung 1.1 ist ein vereinfachter Materialfluss-Prozess abgebildet. In der unteren Hälfte ist die physische Anlage dargestellt, in der oberen Hälfte die Simulation dieser Anlage. Folgendes Szenario wird beschrieben: Ein Produkt (Quadrat in der Abbildung) wird auf einem Förderband transportiert. Am Ende des Förderbandes wird das Produkt von einem Greifer gefasst, um es zu einer weiteren Bearbeitungsstation zu transportieren. Die Regelungs- und Steuerungsfunktion, also die SPS welche eine Steuerungs- und Regelungsaufgabe ausführt, aktiviert das Förderband. Sobald das Produkt durch den Sensor (Dreieck) erfasst wird, hält die Regelungs- und Steuerungsfunktion das Förderband an. Daraufhin fährt die Regelungs- und Steuerungsfunktion den Greifer herunter.

Die Anlage wird nun mit einer Simulation dieser Anlage zusammengeschaltet. Die Signale der Regelungs- und Steuerungsfunktion für das Förderband und den Greifer werden auch in das Simulationsmodell transportiert. Ob sich in der Simulation ein Modell des Sensors befindet oder nicht ist für die betriebsparallele Simulation unerheblich, denn es kann nicht mit der Regelungs- und Steuerungsfunktion verbunden werden. Das führt dazu, dass

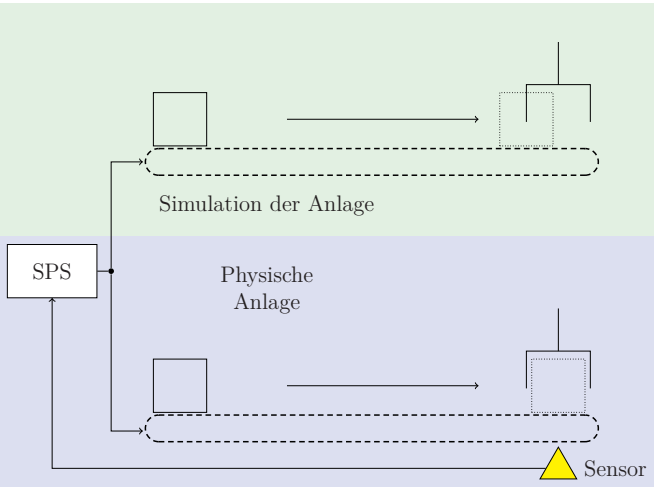


Abbildung 1.1.: Beispiel zur Motivation der Zustandssynchronisierung

die simulierte Anlage mit einem offenen Wirkungsweg betrieben wird. Die Regelungs- und Steuerungsfunktion kann also nicht auf geringfügige Abweichungen vom Zeitverhalten der Simulation reagieren.

In Abbildung 1.1 ist eine solche Abweichung des Zeitverhaltens dargestellt. Das simulierte Förderband bewegt sich etwas langsamer. Die Folge davon ist, dass das Produkt in der physischen Anlage schneller den Platz unter dem Greifer erreicht. Die Regelungs- und Steuerungsfunktion stoppt nun aufgrund des Sensorsignals beide Bänder. Ergebnis ist, dass in der Simulation der Greifer mit dem Produkt kollidiert. Die betriebsparallele Simulation kann zum Erkennen genau solcher Fehlerfälle durchgeführt werden. In diesem Fall tritt die Kollision jedoch ausschließlich in der Simulation durch eine von der Anlage abweichende Parametrierung des Förderbandes auf. Ein so aufgebautes Simulationsmodell zur Überwachung der Anlage würde also zu falschen Alarmmeldungen oder unnötigen Aktionen führen.

An diesem Problem soll diese Arbeit ansetzen: Der Zustand der Simulation, in diesem Fall beispielsweise die Produktposition oder die Förderbandgeschwindigkeit, soll an die physische Anlage angepasst werden. So behält die betriebsparallele Simulation bei abweichenden oder sich ändernden Parametern und physikalischem Verhalten ihre Aussagekraft. Diese Anpassung wird hier als Zustandssynchronisierung bezeichnet und bildet den Untersuchungsgegenstand der Arbeit.

1.3. Struktur der Arbeit

Zur Untersuchung der oben beschriebenen Problemstellung ist die Arbeit im Weiteren wie folgt strukturiert: Zunächst wird in Kapitel 2 der Stand der Wissenschaft dargelegt und analysiert. Aus dieser Analyse wird die dieser Arbeit zugrundeliegende wissenschaftliche Fragestellung abgeleitet. In Kapitel 3 wird eine Beschreibung und Formalisierung der zu untersuchenden Co-Simulation gegeben. Aufbauend auf der formalen Beschreibung der Co-Simulation wird ein Standard-Co-Simulationsmasteralgorithmus gezeigt und eine Graphen-Darstellung der Co-Simulation eingeführt. Diese bilden die Grundlage für die weiteren Betrachtungen. In Kapitel 4 wird das Konzept der Zustandssynchronisierung ausführlich vorgestellt. Dazu wird das Optimierungsproblem aufgestellt und die Zielfunktion definiert. Darüber hinaus werden Betrachtungen für die Echtzeitfähigkeit der Zustandssynchronisierung und die Zeitsynchronisation vorgenommen. Der vorgestellte Standardalgorithmus des Co-Simulationsmasters wird um die Methoden der Zustandssynchronisierung erweitert. Der resultierende Algorithmus und die Methodik werden in Kapitel 5 mit Hilfe verschiedener physischer Systeme validiert. Dort wird zudem eine Übersicht über die im Rahmen der Arbeit vorgenommenen Implementierungen gegeben. Es wird gezeigt, dass die Zustandssynchronisierung erfolgreich und in Echtzeit die simulierte Anlage an die physische Anlage angleichen kann. In Kapitel 6 wird eine Zusammenfassung der Arbeit gegeben, die Lessons-Learned diskutiert und ein Ausblick aufgezeigt.

2. Stand der Wissenschaft

Dieses Kapitel gibt einen Überblick zur betriebsparallelen Simulationen unter Nutzung von Simulationsmodellen aus der virtuellen Inbetriebnahme sowie der Synchronisation der simulierten Anlage mit der physischen Anlage geben. Es wird gezeigt, dass dies ein aktuell verfolgtes Ziel im Engineering industrieller Anlagen ist und dass wichtige Herausforderungen bei der Synchronisation liegen. Aus einer Gegenüberstellung der Herausforderungen auf der einen Seite und den Ergebnissen aus der Literaturrecherche auf der anderen Seite wird der Forschungsschwerpunkt dieser Arbeit abgeleitet.

2.1. Überblick betriebsparallele Simulation

Laut (VDI/VDE 3693 Blatt 1, 2016) ist die virtuelle Inbetriebnahme im engeren Sinne der Test des Automatisierungssystems mit Hilfe von Simulationsmodellen. In vielen Branchen, wie beispielsweise Fertigung (Damrath u. a., 2015), Wassermanagement (Hübner, Suchold und Alex, 2018) und der Prozessindustrie (Chan und Krauss, 2014), ist es damit üblich, während des Engineerings ein Simulationsmodell der automatisierungstechnischen Anlage aufzubauen. Die virtuelle Inbetriebnahme führt letztendlich zu einer Hardware-in-the-Loop Simulation zum Test der Anlagensteuerung. Daher muss dieses Simulationsmodell so präzise arbeiten, dass die Regelungs- und Steuerungsfunktion diese Simulation nicht von der physischen Anlage unterscheiden kann (Drath, Weber und Mauser, 2008). Darüber hinaus muss die Simulation echtzeitfähig sein. Die Simulationszeit darf weder signifikant langsamer, noch signifikant schneller als die Zeit in der physischen Anlage vergehen, da nach Möglichkeit keine Zeitkonstanten im Programm der Regelungs- und Steuerungsfunktion angepasst werden sollen.

In den letzten Jahren wurden die Simulationsmodelle für die virtuelle Inbetriebnahme zunehmend auf Basis physikalischer Verhaltensmodelle aufgebaut (Rodriguez-Guerra u. a., 2019), (Puntel Schmidt, 2017), (Nicolae u. a., 2018), (Vathoopan u. a., 2017), (Brandenbourger, Vathoopan und Zoitl, 2016).

In (Süß u. a., 2016) wurde vorgeschlagen, das Simulationsmodell für die virtuelle Inbetriebnahme in einzelne Simulationskomponenten aufzuteilen. Dieses Verfahren wird *Co-Simulation* genannt. Gegenüber der Nutzung monolithischer Simulationsmodelle hat dies folgenden Vorteile:

- Die Dekomposition erhöht die Wiederverwendbarkeit: Simulationskomponenten für

beispielsweise einen Sensor müssen nicht für jede Anlage neu entwickelt werden. Stattdessen kann eine Bibliothek aufgebaut werden.

- Von den Simulationskomponenten müssen nur Eingänge, Ausgänge und Parameter bekannt sein. Das interne Verhalten, wie mathematische Modelle, Konstanten, Zusammenhänge, können verborgen werden. Dadurch werden Hersteller automatisierungstechnischer Komponenten motiviert, neben den physischen Komponenten auch Simulationskomponente anzubieten. Diese können durch die Möglichkeit des Know-How-Schutzes deutlich genauer sein, da der Hersteller detailliertes Komponentenwissen einbringt.

- In einer Co-Simulation können Simulationskomponenten einzelner Sensoren und Aktoren mit einer 3D-Festkörperphysiksimulation kombiniert werden.

In (Scheifele, Verl und Riedel, 2018) wird gezeigt, dass solche Co-Simulationen für die virtuelle Inbetriebnahme durchaus in Echtzeit durchgeführt werden können. Dazu wurde dort die Verteilung der Co-Simulation auf mehrere Rechenkerne und Betriebssysteme erforscht. Es konnte für Beispiele aus dem Materialfluss gezeigt werden, dass eine Co-Simulation verteilt auf einen Rechenkern mit *Windows* und einem Rechenkern mit *RTOS* Zykluszeiten von $16ms$ beziehungsweise $1ms$ in Echtzeit erreichen konnte. Einen ähnlichen Fokus haben (Lämmle und Oppelt, 2018). Auch dort werden die Echtzeiteigenschaften von Co-Simulationen für die virtuelle Inbetriebnahme untersucht. Dabei wird eine Methode zur Kopplung von Modellen aus unterschiedlichen Werkzeugen zu einer deterministischen, echtzeitfähigen Co-Simulation einer industriellen Anlage entwickelt. In (Tobias Jung, Jazdi und M. Weyrich, 2017) werden verschiedene Simulationswerkzeuge für den Einsatz bei Anwendungen im Internet der Dinge (IOT) untersucht. Auch dort ist die Echtzeitfähigkeit in wichtiges Kriterium. Als Möglichkeit der Umsetzung einer echtzeitfähigen Co-Simulation wird ein agentenbasiertes System vorgeschlagen.

Die in der virtuellen Inbetriebnahme genutzten Modelle finden auch in weiteren Phasen des Engineerings Anwendung. Dafür wurde in den letzten Jahren das Konzept des *digitalen Zwillings* vorgeschlagen (Auris, Zipper u. a., 2018), (Hübner und Alex, 2018). Dieser soll physische Assets, wie Anlagen oder Geräte, im gesamten Lebenszyklus digital repräsentieren (Klein u. a., 2019), (Kritzinger u. a., 2018). Auch in der Betriebsphase eines Assets werden für den Einsatz von digitalen Zwillingen Vorteile prognostiziert (Negri, Fumagalli und Macchi, 2017), (Pfeiffer, 2018):

- Optimierung des Prozesses, zum Beispiel Zeiten oder Energieverbrauch

- Vorbereiten und Validierung von Änderungen am Asset (Umbauten, Anlagenrekonfiguration)

- Fehlerdetektierung und Fehlerdiagnose

Ebenfalls sehen die Aktivitäten im Bereich Industrie 4.0 den Einsatz digitaler Zwillinge vor: In (Plattform Industrie 4.0, 2019) wird das Konzept der Verwaltungsschale als Realisierung des Konzeptes digitaler Zwilling im Rahmen von Industrie 4.0 aufgefasst, wobei Simulation als eine der Grundlagen hierfür gesehen wird. Auch in der *Namur Open Architecture* wird betriebsparallele Simulation Teil der zentralen Monitoring- und Optimierungsdienste sein (Klettner u. a., 2017), (Tauchnitz u. a., 2019).

Generell wird davon ausgegangen, das digitale Zwillinge, welche zur Betriebsphase Anwendung finden, ein Simulationsmodell des Assets enthalten (Glaessgen und Stargel, 2012), (Rosen u. a., 2019), (Vögeli, Göhner und Michael Weyrich, 2018), (Klein u. a., 2019).

In (Oppelt, Barth und Urbas, 2015) wird eine global angelegte Umfrage beschrieben. Diese stellt den damaligen Stand der Nutzung von Simulation in der Prozessindustrie dar und schildert (aus der damaligen Sicht) zukünftige Entwicklungen. Die Befragten sehen die Integration von Simulationsmodellen aus verschiedenen Werkzeugen und die Kopplung unterschiedler Simulationswerkzeuge als wichtig an. Von vier dargestellten Anwendungsfällen beziehen sich zwei Anwendungsfälle auf die Betriebsphase: Training des Anlagenpersonals und Anlagenoptimierung. Zum Zeitpunkt der Studie lag der Einsatz von Simulation für diese Anwendungsfälle jedoch bei 20%, jedoch sahen 82% der Befragten eine zukünftige Zunahme beziehungsweise deutliche Zunahme der Wichtigkeit von Simulation in der Betriebsphase.

Simulation als eine der zentralen Komponenten des Lebenszyklus industrieller Anlagen wird auch in (Markus Graube und Leon Urbas, 2018) untersucht, wobei auch die Rolle der Co-Simulation für die Betriebsphase herausgestellt wird.

(Tobias Jung, Shah und Michael Weyrich, 2018) beschreiben die Anforderungen zur Umsetzung heterogener simulationsbasierter digitaler Zwillinge für Komponenten im Internet der Dinge (IOT). Als eine Anforderung wird die lebenszyklusübergreifende Nutzung von Simulation genannt. Dabei wird ein *Plug and Simulate* Konzept vorgestellt, mit welchen Simulationen unterschiedlicher Komponenten dynamisch zur Betriebsphase eine Co-Simulation bilden können. In dem Beitrag wird vor allem die dynamische Orchestrierung der Co-Simulation durch die Realisierung als Multi-Agentensystem untersucht. Zum Erreichen der Echtzeitfähigkeit der Co-Simulation wurde ein lernender Algorithmus zur Reduzieren der Nachrichten implementiert. Als offene Punkte werden die Synchronisierung und die Interaktion mit der physischen Anlage genannt.

Die Verbindung einer industriellen Anlage mit einer Simulation wird als betriebsparallele Simulation bezeichnet (Kain und Schiller, 2010). In (Bergs und Heizmann, 2019) wird eine Methode gezeigt, welche betriebsparallele Simulation durch die Kombination von White-Box und datengetriebenen Modellen ermöglicht. Da Whitebox Modelle möglicherweise hohe Rechenzeit bedürfen und dadurch von den Autoren als nicht geeignet für die Echtzeitsimulation angesehen werden, werden diese durch sogenannte *Surrogat*

Modelle angenähert. Gleichzeitig wird ein auf Machine-Learning basierendes, datengetriebenes Modell mit Hilfe des White-Box Modells erzeugt. Die Kombination erlaubt dann eine echtzeitfähige und betriebsparallele Simulation. Weitere Beispiele für den Einsatz von White-Box Modellen in der betriebsparallelen Simulation finden sich in (Hübner, Suchold und Alex, 2018) zur Optimierung im Wassermanagement.

Auch (Härle, Barth und Fay, 2018) nennen betriebsparallele Simulation als wichtiges Szenario von Simulation im industriellen Einsatz. Darin wird eine Möglichkeit beschrieben, solche Simulationen auf preisgünstige, jedoch ausreichend leistungsfähige Einplatinenrechner zu verteilen. Jedem Feldgerät einer industriellen Anlage könnte solch ein Einplatinenrechner zugeordnet werden. Der Beitrag beschreibt weiterhin, wie eine Co-Simulation auf dieser Art von Hardware-Verbund verteilt werden kann und dabei echtzeitfähig bleibt.

In (Putman u. a., 2017) wird eine Methodik vorgeschlagen, mit der eine physische Anlage und deren Simulationsmodell mit Hilfe eines *Virtual Fusion Filters* verbunden werden. Dieser nimmt die Messwerte der physischen Anlage und deren simulierte Pendants auf. Die Aufnahmen der physischen Anlage und der simulierten Werte werden zeitlich zueinander zugeordnet und die Differenz berechnet. Diese Differenz wird statt der eigentlichen Messwerte in die Anlagensteuerung gegeben. Der Anwendungsfall liegt bei der Simulation virtueller Werkstücke in physischen automatisierungstechnischen Anlagen. Jedoch wird bei dieser Methodik die Übereinstimmung des Verhaltens und der Parameter des physischen Systems und des simulierten Systems als notwendige Voraussetzung genannt. Wie sich die Methodik verhält, wenn diese Voraussetzung nicht erfüllt sind, wird nicht weiter untersucht.

In (Kain, Dominka u. a., 2009) wird eine Architektur vorgeschlagen, womit die Simulationsmodelle, welche bereits für die virtuelle Inbetriebnahme aufgebaut wurden, zur Betriebszeit einer Anlage weiter genutzt werden können, nennt aber die Synchronisierung von simulierter und physischer Anlage als Herausforderung. Konkret wird davon ausgegangen, dass im Falle der Betrachtung von geschlossenen Regelkreisen als Anlage / Simulationsgegenstand eine zusätzliche Steuerung genutzt werden muss, um den Regelkreis in der simulierten Anlage zu schließen.

Auch (Kapp, 2011) beschreibt die Synchronisation von betriebsparalleler Simulation und dem Prozess als Voraussetzung für die dort entwickelte Architektur zur betriebsparallelen Fabriksimulation. Jedoch wird die Synchronisation zur Laufzeit als zu Aufwändig angesehen und die Erweiterbarkeit und Adaptierbarkeit erschwerend angesehen. Stattdessen wird die Synchronisierung lediglich bei der Initialisierung der Modelle, der erstmaligen Verbindung der betriebsparallelen Simulation und Prozess, durch einmaliges Setzen der Zustände durchgeführt. Die Untersuchung der Synchronisation aus Sicht der Initialisierung der Modelle geschah auch in (Hanisch, Tolujew und Schulze, 2005), geht aber von modellspezifischen sowie verhaltensspezifischen Verfahren aus.

Auch (Rosen u. a., 2019) nennt die Synchronisierung als zukünftige Herausforderung zur

Nutzung betriebsparalleler Simulation und sieht dabei Online-Parameterschätzverfahren, zusätzliche Beobachter oder eine Verbindung von Black- und Whitebox Modellen als Lösung, jedoch ohne detailliert auf diese Ansätze einzugehen.

(Ashtari Talkhestani, T. Jung u. a., 2019) nennen Synchronisierung mit dem physischen Asset, aktiver Datenabgriff und Möglichkeit zur Simulation des Verhaltens des physischen Assets als die Voraussetzungen für digitale Zwillinge. Die Synchronisation mit dem physischen Asset wird als wichtige Voraussetzung für den Erfolg des digitalen Zwillings gesehen. Daher wird das als *Ankerpunktmethode* bezeichnete Konzept zur Synchronisierung vorgestellt. Dieses allgemeingültige Konzept wird zum Beispiel in (Ashtari Talkhestani, Jazdi u. a., 2018) näher beschrieben. Die *Ankerpunktmethode* ist ein Ansatz zur Synchronisierung mittels Korrelation der Informationen zu einer mechatronischen Komponente aus unterschiedlichen Teilmodellen des Engineerings industrieller Anlagen. Beispiele für solche Informationen sind Signale im Steuerungsprogramm oder Objekte in einem 3D-Scan des Assets. Der Fokus liegt darauf, die Teilmodelle zu aktualisieren auf der Basis der Erkennung von Unterschieden zwischen den aktuellen Daten eines Teilmodells und der zu einem früheren Zeitpunkt gespeicherten Version. Der Aspekt der Synchronisierung der Simulationsmodelle mit der physischen Anlage zur Laufzeit wird nicht näher untersucht.

Die Aufgabe, aktuelle Parameterwerte einer mechatronischen Komponente mit Hilfe eines Simulationsmodells zu bestimmen ist als Parameterschätzung bekannt. Es gibt diverse Arbeiten, welche Parameterschätzung für Modelle nach dem *FMI* Standard (FMI 2.0, 2014), welcher auch für die Co-Simulation der virtuellen Inbetriebnahme genutzt wird, durchführen. In (Kampfmann, Mösch und Menager, 2017) wird eine *FMI*-basierte Werkzeugkette vorgestellt. Es wird dargestellt, wie sich mit dieser Werkzeugkette offline die Parameter eines Roboters schätzen lassen. Dazu wird eine Messung der Eingänge und Ausgänge des Roboters mit einem Simulationsmodell und einem Optimierungsalgorithmus kombiniert. Anders als die Co-Simulation der virtuellen Inbetriebnahme handelt sich bei der Simulation nicht um eine Black-Box Co-Simulation. Ähnliche Ansätze zur Offlineparameterschätzung eines einzelnen Modells verfolgen auch (Bonilla u. a., 2017), (Vanfretti u. a., 2016). (Gedda u. a., 2012) nutzt für dieses Optimierungsproblem ableitungsfreie Optimierungsalgorithmen.

In (Otto, Vogel-Heuser und Niggemann, 2018) wird der *CyberOpt Online* Ansatz vorgestellt, um Geschwindigkeits-Parameter von cyber-physischen Produktionssystemen ohne vordefinierte Modelle zu schätzen. Dabei handelt es sich um einen Datengetriebenen Ansatz, welcher *Process Mining* und Surrogat Modelle nutzt, um die Parameter Online zu bestimmen.

Auch in (Biesinger u. a., 2018) wird ein datengetriebener Ansatz verfolgt. Das Ziel dabei ist es, einen digitalen Zwilling automatisch durch die Kombination von Offlinedaten des Engineerings und Onlinedaten zu bilden. Dazu werden die Steuerungs- und Roboterprogramme analysiert, um daraus beispielsweise Prozess-Zykluszeiten zu gewinnen.

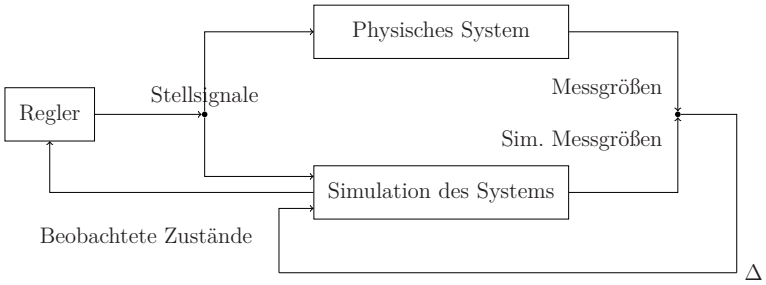


Abbildung 2.1.: Geregeltes System mit Beobachter

Selbst aktuelle, präzise Simulationsmodelle, besitzen Einschränkungen, welche den Abgleich zwischen physischem System und den Simulationsmodellen erschweren (Auris, Fisch u. a., 2018). Effekte wie Rauschen der Signale, Alterung oder Abnutzung lassen sich in Simulationsmodellen, welche auf Differenzialgleichungen oder auf differential-algebraischen Systemen basieren, nur schwer Modellieren. Zudem müssen alle eventuell möglichen so entstehenden Fehler zusätzlich zum nominalen Verhalten implementiert, parametriert und aktiviert werden (Gundermann u. a., 2019).

In der Regelungstechnik existiert das Konzept des Beobachters, siehe z.B. (Lunze, 2008), mit welchem interne, nicht messbare, Zustände geschätzt werden können. Hierbei wird eine Simulation betriebsparallel durchgeführt. Die Reglerausgänge werden sowohl in die Simulationsmodelle, als auch in das physische System gegeben. Nicht messbare Zustände sind im Simulationsmodell hinterlegt und können daher abgegriffen werden. Die Messgrößen des physischen Systems werden mit ihren Entsprechungen der Simulationsmodelle verglichen und die Differenz, der Beobachterfehler, als Korrektur zurück in das Simulationsmodell geführt. In Abbildung 2.1 ist ein Schema von Beobachtern dargestellt. Anhand dieses Schemas ist erkennbar, dass Beobachter eine Art von betriebsparalleler Simulation sind, welche sich mit dem physischen System synchronisieren.

Es existiert eine Vielzahl von Entwurfsverfahren für Beobachter um Zustände und Parameter zu schätzen. Beispielsweise kann das in (Bonvini, Wetter und Sohn, 2014) vorgestellte Verfahren den *FMI*-Standard nutzten. Allgemein haben Beobachter die gemeinsame Eigenschaft, dass speziell entwickelte White-Box Modelle zum Einsatz kommen (Radke und Gao, 2006). Der in (Han, 2009) beschriebene *extended state observers* ermöglicht eine datengetriebene Schätzung der Zustände und kommt ohne eine theoretische Prozessanalyse aus, enthält entsprechend auch kein Modell des Prozesses.

2.2. Schlussfolgerung

Für die virtuelle Inbetriebnahme, und generell im virtuellen Engineering, entstehen hochwertige Simulationsmodelle, welche sowohl echtzeitfähig als auch präzise sind (Süß u. a., 2016). Diese können in digitalen Zwillingen eingesetzt werden, um über das Engineering hinausgehende Mehrwerte zu erzielen. Ein solcher Mehrwert entsteht zur Betriebszeit industrieller Anlagen. White-Box Simulation zur Betriebszeit wird bereits vielfältig eingesetzt und untersucht, beispielsweise auch im Zusammenhang mit dem digitalen Zwilling. Während des Engineerings industrieller Anlagen werden jedoch vermehrt Black-Box Modelle genutzt, da sich so Modellierungsaufwand von den Anlagenherstellern zu den Komponentenherstellern verlagern lässt. Die Komponentenhersteller besitzen das notwendige Wissen über ihre Komponenten und können es aufgrund des Black-Box Ansatzes schützen. Die Nachnutzung diese (Black-Box) Modelle aus der virtuellen Inbetriebnahme wird als wirtschaftlich sinnvoll angesehen, jedoch aktuell nicht praktiziert. Der Aspekt der Echtzeitfähigkeit der betriebsparallelen Simulation wird derzeit aus verschiedenen Perspektiven untersucht, wobei bereits einige vielversprechende Lösungsansätze entwickelt wurden. Das Problem der Synchronisierung zwischen physischer Anlage und simulierter Anlage wird in der Literatur als Herausforderung beschrieben und ist bis dato noch nicht vollständig gelöst.

Zusammenfassend kann gesagt werden, dass die Zielstellung dieser Arbeit, die Nachnutzung der Modelle aus der virtuellen Inbetriebnahme zur synchronisierten betriebsparallelen Simulation, noch nicht vom aktuellen Stand der Wissenschaft abgebildet ist. Die relevante Literatur im Bereich der betriebsparallelen Simulation ist in Tabelle 2.1 dargestellt. Die Literatur wird den folgenden, für die Zielstellung relevanten, Kategorien zugeordnet:

(a) Nutzung von Simulationsmodellen aus der virtuellen Inbetriebnahme

(b) Nutzung von Black-Box Modellen

(c) Ermittlung nicht messbarer Zustände

(d) Synchronisierung von physischer Anlage und simulierter Anlage

Mit ✓werden diejenigen Eigenschaften markiert, welche in der jeweiligen Arbeit behandelt werden, mit - jene Eigenschaften, welche nicht untersucht wurden. Ist die Bewertung in Klammern angegeben, sind diese nur teilweise Beschrieben, oder es wurden andere Engineering-Modelle als Verhaltensmodelle beziehungsweise Simulationen untersucht (siehe zum Beispiel *Ankerpunktmethode*).

Tabelle 2.1.: Einordnung in den Stand der Wissenschaft

Literatur	(a)	(b)	(c)	(d)
(Ashtari Talkhestani, Jazdi u. a., 2018)	✓	-	-	(✓)
(Bergs und Heizmann, 2019)	(✓)	✓	-	-
(Biesinger u. a., 2018)	-	✓	(✓)	-
(Han, 2009)	-	✓	-	✓
(Hanisch, Tolujew und Schulze, 2005)	-	-	✓	(✓)
(Härle, Barth und Fay, 2018)	✓	-	-	-
(Hübner, Suchold und Alex, 2018)	✓	-	✓	(✓)
(Kain, Dominka u. a., 2009)	✓	-	✓	-
(Kapp, 2011)	-	-	✓	(✓)
(Otto, Vogel-Heuser und Niggemann, 2018)	-	✓	-	-
(Putman u. a., 2017)	-	-	✓	(✓)
(Radke und Gao, 2006)	-	-	✓	✓
(Rosen u. a., 2019)	✓	-	✓	-
(Scheifele, Verl und Riedel, 2018)	✓	✓	-	-
(Tobias Jung, Shah und Michael Weyrich, 2018)	✓	-	-	-

2.3. Formulierung der wissenschaftlichen Fragestellung

Aus Tabelle 2.1 wird ersichtlich, dass die Ziele dieser Arbeit und die sich so ergebende Kombination von Anforderungen aktuell nicht vom Stand der Wissenschaft abgedeckt werden. Speziell die Anforderung der Synchronisierung einer aus Black-Box Modellen der virtuellen Inbetriebnahme aufgebauten simulierten Anlage mit der physischen Anlage bleibt offen. Für diese Arbeit leitet sich daraus die folgende wissenschaftliche Fragestellung ab:

> "Wie kann der Zustand einer Co-Simulation einer industriellen Anlage, von der nur die Struktur sowie die Ein- und Ausgänge der einzelnen Simulationskomponenten, nicht jedoch deren internes Verhalten, bekannt sind, mit der physischen Anlage synchronisiert werden?"

Der Schwerpunkt soll dabei auf der Entwicklung eines Co-Simulationsalgorithmus liegen, welcher diese Synchronisierung zur Laufzeit und in Echtzeit durchführen kann.

3. Formale Beschreibung der Blackbox Co-Simulation

In diesem Kapitel soll die betrachtete Co-Simulation, der Co-Simulationsmaster und die Simulationskomponenten genauer beschrieben werden. Die Darstellungen erfolgen anhand einer formalen Beschreibung, einer Graphen-Darstellung und anhand eines Sequenzdiagramms. Zudem wird ein Standard-Algorithmus für einen Co-Simulationsmaster vorgestellt.

3.1. Einführung Co-Simulation

3.1.1. Technischer Hintergrund

Wie in Kapitel 1 beschrieben, sollen die Simulationsmodelle aus der virtuellen Inbetriebnahme zur Laufzeit der Anlage weitergenutzt werden. Hier wird davon ausgegangen, dass dies eine Co-Simulation nach dem *FMI for Co-Simulation* Standard ist (FMI 2.0, 2014). Dieser wurde im Rahmen des ITEA 2 MODELISAR Verbundprojektes (07006) zur Realisierung von „Software-, Modell- und Hardware-in-the-Loop Simulationen“ zur Unterstützung der Automobilentwicklung entworfen (Chombart, 2012). Seitdem wird die Anwendung ausgedehnt auf den Einsatz in der virtuellen Inbetriebnahme, siehe Abschnitt 2.1.

In einer Co-Simulation nach *FMI* wird das Simulationsmodell eines Gesamtsystems in mehrere Teil-Simulationsmodelle der Teilsysteme zerlegt. Ein Simulationsmodell eines Teilsystems wird *FMU* (Functional Mockup Unit) genannt. Im weiteren Verlauf dieser Arbeit werden Simulationsmodelle eines Teilsystems technologieunabhängig als Simulationskomponente bezeichnet, in Anlehnung an (Gomes u. a., 2017). Der Grund ist, dass neben FMI weitere Mechanismen zur Co-Simulation, zum Beispiel *High Level Architecture* (IEEE 1516-2010, 2010) existieren.

Ein als *Co-Simulationsmaster* bezeichneter Algorithmus organisiert die einzelnen Simulationskomponenten, deren Lebenszyklus und den Austausch von Informationen zwischen Simulationskomponenten. Auf diese Weise kommt die Gesamtsystemsimulation zu Stande.

3.1.2. Simulationskomponenten

In Abbildung 3.1 wird eine Simulationskomponente schematisch dargestellt. Die in dieser Arbeit zu entwickelnde Zustandssynchronisierung soll mit den Modellen aus der virtuellen

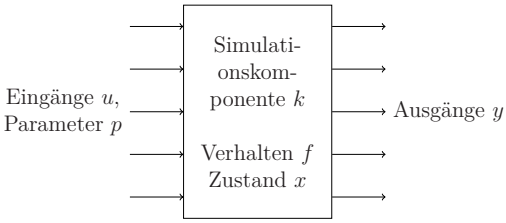


Abbildung 3.1.: Schema Simulationskomponente

Inbetriebnahme nach dem *FMI*-Standard (FMI 2.0, 2014) funktionieren. Daher werden die grundlegenden Eigenschaften der hier betrachteten Simulationskomponenten in Anlehnung an diesen Standard definiert:

- Eine Simulationskomponente ist in sich abgeschlossen, sie kann keine Änderungen an der Co-Simulation außerhalb des eigenen Simulationsmodells vornehmen. Das bedeutet, eine Simulationskomponente beeinflusst andere Simulationskomponenten nicht direkt, sondern nur indirekt über den durch den Co-Simulationsmaster gesteuerten Variablenaustausch.

- Eine Simulationskomponente besitzt Variablen und Parameter.

- Variablen können Eingänge u sein. Über diese Eingänge gelangen Informationen in eine Simulationskomponente.

- Variablen können Ausgänge y sein. Über diese Ausgänge gelangen Information aus einer Simulationskomponente nach außen.

- Parameter erlauben die Konfiguration einer Simulationskomponente. Die Konfiguration kann nur in Vorbereitung der Simulation durchgeführt werden, jedoch nicht während der Simulation.

- Das interne Verhalten f der Simulationskomponente ist im Allgemeinen nicht bekannt. Simulationskomponenten besitzen daher Black-Box-Verhalten. Modellgleichungen, Konstanten, Tabelle etc. liegen im ausführbaren Binärformat vor (zum Beispiel *.dll* Datei unter Windows und *.so* Datei unter Linux).

- Der interne Zustand x der Simulationskomponente kann als unstrukturierte Binärdaten exportiert und importiert werden. Der Aufbau der Binärdaten ist nur innerhalb dieser Instanz der Simulationskomponente bekannt beziehungsweise nutzbar. In der technologischen Umsetzung finden sich solche Ex- und Import-Funktionalitäten zum Beispiel in (FMI 2.0, 2014) (Funktionen *fmi2GetFMUstate*, *fmi2SetFMUstate*, *fmi2FreeFMUstate*), jedoch optional.

- Die Simulationskomponenten besitzen die Fähigkeit der Selbstbeschreibung, um dem Co-Simulationsmaster die zum Einbinden der Simulationskomponente in die Co-Simulation notwendigen Informationen bereitstellen zu können. Diese Informationen sind die Modellidentifikation, Eingänge, Ausgänge und Parameter.

3.1.3. Co-Simulationsmaster

Die Aufgabe des Co-Simulationsmasters ist die Durchführung der Co-Simulation. Dazu wird dieser entsprechend des zu simulierenden Setups parametriert.

Der Co-Simulationsmaster organisiert den Variablenaustausch und koordiniert alle Simulationskomponenten. Dazu gehört neben der eigentlichen Simulation auch das Management des Lebenszyklus der Simulationskomponenten, das Logging, das Speichern der Simulationsergebnisse zur späteren Auswertung und die Realisierung der Echtzeitfähigkeit der Simulation.

Zur Umsetzung dieser Aufgaben muss der Co-Simulationsmaster die folgenden grundlegenden Schritte durchführen:

1. Einlesen einer Beschreibungsdatei über den Aufbau der Co-Simulation und der beteiligten Simulationskomponenten sowie deren Parameter
2. Laden und Instanziieren der Simulationskomponenten
3. Parametrieren der Simulationskomponenten
4. Durchführen der Co-Simulation, Organisation des Variablenaustausches
5. Überprüfen auf Erreichen des Simulationsendes
6. Freigeben und Entladen der Simulationskomponenten

Zur Synchronisation des Zustandes zwischen simulierter Anlage und physischer Anlage soll in Punkt 4 eingegriffen werden.

3.2. Modell einer Co-Simulation

In diesem Abschnitt werden die vorherigen Beschreibungen formalisiert. Dafür wird ein Modell des Algorithmus des Co-Simulationsmasters in Gleichung 3.2 in Anlehnung an (Kübler und Schiehlen, 2000) und (FMI 2.0, 2014) aufgestellt.

$$x_{i+1}^{(k)} = f^{(k)}(x_i^{(k)}, u_i^{(k)}, t_i) \tag{3.1}$$

$$y_i^{(k)} = h^{(k)}(x_i^{(k)}) \tag{3.2}$$

$f^{(k)}$ steht für das zeitvariante Verhalten der k-ten Simulationskomponente, wobei $1 \leq k \leq K$ gilt und K die Gesamtzahl der Simulationskomponenten repräsentiert. $x^{(k)}$ sind die internen Zustände der k-ten Simulationskomponente. $y^{(k)} \in \mathcal{Y}$ sind deren Ausgänge, wobei $\mathcal{Y}$ die Menge aller Ausgänge aller Simulationskomponenten ist welche dem Co-Simulationsmaster bekannt sind. $u^{(k)} \in \mathcal{U}^{(k)}$ sind die Eingänge der Simulationskomponente k. Aus der Sicht des Co-Simulationsmasteralgorithmus sind $x^{(k)}$ unbekannt. Mit $h^{(k)}$ wird die Abbildung von internen Zuständen einer Simulationskomponente auf deren Ausgänge bezeichnet. Diese ist, wie auch das interne Verhalten, dem Co-Simulationsmaster unbekannt. T_S stellt die Makroschrittweite dar. Die Makroschrittweite gibt diejenige Schrittweite an, mit der die Co-Simulation voranschreitet. Jede Simulationskomponente kann intern den Makroschritt in kleinere Schritte aufteilen. Damit kann die Simulationszeit durch die Reihe $t_i = t_\mathrm{start} + \sum_1^i T_\mathrm{S}$ ausgedrückt werden. Dabei gilt, dass i den i-ten Simulationsschritt angibt. Die Gesamtzahl der Simulationsschritte ergibt sich als $n = \frac{t_\mathrm{end} - t_\mathrm{start}}{T_\mathrm{S}}$. Einzelne Ausgänge einer Simulationskomponente können mit einzelnen Eingänge einer oder mehrerer anderer Simulationskomponenten verbunden werden. Darum wird eine Abbildung definiert: $M^{(k)} : \mathcal{Y} \to \mathcal{U}^{(k)}$. Diese Abbildung bildet von der Menge aller in der Co-Simulation vorliegenden Ausgänge aller Simulationskomponenten auf die Eingänge einer Simulationskomponente ab. Die Messwerte der physischen Anlage entsprechen einer Teilmenge der Ausgänge der Simulationskomponenten.

Im weiteren Verlauf dieser Arbeit werden die Größen des Modells der Co-Simulation wie oben beschrieben benutzt. Die korrespondierenden Größen der physischen Anlage werden mit demselben Symbol ergänzt um ein $\hat{}$ dargestellt. Beispielweise wird der Eingang in die physische Anlage mit $\hat{u}$ dargestellt, die Zustände der physischen Anlage $\hat{x}$ und die Ausgänge der physischen Anlage $\hat{y}$.

Im Rahmen dieser Arbeit wird der Co-Simulationsmasteralgorithmus aus (FMI 2.0, 2014) betrachtet. Mit der Hilfe des vorgestellten Formalismus wird ein Simulationsschritt wie in Algorithmus 1 dargestellt definiert.

Algorithmus 1 Ausführung eines Schrittes der Co-Simulation nach (FMI 2.0, 2014)

function SIMULATIONSSCHRITT(i)
  for all $k \in K$ **do**
    $x_{i+1}^{(k)} = f^{(k)}(x_i^{(k)}, u_i^{(k)}, t_i)$ ▷Ausführen der Simulationskomponente, f und x intern
    $y_{i+1}^{(k)} = h^{(k)}(x_i^{(k)})$ ▷Lesen der Ausgänge, h und x intern
    $u_{i+1}^{(k)} = M^{(k)}(y_{i+1}^{(1)}, \ldots, y_{i+1}^{(K)})$ ▷Zuweisen der Eingänge
  end for
end function

Der Schritt *Ausführen der Simulationskomponente* geschieht dabei innerhalb der Blackbox Simulationskomponente. Der Co-Simulation liegt von dieser Gleichung lediglich das Ergebnis nach dem Ausführen des Schrittes *Lesen der Ausgänge* vor. Die Co-Simulation

Algorithmus 2 Co-Simulationsmaster nach (FMI 2.0, 2014)

for all $k \in K$ **do**
  INITIALISIERE($u_0^{(k)}, p^{(k)}$) ▷Setzen der Startwerte und Parameter
end for
for all $i \in [0, n]$ **do**
  SIMULATIONSSCHRITT(i)
end for

über die gesamte Zeitspanne erfolgt wie in Algorithmus 2 beschrieben.

Es wird ein Algorithmus ohne Schrittweitensteuerung genutzt. Die Schrittweitensteuerung hat keinen Einfluss auf das im Rahmen der Arbeit zu entwickelte Synchronisierungskonzept. Diese nicht zu nutzen erleichtert jedoch die Implementierung und die Validierung des Konzeptes erheblich.

3.2.1. Repräsentation einer Co-Simulation als Graph

Eine Co-Simulation kann als gerichteter Graph verbundener Simulationskomponenten verstanden werden. Die Simulationskomponenten werden von Knoten repräsentiert. Alle Verbindungen von den Ausgängen einer Simulationskomponente zu den Eingängen einer zweiten Simulationskomponente werden durch eine gerichtete Kante im Graph beschrieben. Im weiteren Verlauf dieser Arbeit werden die Co-Simulationen als Graph veranschaulicht. In Abbildung 3.2 ist ein Beispiel eines solchen Graphen dargestellt. Mit k_{input} werden diejenigen Komponenten bezeichnet, welche die Eingänge von dem Optimierungsalgorithmus zur Zustandssynchronisierung erhalten. Mit k_{output} werden diejenigen Komponenten bezeichnet, welche y für die Berechnung von $\Delta y = y - \hat{y}$ bereitstellen. Alle weiteren Simulationskomponenten werden durchnummeriert, zum Beispiel $k_1, \dots k_{\text{K}}$. Wege werden mit $W = \{k_{\text{input}}, k_{\text{output}}\}$ (hier am Beispiel des Wegs von k_{input} zu k_{output}) dargestellt. M ist in dem Graph nicht darstellbar, denn es beschreibt die Abbildung von Ausgängen *aller* Simulationskomponenten auf die Eingänge einer weiteren Simulationskomponente. Die Kanten im Graph hingegen beschreiben die Abbildung von Ausgängen *einer* Simulationskomponente auf die Eingänge einer weiteren Simulationskomponente. Zusätzlich sind zum besseren Verständnis x, f, u und y in Abbildung 3.2 eingetragen. Bei weiteren Nutzungen der Graphen werden diese jedoch aus Gründen der Übersichtlichkeit weggelassen.

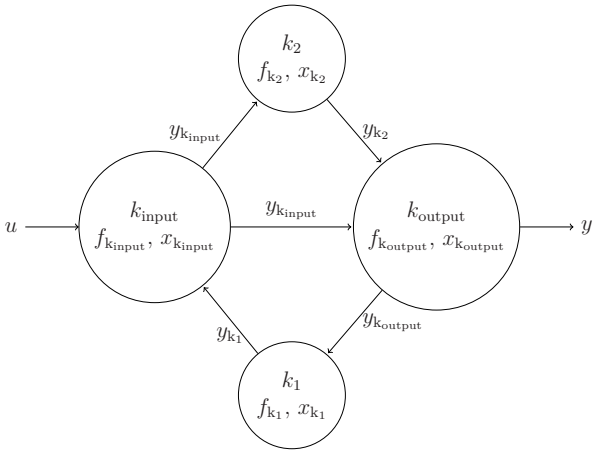


Abbildung 3.2.: Beispielgraph von vier Simulationskomponenten

3.3. Sequenzdiagramm einer Co-Simulation

In Abbildung 3.3 ist ein Sequenzdiagramm des Ablaufs einer Co-Simulation dargestellt. Dort sind Algorithmus 1 und Algorithmus 2 eingeordnet und die einzelnen Funktionsaufrufe sind zwischen Co-Simulationsmaster und Simulationskomponenten aufgeteilt.

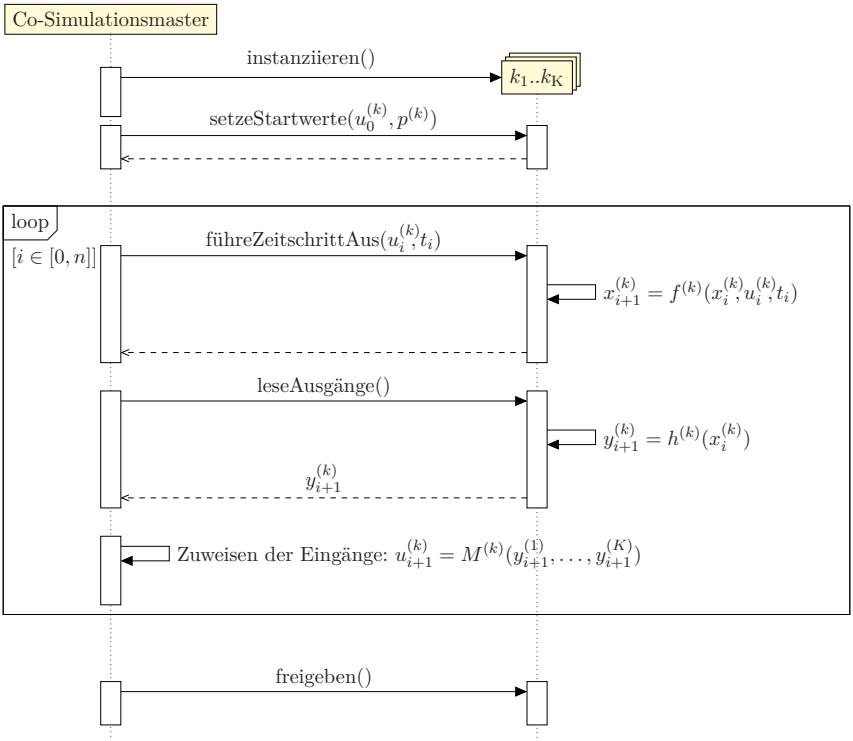


Abbildung 3.3.: Sequenzdiagramm einer Co-Simulation

4. Konzept Zustandssynchronisierung

In diesem Kapitel wird das Konzept zur Zustandssynchronisierung eingeführt. Dafür wird zunächst eine geeignete Architektur dargestellt und das Konzept daran vorgestellt. Im Anschluss wird eine Formalisierung vorgenommen wobei die Formulierung der Zustandssynchronisierung als Optimierungsproblem erfolgt. Möglichkeiten zur Optimierung werden aufgezeigt, so dass die Synchronisierung echtzeitfähig durchgeführt werden kann. Außerdem werden Aspekte der zeitlichen Synchronisierung beschrieben.

4.1. Prinzip der Zustandssynchronisierung

Die in Abschnitt 1.2 vorgestellten Probleme bei der betriebsparallelen Simulation beruhen darauf, dass die Regelungs- und Steuerungsfunktion aufgrund des fehlenden Feedbacks von der simulierten Anlage diese nicht korrekt regeln beziehungsweise steuern kann. Eine Möglichkeit wäre, eine zweite Einheit der Regelungs- und Steuerungsfunktion zu verwenden, um die simulierte Anlage zu beeinflussen. Dieses entspräche einer zweiten SPS. Beide Exemplare der Regelungs- und Steuerungsfunktion müssten dann miteinander synchronisiert werden (Kain, Dominka u. a., 2009).

Das Ziel dieser Arbeit ist es, ohne eine zweite Einheit der Regelungs- und Steuerungsfunktion auszukommen. Dazu wird ein zusätzlicher Regelungs- und Steuerungsfunktionsbaustein in die Co-Simulation eingebracht. Dieser wird hier *Adaption* genannt und als Simulationskomponente realisiert. Er enthält nicht eine Kopie der Regelungs- und Steuerungsfunktion, sondern enthält einen generischen Algorithmus, der auf Basis der Signale der Regelungs- und Steuerungsfunktion funktioniert. Statt das Stellsignal direkt von der Kommunikationssimulationskomponente an die simulierte Anlage zu geben, wird es an die Adaption gegeben, welche es dann weiterleitet. Die Adaption manipuliert das Signal so, dass der Unterschied zwischen physischer Anlage und simulierter Anlage verschwindet. In Abbildung 4.1 ist die Architektur dargestellt, mit welcher eine betriebsparallele Simulation betrieben werden kann. So können die Probleme des Beispiels in Abbildung 1.1 vermieden werden. Das bedeutet, mit jeder neuen Nachricht der Regelungs- und Steuerungsfunktion muss folgendes Problem gelöst werden:

> Suche Werte für diejenigen Eingänge der Simulationskomponenten, welche den Aktoren der physischen Anlage entsprechen, so dass $\Delta y = 0$ gilt.

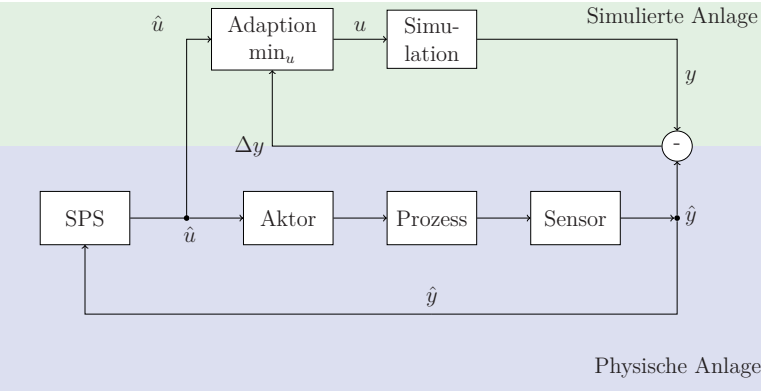


Abbildung 4.1.: Architektur für die Zustandssynchronisierung

Werden diese Werte gefunden, lassen sich die in Abschnitt 1.2 beschriebenen Probleme vermeiden. Zur Umsetzung der Suche können *Optimierungsalgorithmen* genutzt werden.

Das vorgehen kann so verstanden werden, dass in *Adaption* das Signal der Regelungs- und Steuerungsfunktion $\hat{u}$ derart angepasst wird, dass der Unterschied zwischen den Reaktionen der physischen und der simulierten Anlage minimiert wird. Die dabei zugrundeliegende Annahme ist, dass das Verhalten des Simulationsmodells der Anlage bereits eine sehr hohe Ähnlichkeit mit dem Verhalten der physischen Anlage aufweist. Diese Annahme wird dadurch gerechtfertigt, dass die simulierte Anlage jene Simulationsmodelle enthält, welche bereits für die virtuelle Inbetriebnahme zum Einsatz kommen, siehe Abschnitt 2.1. Die hohe Übereinstimmung zwischen beiden Systemen und die Gleichheit der Ausgänge sollen dafür sorgen, dass sich die physische und die simulierte Anlage im selben Arbeitspunkt befinden.

Zentraler Bestandteil ist die *Adaption* genannte Komponente, welche einen Optimierungsalgorithmus ausführt. Dieser soll die Differenz der Messwerte der physischen Anlage und der entsprechenden Ausgänge der Simulationsmodelle minimieren. Die Zielfunktion wird durch die Co-Simulation gebildet. Also Optimierungsvariablen gibt der Optimierungsalgorithmus Eingänge ausgewählter Simulationskomponenten vor. Dabei ist die oben beschriebene Vorgehensweise denkbar, die Signale der Regelungs- und Steuerungsfunktion entsprechend zu manipulieren. Alternativ ist es möglich, von den Signalen der Regelungs- oder Steuerungsfunktion unbeeinflusste Eingänge oder Parameter als Optimierungsvariablen zu nutzen um $\Delta y = 0$ zu erreichen.

Wenn jedoch $\Delta y = 0$ erreicht wird, ist der Unterschied zwischen simulierter Anlage und physischer Anlage immer null. Es können also keine Unterschiede mehr erkannt werden. Um die Verhalten von physischer und simulierter Anlage weiterhin vergleichen zu können, können diese Informationen aus der Adaption der Stellsignale gewonnen werden.

Für das Beispiel in Abschnitt 1.2, in dem das simulierte Förderband langsamer ist als das physische Förderband, gilt folgende Argumentation: Das Ausschalten des simulierten Förderbandes muss verzögert werden, genauso wie das Aktivieren des simulierten Greifers. Zur Diagnose einer möglichen Ursache könnte also wie folgt vorgegangen werden, wobei Methoden zur Diagnose kein Bestandteil dieser Arbeit sind:

Förderband später angehalten
⇒ Greifer wurde später erreicht
⇒ Simuliertes Förderband ist langsamer

4.2. Realisierung der Zustandssynchronisierung mit Hilfe eines Optimierungsalgorithmus

In diesem Abschnitt erfolgt die Formalisierung des in Abschnitt 4.1 entwickelten Konzeptes zur Zustandssynchronisierung. Dazu wird ein Optimierungsproblem formuliert.

Aus Gleichung 3.2 ist erkennbar, dass die hier betrachteten Simulationskomponenten nicht sprungfähig sind. Wird ein Eingang in Simulationskomponente k_{input} gesetzt, kann ein Ausgang der Simulationskomponente k_{output} erst nach Durchführung der Co-Simulation beeinflusst werden. Im Fall von $k_{\text{input}} = k_{\text{output}}$ ist wenigstens ein Simulationsschritt erforderlich. Falls $k_{\text{input}} \neq k_{\text{output}}$, sind mindestens zwei Simulationsschritte erforderlich. Diese Eigenschaft wird *Distanz* T_{D} genannt. Die *Distanz* ergibt sich aus dem Graph der Verbundenen Simulationskomponenten. Formal kann die *Distanz* nach (Krumke und Noltemeier, 2012) folgendermaßen ermittelt werden: Die Länge c eines Weges $\{k_1, k_2\}$ zwischen zwei Knoten wird durch Aufsummierung der Kantengewichte e_i zwischen den am Weg beteiligten Knoten ermittelt:

$$c(\{k_1, k_2\}) = \sum_{k=k_1}^{k_2} e_{\text{k}} \tag{4.1}$$

Aufgrund der Funktionsweise der Co-Simulation, wie sie Gleichung 3.2 beschrieben ist gilt, dass $e_i = 1$. Damit ergibt sich T_{D} zu:

$$\begin{aligned} T_{\text{D}} &= \inf\{c(\{u, y\})\} \\ &= \widetilde{c}(\{u, y\}) \end{aligned} \tag{4.2}$$

T_{D} entspricht der Länge des kürzesten Weges von u nach y. Die Länge eines kürzesten Weges allgemein zwischen zwei Knoten wird darüber hinaus mit $\widetilde{c}(\{\dots\})$ bezeichnet.

Der Optimierungsalgorithmus muss nicht nach *Distanz* Schritten aufhören. Das weitere Verhalten der simulierten Anlage, auch deutlich über die aktuelle physische Zeit hinaus,

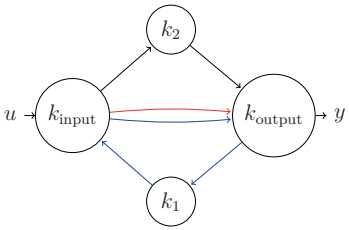


Abbildung 4.2.: Graph von vier Simulationskomponenten — roter Weg: T_{D}, blauer Weg: Vorschlag für T_{P}

kann für die Synchronisierung mit betrachtet werden. Diese Eigenschaft wird *Prädiktion* T_{P} genannt. Es gilt automatisch $T_{\text{D}} \leq T_{\text{P}}$. In Abbildung 4.2 ist ein Beispiel für einen Graphen einer Co-Simulation abgebildet. Die roten Kanten stehen für den minimalen Weg vom relevanten Eingang bis zum relevanten Ausgang der Simulationskomponenten. T_{D} wird bestimmt durch die Anzahl der Knoten entlang dieses Wegs. T_{P} könnte entsprechend der Anzahl der Knoten entlang des blauen Wegs gewählt werden.

Mit Hilfe dieser Zusammenhänge und den Definitionen aus Abschnitt 3.2 kann nun die Zielfunktion J für das Optimierungsproblem formuliert werden. Diese ist durch Gleichung 4.3 gegeben.

$$\begin{aligned} &\min_{(u_j)\in\mathcal{U}} J \\ &= \min_{(u_j)\in\mathcal{U}} \sum_{j=i+T_{\text{D}}}^{i+T_{\text{P}}} \left(y_j - \hat{y}_{(j-\Delta_{\text{o}})}\right)^2 \\ &= \min_{(u_j)\in\mathcal{U}} \sum_{j=i+T_{\text{D}}}^{i+T_{\text{P}}} \left(\Delta y_j\right)^2 \end{aligned} \tag{4.3}$$

Die Zielfunktion wird für jeden Simulationsschritt der Co-Simulation optimiert. Dabei wird die Co-Simulation für T_{P} Schritte ausgeführt. Ergebnisse der Co-Simulation vor T_{D} beeinflussen den Wert der Zielfunktion nicht. Ergebnisse vor T_{D} können aufgrund des Co-Simulationsmasteralgorithmus nicht durch die vom Optimierungsalgorithmus angelegten Eingänge verursacht sein. Anwendungsfälle für die Wahl $T_{\text{P}} > T_{\text{D}}$ können sein: Berücksichtigung von Schleifen im Co-Simulationsgraph, mehrere Wege von k_{input} nach k_{output}, der Einsatz für ereignisdiskreten Systeme und die Verringerung von Schwingungen. Wird $T_{\text{P}} > T_{\text{D}}$ gewählt, gibt es zwei Möglichkeiten, die Optimierung durchzuführen: Es kann weiterhin eine statische Optimierung durchgeführt werden. Bei dieser wird ein u gesucht, welches die Zielfunktion über mehrere Simulationsschritte hinweg minimiert. Alternativ kann eine dynamische Optimierung durchgeführt werden. Dann wird eine Folge (u_j) gesucht, welche die Zielfunktion minimiert.

Im Falle des statischen Optimierungsproblems wird in Gleichung 4.3 eine konstante

Folge (u_j) betrachtet, was als u geschrieben werden kann. Zur Lösung des dynamischen Optimierungsproblems kann die Zielfunktion von Gleichung 4.3 in die folgende Form umformuliert werden.

$$\min_{\begin{bmatrix} u_{i+T_\mathrm{D}} \\ u_{i+T_\mathrm{D}+1} \\ \cdots \\ u_{i+T_\mathrm{P}} \end{bmatrix} \in \mathcal{U}^{T_\mathrm{P}-T_\mathrm{D}+1}} \left\{ (\Delta y_{i+T_\mathrm{D}})^2 + (\Delta y_{i+T_\mathrm{D}+1})^2 + \cdots + (\Delta y_{i+T_\mathrm{P}})^2 \right\} \tag{4.4}$$

Pro Variable zur Optimierung wird stattdessen für einen Vektor der Länge $T_\mathrm{P} - T_\mathrm{D} + 1$ optimiert.

Der Fehler wird aus den Ergebnissen der Simulation y und den entsprechenden Messungen $\hat{y}$ berechnet. Die entsprechenden Messungen finden sich nicht zwangsläufig zur Zeit j aufgrund der unterschiedlichen Totzeiten innerhalb der physischen Anlage und der Co-Simulation. Daher wird die Messung $\hat{y}$ zum Zeitpunkt $j - \Delta_\mathrm{o}$ genommen. Δ_o soll den Unterschied der Totzeiten kompensieren. In Abschnitt 4.6 wird dieser Aspekt genauer untersucht.

Das Abbruchkriterium für den Optimierungsalgorithmus tritt ein, wenn die Differenz zwischen physischer und simulierter Anlage klein genug ist, also $\sum_{j=i+T_\mathrm{D}}^{i+T_\mathrm{P}} (\Delta y_j)^2 < \varepsilon$ gilt. Der Wert für ε muss entsprechend parametriert werden.

Durch die Formulierung der Zustandssynchronisierung als Optimierungsproblem ergibt sich der Vorteil, dass bei Bedarf zusätzliche Anforderungen leicht berücksichtigt werden können: Zum Beispiel kann die Zielfunktion angepasst werden, falls Nebenbedingungen berücksichtigt werden sollen. So können obere und untere Schranken für Variablen festgelegt werden, falls zu niedrige oder zu hohe Werte für die Simulationskomponente Probleme darstellen. Zudem lassen sich leicht weitere Strafterme zur Zielfunktion hinzufügen, zum Beispiel um $\Delta u = u - \hat{u}$ möglichst klein zu halten.

4.3. Einbetten der Optimierung in den Co-Simulationsmasteralgorithmus

Nachfolgend wird der Algorithmus für die Durchführung von einem Schritt der Co-Simulation mit Zustandssynchronisierung beschrieben. In diesem Zusammenhang gibt es drei Arten von Iterationen. Bei der *Iteration der Co-Simulation* werden alle Simulationskomponenten für einen Zeitschritt berechnet. Dabei schreitet die Simulationszeit voran. Bei der *Iteration des Optimierungsalgorithmus* wird die Optimierung ausgeführt. Dabei werden Eingänge gesetzt. Dann werden pro Iteration des Optimierungsalgorithmus T_P Iterationen der Co-Simulation ausgeführt. Anschließend werden die Ausgänge gelesen

und bewertet, ob das Optimum erreicht ist. Nach jeder Iteration des Optimierungsalgorithmus wird die Co-Simulation zurückgesetzt auf den Zustand vor der Iteration des Optimierungsalgorithmus. Bei einer *Iteration der betriebsparallelen Simulation* schreitet diese in Echtzeit voran. Mit jeder Iteration der betriebsparallelen Simulation müssen so viele Iterationen des Optimierungsalgorithmus ausgeführt werden, bis die optimale Zustandssynchronisierung erreicht ist, oder bis abgebrochen wird. Ohne Zustandssynchronisierung entspricht eine Iteration der betriebsparallelen Simulation einer Iteration der Co-Simulation.

Algorithmus 3 Algorithmus Co-Simulationsmaster mit Zustandssynchronisierung

```
for all k ∈ K do
    INITIALISIERE(u_0^(k), p^(k))                ▷Setzen der Startwerte und Parameter
end for
for all i ∈ [0, n] do
    SPEICHERE DEN ZUSTAND ALLER SIMULATIONSKOMPONENTEN
    INITIALISIERE DEN OPTIMIERUNGSALGORITHMUS(U_i, ŷ_i)
    optimal ← Δy_i < ε
    while nicht optimal do
        for all j ∈ [i, T_D) do                  ▷Beeinflussen Ergebnis der Optimierung nicht
            SIMULATIONSSCHRITT(j)
        end for
        for all j ∈ [T_D, T_P) do                ▷Beeinflussen Ergebnis der Optimierung
            SIMULATIONSSCHRITT(j)
            Δy_j ← y_j − ŷ_{j−Δ_o}               ▷Ermittle Vektor der Differenzen Simulation — Anlage
        end for
        optimal ← Σ_{j=i+T_D}^{i+T_P} (Δy_j)^2 < ε
        if nicht optimal then
            LADE GESPEICHERTEN ZUSTAND
            u_m ←ITERATION DES OPTIMIERERS(Σ_{j=i+T_D}^{i+T_P} (Δy_j)^2)
        end if
    end while
    LADE GESPEICHERTEN ZUSTAND
    SIMULATIONSSCHRITT(i)                        ▷Simulationsschritt i mit optimalen u
end for
```

Der Algorithmus 3 beschreibt eine Iteration der betriebsparallelen Simulation zur Umsetzung der Optimierung zur Zustandssynchronisierung. Algorithmus 3 erweitert Algorithmus 2. Der Algorithmus des Co-Simulationsmasters in Algorithmus 3 ist so aufgebaut, dass für die Zustandssynchronisierung die folgende Schnittstelle zu einem Optimierer existieren muss:

- Initialisierung des Optimierungsalgorithmus
- Austausch von Variablen zwischen Optimierungsalgorithmus und Co-Simulation
- Ausführen einer Iteration des Optimierungsalgorithmus

Es bietet sich an, den Optimierer als Simulationskomponente zu realisieren. So wird der Austausch von Variablen zwischen Optimierungsalgorithmus und den weiteren Simulationskomponenten vom Co-Simulationsmaster verwaltet. Die Initialisierung und Ausführung der Optimierung sowie das Prüfen des Abbruchkriteriums kann durch Variablen abgebildet werden. Den Optimierer als Simulationskomponente zu realisieren hat den Vorteil, dass verschiedene Optimierungsalgorithmen und Synchronisierungsstrategien genutzt werden können. Je nach Problem kann die passende Strategie ausgewählt werden, ohne die restliche Co-Simulation zu ändern. Ein Nachteil ist, dass die Schnittstelle zwischen Co-Simulationsmaster und Optimierer exakt bekannt sein muss, zum Beispiel: Was sind die Namen der Variablen? Wird die Initialisierung durch eine steigende Flanke ausgelöst?

Die Schnittstelle ist jedoch so allgemein, dass eine weitere Möglichkeit der Realisierung darin besteht, in einer Simulationskomponente des Optimierers mehrere Optimierungsalgorithmen zu integrieren. Der letztendlich eingesetzte Optimierungsalgorithmus kann dann durch eine Variable ausgewählt werden, welche wie alle anderen Parameter zu Beginn der Co-Simulation gesetzt wird.

In Abbildung 4.3 wird ein Sequenzdiagramm der Co-Simulation mit Zustandssynchronisierung gezeigt. Dabei ist der Optimierer als separate Simulationskomponente realisiert. Daher werden alle Funktionsaufrufe an die Simulationskomponenten auch an den Optimierer gesendet. Aus Gründen der Übersichtlichkeit werden die meisten dieser Funktionsaufrufe im Sequenzdiagramm allerdings ausgelassen. Die Simulationskomponente des Optimierers wird bei diesen Funktionsaufrufen keine Aktion durchführen.

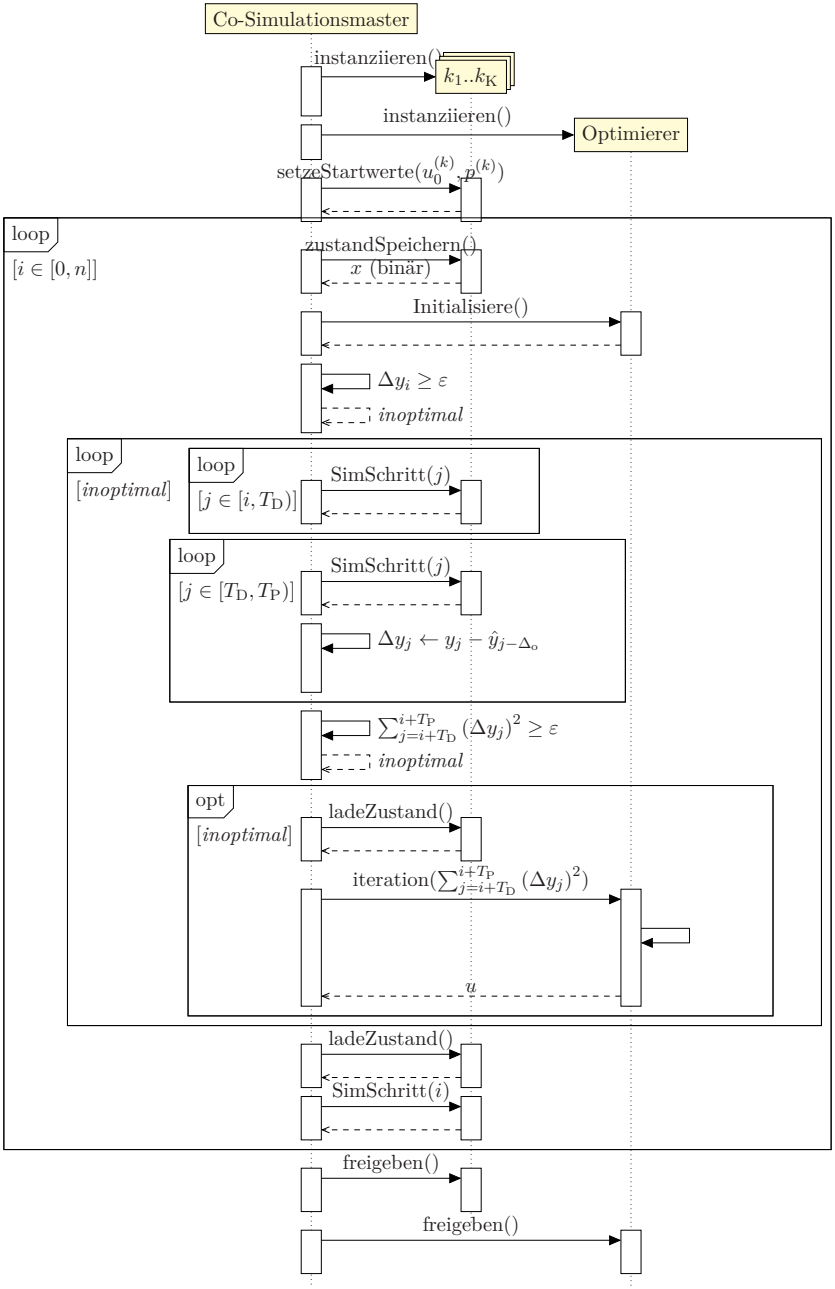


Abbildung 4.3.: Sequenzdiagramm der Co-Simulation mit Zustandssynchronisierung

4.4. Verbesserung der Performance der Zustandssynchronisierung

Die Anwendung von Optimierung zur Zustandssynchronisierung führt zwangsläufig zu einer höheren Zahl von notwendigen Simulationsschritten, da für die Co-Simulation mehrere Iterationen pro Iteration des Optimierungsalgorithmus durchgeführt werden müssen.

Um die benötigte Zeit zur Durchführung der Optimierung zu reduzieren gibt es zwei Möglichkeiten: Verringerung der Anzahl der notwendigen Iterationen und Verringerung der Zeit pro Iteration. Die nachfolgenden Betrachtungen wurden erstmals in (Zipper und Diedrich, 2020) veröffentlicht.

Die Zahl der Iterationen der Co-Simulation ist im Allgemeinen vom Optimierungsalgorithmus abhängig. Die Wahl des Optimierungsalgorithmus und der Startwerte ist daher wichtig und von den konkreten Simulationsmodellen und der physischen Anlage abhängig. Zu beachten ist, dass Optimierungsalgorithmen, welche eine Ableitung der Zielfunktion benötigen, zu zusätzlichen Iterationen der Co-Simulation führen. Unterschiede zwischen physischer und simulierter Anlage können auch durch Rauschen der betrachteten Signale zustande kommen. Die Zahl der Iterationen des Optimierungsalgorithmus lässt sich beispielsweise durch Glätten der Signale reduzieren.

In diesem Abschnitt 4.4 sollen Möglichkeiten aufgezeigt werden, die Anzahl der Iterationen der Co-Simulation zu verringern, ohne das Ergebnis der Zustandssynchronisierung zu beeinflussen oder zu ändern. Diese Performance-Steigerungen funktionieren unabhängig vom gewählten Optimierungsalgorithmus und dem Verhalten der Simulationsmodelle.

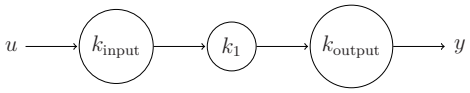
Die Laufzeit einer Blackbox Simulationskomponente lässt sich nicht beeinflussen. Für eine Iteration des Optimierungsalgorithmus müssen mehrere (insgesamt T_P) Iterationen der Co-Simulation ausgeführt werden. In jeder Iteration der Co-Simulation müssen alle (K) Simulationskomponenten ausgeführt werden, insgesamt sind dies $T_\mathrm{P} \cdot K$ Iterationen der Co-Simulation. Um diese Zahl zu reduzieren werden die nachstehenden Fragen beantwortet:

1. Unter welchen Bedingungen kann die Ausführung einer Simulationskomponente das Ergebnis der Optimierung beeinflussen?

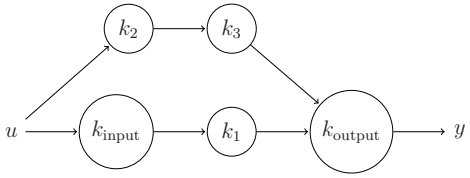
2. Unter welchen Bedingungen liefern zwei Ausführungen einer Simulationskomponente das identische Ergebnis?

Mit Hilfe dieser Fragen lassen sich zwei, sich ergänzende, Ansätze ableiten.

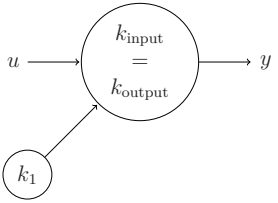
Für die Anzahl der Ausführungen von Simulationskomponenten während der Optimierung sind die Teilgraphen der Co-Simulation relevant, welche Simulationskomponenten enthalten, die durch die Optimierung beeinflusst werden. In Abbildung 4.4 werden dazu einige Beispiele von Co-Simulationssetups vorgestellt, welche die möglichen Strukturen eines Co-Simulationsgraphen darstellen.



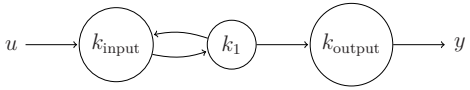
(a) Sequenzielle Anordnung von Simulationskomponenten



(b) Parallele Anordnung von Simulationskomponenten



(c) Von u nicht beeinflusste Knoten



(d) Zyklus von Simulationskomponenten

Abbildung 4.4.: Beispiele von Co-Simulationen

Abbildung 4.4a zeigt ein Beispiel für eine Co-Simulation, in der die Simulationskomponenten zwischen u und y sequenziell angeordnet sind. Nach Gleichung 4.1 gilt $T_{\text{D}} = 3$, da sich zwischen u und y drei Simulationskomponenten befinden. Daher müssen drei Iterationen der Co-Simulation ausgeführt werden wobei mit jeder Iteration der Co-Simulation jede der Simulationskomponenten ausgeführt werden muss. Daher ist die Gesamtzahl der zusätzlichen Ausführungen von Simulationskomponenten pro Iteration des Optimierungsalgorithmus $3 \cdot 3 = 9 = T_{\text{D}}^2$. T_{D} wirkt sich also quadratisch auf die Zahl der zusätzlichen Ausführungen von Simulationskomponenten aus. Die Skalierbarkeit der Zustandssynchronisierung wird dadurch beeinträchtigt, denn die benötigte Rechenzeit für die Zustandssynchronisierung steigt mit zunehmende Zahl der Simulationskomponenten in der Co-Simulation. Ein praktisches Beispiel für solch eine Struktur findet sich, wenn sich in der Signalkette zwischen u und y mehrere Simulationskomponenten befinden.

Abbildung 4.4b zeigt ein Beispiel für eine Co-Simulation, in dem mehrere Wege von u nach y existieren. Nach Gleichung 4.1 ergibt sich $T_{\text{D}} = 3$, da der längste Weg von u nach y drei Simulationskomponenten umfasst, über k_{input}, k_1 und k_{output}. Daher müssen drei Iterationen der Co-Simulation pro Iteration des Optimierungsalgorithmus ausgeführt werden, bei jeder Iteration der Co-Simulation müssen fünf Simulationskomponenten aus-

geführt werden. Die Zahl der zusätzlichen Ausführungen von Simulationskomponenten ergibt sich also zu $3 \cdot 5 = 15$. Parallele Wege gehen also linear ein. Im praktischen Anwendungsfall finden sich solche Strukturen, falls die Zustandssynchronisierung Eingänge oder Parameter mehrerer Simulationskomponenten betrachten soll.

Abbildung 4.4c zeigt ein Beispiel einer Co-Simulation, in welcher sich Simulationskomponenten befinden, die nicht vom Optimierungsalgorithmus beeinflusst werden. Nach Gleichung 4.1 ist $T_{\mathrm{D}} = 1$. Mit jeder Iteration der Co-Simulation müssen zwei Simulationskomponenten ausgeführt werden, die zusätzliche Zahl an Ausführungen von Simulationskomponenten pro Iteration des Optimierungsalgorithmus ergibt sich daher zu $1 \cdot 2 = 2$. Mit u nicht verbundene Simulationskomponenten gehen linear ein. In der Praxis existieren solche Strukturen beispielsweise durch Simulationskomponenten zur Datenaufnahme aus dem physischen System.

In Abbildung 4.4d ist ein Beispiel einer Co-Simulation mit Zyklen dargestellt. Auch hier ergibt Gleichung 4.1 $T_{\mathrm{D}} = 3$. Die Zahl der zusätzlichen Ausführungen pro Iteration des Optimierungsalgorithmus ist also gleich wie im Beispiel aus Abbildung 4.4a. Im praktischen Einsatz lässt sich eine solche Co-Simulation beispielsweise für Systeme finden, in denen ein Regler eine weitere Komponente regelt, oder bei der bidirektionalen Kopplung eines Verhaltensmodells einer mechatronischen Komponente und einer 3D- und Physikengine.

Die Reduzierung der Ausführungen der Simulationskomponenten soll die an den Beispielen gezeigten, zusätzlichen Ausführungen reduzieren und die folgenden Randbedingungen beachten:

1. Es werden keine Maßnahmen genutzt, welche das Ergebnis einer oder mehrerer Simulationskomponenten schätzen oder lernen.

2. Wird ein Ergebnis einer Simulationskomponente zu einem Zeitpunkt benötigt, muss diese für diesen Zeitpunkt und alle vorhergehenden Zeitpunkte ausgeführt werden. Grund hierfür ist der beliebige, unbekannte, interne Zustand der Simulationskomponente.

3. Die Performance-Optimierungen setzen kein bestimmtes Verhalten der Simulationskomponenten voraus.

4. Die Performance-Optimierungen sind unabhängig vom Optimierungsalgorithmus.

4.4.1. Ansatz 1: Unter welchen Bedingungen kann die Ausführung einer Simulationskomponente das Ergebnis der Optimierung beeinflussen?

Diese Frage zielt darauf ab, die Ausführung jener Simulationskomponenten zu vermeiden, deren Ausgänge das Ergebnis der Iteration des Optimierungsalgorithmus (y) bis zum

Erreichen von T_P nicht mehr beeinflussen können.

Ein Beispielszenario hierfür ist in Abbildung 4.4c abgebildet. In diesem Beispiel ist eine Co-Simulation bestehend aus zwei Simulationskomponenten abgebildet. Es gelte $T_\mathrm{D} = T_\mathrm{P} = 1$, d.h. für eine Iteration des Optimierungsalgorithmus ist auch eine Iteration der Co-Simulation notwendig, denn nach einer Ausführung von k_input wirkt u bereits auf y. Da die Ausgänge von k_1 die Eingänge von k_input sind, brauchen sie zwei Iterationen bis sie y beeinflussen. Während der Optimierung können also die Ausgänge von k_1 y nicht beeinflussen, es kann daher auf die Ausführung von k_1 verzichtet werden.

Diese Argumentation kann auch auf die weiteren Beispiele von Abbildung 4.4 angewendet werden. In Beispiel aus Abbildung 4.4a ist eine Co-Simulation bestehend aus drei Simulationskomponenten abgebildet. Es gelte $T_\mathrm{D} = T_\mathrm{P} = 3$, d.h. für eine Iteration des Optimierungsalgorithmus sind drei Iterationen der Co-Simulation notwendig. Das bedeutet, u braucht drei Iterationen bis es y beeinflusst hat, die Ausgänge von k_input brauchen zwei Iterationen und die Ausgänge von k_1 eine Iteration. Sobald also bereits eine Iteration der Co-Simulation ausgeführt wurde, sind nur noch zwei weitere bis T_P zu erledigen. Da die in der zweiten Iteration entstehenden Ausgänge von k_input jedoch zusätzlich noch zwei Iterationen benötigen, können diese y nicht mehr beeinflussen. Das bedeutet, in den zwei verbleibenden Iterationen der Co-Simulation kann k_input ignoriert werden ohne das Ergebnis dieser Iteration des Optimierungsalgorithmus zu verfälschen. Analog dazu kann mit k_1 ab der zweiten Iteration der Co-Simulation verfahren werden. Dadurch sinkt die Zahl der durchzuführenden Iterationen der Co-Simulation pro Iteration der Optimierung von $T_\mathrm{P} \cdot K = 9$ auf $3 + 2 + 1 = 6$.

Mit Gleichung 4.1 zur Beschreibung der Länge eines Weges in einem Graphen kann für eine Simulationskomponente die maximale Anzahl an zusätzlichen Ausführungen pro Iteration der Co-Simulation (Γ_k) wie in Gleichung 4.5 ausgedrückt werden.

$$\Gamma_\mathrm{k} = \begin{cases} T_\mathrm{P} - \widetilde{c}(\{k, k_\mathrm{output}\}) & k_\mathrm{output} \text{ von } k \text{ erreichbar} \\ 0 & \text{sonst} \end{cases} \tag{4.5}$$

4.4.2. Ansatz 2: Unter welchen Bedingungen liefern zwei Ausführungen einer Simulationskomponente das identische Ergebnis?

Diese Frage zielt darauf ab, Informationen aus vorrangegangen Iterationen der Co-Simulation zu nutzen und so die wiederholte Ausführung von Simulationskomponenten zu vermeiden.

In jeder Iteration der Co-Simulation werden die Ausgänge auf die entsprechenden Eingänge übertragen, siehe Algorithmus 2. So ergibt sich über mehrere Iterationen der Co-Simulation hinweg eine Folge von Eingängen in eine Simulationskomponente. Eine Simu-

lationskomponente wird bei gleichem Anfangszustand und gleicher Folge von Eingängen auch immer dieselbe Folge von Ausgängen produzieren. Wird die Folge der Eingänge und die daraus resultierende Folge von Ausgängen gespeichert, kann dies für eine Steigerung der Performance genutzt werden: Tritt in einer Iteration des Optimierungsalgorithmus die gleiche Folge von Eingängen auf, wie in einer vorhergehenden Iteration des Optimierungsalgorithmus, kann auf die Ausführung der Simulationskomponente verzichtet werden und stattdessen die gespeicherten Werte an die nachfolgenden Simulationskomponenten kommuniziert werden.

Sobald sich die Folge von Eingängen aber in mindestens einem Wert unterscheidet, muss die Simulationskomponente für alle Iterationen der Co-Simulation ausgeführt werden. Erst bei der letzten Iteration der Co-Simulation ist die komplette, aktuelle Folge der Eingänge bekannt. Soll also dieser Ansatz verwendet werden, muss die in Algorithmus 3 vorgestellte Abarbeitung der Co-Simulation dahingehend angepasst werden.

Eine Simulationskomponente wird während einer Iteration der Co-Simulation nur dann ausgeführt, wenn sich die bis zu dieser Iteration der Co-Simulation entstandene Teilfolge von Eingängen von allen gespeicherten Teilfolgen der gleichen Länge unterscheidet. Tritt ein Unterschied erst ab einer späteren Iteration der Co-Simulation auf, müssen die zuvor übersprungenen Iteration der Co-Simulation für diese Simulationskomponente nachgeholt werden. Dafür wird eine bis zum Auftreten des Unterschieds passende Folge von Eingängen genutzt. Es ist ausreichend, die Ausführungen nur für die betreffende Simulationskomponente nachzuholen, da deren Ausgänge für die nachzuholenden Ausführungen bereits bekannt sind und an die weiteren Simulationskomponenten kommuniziert wurden. Beim Nachholen der Ausführungen geht es alleine darum, den internen Zustand der betreffenden Simulationskomponente korrekt zu aktualisieren.

Werden mit der Folge der Eingänge neben den Ausgängen zusätzlich auch die internen Zustände gespeichert, lässt sich das Nachholen der Ausführungen umgehen: Statt eine Ausführung einer Simulationskomponente nachzuholen, wird deren interner Zustand entsprechend des zuvor gespeicherten, aus der gleichen Folge von Eingängen resultierenden, Zustandes gesetzt.

Zur Veranschaulichung hierfür kommt das in Abbildung 4.4a gezeigte Beispiel zum Einsatz. In Tabelle 4.1 wird das Vorgehen für eine Simulationskomponente veranschaulicht: Zu einem Zeitpunkt während der Simulation wird der Optimierungsalgorithmus für sechs Iterationen ausgeführt, siehe Spalte n. In den Spalten $u_{\mathrm{t}=1}$, $u_{\mathrm{t}=2}$ und $u_{\mathrm{t}=3}$ sind die Eingänge für die Iterationen der Co-Simulation dargestellt, zur Verdeutlichung seien in diesem Beispiel nur die Werte $\circ$ und $\star$ möglich.

Nachfolgend wird anhand von Tabelle 4.1 erläutert, aus welchem Grund die Simulationskomponente wie oft ausgeführt werden muss, in Abhängigkeit von der Folge der Eingänge.

1. In der ersten Iteration des Optimierungsalgorithmus muss die Simulationskompo-

Tabelle 4.1.: Veranschaulichung Ansatz 2 für Beispiel aus Abbildung 4.4a

n	$u_{t=1}$	$u_{t=2}$	$u_{t=3}$	Ausführungen	Ausführungen (Zustände speichern)
1	○	○	○	3	3
2	⋆	⋆	⋆	3	3
3	○	○	○	0	0
4	○	⋆	⋆	3	2
5	⋆	○	⋆	3	2
6	⋆	⋆	○	3	1

nente für jede Iteration der Co-Simulation ausgeführt werden, also dreimal. Da dies die erste Iteration ist, wurde noch keine Folge von Eingängen gespeichert.

2. Auch in der zweiten Iteration des Optimierungsalgorithmus muss die Simulationskomponente dreimal ausgeführt werden, da die komplette Folge der Eingänge zur ersten Iteration verschieden ist.

3. In der dritten Iteration des Optimierungsalgorithmus muss die Simulationskomponente nicht ausgeführt werden, da die komplette Folge der Eingänge der Folge der ersten Iteration des Optimierungsalgorithmus gleicht.

4. In der vierten Iteration des Optimierungsalgorithmus muss die Simulationskomponente dreimal ausgeführt werden, da die Folge der Eingänge teilweise abweicht. Auch wenn in der ersten Iteration der Co-Simulation die Eingänge noch gleich sind, muss die Simulationskomponente trotzdem auch in der ersten Iteration der Co-Simulation ausgeführt werden. Der Grund dafür ist, dass bei dieser Ausführung der interne Zustand der Simulationskomponente verändert wird, so dass die Iterationen zwei und drei der Co-Simulation andere Ergebnisse liefern würden. Durch die Nutzung des zuvor gespeicherten internen Zustands kann die Anzahl der Ausführungen auf 2 reduziert werden.

5. Analog verhält es sich für die Iteration fünf des Optimierungsalgorithmus, so dass dort die Simulationskomponente dreimal ausgeführt werden muss. Durch die Nutzung des zuvor gespeicherten internen Zustands kann die Anzahl der Ausführungen auf 2 reduziert werden.

6. Analog verhält es sich für die Iteration sechs des Optimierungsalgorithmus, so dass dort die Simulationskomponente dreimal ausgeführt werden muss. Durch die Nutzung des zuvor gespeicherten internen Zustands kann die Anzahl der Ausführungen auf 1 reduziert werden.

Dieser Ansatz funktioniert zur Verbesserung der Performance der Optimierung sowohl

aufgrund der Daten, als auch aufgrund der Struktur der Co-Simulation. Nachfolgend werden die durch die Struktur bedingten Verbesserungspotenziale vorgestellt.

Vom Optimierungsalgorithmus unbeeinflusste Simulationskomponenten

Ein Beispiel für die Anwendung dieses Ansatzes für eine Verbesserung der Performance aufgrund der Struktur der Co-Simulation ist in Abbildung 4.4c dargestellt. Der Optimierungsalgorithmus versucht durch Variation von u ein optimales y zu finden. Die Simulationskomponente k_1 ist nicht direkt oder indirekt von u beeinflusst. Dadurch führt die Ausführung dieser Simulationskomponente in jeder Iteration des Optimierungsalgorithmus zu denselben Ausgängen. In einer so strukturierten Co-Simulation müssen die von u nicht erreichbaren Simulationskomponenten demnach nur bei der ersten Iteration des Optimierungsalgorithmus ausgeführt werden. Darüber hinaus entfällt für diese Simulationskomponenten die Anforderung, dass der interne Zustand gespeichert werden muss.

Redundante Ausführungen aufgrund der Distanz

Ein weiteres Potenzial die Performance zu verbessern bringt der Ansatz, wenn $T_\mathrm{D} > 1$ ist. Für eine Simulationskomponente k wird ihr Eingang vom Optimierer erst ab der $(\widetilde{c}(\{k_\mathrm{input}, k\}) + 1)$-ten Iteration aktiv (siehe Gleichung 4.1). Das bedeutet, dass in jeder Iteration des Optimierers die ersten

$$\widetilde{c}(\{k_\mathrm{input}, k\}) \tag{4.6}$$

Ausführungen identisch sind. Demnach können nach der ersten Iteration des Optimierungsalgorithmus für einen Zeitschritt die Ergebnisse für diese Ausführungen nachgenutzt werden, da sie als wiederholte Folge von Eingängen erkannt werden. Dies kann am Beispiel Abbildung 4.4a, Simulationskomponente k_output nachvollzogen werden: In einer Iteration des Optimierungsalgorithmus hat u in k_output erst bei der dritten Iteration der Co-Simulation einen Effekt. Generell kann aufgrund des internen Zustands nicht auf die ersten zwei Iterationen der Co-Simulation verzichtet werden. Jedoch sind die Eingänge von k_output in den ersten zwei Iteration der Co-Simulation immer gleich. Deren Werte hängen von den vorhergehenden Iterationen der betriebsparallelen Simulation ab. Daher sind die Ergebnisse — Ausgänge und interne Zustände — noch gespeichert aufgrund von Ansatz 2. Die ersten zwei Ausführungen können daher übersprungen werden.

Redundante Ausführungen aufgrund dynamischer Optimierung

Analog zur vorrangegangen Argumentation lassen sich Ausführungen der Simulationskomponenten bei der Nutzung von dynamischer Optimierung einsparen: Mit $T_\mathrm{P} > T_\mathrm{D}$ wird aus Sicht der aktuellen Simulationszeit das zukünftige Verhalten simuliert. Es besteht die

Möglichkeit, dass die so erhaltenen Folgen von Eingängen in einer darauffolgenden Iteration der betriebsparallelen Simulation erneut auftreten. Ob dies tatsächlich geschieht, hängt vom Verhalten der physischen Anlage und der Datenerfassung ab: Wird während der Prädiktion ein Datenabgriff erlaubt, also auf zu den Simulationszeiten passende Werte der physischen Anlage gewartet, kommen die prädizierten Folgen von Eingängen garantiert in der darauffolgenden Iteration der betriebsparallelen Simulation vor. Es lassen sich dann die ersten $T_\mathrm{P} - T_\mathrm{D}$ Ausführungen von Simulationskomponenten einsparen. Wird nicht auf die physische Anlage gewartet, sondern der letzte Messwert der physischen Anlage genutzt während der Prädiktion, sind Einsparungen vom Verhalten der physischen Anlage abhängig. Einsparungen können dann beispielsweise auftreten, wenn die Makroschrittweite der Co-Simulation kleiner ist als die Abtastrate der Datenerfassung am physischen Prozess.

4.4.3. Kombination von Ansatz 1 und Ansatz 2

Die Kombination der vorgestellten Ansätze, welche in (Zipper und Diedrich, 2020) erstmalig vorgestellt wurde, wird hier noch einmal beschrieben. Diese Kombination ermöglicht es, die Ausführungen der Simulationskomponenten während der Optimierung deutlich zu reduzieren, ohne das Verhalten zu beeinflussen. Anhand der Beispiele aus Abbildung 4.4 kann nachvollzogen werden, dass die Anzahl der Ausführungen von Simulationskomponente während der Optimierung mindestens quadratisch mit T_D anwächst.

Durch die Kombination von Gleichung 4.5 und Gleichung 4.6 kann diese Zahl auf exakt eine Ausführung pro Simulationskomponente reduziert werden für $T_\mathrm{P} = T_\mathrm{D}$ und je eine weitere Ausführung für $T_\mathrm{P} > T_\mathrm{D}$. Die Anzahl ist daher ab der zweiten Iteration des Optimierungsalgorithmus linear in T_D. Das kann durch folgende Überlegungen nachvollzogen werden:

Nach dem Optimalitätsprinzip von Bellman (Bellman und Dreyfus, 1962) kann der kürzeste Weg in einem Graph durch eine Summe aus den kürzesten Teilstrecken ausgedrückt werden. Für die Position von k im Graph gibt es exakt drei Möglichkeiten:

1. k liegt auf gar keinen Weg $\{k_\mathrm{input}, k_\mathrm{output}\}$. Dann muss es aufgrund mangelnden Einflusses auf die Zielfunktion oder mangelnder Beeinflussbarkeit durch den Optimierungsalgorithmus spätestens ab der zweiten Iteration des Optimierungsalgorithmus nicht mehr ausgeführt werden.

2. k liegt auf dem kürzesten Weg $\{k_\mathrm{input}, k_\mathrm{output}\}$. Dann entspricht die Summe der Längen der Wege $\{k_\mathrm{input}, k\}$ und $\{k, k_\mathrm{output}\}$ gleich der Länge des kürzesten Weges $\{k_\mathrm{input}, k_\mathrm{output}\}$.

$$\widetilde{c}(\{k_\mathrm{input}, k_\mathrm{output}\}) = \widetilde{c}(\{k_\mathrm{input}, k\}) + \widetilde{c}(\{k, k_\mathrm{output}\})$$

3. k liegt auf einem längeren Weg als dem kürzesten Weg $\{k_\mathrm{input}, k_\mathrm{output}\}$. Dieser längere

Weg ist um C_k länger als der kürzeste Weg $\{k_\mathrm{input}, k_\mathrm{output}\}$. Dann gilt

$$\widetilde{c}(\{k_\mathrm{input}, k_\mathrm{output}\}) + C_\mathrm{k} = \widetilde{c}(\{k_\mathrm{input}, k\}) + \widetilde{c}(\{k, k_\mathrm{output}\})$$
$$\Rightarrow$$
$$\widetilde{c}(\{k_\mathrm{input}, k_\mathrm{output}\}) = \widetilde{c}(\{k_\mathrm{input}, k\}) + \widetilde{c}(\{k, k_\mathrm{output}\}) - C_\mathrm{k}$$

Γ_k aus Gleichung 4.5 gibt die aus Ansatz 1 folgende maximale Zahl an Ausführungen einer Simulationskomponente pro Iteration des Optimierungsalgorithmus an. Ansatz 2, Gleichung 4.6, stellt ein Maß für redundante Ausführungen dar. Die Differenz aus beiden bildet die maximale Anzahl an Ausführungen durch die Kombination von Ansatz 1 und Ansatz 2.

Maximale Anzahl von Ausführungen nach Ansatz 1 und Ansatz 2

$$= \Gamma_\mathrm{k} - \widetilde{c}(\{k_\mathrm{input}, k\}) \tag{4.7}$$
$$= T_\mathrm{P} - \widetilde{c}(\{k, k_\mathrm{output}\}) - \widetilde{c}(\{k_\mathrm{input}, k\}) \tag{4.8}$$
$$= (T_\mathrm{P} - T_\mathrm{D}) + T_\mathrm{D} - \widetilde{c}(\{k, k_\mathrm{output}\}) - \widetilde{c}(\{k_\mathrm{input}, k\}) \tag{4.9}$$
$$= (T_\mathrm{P} - T_\mathrm{D}) + \widetilde{c}(\{u, y\}) - \widetilde{c}(\{k, k_\mathrm{output}\}) - \widetilde{c}(\{k_\mathrm{input}, k\}) \tag{4.10}$$
$$= (T_\mathrm{P} - T_\mathrm{D}) + 1 + \widetilde{c}(\{k_\mathrm{input}, k_\mathrm{output}\}) - \widetilde{c}(\{k, k_\mathrm{output}\}) - \widetilde{c}(\{k_\mathrm{input}, k\}) \tag{4.11}$$
$$= (T_\mathrm{P} - T_\mathrm{D}) + 1 + \widetilde{c}(\{k_\mathrm{input}, k\}) + \widetilde{c}(\{k, k_\mathrm{output}\}) - C_\mathrm{k} - \widetilde{c}(\{k, k_\mathrm{output}\}) \quad - \widetilde{c}(\{k_\mathrm{input}, k\}) \tag{4.12}$$
$$= (T_\mathrm{P} - T_\mathrm{D}) + 1 - C_\mathrm{k} \tag{4.13}$$

Der kürzeste Weg $\{u, y\}$ entspricht dem kürzesten Weg von $\{k_\mathrm{input}, k_\mathrm{output}\} + 1$, denn von k_output nach y ist noch eine Ausführung von k_output erforderlich. Dadurch ergibt sich Gleichung 4.11. Aus Gleichung 4.13 ist die Anzahl an Ausführungen einer Simulationskomponente ab der zweiten Iteration des Optimierungsalgorithmus ersichtlich. Bei $T_\mathrm{P} = T_\mathrm{D}$ wird maximal eine Ausführung benötigt. Für jeden zusätzlichen Simulationsschritt bei einer Prädiktion kommt eine Ausführung hinzu, wobei Simulationskomponenten, welche sich nicht auf dem kürzesten Weg befinden erst ab einem Schritt $>= T_\mathrm{P} - T_\mathrm{D} - C_\mathrm{k}$ ausgeführt werden müssen.

4.4.4. Anwendung von Ansatz 1 und Ansatz 2

In Tabelle 4.2 sind die reduzierten Anzahlen zusätzlicher Iteration für die Beispiele aus Abbildung 4.4 dargestellt. Wie in Tabelle 4.2 zu sehen ist, kann durch Ansatz 1 und Ansatz 2 die Zustandssynchronisierung von einem Problem mit mindestens quadratischer Komplexität bezüglich T_D in ein Problem geringerer Komplexität überführt werden. Zudem lässt sich die zusätzliche Anzahl an Ausführungen abhängig vom genauen Aufbau der Co-Simulation signifikant verringern. Die Reduzierung durch Ansatz 2 erfolgen teilweise erst bei der zweiten Iteration der Co-Simulation. Dies wird in Tabelle 4.2 durch die Angabe $\rightarrow 3$ beziehungsweise $\rightarrow 5$ dargestellt.

Tabelle 4.2.: Gegenüberstellung der zusätzlichen Anzahl an Ausführungen pro Iteration des Optimierungsalgorithmus ohne und mit Reduzierung ($T_\mathrm{P} = T_\mathrm{D}$)

Beispiel	Ohne Reduzierung	Mit Reduzierung
Abbildung 4.4a	9	$\rightarrow 3$
Abbildung 4.4b	15	$\rightarrow 5$
Abbildung 4.4c	2	1
Abbildung 4.4d	9	$\rightarrow 3$

4.5. Integration der Performanceverbesserungen in den Algorithmus

Die Verbesserungen der Performance aus Abschnitt 4.4 betreffen den Algorithmus zur Ausführung eines Schrittes der Co-Simulation. Daher integriert Algorithmus 4 diese Verbesserungen in Algorithmus 1. j bezieht sich dabei auf den Zeitpunkt während der Iteration des Optimierungsalgorithmus. i bezieht sich auf den Zeitpunkt der Simulation.

Algorithmus 4 Ausführung eines Schrittes der Co-Simulation mit Performanceverbesserungen

function SIMULATIONSSCHRITT(i, j)
  for all $k \in K$ **do**
    if $j - i < \Gamma_\mathrm{k}$ **then** ▷Ansatz 1 (Unterabschnitt 4.4.1)
      if $(u_0^{(k)}, u_1^{(k)}, \ldots, u_j^{(k)})$ gespeichert **then** ▷Ansatz 2 (Unterabschnitt 4.4.2)
        $x_{j+1}^{(k)} \leftarrow$ LADEZUSTAND$((u_0^{(k)}, u_1^{(k)}, \ldots, u_j^{(k)}))$ ▷Ansatz 2 (Unterabschnitt 4.4.2)
      else
        $x_{j+1}^{(k)} = f^{(k)}(x_j^{(k)}, u_j^{(k)}, t_j)$
        SPEICHERE$(x_{j+1}^{(k)}, (u_0^{(k)}, u_1^{(k)}, \ldots, u_j^{(k)}))$ ▷Ansatz 2 (Unterabschnitt 4.4.2)
      end if
      $y_{j+1}^{(k)} = h^{(k)}(x_j^{(k)})$
      $u_{j+1}^{(k)} = M^{(k)}(y_{j+1}^{(1)}, \ldots, y_{j+1}^{(K)})$
    end if
  end for
end function

4.6. Zeitliche Synchronisierung

Neben dem Problem der Zustandssynchronisierung steht bei der betriebsparallelen Simulation die Herausforderung der Zeitsynchronisation an. Dabei werden Unterschiede des zeitlichen Verhaltens von physischer und simulierter Anlage untersucht, welche nicht

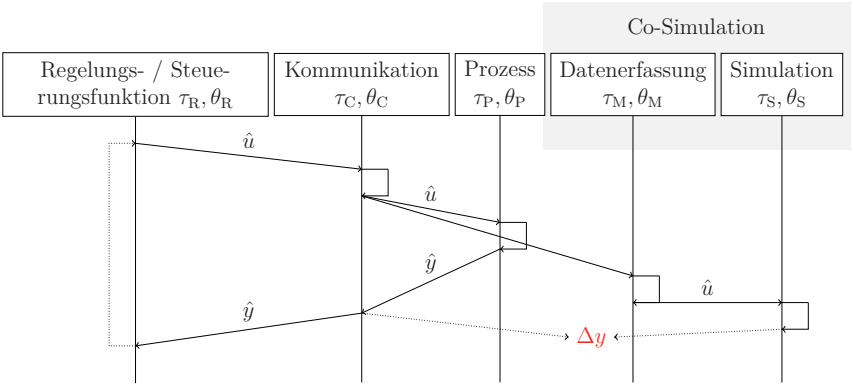


Abbildung 4.5.: Durch Vernetzung bedingte Zyklus- und Totzeiten

durch die verwendeten Modelle entstehen, sondern durch die Kommunikation zwischen den an der betriebsparallelen Simulation beteiligten Entitäten. Die Untersuchungen wurden in (Zipper und Diedrich, 2018a) erstmals veröffentlicht.

Es sollen nachfolgend die Signalketten der physischen Anlage und der simulierten Anlage ausgehend vom Prozess untersucht werden. Die Signalkette kann wie folgt aufgeschrieben werden:

Prozess → Sensor → Regelungs- und Steuerungsfunktion → Aktor → Prozess

Entlang dieser Signalkette werden die Auswirkungen bei einer Änderung eines Signals verzögert. Ursachen für solche Verzögerungen können sein: Analog-Digital-Wandlung, Verarbeitungszeiten innerhalb der Komponenten, Kommunikation zwischen Komponenten und Digital-Analog-Wandlung. Die Signalkette der physischen Anlage umfasst nur physische Komponenten während die Signalkette der simulierten Anlage zusätzlich simulierte Versionen aller Komponenten beinhaltet, die Regelungs- und Steuerungsfunktion ausgenommen. Dabei werden die in Abbildung 4.1 dargestellten Komponenten betrachtet. Zusätzlich wird das Kommunikationssystem zwischen den Komponenten und eine Datenerfassungskomponente betrachtet. Die Datenerfassungskomponente ist die technische Realisierung der Verbindung von physischer und simulierter Anlage. Die Zeitbasis der physischen und simulierten Anlage beziehungsweise der darin befindlichen Komponenten sind unterschiedlich. Zudem sind die Totzeiten der physischen und simulierten Anlage unterschiedlich. Das führt zu unterschiedlichen Verzögerungen entlang der Signalketten. Diese haben Auswirkungen auf alle Vergleiche zwischen Messungen der physischen Anlage und der simulierten Anlage.

Für die folgenden Betrachtungen werden die in Tabelle 4.3 definierten Symbole genutzt.

Die zwischen zwei verbundenen Systemen auftretenden Verzögerungen können mit den Verzögerungseigenschaften Totzeit τ und Zykluszeit θ beschrieben werden (Höme, Palis

Tabelle 4.3.: Übersicht über Symbole zur Beschreibung der zeitlichen Synchronisation

Symbol	Description	Symbol	Beschreibung
R	Regelungs- und Steuerungsfunktion	C	Kommunikation
P	physische Anlage	M	Sensor
S	Co-Simulation	A	Aktor
$\alpha_{\text{Quelle,Ziel}}$	Verzögerung Quelle $\rightarrow$ Ziel		
τ	Totzeit	θ	Zykluszeit
$\alpha_1 * \alpha_2$	Faltung von α_1 und α_2	α^n	n-fache Faltung von α mit sich selbst

und Diedrich, 2014). Diese gilt auch für das Kommunikationssystem selbst. In Abbildung 4.5 sind die Verzögerungen dargestellt, welche entlang der oben gezeigte Signalkette sowie innerhalb der Co-Simulation auftreten.

Für die weiteren Betrachtungen wird die abstrakte Funktion $\alpha_{\text{Quelle,Ziel}}$ eingeführt. Diese ermöglicht die Beschreibung der Verzögerung der Ankunft eines Datums im Ziel ausgehend von den Verzögerungseigenschaften des Quellsystems und des Zielsystems. α hängt also von den Totzeiten und Zykluszeiten des Quell- und Zielsystems ab. Letztendlich wird α eine Wahrscheinlichkeitsdichtefunktion repräsentieren, so dass die Gesamtverzögerung entlang der Signalkette durch Faltung und den Operator $*$ dargestellt wird.

Angenommen zu einer Zeit t_i wird eine Nachricht $\hat{u}$ von der Regelungs- und Steuerungsfunktion zur physischen Anlage über das Kommunikationssystem transportiert. Die physische Anlage empfängt die Nachricht bei $\alpha_{\text{CP}}(t_i)$. Dieselbe Nachricht wird von einer Simulationskomponente zur Zeit $(\alpha_{\text{CM}} * \alpha_{\text{MS}})(t_i)$ empfangen, da sie zusätzlich durch die Datenerfassungskomponente geht.

Nachdem die simulierte oder die physische Anlage einen Eingang erhalten haben, ist eine Reaktion auf diesen Eingang über die Sensoren beobachtbar basierend auf deren internen Verhalten. Nur der Sensorausgang $\hat{y}$ der physischen Anlage wird zurück zur Regelungs- und Steuerungsfunktion kommuniziert. Nach dem Schema in Abbildung 4.1 werden die Ausgänge der physischen Anlage mit den Ausgängen der simulierten Anlage verglichen um Δy zu erhalten.

Die Signale der Regelungs- und Steuerungsfunktion werden zu diskreten Zeitpunkten von der Anlage empfangen. Entsprechend werden diese in der Anlage eine bestimmte Zeit gehalten, bis der nächste Eingang empfangen wird. Da die simulierte Anlage durch das selbe Signal geregelt oder gesteuert wird, muss die Co-Simulation über die exakte Zeitspanne durchgeführt werden um einen Ausgang zu erhalten, welcher äquivalent zum Ausgang der physischen Anlage ist. Zudem ist die Kopplung von simulierter und physischer Anlage asynchron, denn deren jeweilige Zeitbasen sind unterschiedlich. Daher entspricht der k-te gemessene Ausgang der physischen Anlage nicht dem k-ten Ausgang der simulierten An-

lage. Die Zeiten, welche für die Kommunikation zwischen den verschiedenen Elementen benötigt werden, müssen berücksichtigt werden. So wird ein geeignetes Paar von Ausgängen für die Berechnung von Δy gefunden.

Die Ausgänge der physischen Anlage werden über das Kommunikationssystem zur Regelungs- und Steuerungsfunktion gesendet. Die Regelungs- und Steuerungsfunktion berechnet daraufhin einen neuen Eingangswert basierend auf dem Ausgang. Der neue Eingangswert wird über das Kommunikationssystem zur physischen Anlage gesendet.

Ausgehend von der Zeit t_i wird ein Ausgang von der physischen Anlage nach einer Zeit $(\alpha_{\mathrm{CP}} * \alpha_{\mathrm{PC}})(t_i)$ gemessen. Der nächste Eingang zur physischen Anlage liegt zum Zeitpunkt $(\alpha_{\mathrm{CP}} * \alpha_{\mathrm{PC}} * \alpha_{\mathrm{CR}} * \alpha_{\mathrm{RC}} * \alpha_{\mathrm{CP}})(t_0)$ an. Ausgehend von der Zeit t_i liegt der resultierende Ausgang aus der Simulation zur Zeit $(\alpha_{\mathrm{CM}}^{T_{\mathrm{D}}} * \alpha_{\mathrm{MS}} * \alpha_{\mathrm{SS}})(t_0)$ vor.

Zusammengefasst kann festgehalten werden: Selbst, wenn ein perfektes Simulationsmodell der physischen Anlage genutzt wird, wird sich das zeitliche Verhalten (Δy) zwischen simulierter und physischer Anlage unterscheiden. Um ein geeignetes Paar von Ausgängen zu erhalten müssen die folgenden zwei Aspekte berücksichtigt werden.

Bestimmung der Eingangsverzögerung Die *Eingangsverzögerung* beschreibt die Zeitspanne, für welche die Co-Simulation für einen Eingang der Regelungs- und Steuerungsfunktion ausgeführt werden muss. Sie ist nach Gleichung 4.14 definiert:

$$\Delta_i = \underbrace{\alpha_{\mathrm{CP}}(t_i)}_{u \text{ in phys. Anlage}} - \underbrace{(\alpha_{\mathrm{CM}} * \alpha_{\mathrm{MS}})(t_i)}_{u \text{ in Simulation}} \tag{4.14}$$

Bestimmung der Ausgangsverzögerung Die *Ausgangsverzögerung* beschreibt die unterschiedlichen Zeitpunkte, zu denen Reaktionen der physischen und der simulierten Anlage auftreten, welche durch das selbe Eingangssignal ausgelöst wurden. Sie ist nach Gleichung 4.15 definiert:

$$\Delta_{\mathrm{o}} = \underbrace{(\alpha_{\mathrm{CP}} * \alpha_{\mathrm{PC}})(t_i)}_{\hat{y} \text{ verfügbar}} - \underbrace{(\alpha_{\mathrm{CM}}^{n} * \alpha_{\mathrm{MS}} * \alpha_{\mathrm{SS}})(t_i)}_{y \text{ verfügbar}} \tag{4.15}$$

Für die Berechnung der Ausgangsverzögerung und der Eingangsverzögerung werden die in Abbildung 4.5 dargestellten Zeiten berücksichtigt.

Zur konkreten Berechnung wird das *Modell der asynchronen Zyklen* genutzt, welches in (Höme, Palis und Diedrich, 2014) beschrieben wird. Das Gesamtsystem wird in Teilsysteme und das Kommunikationssystem zwischen diesen Teilsystemen zerlegt. Zwischen den Teilsystemen können Kommunikationsverbindungen hergestellt werden. Die Kommunikationszeiten werden entlang einer Signalkette modelliert. Eine Signalkette reicht von ihrer Quelle zu ihrem Ziel über alle involvierten Teilsysteme. Die Verbindung zweier Teilsysteme kann dabei entweder synchron oder asynchron sein. Teilsysteme, welche auf

Tabelle 4.4.: Klassifikation von Verbindungen zwischen Teilsystemen

Verbindung			Sync./ Async.
Regelungs- und Steuerungsfunktion	→	physische Anlage	sync. oder async.
Regelungs- und Steuerungsfunktion	→	Signalerfassung	async
physische Anlage	→	Regelungs- und Steuerungsfunktion	sync. oder async.
Signalerfassung	→	Simulation	sync
Simulationskomponente	→	Simulationskomponente	sync

einer gemeinsamen Zeitbasis beruhen, sind synchron. Teilsysteme mit unterschiedlichen Zeitbasen sind asynchron zueinander.

In (Höme, Palis und Diedrich, 2014) wird dieser Formalismus zur Berechnung der Klemme-Klemme-Reaktionszeit genutzt. Hier wird der Formalismus zur Berechnung der Eingangsverzögerung und Ausgangsverzögerung adaptiert. In Abbildung 4.5 ist auch die Dekomposition in Teilsysteme dargestellt. Der nächste Schritt ist die Klassifikation der Verbindungen zwischen Teilsysteme. Diese Klassifikation ist in Tabelle 4.4 dargestellt. Die Verbindungen innerhalb der Co-Simulation zwischen den Simulationskomponenten sind synchron. Der Grund dafür liegt in der Vorgabe der globalen Simulationszeit durch den Co-Simulationsmaster (siehe Algorithmus 2 und Algorithmus 3). Die Verbindung zwischen der Datenerfassungskomponente und den weiteren Simulationskomponenten ist ebenfalls synchron, da die Datenerfassungskomponente als Simulationskomponente realisiert wird und so Teil der Co-Simulation ist. Die Verbindung der physischen Anlage und der Regelungs- und Steuerungsfunktion ist synchron oder asynchron, abhängig vom verwendeten Synchronisationsschema.

Nach (Höme, 2016) und (Cristian und Fetzer, 1999) können die Verzögerungszeiten einzelner Verbindungen zwischen zwei Systemen wie folgt ausgedrückt werden:

- Die Verteilungsdichtefunktion der Gleichverteilung zur zeitlichen Beschreibung des Informationstransports kann zwischen asynchron verbundenen Systemen genutzt werden. Die Verteilungsdichtefunktionen wird mit der Zykluszeit des Zielsystems parametriert, nicht jedoch mit der Verarbeitungszeit (Höme, 2016, S. 74).

- Synchron verbundene Teilsysteme werden zu einem Teilsystem gruppiert. Die resultierende Verzögerung ergibt sich aus der Summe der Verzögerungen der einzelnen Teilsysteme.

- Für die zeitliche Beschreibung von Verarbeitungszeiten kann die Verteilungsdichtefunktion der δ-Verteilung dienen.

Anhand dieser Vorgaben kann die oben deklarierte α-Funktion in Gleichung 4.16 definiert werden. Für Verarbeitungszeiten wird die δ-Verteilungsdichtefunktion genutzt. Transitionen zwischen zyklischen Systemen werden durch die Verteilungsdichtefunktion der Gleichverteilung $\mathcal{U}$ ausgedrückt.

$$\alpha_{\bullet\circ}(t) = \begin{cases} \delta(t - (\tau_\bullet + \tau_\circ)) & \bullet \text{ zu } \circ \text{ synchron} \\ \mathcal{U}(t, \theta_\circ) & \text{sonst} \end{cases} \tag{4.16}$$

Liegt eine passende Parametrierung für die betriebsparallele Simulation vor, können Δ_i und Δ_o mit Hilfe der Gleichung 4.16 geschätzt werden. Um einen fixen Wert zu erhalten, findet der Modalwert der resultierenden Verteilungsdichtefunktion Anwendung.

4.7. Diskussion der Methodik

In diesem Kapitel wurde die Zustandssynchronisierung vorgestellt. Das Ergebnis davon ist ein Algorithmus, welcher in einem Co-Simulationsmaster integriert wird. Dieser Algorithmus führt kontinuierlich ein Optimierungsalgorithmus aus. Mithilfe dieses Optimierungsalgorithmus wird die Zustandssynchronisierung erreicht. Das übergeordnete Ziel dabei ist, das Auseinanderlaufen der Verhalten der physischen Anlage und der simulierten Anlage zu verhindern, also die Differenz der Messwerte der physischen Anlage und der entsprechenden Variablen der simulierten Anlage innerhalb eines definierten Bereiches bleibt. Dabei bleibt die simulierte Anlage auch bei Änderungen der Anlage aussagefähig. Zudem kann gezeigt werden, dass die Performance der Zustandssynchronisierung durchaus echtzeitfähig realisiert werden kann. So kann sichergestellt werden, dass die betriebsparallele Simulation jederzeit realistische Aussagen über die physische Anlage liefert. Ohne eine kontinuierliche Synchronisierung wäre die Aussagefähigkeit der betriebsparallelen Simulation nicht gegeben, sobald Änderungen an der physischen Anlage auftreten. Der Unterschied zwischen physischer und simulierte Anlage lässt sich nicht mehr am Unterschied des Verhaltens feststellen, sondern am Verhalten der Zustandssynchronisierung.

Die Zustandssynchronisierung kann auf zwei verschiedene Arten eingesetzt werden. Wird das Signal der Regelungs- und Steuerungsfunktion in den Optimierer gegeben und wird der Ausgang des Optimierungsalgorithmus in die den Aktoren entsprechenden Eingänge der Simulationskomponenten gegeben, lässt sich ein Zustandsbeobachter realisieren. Diese Möglichkeit wird in Abbildung 4.6 beschrieben. Der Unterschied zwischen physikalischer und simulierter Anlage lässt sich aus der Betrachtung von Δu gewinnen, also dem Unterschied der Stellsignale der Regelungs- und Steuerungsfunktion und deren Manipulation durch den Optimierer.

Wird das Signal der Regelungs- und Steuerungsfunktion mit den entsprechenden Eingängen der Simulationskomponenten verbunden und wird der Optimierer mit Parametern

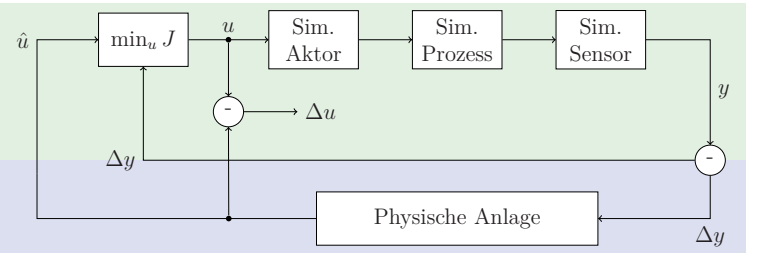


Abbildung 4.6.: Einsatz der Zustandssynchronisierung als Beobachter

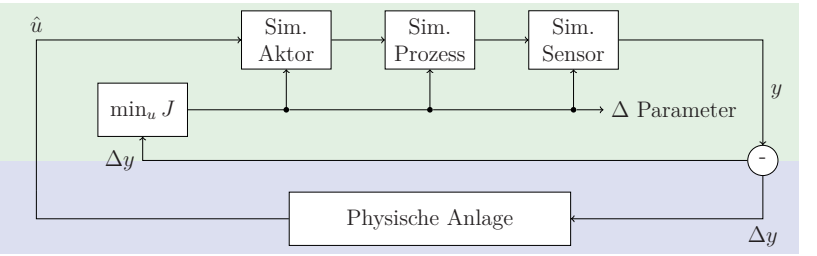


Abbildung 4.7.: Einsatz der Zustandssynchronisierung als Parameteroptimierung

von Simulationskomponenten verbunden, findet eine Parameteroptimierung statt. Diese Möglichkeit ist in Abbildung 4.7 dargestellt. Der Unterschied zwischen physikalischer und simulierter Anlage lässt sich aus der Änderung der entsprechenden Parameter durch den Optimierer feststellen.

Das Tuning der betriebsparallelen Simulation kann vorbereitend offline erfolgen. Dies ist möglich, da das vorgestellte Konzept vollständig rückwirkungsfrei bezüglich der physischen Anlage ist. Für das offline Tuning müssen die entsprechenden Signale der physischen Anlage aufgezeichnet werden. Diese Aufzeichnung kann dann mit der Co-Simulation und der aktivierten Zustandssynchronisierung genutzt werden, um den geeignetsten Optimierungsalgorithmus zu ermitteln und eine optimale Parametrierung dieser Optimierung herauszufinden.

Die Formulierung als Optimierungsproblem birgt zudem den Vorteil, dass abhängig vom konkreten Szenario Anpassungen möglich sind. So können beispielsweise Nebenbedingungen einfach berücksichtigt werden. Außerdem können Kosten für Abweichungen der Eingänge, also den Standardparametern beziehungsweise im Fall der Realisierung als Beobachter den Steuersignalen, formuliert werden.

5. Validierung

In diesem Kapitel wird die Validierung der Zustandssynchronisierung an verschiedenen Systemen dargestellt. Dazu wird zunächst kurz auf die Umsetzung der zuvor entwickelten Algorithmen und Methoden eingegangen. Anschließend erfolgt die Validierung an drei physischen Systemen. Die zeitliche Synchronisation wird simulativ validiert. Neben der Validierung zeigt dieses Kapitel auch den Einsatz und die Möglichkeiten der Parametrierung der Zustandssynchronisierung für diese unterschiedlichen Systeme.

5.1. Umsetzung des Co-Simulationsmasters

Zur Validierung der Zustandssynchronisierung sind Implementierungen für den Co-Simulationsmaster und die Simulationskomponente der Optimierung notwendig. Diese Implementierungen werden nach dem *FMI 2.0* Standard (FMI 2.0, 2014) umgesetzt in C++. In *FMI* werden Simulationskomponenten als *Functional Mockup Unit — FMU* bezeichnet.

Als Technologie wird die Middleware DOME genutzt (*DOME* 2020). Dort werden Applikationen in Funktionsblöcke zerlegt. Der Vorteil für die Implementierung der Zustandssynchronisierung ist, dass diese Funktionsblöcke innerhalb eines Prozesses laufen können. Darüber hinaus können die Funktionsblöcke auf mehrere Prozesse beziehungsweise auf mehrere Rechner verteilt werden. Diese Möglichkeit der Verteilung erleichtert die notwendigen Implementierungen für die Validierung am System in Abschnitt 5.6.

Daher wird eine Klasse für einen Funktionsblock eines Co-Simulationsmasters erstellt, welcher den Algorithmus aus Abschnitt 4.3 implementiert. Weiterhin wird eine Klasse für generische Funktionsblöcke erstellt, welche eine FMU laden und in die Co-Simulation einbinden kann. Zusätzlich werden die Performanceverbesserungen aus Abschnitt 4.4 dort implementiert. Mehr Informationen über die Entwicklung von verteilten Systemen mit *DOME* lassen sich zum Beispiel in (Riedl u. a., 2014) finden. Die Konfiguration der Co-Simulation wird über *Lua* Skripte erledigt. So wird festgelegt, welche FMUs geladen werden, wie die Zuweisen der Parameter erfolgt, wie Variablen verbunden werden sollen sowie die Auswahl und Parameteroptimierung der Optimierung.

5.2. Umsetzung der Simulationskomponente des Optimierers

Die Optimierung wird in einer extra Simulationskomponente als *FMU* implementiert. Wie bereits in Abschnitt 4.3 beschrieben, implementiert diese Simulationskomponente die folgenden Funktionalitäten:

- Ausführung der Initialisierung des Optimierungsalgorithmus
- Ausführen einer Iteration des Optimierungsalgorithmus
- Austausch von Variablen zwischen Optimierungsalgorithmus und Co-Simulation
- Überprüfen der Abbruchbedingung
- Überwachung auf Fehler — Fehlermeldung vom Optimierungsalgorithmus, numerische Probleme, Inkonsistenzen

Die Variablen, welche diese Simulationskomponente zur Optimierung austauschen muss, sind u, $\hat{u}$, y und $\hat{y}$. Diese werden vom Typ `fmi2Real` umgesetzt mit der Kausalität Eingang beziehungsweise Ausgang.

Das Ausführen von Funktionen ist in *FMI* nicht vorgesehen. Daher wird die Ausführung der Initialisierung des Optimierungsalgorithmus und das Ausführen einer Iteration des Optimierungsalgorithmus an je eine boolesche Variable geknüpft. Das Schreiben von booleschen Variablen erfolgt in *FMI*, indem eine *FMU* eine Funktion `fmi2SetBoolean` implementiert, welche für einen Schreibvorgang aufgerufen wird. Dies ist der Einstiegspunkt für die beiden Funktionen zum Ausführen: Wird `fmi2SetBoolean` für eine entsprechende Variable aufgerufen, wird auf eine steigende Flanke geprüft und die Funktion ausgeführt.

Eine Variable `algorithm` vom Typ `fmi2Integer` erlaubt die Auswahl eines Optimierungsalgorithmus. Dadurch kann eine Simulationskomponente zur Optimierung mit mehreren unterschiedlichen Optimierungsalgorithmen genutzt werden. Weiterhin ist es so auch möglich auf einen anderen Algorithmus zu wechseln falls keine Lösung gefunden wird.

In Tabelle 5.1 sind die umgesetzten Algorithmen aufgelistet. Das Ziel ist es, sowohl Algorithmen zu testen, welche keine Ableitung der Zielfunktionen benötigen, sowie Algorithmen, welche eine Ableitung benötigen.

Die Optimierungsalgorithmen *Nelder-Mead-Simplex* (Nelder und Mead, 1965) und *BOBYQA* (Powell, 2009) benötigen keine Ableitung der Zielfunktion. Dies ist von Vorteil, da die Ableitung nicht bekannt ist. Über die *FMI* Interfaces `fmi2GetDirectionalDerivative` und `fmi2GetRealOutputDerivatives` können Simulationskomponenten Ableitungen bekannt geben. Diese Interfaces geben jedoch in der Regel keine Ableitungen bezüglich der zu optimierenden Variablen und Parameter zurück.

Tabelle 5.1.: Umgesetzte Optimierungsalgorithmen

Nr.	Algorithmus	Implementierung
1	Nelder-Mead-Simplex	Eigene Implementierung nach (Nelder und Mead, 1965)
100	L-BFGS	(Bochkanov, 2020)
1000	BOBYQA	(King, 2009)

Zudem muss für eine Ableitung der Zielfunktion nicht nur die Ableitung einer Simulationskomponente bekannt sein. Vielmehr muss diese Ableitung bezüglich der gesamten Co-Simulation betrachtet werden.

L-BFGS ist in vielen Optimierungsbibliotheken für C++ zu finden. Der Algorithmus *L-BFGS* ist ein Quasi-Newton-Verfahren und benötigt neben der Zielfunktion die Ableitung erster Ordnung der Zielfunktion als Eingabe. Falls die erste Ableitung nicht verfügbar ist, muss sie numerisch aus Auswertungen der Zielfunktion approximiert werden. In der Implementierung von (Bochkanov, 2020) geschieht dies durch vier zusätzliche Funktionsauswertungen pro Optimierungsvariable. Es werden also fünf Ausführungen der Co-Simulation pro Iteration des Optimierungsalgorithmus benötigt.

Die ausgewählten Algorithmen werden vielfältig eingesetzt zur numerischen Optimierung. Sie sind in vielen freien Softwarebibliotheken zur Optimierung enthalten und werden in einer Vielzahl verschiedener Domänen genutzt (beispielsweise (Bochkanov, 2020), (King, 2009), (Gough, 2009), (Melo und Iacca, 2014) oder (Nash, Varadhan u. a., 2011)).

Die Untersuchungen der Performance verschiedener Optimierungsalgorithmen und die Ergebnisse der Ansätze zur Steigerung der Performance werden anhand von *L-BFGS* und *BOBYQA* am Motorsystem dargestellt. Der *Nelder-Mead-Simplex* wird für die Validierung am Zylindersystem genutzt aus dem technischen Grund, dass sich die bereits für Linux umgesetzte Implementierung im Gegensatz zu den weiteren umgesetzten Verfahren leichter auf Windows portieren ließ.

Unabhängig vom verwendeten Optimierungsalgorithmus erfolgt durch die Simulationskomponente des Optimierungsalgorithmus eine Überprüfung des Abbruchkriteriums. Sobald diese erfüllt ist, wird die Optimierung als erfolgreich angesehen, unabhängig zum Beispiel vom Gradienten der Zielfunktion.

5.3. Metriken für die Auswertung der Validierung

Die Validierung wird an verschiedenen Systemen durchgeführt. Dazu werden Simulationsergebnisse ohne aktivierte Zustandssynchronisierung mit Simulationsergebnissen bei aktivierter Zustandssynchronisierung verglichen. Zudem werden anhand des Motorsys-

tems verschiedene Anwendungen der Zustandssynchronisierung untereinander evaluiert.

Zur Realisierung dieser Vergleiche werden folgend einige Metriken eingeführt: Die Zahl der Iterationen der einzelnen Simulationskomponenten, die mittlere quadratische Abweichung sowie die maximale Abweichung.

5.3.1. Anzahl Iterationen der Simulationskomponenten

Die Anzahl der Iterationen N der Simulationskomponenten berechnet sich aus der Summe der Iterationen jeder einzelnen Simulationskomponente und wird in Gleichung 5.1 umgesetzt. Die Anzahl der Iterationen jeder einzelnen Simulationskomponente wird bei Beginn der Co-Simulation mit 0 initialisiert und bei jeder Iteration der Co-Simulation um eins erhöht. In der Implementierung in *FMI* findet daher eine Erhöhung bei jedem Aufruf von *fmi2DoStep* einer *FMU* statt. Dabei ist zu beachten, dass hier die Simulationskomponente des Optimierers nicht berücksichtigt wird, denn diese ist eine Implementierungsstrategie für die Kapselung des Optimierungsalgorithmus und führt in *fmi2DoStep* keine Berechnungen durch.

$$N_{\text{ges}} = \sum_{k=1}^{K} N_{\text{k}} \tag{5.1}$$

Die Zahl der Iterationen der einzelnen Simulationskomponenten soll als Benchmark für die Performance der Zustandssynchronisierung dienen. Diese Zahl ist unabhängig von der eingesetzten Hardware und weiteren Optimierungen am Quelltext der Implementierung. Außerdem variiert diese Zahl nicht mit jeder Durchführung eines Szenarios der Validierung. So kann die Performance qualitativ sowie wiederholbar abgeschätzt werden.

5.3.2. Mittlere quadratische Abweichung

Als eine Metrik zum Vergleich zwischen der betriebsparallelen Simulation ohne und mit Zustandssynchronisierung soll die mittlere quadratische Abweichung (*MSE*, aus dem Englischen Mean-Squared-Error) genutzt werden. Diese berechnet sich nach (Botchkarev, 2018) wie folgt:

$$MSE = \frac{1}{n} \sum_{i=0}^{n} (y_i - \hat{y}_i)^2 \tag{5.2}$$

Die mittlere quadratische Abweichung wird aus den Differenzen zwischen den Variablen der Ausgänge des physischen Systems und des simulierten Systems gebildet. Generell dient sie zur Bewertung von Modellen (zum Beispiel (Botchkarev, 2018) oder (Wu, McAuley und Harris, 2011)). Hier wird die Summe der Abweichungen auf die Anzahl

der Messpunkte normiert. So soll eine Aussage über Durchschnittswerte der Güte der Zustandssynchronisierung getroffen werden. Ein hoher Wert der mittleren quadratischen Abweichung ist ein Indiz dafür, dass die Zustandssynchronisierung während der gesamten Zeit die betriebsparallele Simulation unzureichend synchronisiert.

5.3.3. Maximale absolute Abweichung

Als weitere Metrik zum Vergleich zwischen der betriebsparallelen Simulation ohne und mit Zustandssynchronisierung wird die maximale absolute Abweichung genutzt. Diese berechnet sich nach (Botchkarev, 2018) wie folgt:

$$MAX = \max_{i \in 0..n} |y_i - \hat{y}_i| \tag{5.3}$$

Die maximale absolute Abweichung gibt den maximalen Unterschied zwischen den Variablen der Ausgänge des physischen Systems und des simulierten Systems an. Eine hohe maximale absolute Abweichung ist ein Indiz dafür, dass die Zustandssynchronisierung zu bestimmten Zeitpunkten die betriebsparallele Simulation unzureichend synchronisiert.

5.4. Wertekontinuierliches System: Motorsystem

5.4.1. Beschreibung des Systems

Das Motorsystem wurde erstmals in (Zipper, 2019) vorgestellt und ist in Abbildung 5.1 abgebildet. Das Motorsystem besteht aus folgenden, für die Validierung relevanten, Komponenten:

(1) Schalter zur Drehzahlwahl

(2) Antriebsregler und Microcontroller zur Messwerterfassung

(3) Bremshebel

(4) Scheibenbremse

(5) Drehzahlsensor

(6) Motor

(7) Temperatursensor

An dem Motor (6) ist ein Rohr angebracht. Diesem rotierenden Rohr steht ein festes Rohr gegenüber. An dem rotierenden Ende ist eine Bremsscheibe (4) befestigt. Mit Hilfe des Hebels (3) kann der Motor so gebremst werden. In unmittelbarer Nähe zur Bremsscheibe

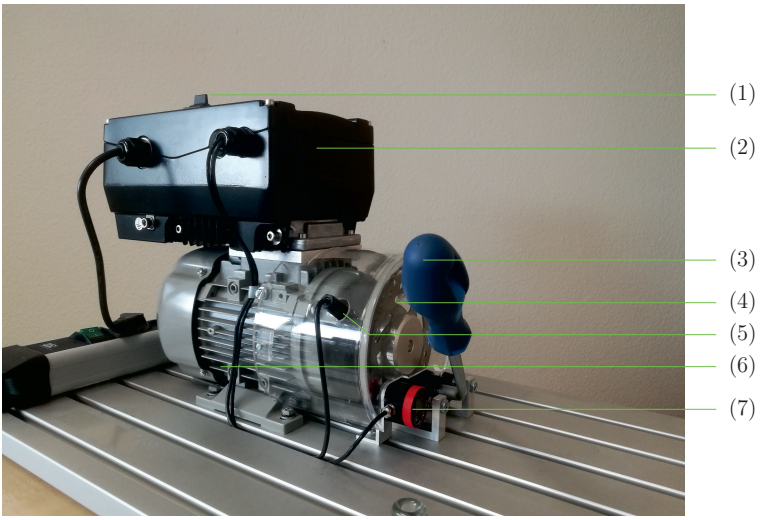


Abbildung 5.1.: Motorsystem (Zipper und Diedrich, 2020)

ist ein Temperatursensor angebracht (7), welcher im Rahmen der Validierung allerdings ungenutzt bleibt. Des Weiteren ist am rotierenden Rohr ein Drehzahlsensor installiert (5). Der Motor wird über einen Antriebsregler (2) geregelt. Der Antriebsregler nimmt weiterhin Steuersignale zum Einstellen der gewünschten Drehzahl über den Schalter (1) entgegen. Darüber hinaus liefert der Antriebsregler verschiedene Messwerte. Darunter sind zum Beispiel die aktuelle Drehzahl, das aktuell anliegende Drehmoment und der Strom. Alle verfügbaren Messwerte werden zentral über einen Microcontroller erfasst. Dieser Mikrocontroller ist innerhalb des Gehäuses des Antriebsreglers (2) installiert und stellt diese Werte per seriellem Anschluss über USB bereit. Dieser USB Anschluss befindet sich auf der Rückseite des Gehäuses des Antriebsreglers und ist daher nicht in Abbildung 5.1 zu sehen. Über die Verbindung mit USB können die Motordaten in Echtzeit zum Beispiel an einen PC übermittelt werden. Der Takt der Datenübertragung schwankt zwischen 25 ms und 40 ms.

Die Informationsflüsse des Motorsystems sind in Abbildung 5.2 dargestellt. Einfluss auf das Verhalten des Motors kann mechanisch durch Bremsen über (a) erfolgen und durch die Wahl der Soll-Drehzahl bei (b). Der Antriebsregler als Black-Box regelt den Motor so, dass die eingestellte Soll-Drehzahl auch bei Anliegen eines Bremsmomentes möglichst erreicht wird. Die Ist-Drehzahl wird durch den Drehzahlsensor gemessen.

Der Antriebsregler ermöglicht die Ausgabe verschiedener Parameter und Messdaten über eine proprietäre Schnittstelle (d). Der Microcontroller greift hier nur das Bremsmoment ab. Die Ist-Drehzahl wird durch eine elektrische Anbindung des Drehzahlsensors

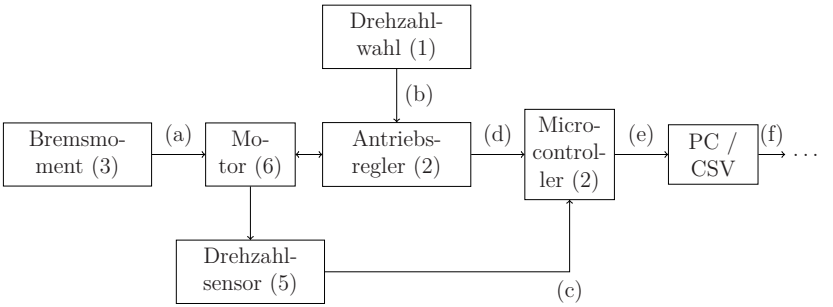


Abbildung 5.2.: Informationsflüsse Motorsystem

bei (c) realisiert. Der Microcontroller stellt alle Daten über USB (e) per seriellem Protokoll bereit. Auf einem PC läuft eine Software, welche den Daten-Stream von der seriellen Schnittstelle ausließt und in eine CSV Datei speichert. Die Anbindung der Co-Simulation an das Motorsystem wird über diese CSV Datei realisiert (f). Die CSV Datei findet sich im Dateisystem des PC, wird jedoch durch den Befehl *mkfifo* erzeugt (MacKenzie, 2010). Auf diese Weise erfolgt die Anbindung der Co-Simulation an das Motorsystem über eine *Shared Memory* Schnittstelle ohne tatsächlichen Zugriff auf die Festplatte des PCs.

5.4.2. Aufbau der Co-Simulation

Die Co-Simulation besteht aus drei Simulationskomponenten:

1. Verhalten des Motors
2. Verhalten des Antriebsreglers, als PID Regler angenommen
3. Erfassung der Messwerte

Das für das Modell des geregelten Motors zugrundeliegende Blockschaltbild ist in Abbildung 5.3 dargestellt. Dieses wird in zwei Simulationskomponenten aufgeteilt. Das Modell wurde in studentischer Zusammenarbeit mit einem Mechatroniker erstellt, welcher den theoretischen Hintergrund im Bereich Motoren zur Modellierung beitragen konnte.

Die Kopplung mit dem physischen System erfolgt über eine CSV Datei. Dafür wird eine Simulationskomponente genutzt, welche Zeitreihen aus CSV Dateien einlesen und der Co-Simulation bereitstellen kann. Entweder kann die Anbindung des physischen Motors über eine bereits existierende Aufzeichnung erfolgen, oder die CSV Datei wird online kontinuierlich erzeugt. Bei der Inbetriebnahme kann die verwendete CSV Datei leicht erstellt werden. Hier werden aufeinanderfolgende Datensätze als Zeile abgespeichert. Eine Zeile besteht dabei aus mehreren Spalten, wobei jede Spalte eine gemessene Variable

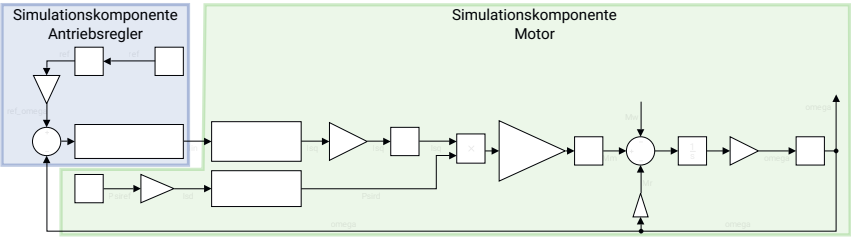


Abbildung 5.3.: Blockschaltbild Motorsimulation in Anlehnung an (Riefenstahl, 2000)

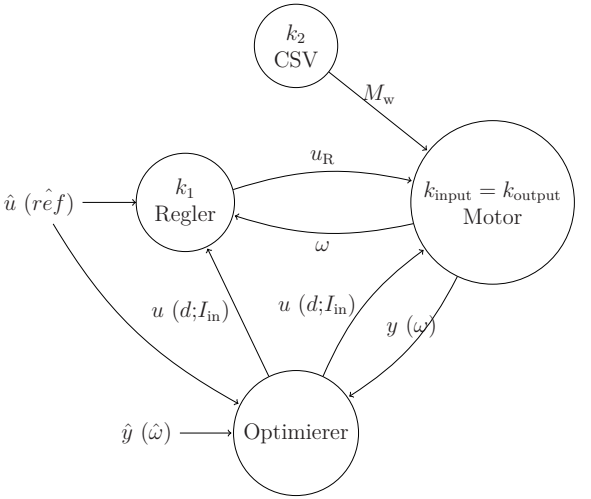


Abbildung 5.4.: Co-Simulation des Motorsystems

repräsentiert. Üblicherweise steht in der ersten Spalte ein Zeitstempel der entsprechenden Messung. Über diesen Zeitstempel kann die CSV Datei, also die Aufnahme oder die Online-Messung, mit der Simulationszeit synchronisiert werden.

In Tabelle 5.2 werden die Variablen der Simulationskomponente für den Motor aufgeführt. In Tabelle 5.3 werden die Variablen der Simulationskomponente des Antriebsreglers aufgeführt.

Abbildung 5.4 zeigt die Co-Simulation des Motorsystems. Die Co-Simulation besteht insgesamt aus vier Stimulationskomponenten. Eine der Stimulationskomponenten ist der Optimierungsalgorithmus. Die Co-Simulation des Motorsystems besteht aus einer Simulationskomponente für den Motor und einer Simulationskomponente für den Antriebsregler. Die Kopplung mit dem physischen System erfolgt über die CSV Simulationskomponente. Diese gibt das Bremsmoment, welches vom physischen Antriebsregler ermittelt wird, in die Simulationskomponente für den Motor. Welche Eingänge u im konkreten Fall vom Op-

Tabelle 5.2.: Variablen der Simulationskomponenten des Motorsystems nach (Riefenstahl, 2000)

Variable	Einheit	Richtung	Bedeutung
Z_p		Parameter	Polpaarzahl
L_h	H	Parameter	Hauptinduktivität
L_r	H	Parameter	Rotorinduktivität
R_r	Ω	Parameter	Rotorwiderstand
T_r	s	Parameter	Zeitkonstante Rotor
d	Nm/rad/s	Parameter	Widerstandsbeiwert
J	kg/m^2	Parameter	Massenträgheitsmoment
$ratio$		Parameter	Verhältnis L_r zu L_h
Ψ_ref	V s	Parameter	Referenz-Flussverkettung
Ψ_rd	V s	Parameter	Magnetische Flussverkettung
T_s	s	Parameter	Summenzeitkonstante Umformer
I_sd	A	Parameter	d-Anteil Strom
I_sq	A	Parameter	q-Anteil Strom
I_ges	A	Parameter	Gesamtmotorstrom
M_m	N m	Parameter	Motormoment
M_r	N m	Parameter	Reibmoment
I_in	A	Eingang	Eingangsstrom aus Regler (q-Anteil)
M_w	N m	Eingang	ext. Widerstandsmoment
ω	rad/s	Ausgang	Motordrehzahl

timierer zu den weiteren Simulationskomponenten gegeben werden, hängt vom Szenario ab. So können sowohl die Parameter P, I oder D des Antriebsreglers optimiert werden. Alternativ können auch die Höhe des anliegenden Bremsmomentes oder die Zeitkonstanten des Umformers variiert werden.

5.4.3. Parametrierung der Simulationsmodelle und der Optimierung

Die Parametrierung der Simulationskomponenten ist in Tabelle 5.4 dargestellt. Die Werte der Parameter sind größtenteils der Dokumentation des Motorsystems und der Bediensoftware vom Hersteller des Antriebs entnommen. Für die vom Optimierer während der Zustandssynchronisierung angepassten Variablen ist in Tabelle 5.4 der initiale Wert des

Tabelle 5.3.: Variablen der Simulationskomponenten des Antriebsreglers

Variable	Einheit	Richtung	Bedeutung
P	A/rad/s	Parameter	P-Anteil
I	1/s	Parameter	I-Anteil
D	s	Parameter	D-Anteil
$rise$	Hz/s	Parameter	Anstieg Rampe Frequenz
Wgh	A m/N	Parameter	Frequenzanpassung (Schlupf)
Ψ_{ref}	V s	Parameter	Referenzflussverkettung
ref	Hz	Eingang	Sollwert (Frequenz)
y	rad/s	Eingang	Istwert (Drehzahl)
e	rad/s	Ausgang	Reglerfehler
u_{R}	A	Ausgang	Reglerausgang (Stellstrom)
Ψ_{refPID}	V s	Ausgang	Nennfluss

Tabelle 5.4.: Parametrierung (Startwerte) Motorsystem

Parameter	Wert	Parameter	Wert
P	51,7 A/rad/s	t_{step}	0,001 s
I	309,7/s	$ratio$	0,25
D	0 s	T_{s}	0,0025 s
ref	10 Hz	d	0,0031 Nm/rad/s
Ψ_{ref}	0,5 V s	J	0,0015 kg/m^2
$rise$	12 Hz s	R_{r}	2 Ω
Wgh	0,01 A/Nm	L_{r}	0,025 H
Z_{p}	1	L_{h}	0,02 H

Parameters dargestellt.

Die Darstellung der Ergebnisse erfolgt unter Verwendung *BOBYQA* als Vertreter eines ableitungsfreien Algorithmus und *L-BFGS* als Vertreter für einen Algorithmus, der die erste Ableitung der Zielfunktion nutzt (siehe Tabelle 5.1).

5.4.4. Formulieren von Szenarien für die Validierung

Als Szenario wird angenommen, dass der Motorparameter J statt der eigentlichen Parametrierung von 0,0015 kg/m^2 auf einen Wert von 0,0515 kg/m^2 gesetzt ist. In der Praxis kann ein Unterschied im Massenträgheitsmoment des Motors neben einer fehlerhaften

Parametrierung beispielsweise durch Verschleiß oder Änderungen am Prozess und am Produkt ausgelöst werden.

Zur Validierung werden unterschiedliche Strategien untersucht. Zunächst wird die betriebsparallele Simulation ohne Synchronisierung durchgeführt, um einen Vergleich zu ermöglichen. Die betriebsparallele Simulation mit Synchronisierung wird für zwei verschiedene Fälle durchgeführt. Zum einen als Parameteroptimierung für den Parameter d des Motors, zum anderen als kontinuierliche Modifikation des Eingangsstroms I_{in} des Motors.

Diese zwei Fälle werden zunächst ohne Performanceverbesserung durchgeführt für einen Horizont von eins. Anschließend werden sie mit einem Horizont von eins ($T_{\text{P}} = 1$), mit einem Horizont fünf ($T_{\text{P}} = 5$) ohne dynamische Optimierung und mit einem Horizont von fünf ($T_{\text{P}} = 5$) und mit dynamischer Optimierung untersucht, jeweils mit aktivierter Performanceverbesserung.

Die Berechnung der Metriken aus Abschnitt 5.3 wird wie folgt auf das Motorsystem angewendet: Zur Berechnung der Anzahl der Iterationen N_{ges} werden die Simulationskomponenten *Regler*, *CSV* und *Motor* betrachtet. Zur Berechnungen der MSE und MAX Metriken werden die Motordrehzahlen ω des simulierten Motors und $\hat{\omega}$ (aus (c) in Abbildung 5.2) des physischen Motors betrachtet.

5.4.5. Auswertung

In Abbildung 5.5 ist die betriebsparallele Simulation ohne Synchronisation dargestellt. Es ist zu sehen, dass die Drehzahlkurven der simulierten und der physischen Anlage übereinstimmen. In Abbildung 5.6 ist die betriebsparallele Simulation mit falschen Parameter $J = 0{,}0515\,\text{kg/m}^2$ ohne Synchronisation dargestellt. Die fehlerhafte Parametrierung führt dazu, dass der simulierte Antriebsregler bei anliegendem Moment die Drehzahl nicht im selben Maß stabilisieren kann, wie der physische Antriebsregler.

Ohne Synchronisation benötigt die Simulation aufgrund der zu simulierenden Zeitspanne von 14 s und der Schrittweite von 0,01 s 1401 Schritte. Da drei Simulationskomponenten beteiligt sind, werden insgesamt $3 \cdot 1401 = 4203$ Ausführungen von Simulationskomponenten benötigt. Diese Zahl stellt das Minimum der benötigten Ausführungen von Simulationskomponenten dar.

Bei den Abbildungen (5.7, 5.8, 5.9, 5.10, 5.11 und 5.12) der Synchronisation wird statt des Bremsmomentes die vom Optimierer beeinflusste Größe (d oder I_{in}) dargestellt. Nachfolgend werden die Ergebnisse bei aktivierter Synchronisation beschrieben.

Abbildung 5.7 zeigt die betriebsparallele Simulation mit Synchronisation und bei $T_{\text{P}} = 1$. Es ist zu sehen, dass die Synchronisation funktioniert, da die Drehzahl der Motorsimulation lediglich eine kleine Differenz zur Drehzahl der physischen Anlage aufweist und die Auswirkung der abweichenden Parametrierung ausgeglichen wird. d unterliegt dabei starken Schwankungen und muss mit jedem Zeitschritt signifikant angepasst werden

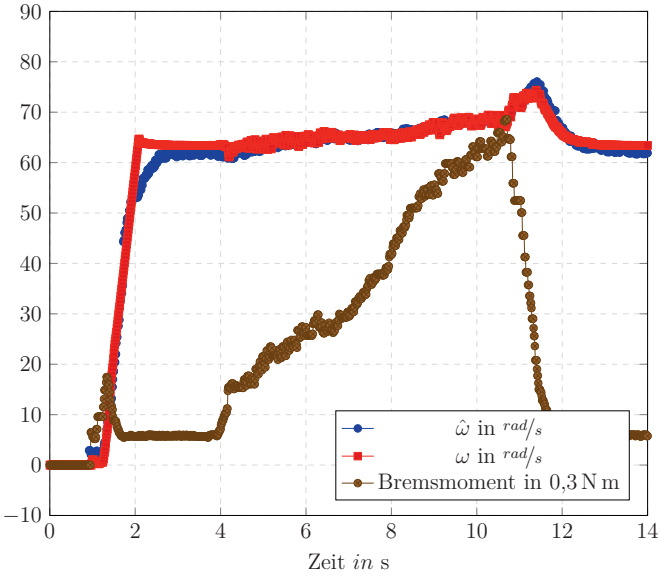


Abbildung 5.5.: Motorsystem ohne Synchronisation

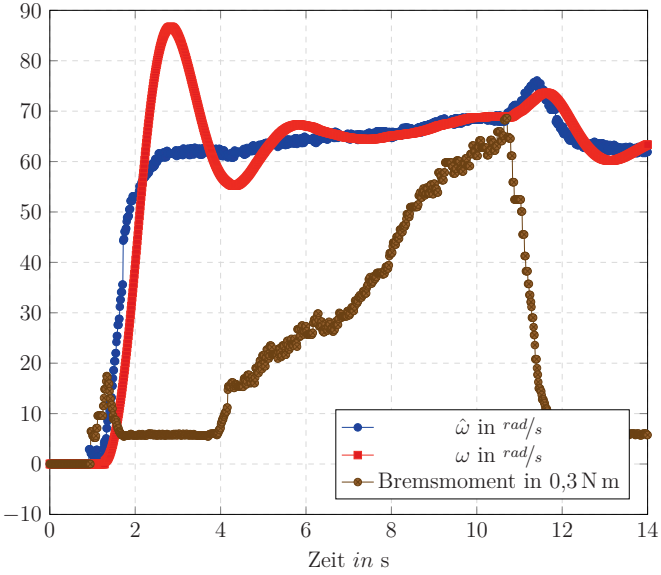


Abbildung 5.6.: Motorsystem ohne Synchronisation mit abweichendem Parameter $J = 0{,}0515\,\mathrm{kg/m^2}$

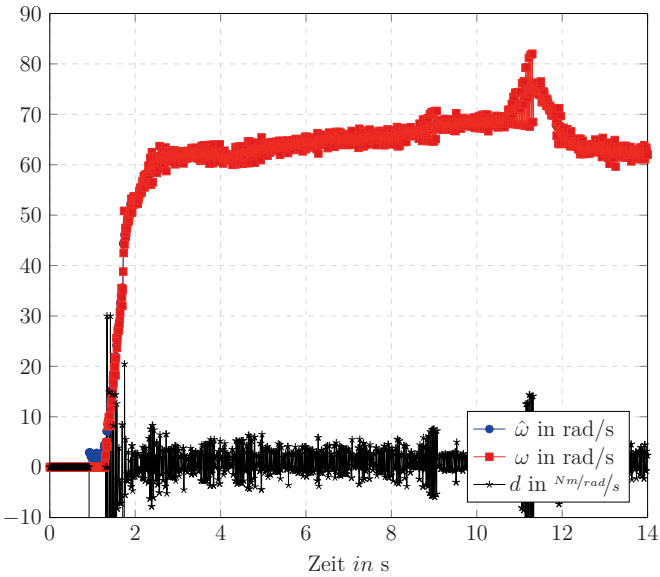


Abbildung 5.7.: Motorsystem mit Synchronisation durch d

durch den Optimierungsalgorithmus. In Abbildung 5.8 ist die betriebsparallele Simulation mit Synchronisation und $T_{\mathrm{P}} = 5$ dargestellt, ohne dynamische Optimierung. Auch hier funktioniert die Synchronisation und d schwankt dabei weniger stark. Abbildung 5.9 zeigt die betriebsparallele Simulation mit Synchronisation und $T_{\mathrm{P}} = 5$, mit dynamischer Optimierung. Hier wird die betriebsparallele Simulation am besten mit der physischen Anlage synchronisiert. Dabei sind die Schwankungen von d geringer als in den vorherigen Szenarien. Die Abbildungen 5.10, 5.11 und 5.12 zeigen die Synchronisation bei Nutzung des Stroms I_{in}.

Insgesamt ist erkennbar, dass die Synchronisation mit dem Parameter d besser funktioniert als mit dem Strom I_{in}. Bei Nutzung des Stroms funktioniert die Synchronisation nur mit Hilfe der dynamischen Optimierung. Auch dann werden große Änderungen an I_{in} zur Realisierung der Synchronisation notwendig.

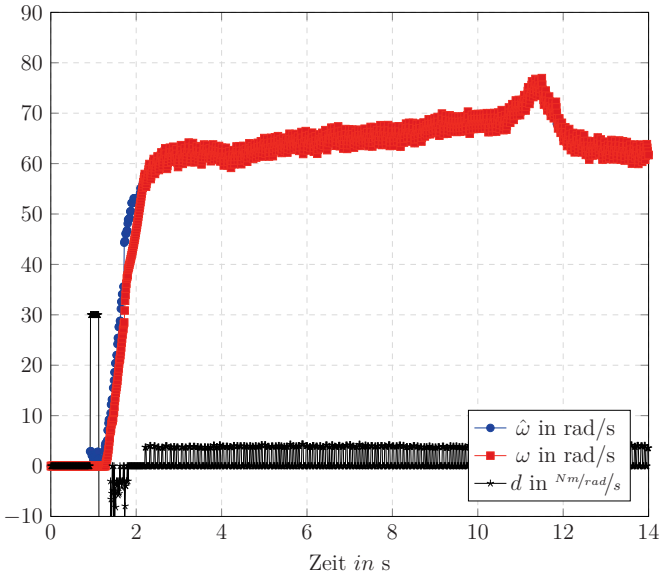


Abbildung 5.8.: Motorsystem mit Synchronisation durch d und $T_{\mathrm{P}} = 5$

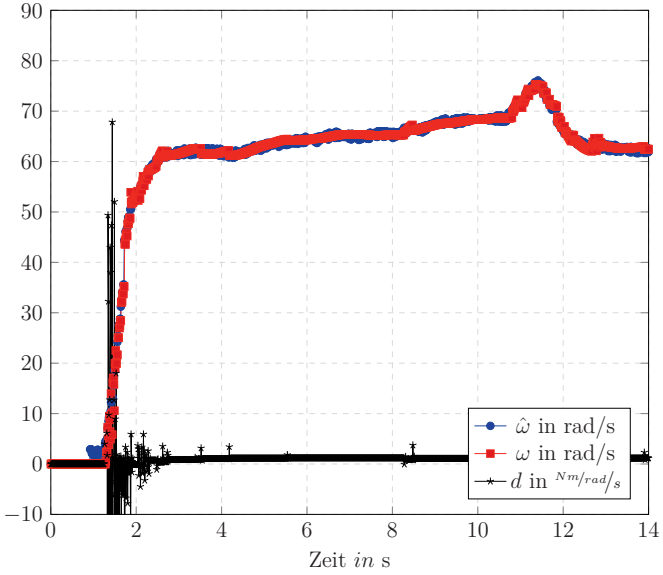


Abbildung 5.9.: Motorsystem mit Synchronisation durch d und $T_{\mathrm{P}} = 5$ und dynamischer Optimierung

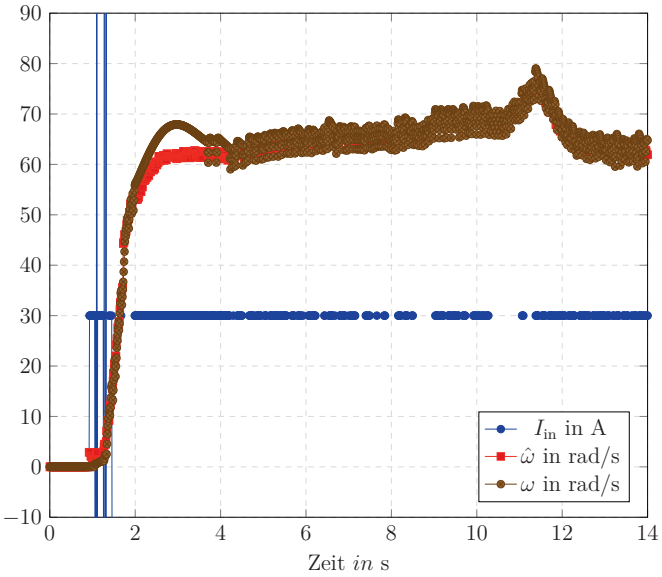


Abbildung 5.10.: Motorsystem mit Synchronisation durch I_{in}

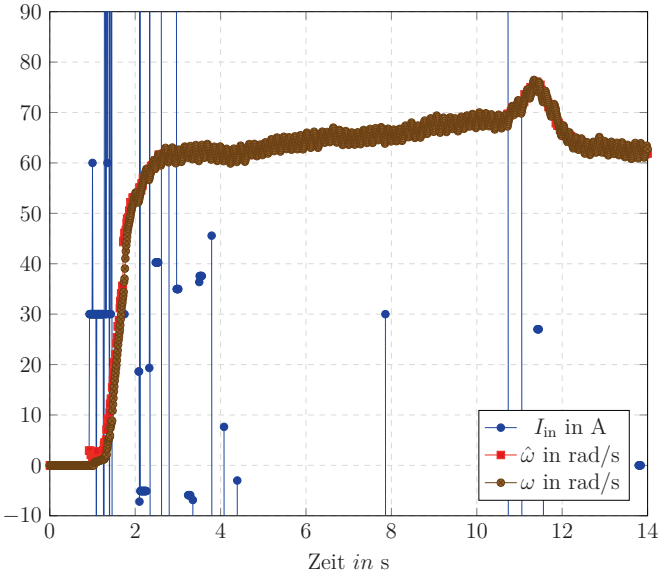


Abbildung 5.11.: Motorsystem mit Synchronisation durch I_{in} und $T_{\text{P}} = 5$

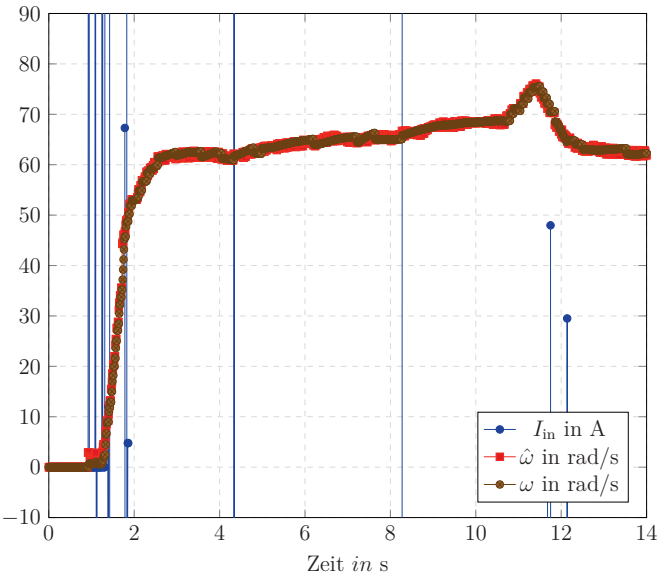


Abbildung 5.12.: Motorsystem mit Synchronisation durch I_{in} und $T_{\text{P}} = 5$ und dynamischer Optimierung

In Tabelle 5.5 befindet sich die Auswertung der unterschiedlichen Szenarien anhand der zuvor eingeführten Metriken. Eine Iteration einer Simulationskomponente für das Motorsystem benötigt etwa 0,05 ms auf dem Simulationsrechner (parallele Ausführung der Simulationskomponenten, Hardware: Intel e3 Quad-Core CPU, 3,3 GHz, 16 GB RAM, Linux). Über die gesamte Simulationszeit von 14 s betrachtet sind somit alle Fälle echtzeitfähig.

Tabelle 5.6 stellt die notwendigen Ausführungen der Simulationskomponenten bei gleichen $MSE = 1,5$ unter Nutzung des Parameters d zur Zustandssynchronisierung gegenüber. Zur Erzeugung gleicher *MSE* wird die Abbruchbedingung entsprechend angepasst. Es werden sowohl die benötigten Ausführungen absolut dargestellt als auch das Verhältnis der benötigten Ausführungen zu der minimalen Anzahl an Ausführungen, also Ausführungen/$(3 \cdot 1401)$.

Das beste Ergebnis wird für $T_{\text{P}} = 2$ und der Nutzung des Parameters d und dynamischer Optimierung erreicht (Fall *e)*). Dort sind die Unterschiede zur physischen Anlage minimal. Das lässt sich dadurch erklären, dass durch die dynamische Optimierung das zukünftige Verhalten mit betrachtet werden kann. So werden die Auswirkungen von Rauschen oder Ausreißern in den Messdaten von der physischen Anlage abgemildert, sichtbar im Vergleich von Abbildung 5.7 und Abbildung 5.9. Zudem treten die niedrigsten Anzahlen an Ausführungen der Simulationskomponenten auf. Das lässt sich dadurch erklären, dass die

Tabelle 5.5.: Auswertung Motorsystem

Szenario	Nr.	MSE in rad/s^2	MAX in rad/s	N_{ges}
Ohne Sync	-	55,14	27,74	4203
Parameter d				
$T_{\text{P}} = 1$	a)	1,43	7,02	19 875
$T_{\text{P}} = 1$ schnell	b)	1,43	7,02	9345
$T_{\text{P}} = 5$, u statisch	c)	4,45	15,81	32 818
$T_{\text{P}} = 5$, u dynamisch	d)	0,69	9,12	48 616
$T_{\text{P}} = 2$, u dynamisch	e)	0,53	4,5	16 096
Strom $\mathbf{I}_{\text{in}}$				
$T_{\text{P}} = 1$	f)	5,70	7,60	82 254
$T_{\text{P}} = 1$ schnell	g)	5,70	7,60	30 454
$T_{\text{P}} = 5$, u statisch	h)	1,74	10,64	41 929
$T_{\text{P}} = 5$, u dynamisch	i)	0,57	9,10	34 570
$T_{\text{P}} = 2$, u dynamisch	j)	66,57	65,42	44 180

Tabelle 5.6.: Anzahl der Ausführungen von Simulationskomponenten (N_{ges}) nach Gleichung 5.1 mit und ohne Performanceverbesserung bei Nutzung des Parameters d zur Zustandssynchronisierung

	a)/b)	c)	d)	e)
BOBYQA				
N_{ges} ohne	19 626	96 783	31 293	9694
N_{ges} mit	8604	45 495	11 639	5804
Verhältnis ohne	4,67	23,04	7,45	2,31
Verhältnis mit	2,04	10,83	2,77	1,38
L-BFGS				
N_{ges} ohne	229 638	920 958	769 653	55 233
N_{ges} mit	16 902	70 390	53 268	8148
Verhältnis ohne	54,68	219,28	183,25	13,15
Verhältnis mit	4,02	16,76	12,68	1,94

Optimierung gut funktioniert und so die Ausgänge des simulierten Systems stets nah am physischen System bleiben. Weiterhin trägt vor allem Ansatz 2 aus Unterabschnitt 4.4.2 zur Reduzierung der Ausführungen bei.

Neben der Nutzung von *BOBYQA* aus (King, 2009) wurde ebenfalls *L-BFGS* aus (Bochkanov, 2020) getestet. Damit sind die notwendigen Ausführungen von Simulationskomponenten bei gleicher Qualität der Zustandssynchronisierung stets höher. Hervorzuheben ist, dass die Nutzung der Prädiktion, als statische Optimierung oder als dynamische Optimierung, die Anzahl der Ausführungen stark erhöht, bedingt durch das approximieren der Ableitung. Erwähnenswert ist, dass diese zusätzlichen Ausführungen zu einem erheblichen Teil durch die Performanceverbesserung reduziert werden können: Einerseits, da aufgrund von Ansatz 1 aus Unterabschnitt 4.4.1 zwei der drei Simulationskomponente während der Optimierung nicht ausgeführt werden. Andererseits werden durch das approximieren der Ableitungen Folgen von Eingängen produziert, welche in weiteren Iterationen durch Ansatz 2 aus Unterabschnitt 4.4.2 wiedergenutzt werden können ohne die Simulationskomponente erneut auszuführen.

5.5. Hybrides System: Transportsystem

5.5.1. Beschreibung des Systems

Zur Validierung des Konzeptes der Zustandssynchronisierung erfolgte auch eine Untersuchung an dem Transportsystem in Abbildung 5.13. Im Gegensatz zum vorherigen System in Abschnitt 5.4 handelt es sich bei dem Transportsystem um ein hybrides System. Die Stellsignale sind binär. Die Messsignale sind kontinuierlich. Zudem erfolgt die Kommunikation zwischen Prozess und Regelungs- und Steuerungsfunktion nachrichtenbasiert. Die Validierung der Methodik der Zustandssynchronisierung an diesem Transportsystem wurde in (Zipper und Diedrich, 2018b) vorgestellt. Das System besteht aus folgenden Elementen:

(1) Optischer Sensor für die Aktivierung von Band 2

(2) Band 2

(3) Band 1

(4) Rutsche

(5) Motor für Band 2

(6) Motor für Band 1

(7) Produkte

(8) Optischer Sensor für die Aktivierung von Band 1

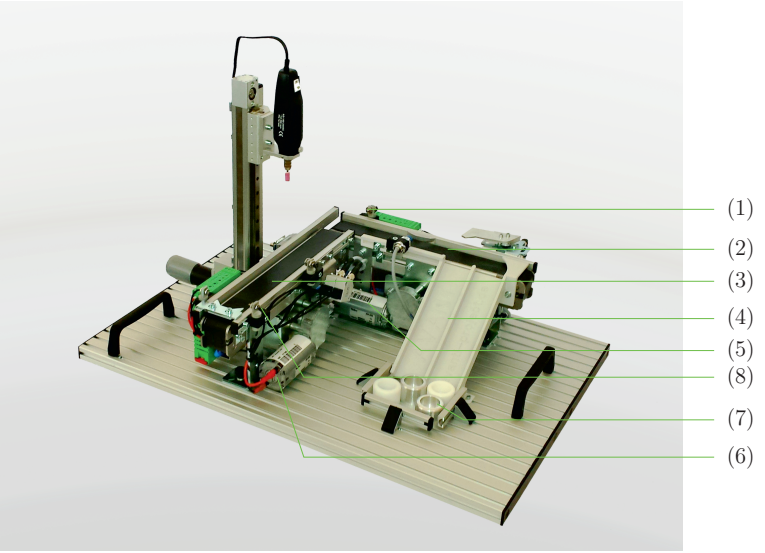


Abbildung 5.13.: Transportsystem

(9) Kamera zur Erfassung der Produktposition

Das Transportsystem stellt einen Fertigungsprozess nach. Produkte können über zwei Förderbänder (2) und (3) transportiert werden. Am Ende des Fertigungsprozesses werden die Produkte über eine Rutsche (4) aus dem Transportsystem heraus transportiert. Während sich ein Produkt in der Mitte des ersten Förderbandes befindet, kann es von der Bohrmaschine bearbeitet werden. Diese Funktion wird für die Validierung der Synchronisation nicht genutzt. Bevor ein Produkt das Transportsystem über die Rutsche verlässt, kann es sortiert werden, je nachdem ob es aus Metall oder aus Kunststoff ist. Dies wird mithilfe eines induktiven Sensors und Sortierers auf dem zweiten Förderband erreicht. Auch diese Funktion des Demonstrators wird für die Validierung der Synchronisation nicht genutzt. Als Sensoren werden zur Validierung lediglich die optischen Sensoren und eine zusätzlich angebrachte Kamera (9) genutzt (in Abbildung 5.13 nicht enthalten). Am Anfang des ersten Förderbandes befindet sich ein optischer Sensor (8). Mit dessen Hilfe ist bekannt, dass ein Produkt auf Förderband eins liegt. Daraufhin wird dieses Förderband eingeschaltet. Erreicht ein Produkt das Ende des ersten Förderbandes, soll dieses deaktiviert werden und das zweite Förderband soll durch den Sensor (1) aktiviert werden. Dieses bleibt so lange aktiv, bis das Produkt aus dem Transportsystem entfernt wird. Die Förderbänder werden durch die Motoren (5) und (6) angetrieben. Mit Hilfe der Kamera kann das Produkt an jedem Ort des Transportsystems lokalisiert werden.

In Abbildung 5.14 werden die Informationsflüsse zur Validierung des Transportsystems

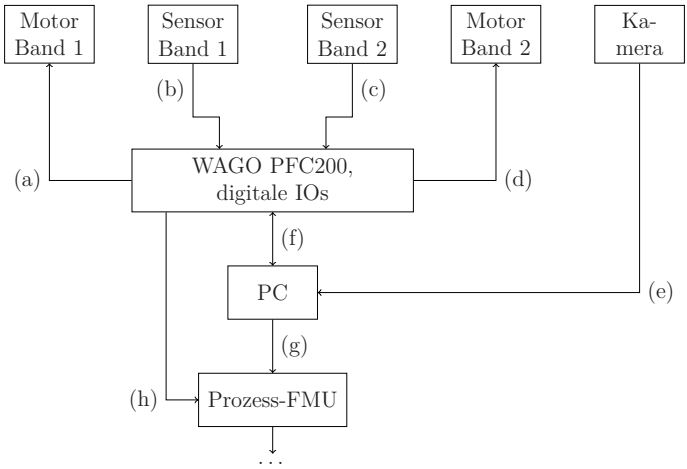


Abbildung 5.14.: Informationsflüsse Transportsystem

dargestellt. Die Sensoren und Aktoren sind über eine digitale Ein-Ausgangsbaugruppe an eine *WAGO PFC200* Steuerung angeschlossen (a)-(d). Die Regelungs- und Steuerungsfunktion ist als verteiltes Steuerungsprogramm durch *DOME* auf der Steuerung implementiert (siehe Beschreibung in Abschnitt 5.1, (*DOME* 2020)). Die Bilderkennung für das Erfassen der Produktposition durch die Kamera ist auf dem PC als Teil des verteilten Steuerungsprogramms implementiert. Die Kamera überträgt das Video per USB (e) zum PC. Für eine auf dem PC befindliche Visualisierung stellt das Steuerungsprogramm Informationen über den Zustand des Transportsystems bereit (f): Geschwindigkeiten der Förderbänder, Zustände der Sensoren, Position und Id der Produkte. Diese Informationen werden ebenfalls von der Simulationskomponente für die Anbindung an die Co-Simulation (*Prozess-FMU*) genutzt (g) und (h).

Durch die Alterung des Demonstrators bedingt, kann es passieren, dass Produkte beim Übergang zwischen Förderband eins und Förderband zwei eine Weile hängen bleiben. Dies kann dazu führen, dass Produkte deutlich später aus dem Transportsystem abtransportiert werden, oder gar auf Förderband zwei verbleiben und manuell entfernt werden müssen. Dieser Alterungsaspekt soll den Nutzen der Synchronisation sichtbar machen und bildet so das Szenario für die Validierung der hybriden Zustandssynchronisierung an dem Transportsystem.

Tabelle 5.7 fasst die Möglichkeiten der Regelungs- und Steuerungsfunktion zur Beeinflussung des Transportsystems abhängig von den optischen Sensoren zusammen. Insgesamt existieren zur Regelungs- und Steuerungsfunktion drei Regeln mit unterschiedlicher Priorität. Regeln mit einer höheren Priorität haben Vorrang.

Der Startzustand für die Validierung ist: Beide Förderbänder sind leer und es wird ein

Tabelle 5.7.: Beispiel Steuerung Transportsystem

Nr.	Bedingung	Aktion	Priorität
1	Produkt auf Band 1	aktiviere Band 1	1
2	Produkt auf Band 2	aktiviere Band 2	1
3	Produkt beim Übergang Band 1 nach Band 2	deaktiviere Band 1	2

Produkt auf Förderband eins gegeben.

5.5.2. Aufbau der Co-Simulation

Die Co-Simulation für das Transportsystem besteht aus fünf Simulationskomponenten:

1. Simulationskomponente für die Verbindung zu den Signalen zwischen der Regelungs- und Steuerungsfunktion und der physischen Anlage.
2. Da diese Kommunikation nachrichtenbasiert ist, also Werte nur einmalig übertragen werden wenn sie sich ändern, wird eine Simulationskomponente zur Umsetzung der nachrichtenbasierten Kommunikation in die zyklische Kommunikation der Co-Simulation notwendig
3. Simulationskomponente für das horizontale Förderband
4. Simulationskomponente für das vertikale Förderband
5. Simulationskomponente für das Verhalten des Produktes

Der Zusammenhang zwischen diesen Simulationskomponenten ist in dem Graphen in Abbildung 5.15 dargestellt. Die Parameter der Simulationskomponenten für die Förderbänder sind in Tabelle 5.8 und die Parameter der Simulationskomponente für das Produkt in Tabelle 5.9 aufgelistet.

5.5.3. Parametrierung der Simulationsmodelle und der Optimierung

Die Parametrierung der Simulationskomponenten ist in Tabelle 5.10 dargestellt. Wo Parameter vom Optimierer während der Zustandssynchronisierung angepasst werden, enthält Tabelle 5.10 den initialen Wert des Parameters. Es gilt $T_{\mathrm{D}} = 2$. Zudem wird $T_{\mathrm{P}} = 2$ gewählt und eine statische Optimierung durchgeführt. Die Variable der Zielfunktion ist hier *enabled* und kann nur die Werte 0 oder 1 annehmen. Daher muss ein Optimierungsalgorithmus maximal zwei Iterationen durchlaufen. Zuerst wird die Lösung der vorherigen Iteration des Optimierungsalgorithmus gewählt. Ist dies nicht optimal, wird die logische Negation der Lösung der vorherigen Iteration des Optimierungsalgorithmus gewählt.

Zur Berechnungen der Metriken MSE und MAX werden die Positionen p_{x} des simulierten Produktes und $\hat{p_{\mathrm{x}}}$ des physischen Produktes (von Abbildung 5.14 (e)) betrachtet.

Tabelle 5.8.: Variablen der Simulationskomponente Förderband

Variable	Einheit	Richtung	Bedeutung
x, y	mm	Parameter	Position
w, h	mm	Parameter	Ausdehnung in x beziehungsweise y Richtung
v_x, v_y	mm/s	Parameter	Geschwindigkeit in x beziehungsweise y Richtung
p_x, p_y	mm	Eingang	Position des Produktes
enabled	bool	Eingang	an / aus
p_v	mm/s	Ausgang	Geschwindigkeit des Produktes

Tabelle 5.9.: Variablen der Simulationskomponente Produkt

Variable	Einheit	Richtung	Bedeutung
$p_\mathrm{v,x}$, $p_\mathrm{v,y}$	mm/s	Eingang	Geschwindigkeit des Produktes
x, y	mm	Ausgang	Position

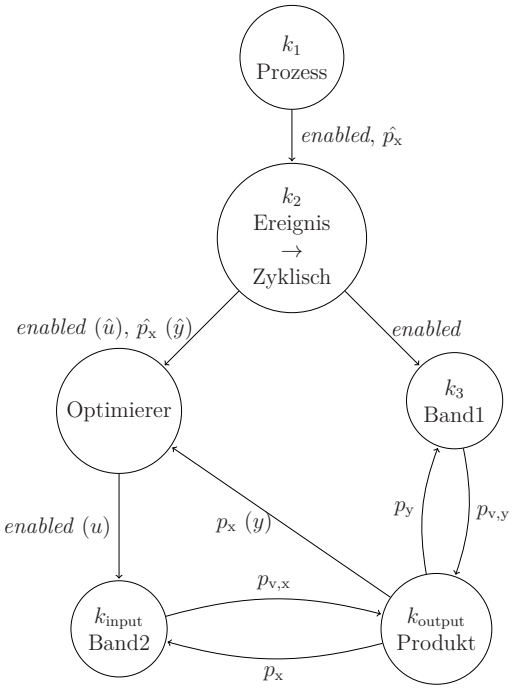


Abbildung 5.15.: Co-Simulation des Transportsystems

Tabelle 5.10.: Parametrierung (Startwerte) Transportsystem

Parameter	Wert	Parameter	Wert
Förderband 1			
x	0 mm	y	0 mm
w	30 mm	h	270 mm
v_{x}	0 mm/s	v_{y}	150 mm/s
Förderband 2			
x	0 mm	y	300 mm
w	270 mm	h	30 mm
v_{x}	70 mm/s	v_{y}	0 mm/s

5.5.4. Auswertung

In Abbildung 5.16 wird die nicht synchronisierte Simulation des Transportsystems dargestellt. Es werden lediglich die für den Transport über Band zwei relevanten Größen gezeigt. Bis ca. 3 s ist das Band ausgeschaltet. x und $\hat{x}$ sind verschieden aufgrund der Initialisierung der Simulationskomponente — an welcher Position genau das Produkt von Band eins auf Band zwei übergeht hängt auch davon ab, wie das Produkt auf Band eins gelegt wird und ist daher etwas zufällig. Sobald Band zwei eingeschaltet wird, bewegt sich das Produkt nur in der simulierten Anlage und hängt in der physischen Anlage eine Weile am Übergang zwischen Band zwei und Band eins. Erst bei ca. 5 s fängt es sich an zu bewegen. Zu sehen ist außerdem, dass die Anstiege der Kurven von x und $\hat{x}$ verschieden sind, also das simulierte Band zwei mit höherer Geschwindigkeit läuft.

Die synchronisierte Simulation ist in Abbildung 5.17 dargestellt. Wieder ist zu sehen, dass zwar das physische Band zwei aktiviert wird, aber das physische Produkt am Übergang hängen bleibt. Durch die Zustandssynchronisierung wird das simulierte Band zwei jedoch weiterhin angehalten. Erst wenn sich das physische Produkt tatsächlich bewegt wird auch das simulierte Band zwei aktiviert. Zudem ist zu sehen, dass die Anstiege der Kurven von x und $\hat{x}$ gleich sind. Dies wird dadurch erreicht, dass der Optimierer das simulierte Band zwei abwechselnd ein und aus schaltet.

Das Funktionieren der Zustandssynchronisierung wird auch durch die Metriken MSE und MAX bestätigt. Diese sind in Tabelle 5.11 dargestellt. Beide Metriken ergeben deutlich niedrigere Abweichung zwischen physischen System und simulierten System bei aktivierter Zustandssynchronisierung.

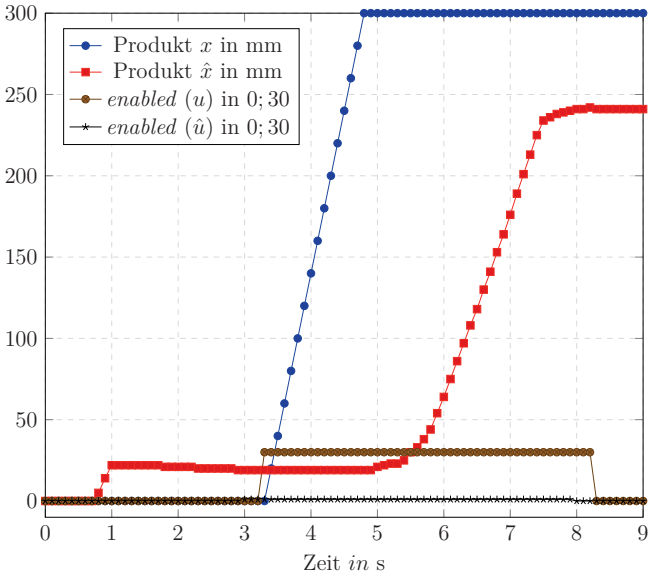


Abbildung 5.16.: Förderband nicht synchronisiert

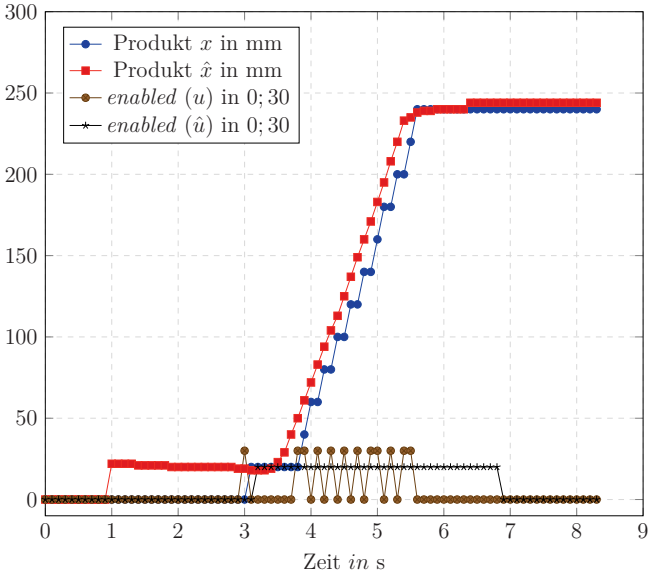


Abbildung 5.17.: Förderband synchronisiert

Tabelle 5.11.: Auswertung Transportsystem

Szenario	MSE in mm^2	MAX in mm
Ohne Zustandssynchronisierung	15 623	281
Mit Zustandssynchronisierung	226	33

Am Transportsystem wird das Verhalten untersucht, welches eintritt wenn das Förderband der physischen Anlage langsamer läuft als das Förderband in der simulierten Anlage und das physische Produkt festhängt. Die Zustandssynchronisierung funktioniert, in dem das Förderband in der simulierten Anlage angehalten wird. Wäre das Förderband in der physischen Anlage schneller als das Förderband in der simulierten Anlage, würde dieser Ansatz nicht funktionieren. Die Zustandssynchronisierung wäre für diesen Fall mit Hilfe der Eingänge *enabled* der Simulationsmodelle der Förderbänder nicht möglich. Eine Möglichkeit, die Zustandssynchronisierung in einem solchen Fall zu erreichen, ist in Abbildung 5.18 dargestellt. Die Simulationsmodelle der Förderbänder haben einen Parameter für die Geschwindigkeit, welcher jedoch nur initial gesetzt werden kann und während der Simulation nicht geändert werden kann. Die Idee ist daher, den Ausgang $p_{\mathrm{v,x}}$ entsprechend so zu manipulieren, dass die Zustandssynchronisierung erreicht wird. Dieses Vorgehen würde für schneller und langsamere physische Förderbänder funktionieren.

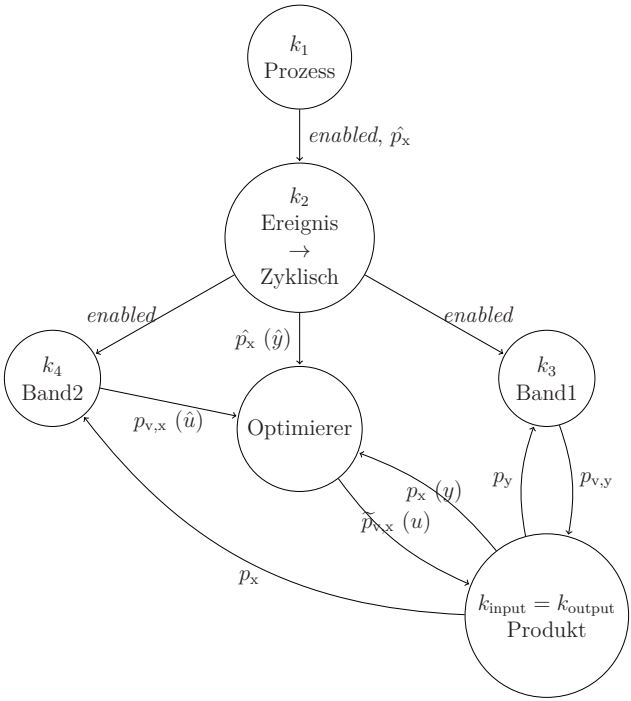


Abbildung 5.18.: Möglichkeit der Zustandssynchronisierung, wenn das physische Förderband schneller ist als das simulierte Förderband

5.6. Hybrides System: Zylindersystem

5.6.1. Beschreibung des Systems

Zur Validierung wird weiterhin der in Abbildung 5.19 und Abbildung 5.20 abgebildete Demonstrator genutzt. Dieser wurde erstmalig in (Zipper, Auris u. a., 2018) vorgestellt.

Die Anlage enthält einen pneumatischen Zylinder, welcher beim Ausfahren den Balken nach oben bewegt und beim Einfahren den Balken nach unten bewegt. In der Abbildung sind der Kompressor und die Elektromotoren aus Gründen der Übersichtlichkeit weggelassen. Die Kraft, welche auf den Zylinder wirkt, kann durch das Verfahren der Masse entlang des Balkens mithilfe des Elektromotors beeinflusst werden. Daher beeinflusst die Position der Masse die Zeit, welche der pneumatische Zylinder zum Ein- und Ausfahren benötigt.

Die Informationsflüsse werden in Abbildung 5.21 dargestellt. Der pneumatische Zylinder wird durch eine Ventilinsel angesteuert (a). Die Regelungs- und Steuerungsfunktion sendet Signale zu dieser Ventilinsel um den Zylinder einzufahren oder auszufahren via

Abbildung 5.19.: Zylindersystem (Quelle Foto: Mercedes-Benz AG)

PROFINET (b). Zwei Drosseln beeinflussen das Verhalten des Zylinders. Drossel eins beeinflusst das Ausfahren und Drossel zwei beeinflusst das Einfahren. Die Ventilinsel sendet Sensorwerte (b), sobald der Zylinder eine Endlage erreicht. Andere Variablen des Zylinders, wie zum Beispiel die Geschwindigkeit oder Kräfte sind nicht messbar. Durch eine interne Software des Herstellers des Demonstrators werden die Stell- und Messsignale bezüglich der Ventilinsel als CSV-Datei aufgezeichnet (c) und (d).

Für die Validierung wird ein Sensor zur Messung des Luftstroms durch den Zylinder hinzugefügt. Solche Sensoren sind normalerweise nicht in den Zylindern enthalten, welche in industriellen Anlagen ihre Anwendung finden. Hier werden sie hinzugefügt, um die Genauigkeit der Synchronisation und der Simulation zu bestimmen, auch hinsichtlich nicht messbarer Zustände. Die Validierung der Synchronisation anhand dieses Zylindersystems wurde erstmals in (Zipper und Diedrich, 2019) vorgestellt.

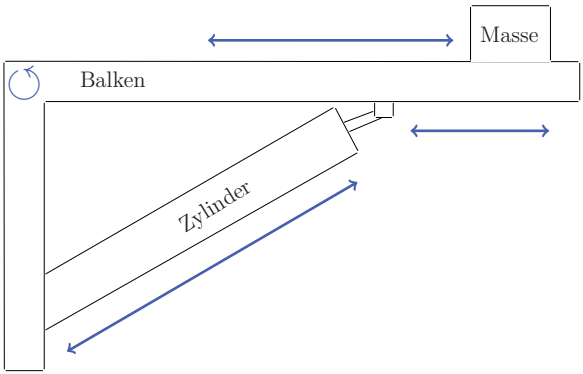


Abbildung 5.20.: Schema Zylindersystem nach (Zipper, Auris u. a., 2018)

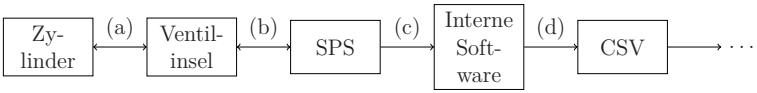


Abbildung 5.21.: Informationsflüsse des Zylindersystems

5.6.2. Aufbau der Co-Simulation

In Tabelle 5.12 werden die Variablen der Simulationskomponente *3D und Physik Engine* aufgelistet, in Tabelle 5.13 sind die Variablen der Simulationskomponente des Zylinders beschrieben. Der Aufbau der Co-Simulation ist in Abbildung 5.22 dargestellt. Die Simulationskomponenten *3D und Physik Engine* sowie *Zylinder* sind Produkte, welche auch für den industriellen Einsatz genutzt werden, zum Beispiel für die virtuelle Inbetriebnahme. Das interne Verhalten dieser Komponenten ist daher gänzlich unbekannt.

Die Kraft, welche der pneumatische Zylinder zum Einfahren und Ausfahren benötigt wird von der Masse des Balkens und der darauf liegenden, verfahrbaren Masse beeinflusst. Zur Umsetzung dieses Zusammenhangs wird eine Simulationskomponente, welche die Signallogik und die physikalischen Zusammenhänge innerhalb des pneumatischen Zylinders modelliert, gekoppelt mit einer Simulationskomponente, welche anhand von CAD-Daten (3D Modelle, Massen, Gelenke) eine Simulation der Physik starrer Körper durchführt. Die Simulationskomponente des Zylinders gibt die aus den anliegenden Signalen der Regelungs- und Steuerungsfunktion und den wirkenden Kräften resultierende Ausfahrposition des Zylinders an die 3D und Physik Engine. Diese berechnet die aufgrund der geometrischen Gegebenheiten resultierende Kraft auf den pneumatischen Zylinder. Diese Kraft wird als Eingang in die Simulationskomponente des Zylinders gegeben.

Eine Schwierigkeit bei der Validierung der Zustandssynchronisierung an dem Zylindersystem besteht darin, dass der Zustand der Simulationskomponente des Zylinders nicht zurückgesetzt werden kann — die optionalen Features von FMI *fmi2GetFMUstate* und

Tabelle 5.12.: Variablen der Simulationskomponente 3D- und Physik Engine

Variable	Einheit	Richtung	Bedeutung
cylinderjointLockPosition	m	Eingang	Position des Zylinders
cylinderjointForce	N	Ausgang	Kraft, welche auf den Zylinder wirkt

Tabelle 5.13.: Variablen der Simulationskomponente Zylinder

Variable	Einheit	Richtung	Bedeutung
throttle extend	1 = 100% offen, 0 =zu	Parameter	Drossel 1, legt Luftdurchflusses beim Ausfahren fest
throttle rectract	1 = 100% offen, 0 =zu	Parameter	Drossel 2, legt Luftdurchflusses beim Einfahren fest
power supply	bar	Parameter	Versorgungsdruck
move out	bool	Eingang	fahre Zylinder aus
move in	bool	Eingang	fahre Zylinder ein
F_{ext}	N	Eingang	Externe Kraft auf den Zylinder
F_{pneu}	N	Ausgang	resultierende Kraft
pos	m	Ausgang	Position
v	m/s	Ausgang	Geschwindigkeit
airflow	l/s	Ausgang	Luftdurchfluss
rectract signal	bool	Ausgang	Eingefahren — Endlage erreicht
extend signal	bool	Ausgang	Ausgefahren — Endlage erreicht

fmi2SetFMUstate werden nicht unterstützt. Um die Validierung dennoch durchführen zu können, werden Anforderungen an die praktische Relevanz zurückgestellt. Statt *einer* Instanz der Simulationskomponente für den Zylinder werden eine große Menge Instanzen dieser Simulationskomponente der Co-Simulation hinzugefügt. Die genaue Anzahl richtet sich nach der Schrittweite, der gesamten Simulationszeit und der durchschnittlich zu erwartenden Anzahl der Iterationen des Optimierungsalgorithmus, zum Beispiel $1/_{0{,}005} \cdot 3 \cdot 20 = 12000$ Instanzen. Mit jeder Iteration der Co-Simulation, die nicht innerhalb einer Iteration der Optimierung stattfindet, werden alle Instanzen simuliert. Davon ist jedoch nur eine Instanz als *aktiv* markiert. Nur die als aktiv markierte Simulationskomponente wird während einer Iteration des Optimierungsalgorithmus ausgeführt. Zwischen zwei Iterationen des Optimierungsalgorithmus müsste normalerweise das Zurücksetzen der Simulationskomponenten erfolgen. Stattdessen wird die aktive Instanz auf eine Instanz gesetzt, welche in der gesamten Simulation bis dahin noch nicht aktiv war. So kann

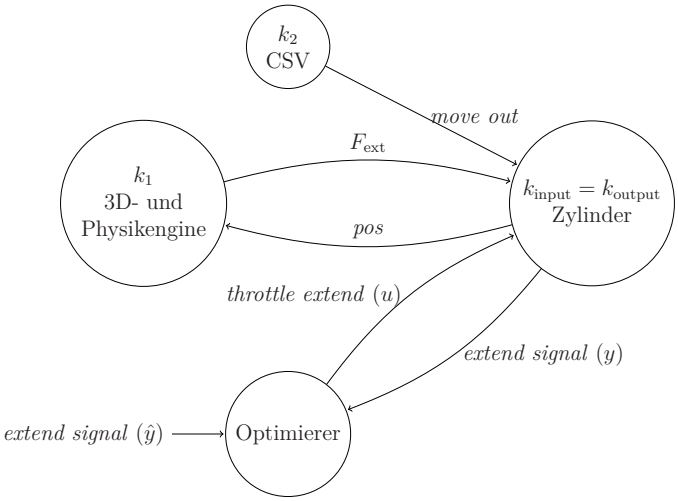


Abbildung 5.22.: Co-Simulation des Zylindersystems

die Zustandssynchronisierung validiert. Für den praktischen Einsatz ist dieses Vorgehen jedoch nicht geeignet, denn durch die hohe Anzahl der parallel ausgeführten Simulationskomponenten ist die Co-Simulation nicht echtzeitfähig.

5.6.3. Parametrierung der Simulationsmodelle und der Optimierung

Für die Messungen wird ein Druck von $4bar$ bereitgestellt und die Masse auf dem Balken zentriert, sowohl in der physischen Anlage, als auch in der simulierten Anlage. Die Drosseln werden in der simulierten Anlage auf ein Prozent geöffnet gestellt. Die tatsächliche Drosselstellung in der physischen Anlage wird nicht gemessen.

Die Drosselstellung beeinflusst die Geschwindigkeit und den Luftdurchfluss während des gesamten Ausfahrvorgangs. Da jedoch für die Synchronisation nur die Variable *extend signal* beim Erreichen des Endpunktes und das dazugehörige Signal der physischen Anlage betrachtet werden, steht die Wirkung der Drosselstellung erst am Ende des Ausfahrvorgangs fest. Dies muss bei der Wahl von T_{D} und T_{P} berücksichtigt werden. Zur Validierung am Zylindersystem wird daher bei der Wahl von T_{D} und T_{P} wie folgt vorgegangen.

Die Grundidee ist es, T_{D} und T_{P} abhängig vom Zeitpunkt des Erreichens der Endlage (hier $T_{extend\ signal}$) festzulegen. Die booleschen Signale über das Erreichen der Endlage werden in 1 und 0 umgerechnet, für Endlage ist erreicht beziehungsweise nicht erreicht. Nun wird ein Zeitfenster T_{F} festgelegt.

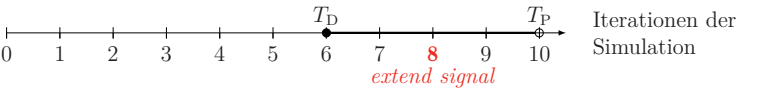


Abbildung 5.23.: Wahl von T_D und T_P zur Synchronisation des Zylindersystems

Tabelle 5.14.: Parametrierung (Startwerte) Zylindersystem

Parameter	Wert	Parameter	Wert
Zylinder			
throttle extend	0,01	*throttle rectract*	0,01
power supply	4 bar		

$$T_D = T_{extend\ signal} - \frac{T_F}{2} \tag{5.4}$$

$$T_P = T_{extend\ signal} + \frac{T_F}{2} \tag{5.5}$$

In Abbildung 5.23 ist ein Zeitstrahl als Beispiel dargestellt. An dem Zeitstrahl stehen die Iterationen der Simulation. In dem Beispiel kommt das Signal des Erreichens der Endlage der physischen Anlage zum Simulationszeitschritt 8. Das Zeitfenster ist hier auf $T_F = 4$ festgelegt. Dadurch ergibt sich für das Beispiel $T_D = 6$ und $T_P = 10$. Nun wird die Zielfunktion der Optimierung in Gleichung 4.3 betrachtet. Da die booleschen Signale zu 0 und 1 umgewandelt werden, kann $(\Delta y)^2$ in jedem Zeitschritt entweder nur 0 oder 1 betragen. $(\Delta y)^2$ ist 0 für den Fall, dass physische und simulierte Anlage dieselbe Aussage über das Erreichen der Endlage treffen. Es ist 1 für den Fall, dass in einem System der Zylinder vollständig ausgefahren ist und in dem anderen System noch nicht. Durch die Summe in der Zielfunktion werden die Unterschiede zwischen den Systemen aufsummiert. Das Optimum beträgt 0. T_D wird kleiner als der eigentliche Zeitpunkt des Erreichens der Endlage gesetzt, um zu bestrafen, wenn in der simulierten Anlage die Endlage eher erreicht wird.

In Tabelle 5.14 sind die Parameter der Simulationskomponenten aufgelistet. Als Optimierungsalgorithmus kommt ein Nelder-Mead Simplex zum Einsatz. Die Simulationsschrittweite beträgt 0,005 s. Es wird $T_F = 10$ genutzt. Es wird eine statische Optimierung durchgeführt, um das Zeitfenster für die Wahl von T_D und T_P umsetzen zu können.

Die Metriken werden sowohl für die Ausgänge, welche die Zustandssynchronisierung betrachtet, als auch für die Zustände ermittelt. Die Berechnung von MSE und MAX für die Ausgänge erfolgt für die Variablen *extended signal* der simulierten und physischen Zylinder. MSE und MAX für die Zustände werden anhand der Variablen *airflow* ermittelt.

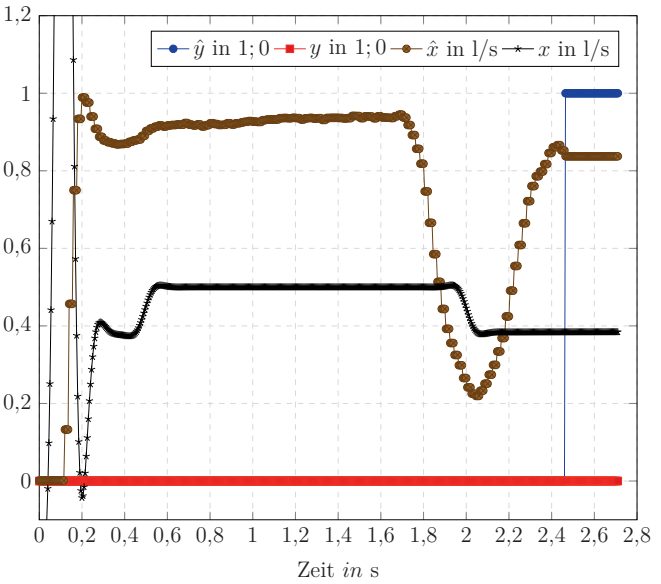


Abbildung 5.24.: Zylindersystem ohne Zustandssynchronisierung, Drossel 1 bei 1%

5.6.4. Auswertung

Abbildung 5.24 stellt das Zylindersystem ohne Zustandssynchronisierung dar. Die abweichend gewählte Parametrierung führt zu einem deutlichen Unterschied des gemessen und simulierten Luftdurchflusses. Darüber hinaus führt die abweichende Parametrierung zu unterschiedlichen Zeiten an denen der physische und der simulierte Zylinder voll ausgefahren sind, also das Signal für das Erreichen der Endlage anliegt. Die Ergebnisse der Validierung der Zustandssynchronisierung sind in Abbildung 5.25 zu sehen. Die Endlagen der Zylinder werden fast zur gleichen Zeit erreicht. Zudem gleichen sich die Kurven des Luftdurchflusses signifikant besser als im Fall ohne Zustandssynchronisierung.

Die Anwendung der Metriken MSE und MAX auf das Zylindersystem ist in Tabelle 5.15 dargestellt. Die Ausgänge, welche von der Zustandssynchronisierung betrachtet werden sind *extended signal*. Für diese ist mit aktivierter Zustandssynchronisierung eine Verbesserung gegenüber dem Szenario ohne Zustandssynchronisierung sichtbar. Für den Anwendungsfall Beobachter sind die Zustände *airflow* interessant. Hier ist für MSE und MAX eine geringfügige Verschlechterung bei aktivierter Zustandssynchronisierung sichtbar. Bei der Analyse der Abbildungen 5.24 und 5.25 ist eine Schwingung der Zustände des simulierten Zylinders zu erkennen. Diese Schwingung tritt unabhängig von der Verwendung der Zustandssynchronisierung auf und ist ursächlich dafür, dass durch die aktivierte Zustandssynchronisierung keine Verbesserung durch die Metriken sichtbar wird.

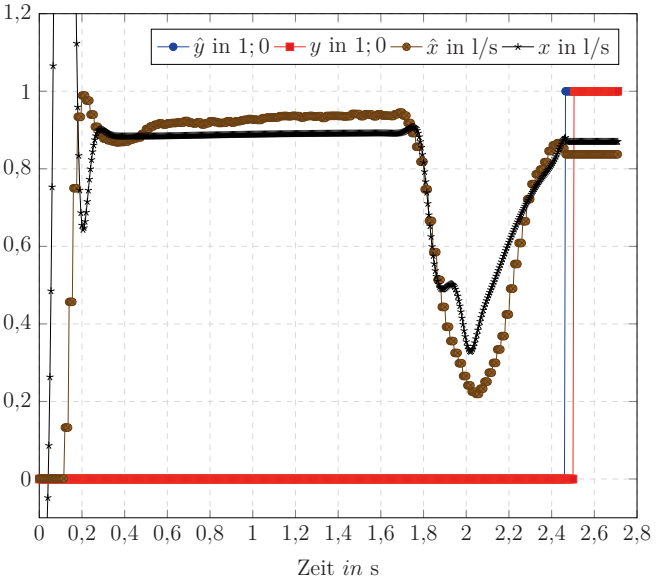


Abbildung 5.25.: Zylindersystem mit Zustandssynchronisierung, Drossel 1 bei 2, 3%

Zum Vergleich können die Metriken ab einen späteren Zeitpunkt nach der Schwingung gebildet werden. In Tabelle 5.15 ist dies für die Zeit ab 0,2 s dargestellt. Daraus ist ersichtlich, dass nach den Schwingungen zu Beginn der Simulation die Zustandssynchronisierung durchaus eine Verbesserung hinsichtlich der Synchronisation der Zustände erreichen kann. Für die tatsächliche Anwendung der betriebsparallelen Simulation in diesem Fall müssten die numerischen Eigenschaften der vorliegenden Co-Simulation jedoch verbessert werden.

Tabelle 5.15.: Auswertung Zylindersystem

extended signal y und $\hat{y}$	MSE	MAX
Ohne Zustandssynchronisierung	0,092	1
Mit Zustandssynchronisierung	0,014	1
airflow x und $\hat{x}$	MSE in $(\mathrm{l/s})^2$	MAX in l/s
Ohne Zustandssynchronisierung	0,38	3,37
Mit Zustandssynchronisierung	0,34	4,16
airflow x und $\hat{x}$ ab 0,2s	MSE in $(\mathrm{l/s})^2$	MAX in l/s
Ohne Zustandssynchronisierung	0,172	1,03
Mit Zustandssynchronisierung	0,006	0,35

5.7. Simulative Validierung der zeitlichen Synchronisation

Das in Abschnitt 4.6 vorgestellte Konzept zur zeitlichen Synchronisation wird, wie in (Zipper und Diedrich, 2018a) beschrieben, simulativ validiert.

Dazu wird das System zweiter Ordnung unter Gleichung 5.6 als physische Anlage genutzt. In Tabelle 5.16 sind die Parameter des Test-Systems und der zeitlichen Parameter für Gleichung 4.14, Gleichung 4.15 und Gleichung 4.16 aufgelistet.

$$\begin{bmatrix} \dot{\hat{x}}_1 \\ \dot{\hat{x}}_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -1/T^2 & -2d/T \end{bmatrix} \begin{bmatrix} \hat{x}_1 \\ \hat{x}_2 \end{bmatrix} + \begin{bmatrix} 0 \\ K/T^2 \end{bmatrix} \hat{u} \tag{5.6}$$

Tabelle 5.16.: Parametrierung zur Validierung der zeitlichen Synchronisation

Parameter	Wert	Parameter	Wert
T_R	0,0001 s	t_R	0,032 s
T_C	0 s	t_C	0,008 s
T_P	0,01 s	t_P	0,0001 s
T_M	0,004 s	t_M	0,004 s
T_S	0,004 s	t_S	0,004 s
T	0,125	K	1
d	0,225	$\hat{u}$	1 (Sprung)

Die physische Anlage wird durch eine Simulation von Gleichung 5.6 gebildet. Die Umsetzung der simulierten Anlage erfolgt auch auf Basis des Systems in Gleichung 5.6. Dieses wird zunächst diskretisiert und in die Form von Gleichung 3.2 gebracht und anschließend

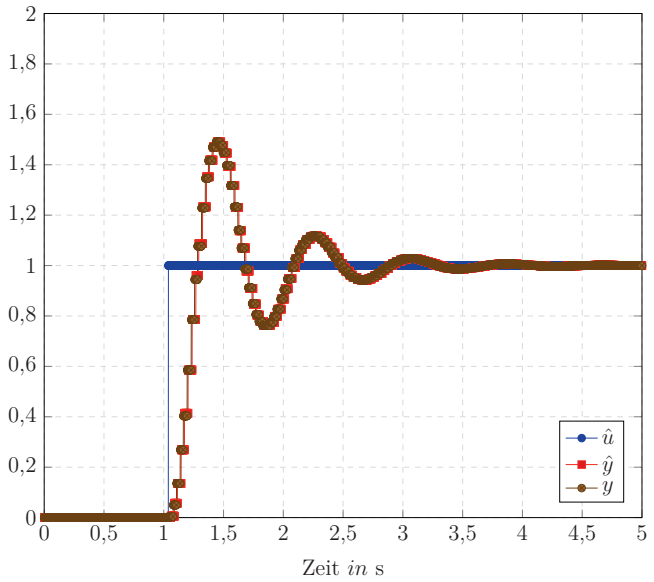


Abbildung 5.26.: Sprungantworten der physischen und der simulierten Anlage

in die Gleichungen entsprechend Algorithmus 2 eingesetzt. Dadurch entsteht eine *Simulation der Co-Simulation*.

Als Eingang wird ein Sprung zur Zeit $1s$ simuliert und in die physische und in die simulierte Anlage gegeben. Von beiden wird die Sprungantwort aufgenommen, welche dann für die Fälle mit und ohne Kompensation der Ausgangsverzögerung verglichen werden. Wie in Abbildung 5.26 zu sehen ist, zeigen physische und simulierte Anlage die gleichen Antworten. Die simulierte Anlage folgt der physischen Anlage.

In Abbildung 5.28 wird Δy zu jedem Zeitpunkt dargestellt. Da physische und simulierte Anlage auf dem selben mathematischen System basieren, würde ein optimaler Vergleich von beiden Systemen zu jedem Zeitpunkt ein $\Delta y = 0$ ergeben. Das bedeutet, dass die Differenz in Abbildung 5.28 als Fehler des Vergleichs interpretiert werden kann. Diese Differenzen treten auf, da der diskrete Algorithmus (Algorithmus 2) der Co-Simulation zum Ausführen der simulierten Anlage genutzt wird.

Zunächst werden die Verteilungsdichtefunktionen anhand von Gleichung 4.15 und Gleichung 4.16 berechnet. Die sich daraus ergebenden Verteilungsdichtefunktionen sind in Abbildung 5.27 dargestellt. Um einen Wert Δ_0 für alle Vergleiche zu erhalten, kann dieser von den Modalwerten der beiden Verteilungsdichtefunktionen gebildet werden. Da die Modalwerte nicht eindeutig sind, werden die am weitesten rechtsliegenden Modalwerte genutzt. Daraus resultiert eine Ausgangsverzögerung von $\Delta_\mathrm{o} = 0{,}008\,\mathrm{s}$.

Die Metriken aus Abschnitt 5.3 werden für die in Abbildung 5.28 gezeigten Differenzen

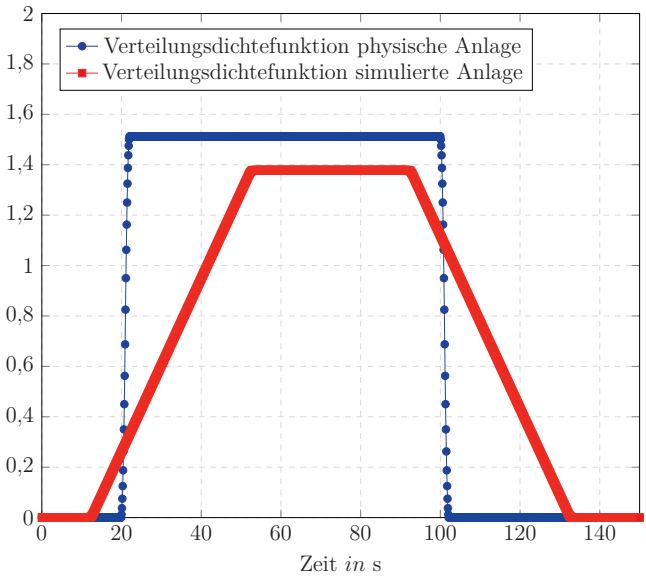


Abbildung 5.27.: Verteilungsdichtefunktion der Verzögerungen

Tabelle 5.17.: Werte der Metriken für die simulative Validierung der Zeitsynchronisation

Szenario	MSE	MAX
Ohne Kompensation	0,001 47	0,18
Mit Kompensation	0,000 27	0,18

Δy berechnet. In Gleichung 5.2 und in Gleichung 5.3 wird jeweils von $y_i - \hat{y}_i$ ausgegangen, wobei $\Delta y = y_i - \hat{y}_i$ gilt. Die Ergebnisse der Berechnung der Metriken sind in Tabelle 5.17 dargestellt.

Mit Kompensation der Ausgangsverzögerung ist die Differenz Δy kleiner als ohne diese Kompensation. Dies wird auch durch die in Tabelle 5.17 dargestellten Angaben bestätigt. Wie zu sehen ist, sind die Werte der absoluten Abweichung gleich. Die Werte der mittleren quadratischen Abweichung sind deutlich niedriger.

Wie in Abbildung 5.28 zu sehen, bleibt auch mit Berücksichtigung der Ausgangsverzögerung ein Fehler. Durch Gleichung 4.15 werden günstige Wertepaare für den Vergleich der physischen und der simulierten Anlage bestimmt. Jedoch sind die Ausgänge des physischen Systems nur zu bestimmten Zeitpunkten verfügbar, basierend auf der Datenerfassungskomponente. Es können Zeitpunkte berechnet werden, an denen keine Daten vorliegen. In diesem Fall muss das nächste verfügbare Datum genutzt werden. Daraus entsteht ein Fehler. Zudem wird durch Gleichung 4.15 ein durchschnittlicher für alle Vergleiche zu

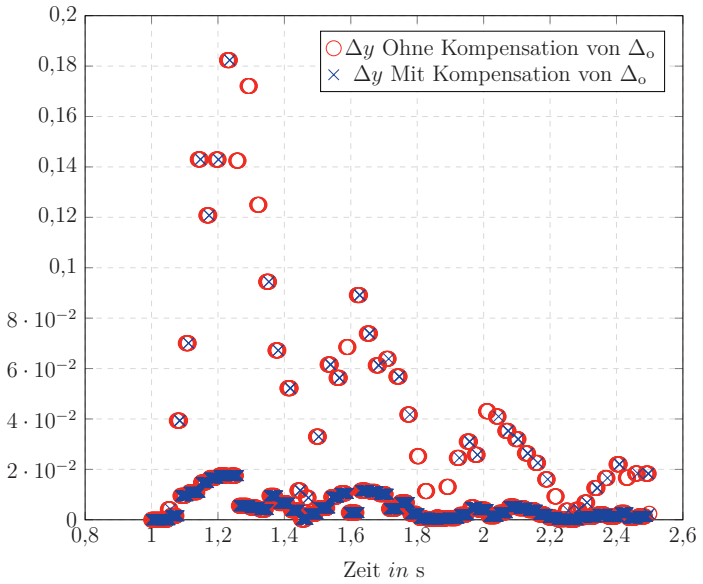


Abbildung 5.28.: Δy ohne Kompensation (o) und mit Kompensation (x)

verwendender Wert berechnet.

5.8. Auswertung der Validierung

Anhand der Validierung an den drei technischen Systemen wird ersichtlich, dass eine betriebsparallele Simulation erfolgreich mit einer physischen Anlage synchronisiert werden kann und dabei weiterhin echtzeitfähig bleibt. Die gezeigten technischen Systeme nutzen unterschiedle Signaltypen — wertekontinuierlich, wertediskret und ereignisdiskret. Dadurch wird die vielfältige Einsetzbarkeit der Zustandssynchronisierung nachgewiesen. Weiterhin wird am Motorsystem die Nutzung dynamischer Optimierung in Verbindung mit den Performanceverbesserungen zur genauen und echtzeitfähigen Synchronisierung gezeigt.

Zudem wird an dem Zylindersystem deutlich, dass für den praktischen Einsatz zwingend die optionalen Features des *FMI* Standards zum Export und Import des Zustandes aus beziehungsweise in die Simulationskomponenten benötigt werden.

6. Zusammenfassung und Ausblick

6.1. Zusammenfassung

Das Ziel dieser Arbeit war die Beantwortung der wissenschaftlichen Fragestellung aus Abschnitt 2.1:

> "Wie kann der Zustand einer Co-Simulation einer industriellen Anlage, von der nur die Struktur sowie die Ein- und Ausgänge der einzelnen Simulationskomponenten, nicht jedoch deren internes Verhalten, bekannt sind, mit der physischen Anlage synchronisiert werden?"

Aus dieser Fragestellung wurden die in Abschnitt Abschnitt 2.2 gezeigten Punkte abgeleitet:

(a) Nutzung von Simulationsmodellen aus der virtuellen Inbetriebnahme

(b) Nutzung von Black-Box Modellen

(c) Ermittlung nicht messbarer Zustände

(d) Synchronisierung von physischer Anlage und simulierter Anlage

Dazu wird zunächst der Stand der Wissenschaft untersucht und diese Arbeit eingeordnet. Gerade im Zusammenhang mit Industrie 4.0 und digitalen Zwillingen gewinnt aktuell die betriebsparallele Simulation weiter an Bedeutung. Zudem wird aus der Analyse aktueller Literatur ersichtlich, dass die Synchronisation dieser Simulation mit der physischen Anlage weiterhin ein offener Punkt ist, für welchen weiterer Forschungsbedarf existiert. Nur so kann bei Abweichungen zwischen der betriebsparallelen Simulation und der physischen Anlage, beispielsweise aufgrund von Alterung, ungenauer Parametrierung oder Änderungen am Prozess, die Simulation aussagefähig bleiben.

Als Grundlage für die weiteren Arbeiten wird anschließend ein formales Modell der Co-Simulation aufgestellt. Dieses richtet sich nach der Funktionsweise des *FMI*-Standards, wie er aktuell für Simulationen in der virtuellen Inbetriebnahme industrieller Anlagen eingesetzt wird. Zudem wird ein Standard-Algorithmus der Co-Simulation aus diesem Standard eingeführt. Das Modell der Co-Simulation betrachtet dabei die Simulationskomponenten als Blackbox, so dass die Punkte (a) und (b) berücksichtigt werden.

Der Hauptteil der Arbeit beschäftigt sich darauf aufbauend mit der Entwicklung der Zustandssynchronisierung. Im Gegensatz zu bisherigen Ansätzen der Synchronisation kommt die Zustandssynchronisierung ohne zusätzliche Steuerungskomponenten beziehungsweise SPS aus. Die Zustandssynchronisierung funktioniert nach dem Prinzip der Rückkopplung des Unterschiedes zwischen betriebsparalleler Simulation und physischer Anlage. Die Synchronisation wird durch einen Optimierungsalgorithmus auf Basis dieses Unterschieds online durchgeführt.

Dafür wird zunächst das Optimierungsproblem formal aufgestellt. Dabei kommt eine dynamische Optimierung zum Einsatz. Dieses Optimierungsproblem wird anschließend in den Standard-Algorithmus der Co-Simulation eingebettet. Dies geschieht so, dass der Algorithmus des Co-Simulationsmasters und der Optimierung getrennt sind. Für die Verbindung von Co-Simulationsmaster und Optimierungsalgorithmus wird eine Schnittstelle definiert. Dadurch ist es möglich, die Optimierung in eine Simulationskomponente auszulagern. Das erlaubt es, bestehende Co-Simulationen mit wenig Aufwand um die Zustandssynchronisierung zu erweitern.

Optimierer finden durch wiederholtes Auswerten einer Zielfunktion ein Optimum. Dies bedeutet in diesem Fall die mehrmalige Ausführung der Co-Simulation. Die Echtzeitfähigkeit der Zustandssynchronisierung wäre dadurch erschwert. Daher werden weiterhin Möglichkeiten erarbeitet, mit denen sich die Anzahl der Ausführungen der Co-Simulation deutlich verringern lassen, ohne dass dies einen Einfluss auf die Funktionsweise der Zustandssynchronisierung hat. Zudem werden Aspekte der zeitlichen Synchronisation erläutert.

Die entwickelten Methoden und Algorithmen werden anschließend validiert. Dazu wird ein Co-Simulationsmasteralgorithmus nach dem *FMI*-Standard implementiert, welcher die Zustandssynchronisierung umsetzen kann. Der Optimierer wird als separate Simulationskomponente implementiert, wobei unterschiedliche Optimierungsalgorithmen genutzt werden.

Das Motorsystem wird zur Validierung am ausführlichsten untersucht. Das Motorsystem besitzt eine hohe Dynamik und reagiert stark auf Änderungen der Parameter. Zudem wurden alle Simulationskomponenten im Rahmen der Arbeit entwickelt. So können im Gegensatz zu industriell eingesetzten Simulationskomponenten alle notwendigen Funktionen des *FMI* Standards, vor allem das Speichern und Zurücksetzens des internen Zustands, genutzt werden. Aus diesem Grund kann an dem Motorsystem ein großer Teil der Methodik getestet werden.

Anhand des Motorsystems kann für unterschiedliche Varianten der Einbindung der Simulationsmodelle und des Optimierers gezeigt werden, dass der Algorithmus der Zustandssynchronisierung funktioniert. Verschiedene Varianten — statische Optimierung ohne Horizont, statische Optimierung über einen zeitlichen Horizont und die dynamische Optimierung über einen zeitlichen Horizont — werden untersucht. Dabei wird festgestellt,

dass die dynamische Optimierung über einen Horizont von zwei Simulationsschritten die besten Ergebnisse liefert, sowohl qualitativ als auch in Bezug zu den benötigten Iterationen. Zudem wurde gezeigt, dass die notwendigen Iterationen durch die vorherigen Betrachtungen zur Echtzeit reduziert werden können. Generell wurde festgestellt, dass die Simulation des Motorsystems in Echtzeit stattfinden kann.

Das Transportsystem und das Zylindersystem werden als weitere Beispiele der Anwendung der Zustandssynchronisierung untersucht. Die Untersuchung zeigt damit die Einsetzbarkeit der Zustandssynchronisierung an unterschiedlichen technischen Systemen und beantwortet somit den Punkt (d) der wissenschaftlichen Fragestellung. Dabei kann an dem Transportsystem die Funktionsfähigkeit der Zustandssynchronisierung in Systemen mit booleschen Signalen und nachrichtenbasierter Kommunikation nachgewiesen werden.

In der Co-Simulation des Zylindersystems werden kommerzielle Simulationskomponenten, wie sie für den industriellen Einsatz finden können, genutzt. Zum einen kann die Zustandssynchronisierung an einem ereignisbasierten Fall demonstriert werden. Zum anderen kann die erfolgreiche Nutzung im Beobachter-Szenario gezeigt werden und erfüllt somit Punkt (c).

Die zeitliche Synchronisation für betriebsparallele Simulationen wird simulativ durchgeführt. So kann bestätigt werden, dass die Kompensation der Ausgangsverzögerung bereits die Qualität des Abgleichs zwischen physischer Anlage und simulierte Anlage verbessern kann.

Zusammenfassend lässt sich aufgrund der erzielten Ergebnisse feststellen, dass die wissenschaftliche Fragestellung in dieser Arbeit positiv beantwortet werden konnte.

6.2. Lessons-Learned

Aus der Umsetzung der Zustandssynchronisierung für die drei Systeme zur Validierung lassen sich Erkenntnisse für das Vorgehen zur Nutzung der Zustandssynchronisierung ableiten.

Zunächst ist es angeraten, Messreihen der Anlage aufzuzeichnen. Anhand dieser kann die Zustandssynchronisierung offline entwickelt werden. So kann die Wahl von u und y entschieden werden. Dabei können typische Änderungen an Parametern oder Eingängen eingefügt werden, wie sie im Betrieb der physischen Anlage auftreten können, beispielsweise durch Abnutzung, Verschleiß oder Störungen. So kann das funktionieren der Zustandssynchronisierung abgestimmt werden.

Dazu können verschiedene Optimierungsalgorithmen getestet werden. Es bietet sich an, die Implementierung der Zielfunktion von der Implementierung des Optimierungsalgorithmus zu trennen, so dass leicht zwischen einer Reihe von Algorithmen umgeschaltet werden kann. Die Nutzung ableitungsfreier Optimierungsalgorithmen ist sinnvoll, da Simulationskomponenten keine Ableitungen in der benötigten Form bereitstellen und die Approxi-

mation der Ableitung die Performance beeinträchtigen kann. Vor allem bei der Nutzung der Prädiktion in Zusammenhang dynamischer oder statischer Optimierung kann sich die Anzahl der zusätzlichen Ausführungen der Simulationskomponenten vervielfachen. Durch die Nutzung der beschriebenen Ansätze zur Reduzierung der Ausführungen können diese jedoch zu einem Teil vermieden werden. Daher sollten auch Algorithmen, welche Ableitungen nutzen in Betracht gezogen werden.

Die Zustandssynchronisierung mit Hilfe der dynamischen Optimierung erzielte die besten Resultate. Durch die Prädiktion ist die Zustandssynchronisierung weniger anfällig für Schwingungen und Ausreißer in den Messwerten der physischen Anlage. Durch die Ansätze zur Reduzierung der Anzahl der zusätzlichen Ausführungen können die durch die dynamische Optimierung anfallenden zusätzlichen Iterationen zu einem großen Teil ausgeglichen werden.

6.3. Ausblick

Weitere Möglichkeiten der Forschung für die Zustandssynchronisierung ergeben sich durch die Kombination mit datenbasierten Verfahren und Verfahren, die zur Simulation *Surrogat Modelle* nutzen. Dadurch könnte einerseits die Zustandssynchronisierung beschleunigt werden. Andererseits wären in solchen Modellen mehr Informationen über das interne Verhalten vorhanden. Dadurch ließe sich die Zustandssynchronisierung verbessern.

Weiterhin könnten Simulationskomponenten von Herstellern mit Informationen über das interne Verhalten angereichert werden. Beispielsweise könnte angegeben werden, in welchen Intervallen eine Simulationskomponente welches Verhalten (z.B. linear, Totzeit oder Anlaufphase) aufweist.

Neben diesen Verbesserungsmöglichkeiten könnten gerade Methoden der künstlichen Intelligenz dafür eingesetzt werden, den zu nutzenden Optimierungsalgorithmus auszuwählen und zu parametrieren. So wäre es möglich, während verschiedener Phasen des Anlagenbetriebs die Zielfunktion oder den Optimierungsalgorithmus anzupassen beziehungsweise unterschiedlich zu parametrieren.

Die Validierung der Zustandssynchronisierung sollte als nächster Schritt an komplexeren industriellen Anlagen erfolgen, zum Beispiel einer Fertigungszelle. So würden sich gerade die Echtzeiteigenschaften und die Robustheit weiter untersuchen lassen. Zunächst müsste für einen industriellen Einsatz eine Implementierung der Zustandssynchronisierung erfolgen, deren Quelltext weit besser optimiert wird, als es im Rahmen dieser Dissertation möglich war.

Generell müsste für einen industriellen Einsatz eine breitere Werkzeugunterstützung für das Erstellen von Simulationskomponenten geschaffen werden, welche die Funktion des Speicherns und Ladens des internen Zustands unterstützten.

Anhang A

Überblick über die im Rahmen der Arbeit erstellten Implementierungen

Zur Validierung der Ergebnisse und der Realisierung von Kapitel 5 wurden die folgenden Komponenten implementiert:

- Verteilter Co-Simulationsmaster für *FMI 2.0 für Co-Simulation*
- Implementierung *Nelder-Mead* Algorithmus
- Simulationskomponenten nach *FMI 2.0 für Co-Simulation*:
  - Optimierung mit Anbindung an verschiedene Optimierungsalgorithmen
  - CSV
  - Motor
  - PID Regler
  - Ereignis → Zyklisch
  - Förderband
  - Produkt
- Simulation in *Octave* für die Validierung der zeitlichen Synchronisation in Abschnitt 5.7

Aufgrund des Umfangs ist der Quelltext nicht in der Arbeit enthalten, kann aber auf Anfrage zur Verfügung gestellt werden.

Eigene Publikationen

Auris, Felix, Holger Zipper, Michael Brandl, Sebastian Süss und Christian Diedrich (2018). „Durchgängige Nutzung von Anlagenmodellen“. In: *atp magazin* 60 (06), S. 90–91.

Riedl, Matthias, Holger Zipper, Marco Meier und Christian Diedrich (2014). „Cyber-physical systems alter automation architectures“. In: *Annual Reviews in Control* 38 (1), S. 123–133.

Strahilov, Anton, Ender Yemenicioglu, Mario Thron, Holger Zipper, Matthias Riedl, Ulf Zimmermann, Ireneus Wior und Sebastian Süß (2016). „Improving the transition and modularity of the virtual commissioning workflow with AutomationML“. In: *4th AutomationML User Conference*.

Süß, Sebastian, Stephan Magnus, Mario Thron, Holger Zipper, Ulrich Odefey, Victor Fäßler, Anton Strahilov, Adam Kłodowski, Thomas Bär und Christian Diedrich (2016). „Test methodology for virtual commissioning based on behaviour simulation of production systems“. In: *Emerging Technologies and Factory Automation (ETFA), 2016 IEEE 21st International Conference on*. IEEE, S. 1–9.

Zipper, Holger (2019). „Gateway zur Erfassung, Auswertung und Bereitstellung von Maschinendaten nach den aktuellen Spezifikationen der Plattform Industrie 4.0“. In: *14. Magdeburger Maschinenbautage MMT, 24.-25.09.2019*. Otto-von-Guericke-Universität Magdeburg.

Zipper, Holger, Felix Auris, Anton Strahilov und Manuel Paul (2018). „Keeping the digital twin up-to-date—Process monitoring to identify changes in a plant“. In: *2018 IEEE International Conference on Industrial Technology (ICIT)*. IEEE, S. 1592–1597.

Zipper, Holger, Alexander Belyaev und Christian Diedrich (2019). „Generische Umsetzung von Verwaltungsschalen auf Basis der aktuellen Handlungsempfehlungen der Plattform Industrie 4.0“. In: *AUTOMATION* VDI-Berichte 2351. VDI Verlag, Düsseldorf, S. 1025–1043.

Zipper, Holger und Christian Diedrich (2018a). „Communication-delay-caused errors in process monitoring scenarios“. In: *Industrial Technology (ICIT), 2018 19th IEEE Conference on*. IEEE.

– (2018b). „Echtzeit-Prozessmonitoring auf Basis standardisierter Simulationsmodelle und Anlagenbeschreibungen“. In: *AUTOMATION* VDI-Berichte 2330. VDI Verlag, Düsseldorf, S. 1131–1141.

Zipper, Holger und Christian Diedrich (2019). „Synchronization of Industrial Plant and Digital Twin“. In: *Emerging Technologies and Factory Automation (ETFA), 2019 IEEE 24st International Conference on*. IEEE.

– (2020). „Effiziente Synchronisation einer industriellen Anlage und deren betriebsbegleitender Simulation“. In: *16. Fachtagung Entwurf komplexer Automatisierungssysteme (EKA)*.

Literaturverzeichnis

Ashtari Talkhestani, B., Nasser Jazdi, Wolfgang Schlögl und Michael Weyrich (2018). „A concept in synchronization of virtual production system with real factory based on anchor-point method“. In: *Procedia CIRP* 67, S. 13–17.

Ashtari Talkhestani, B., T. Jung, B. Lindemann, N. Sahlab, N. Jazdi, W. Schloegl und M Weyrich (2019). „An architecture of an Intelligent Digital Twin in a Cyber-Physical Production System“. In: *at - Automatisierungstechnik, Band 67, Heft 9, Seiten 762–782*. ISSN: ISSN (Print) 0178-2312. DOI: https://doi.org/10.1515/auto-2019-0039. URL: https://www.ias.uni-stuttgart.de/dokumente/publikationen/2019_An_architecture_of_an_Intelligent_Digital_Twin_in_a_Cyber-Physical_Production_System.pdf.

Auris, Felix, Jessica Fisch, Michael Brandl, Sebastian Süß, Abedalhameed Soubar und Christian Diedrich (2018). „Enhancing Data-Driven Models with Knowledge from Engineering Models in Manufacturing“. In: *2018 IEEE 14th International Conference on Automation Science and Engineering (CASE)*. IEEE, S. 653–656.

Bellman, R.E. und S.E. Dreyfus (1962). *Applied Dynamic Programming*. Princeton Legacy Library. Princeton University Press.

Bergs, Christoph und Michael Heizmann (2019). „Kombination unterschiedlicher Modellierungsansätze für die betriebsbegleitende Simulation industrieller Prozesse“. In: *at - Automatisierungstechnik* 67 (3), S. 183–192.

Biesinger, Florian, Davis Meike, Benedikt Kraß und Michael Weyrich (Dez. 2018). „A Digital Twin for the Production Planning based on Cyber-Physical Systems“. In: *12th CIRP Conference on Intelligent Computation in Manufacturing Engineering, CIRP ICME 2018; Procedia CIRP volume, CIRP ICME'18 proceedings*.

Bonilla, Javier, Jose Antonio Carballo, Lidia Roca und Manuel Berenguel (2017). „Development of an open source multi-platform software tool for parameter estimation studies in FMI models“. In: *Proceedings of the 12th International Modelica Conference, Prague, Czech Republic, May 15-17, 2017*. 132. Linköping University Electronic Press, S. 683–692.

Bonvini, Marco, Michael Wetter und Michael D Sohn (2014). „An fmi-based framework for state and parameter estimation“. In: *Proceedings of the 10 th International Modelica Conference; March 10-12; 2014; Lund; Sweden*. 096. Linköping University Electronic Press, S. 647–656.

Botchkarev, Alexei (2018). „Performance metrics (error measures) in machine learning regression, forecasting and prognostics: Properties and typology“. In: *arXiv preprint arXiv:1809.03006*.

Brandenbourger, Benjamin, Milan Vathoopan und Alois Zoitl (2016). „Behavior modeling of automation components using cross-domain interdependencies“. In: *2016 IEEE 21st International Conference on Emerging Technologies and Factory Automation (ETFA)*. IEEE, S. 1–4.

Chan, Rachel und Michael Krauss (2014). „Virtuelle Inbetriebnahme in der Prozessindustrie“. In: *atp magazin* 56 (06), S. 52–57.

Chombart, Patrick (2012). *MODELISAR Innovation report*. Techn. Ber. Dassault Systèmes.

Cristian, Flaviu und Christof Fetzer (1999). „The timed asynchronous distributed system model“. In: *IEEE Transactions on Parallel and Distributed systems* 10 (6), S. 642–657.

Damrath, Felix, Anton Strahilov, Thomas Bär und Michael Vielhaber (2015). „Experimental validation of a physics-based simulation approach for pneumatic components for production systems in the automotive industry“. In: *Procedia CIRP* 31, S. 35–40.

DOME (2020). https://www.ifak-ts.com/en/pf/ifak-dome/.

Drath, Rainer, Peter Weber und Nicolas Mauser (2008). „Virtuelle Inbetriebnahme — ein evolutionares Konzept fur die praktische Einfuhrung“. In: *AUTOMATION, VDI Wissensforum GmbH* 2032, S. 73.

FMI 2.0 (2014). *Functional Mockup Interface for Model Exchange and Co-Simulation 2.0*.

Gedda, Sofia, Christian Andersson, Johan Åkesson und Stefan Diehl (2012). „Derivative-free parameter optimization of functional mock-up units“. In: *Proceedings of the 9th International MODELICA Conference; September 3-5; 2012; Munich; Germany*. 076. Linköping University Electronic Press, S. 819–828.

Glaessgen, Edward H und David Stargel (2012). „The Digital Twin paradigm for future NASA and US Air Force vehicles“. In: *53rd Struct. Dyn. Mater. Conf. Special Session: Digital Twin, Honolulu, HI, US*, S. 1–14.

Gomes, Cláudio, Casper Thule, David Broman, Peter Gorm Larsen und Hans Vangheluwe (2017). „Co-simulation: State of the art“. In: *arXiv preprint arXiv:1702.00686*.

Gough, Brian (2009). *GNU scientific library reference manual*. Network Theory Ltd.

Gundermann, Julia, Artem Kolesnikov, Morgan Cameron und Torsten Blochwitz (2019). „The Fault library-A new Modelica library allows for the systematic simulation of non-nominal system behavior“. In: *Proceedings of the 2nd Japanese Modelica Conference Tokyo, Japan, May 17-18, 2018*. 148. Linköping University Electronic Press, S. 161–168.

Han, Jingqing (2009). „From PID to active disturbance rejection control“. In: *IEEE transactions on Industrial Electronics* 56 (3), S. 900–906.

Hanisch, Andre, Juri Tolujew und Thomas Schulze (2005). „Methoden zur Initialisierung von Online-Simulationsmodellen“. In: *Simulationstechnik, ASIM* 18, S. 388–393.

Härle, Christian, Mike Barth und Alexander Fay (2018). „Einplatinenrechner als Simulationsplattform“. In: *atp magazin* 60 (11-12), S. 56–67.

Höme, Stephan (Apr. 2016). „Analytische Modellierung des Zeitverhaltens von verteilten industriellen Steuerungssystemen“. Diss. Fakultät für Elektrotechnik und Informationstechnik der Otto-von-Guericke-Universität Magdeburg.

Höme, Stephan, Stefan Palis und Christian Diedrich (2014). „Design of communication systems for networked control system running on PROFINET“. In: *Factory Communication Systems (WFCS), 2014 10th IEEE Workshop on*. IEEE, S. 1–8.

Hübner, Christian und Jens Alex (2018). „Digitaler Zwilling im Wassermanagement 4.0“. In: *gwf Wasser|Abwasser* 159 (11), S. 50–57.

Hübner, Christian, Nico Suchold und Jens Alex (2018). „Realisierung überlagerter Regelungsfunktionen im Abwassermanagement auf Grundlage von Simulationsmodellen“. In: *AUTOMATION 2018, Seamless Convergence of Automation & IT*. VDI Verlag GmbH.

IEEE 1516-2010 (2010). *IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA)– Framework and Rules*. New York: Institute of Electrical und Electronics Engineers. ISBN: 978-0-7381-6251-5.

Jung, Tobias, Nasser Jazdi und M. Weyrich (Sep. 2017). „A Survey on Dynamic Simulation of Automation Systems and Components in the Internet of Things“. In: *2017 22nd IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, S. 1–4. ISBN: 978-1-5090-6505-9. DOI: 10.1109/ETFA.2017.8247770. URL: https://doi.org/10.1109/ETFA.2017.8247770.

Jung, Tobias, Payal Shah und Michael Weyrich (Mai 2018). „Dynamic Co-Simulation of Internet-of-Things-Components using a Multi-Agent System“. In: *51st CIRP Conference on Manufacturing Systems, Stockholm*. URL: https://www.ias.uni-stuttgart.de/dokumente/publikationen/2018_Dynamic_Co-Simulation_of_Internet-of-Things-Components_using_a_Multi-Agent-System.pdf.

Kain, Sebastian, Sven Dominka, Martin Merz und Frank Schiller (2009). „Reuse of HiL simulation models in the operation phase of production plants“. In: *Industrial Technology, 2009. ICIT 2009. IEEE International Conference on*. IEEE, S. 1–6.

Kain, Sebastian und Frank Schiller (2010). „Überwachung und Diagnose mit betriebsparallelen Simulationsmethoden“. In: *Integrationsaspekte der Simulation: Technik, Organisation und Personal: Karlsruhe, 7. und 8. Oktober 2010* (131), S. 445.

Kampfmann, Rüdiger, Danny Mösch und Nils Menager (2017). „Parameter Estimation based on FMI“. In: *Proceedings of the 12th International Modelica Conference, Prague, Czech Republic, May 15-17, 2017*. 132. Linköping University Electronic Press, S. 313–319.

Kapp, Ralf (Feb. 2011). „Ein betriebsbegleitendes Fabriksimulationssystem zur durchgängigen Unterstützung der kontinuierlichen Fabrikadaption“. Diss. Universität Stuttgart.

King, Davis E. (2009). „Dlib-ml: A Machine Learning Toolkit“. In: *Journal of Machine Learning Research* 10, S. 1755–1758.

Klein, M., B. Maschler, A. Zeller, B. Ashtari Talkhestani, N. Jazdi und M. Weyrich (2019). „Architektur und Technologiekomponenten eines digitalen Zwillings“. In: *VDI-Berichte Nr. 2351*, S. 89–102.

Klettner, Christian, Thomas Tauchnitz, Ulrich Epple, Lars Nothdurft, Christian Diedrich, Tizian Schröder, Daniel Großmann, Suprateek Banerjee, Michael Krauß, Chris Iatrou u. a. (2017). „Namur open architecture“. In: *atp magazin* 59 (01-02), S. 20–37.

Kritzinger, Werner, Matthias Karner, Georg Traar, Jan Henjes und Wilfried Sihn (2018). „Digital Twin in manufacturing: A categorical literature review and classification“. In: *16th IFAC Symposium on Information Control Problems in Manufacturing INCOM, IFAC-PapersOnLine* 51 (11), S. 1016–1022.

Krumke, S.O. und H. Noltemeier (2012). *Graphentheoretische Konzepte und Algorithmen*. Leitfäden der Informatik. Vieweg+Teubner Verlag. ISBN: 9783834822642. URL: https://books.google.de/books?id=njUpBAAAQBAJ.

Kübler, Ralf und Werner Schiehlen (2000). „Two methods of simulator coupling“. In: *Mathematical and computer modelling of dynamical systems* 6 (2), S. 93–113.

Lämmle, Beat und Mathias Oppelt (2018). „Virtuelle Inbetriebnahme im Takt“. In: *atp magazin* 60 (11-12), S. 90–103.

Lunze, Jan (2008). *Regelungstechnik 2: Mehrgrößensysteme, Digitale Regelung (Springer-Lehrbuch)*. Springer. ISBN: 978-3-540-78462-3.

MacKenzie, David (Juli 2010). *mkfifo(1) - Linux User's Manual*.

Markus Graube und Leon Urbas, Stephan Hensel und (2018). „Informationsmodelle im Lebenszyklus“. In: *atp magazin* 60 (04-05), S. 30–51. ISSN: 2364-3137. DOI: 10.17560/atp.v60i04-05.2345. URL: http://ojs.di-verlag.de/index.php/atp_edition/article/view/2345.

Melo, Vinícius Veloso de und Giovanni Iacca (2014). „A CMA-ES-based 2-stage memetic framework for solving constrained optimization problems“. In: *2014 IEEE Symposium on Foundations of Computational Intelligence (FOCI)*. IEEE, S. 143–150.

Nash, John C, Ravi Varadhan u. a. (2011). „Unifying optimization algorithms to aid software system users: optimx for R“. In: *Journal of Statistical Software* 43 (9), S. 1–14.

Negri, Elisa, Luca Fumagalli und Marco Macchi (2017). „A review of the roles of digital twin in cps-based production systems“. In: *Procedia Manufacturing* 11, S. 939–948.

Nelder, John A und Roger Mead (1965). „A simplex method for function minimization“. In: *The computer journal* 7 (4), S. 308–313.

Nicolae, Maximilian, Stefan Mocanu, Mihai Craciunescu und Radu Dobrescu (2018). „Framework Architecture for Manufacturing Systems Emulation“. In: *2018 22nd In-*

ternational Conference on System Theory, Control and Computing (ICSTCC). IEEE, S. 311–316.

Oppelt, Mathias, Mike Barth und Leon Urbas (2015). „The role of simulation within the life-cycle of a process plant“. In: *Results of a global online survey*.

Otto, Jens, Birgit Vogel-Heuser und Oliver Niggemann (2018). „Online parameter estimation for cyber-physical production systems based on mixed integer nonlinear programming, process mining and black-box optimization techniques“. In: *at — Automatisierungstechnik* 66 (4), S. 331–343.

Pfeiffer, Marcel (2018). „SMART OPTIMIEREN statt investieren“. In: *DIGITAL ENGINEERING Magazin* 8 (8), S. 16–17.

Plattform Industrie 4.0 (2019). *Die Verwaltungsschale im Detail, von der Idee zum implementierbaren Konzept*. Bundesministerium für Wirtschaft und Energie.

Powell, Michael JD (2009). „The BOBYQA algorithm for bound constrained optimization without derivatives“. In: *Cambridge NA Report NA2009/06, University of Cambridge, Cambridge*, S. 26–46.

Puntel Schmidt, Philipp (2017). „Methoden zur simulationsbasierten Absicherung von Steuerungscode fertigungstechnischer Anlagen“. In:

Putman, Nicholas M, Francisco Maturana, Kira Barton und Dawn M Tilbury (2017). „Virtual fusion: a hybrid environment for improved commissioning in manufacturing systems“. In: *International Journal of Production Research* 55 (21), S. 6254–6265.

Radke, Aaron und Zhiqiang Gao (2006). „A survey of state and disturbance observers for practitioners“. In: *2006 American Control Conference*. IEEE.

Riefenstahl, Ulrich (Jan. 2000). *Elektrische Antriebstechnik*. Teubner Verlag. ISBN: 3519064294. URL: https://www.xarg.org/ref/a/3519064294/.

Rodriguez-Guerra, Jorge, Carlos Calleja, Iker Elorza, Ana Maria Macarulla, Aron Pujana und Igor Azurmendi (2019). „A Methodology for Real-Time HiL Validation of Hydraulic-Press Controllers Based on Novel Modeling Techniques“. In: *IEEE Access* 7, S. 110541–110553.

Rosen, Roland, Jens Jäkel, Mike Barth, Oliver Stern, Ronald Schmidt-Vollus, Till Heinzerling, Peter Hoffmann, Christoph Richter, Philipp Puntel Schmidt und Christian Scheifele (2019). „Simulation und Digitaler Zwilling im Engineering und Betrieb automatisierter Anlagen“. In: *VDI-Berichte Nr. 2351*, S. 531–546.

Scheifele, Christian, Alexander Verl und Oliver Riedel (2018). „Echtzeit-Co-Simulation für die Virtuelle Inbetriebnahme“. In: *atp magazin* 60 (11-12), S. 44–55.

Schluse, Michael, Marc Priggemeyer, Linus Atorf und Juergen Rossmann (2018). „Experimentable digital twins—Streamlining simulation-based systems engineering for industry 4.0“. In: *IEEE Transactions on Industrial Informatics* 14 (4), S. 1722–1731.

Bochkanov, Sergey (2020). *ALGLIB*. www.alglib.net.

Tauchnitz, Thomas, Ronny Becker, Christian Diedrich, Tizian Schröder, Daniel Grossmann, Suprateek Banerjee, Leon Urbas und Markus Graube (2019). „NOA — Von Demonstratoren zu Pilotanwendungen“. In: *atp magazin* 61 (1–2), S. 44–55.

Vanfretti, Luigi, Maxime Baudette, Achour Amazouz, Tetiana Bogodorova, Tin Rabuzin, Jan Lavenius und Francisco José Goméz-López (2016). „RaPId: A modular and extensible toolbox for parameter estimation of Modelica and FMI compliant models“. In: *SoftwareX* 5, S. 144–149.

Vathoopan, Milan, Benjamin Brandenbourger, Amil George und Alois Zoitl (2017). „Towards an integrated plant engineering process using a data conversion tool for AutomationML“. In: *2017 IEEE International Conference on Industrial Technology (ICIT)*. IEEE, S. 1205–1210.

VDI/VDE 3693 Blatt 1 (2016). *Virtuelle Inbetriebnahme - Modellarten und Glossar*. VDI/VDE-Gesellschaft Mess- und Automatisierungstechnik.

Vögeli, Desirée, Peter Göhner und Michael Weyrich (2018). „Framework für Agentensysteme zur Parallelisierung von simulationsbasierten Entwicklungsaufgaben“. In: *15. Fachtagung Entwurf komplexer Automatisierungssysteme (EKA)*.

Wu, Shaohua, KB McAuley und TJ Harris (2011). „Selection of simplified models: II. Development of a model selection criterion based on mean squared error“. In: *The Canadian Journal of Chemical Engineering* 89 (2), S. 325–336.

OHNE PROTOTYP GEHT NICHTS IN SERIE.

Unser Podcast ist das Werkzeug, mit dem Sie Ihre Karriere in allen Phasen entwickeln – vom Studium bis zum Chefsessel. Egal, ob Sie Ingenieur*in, Mechatroniker*in oder Wissenschaftler*in sind: Prototyp begleitet Sie. Alle 14 Tage hören Sie die Redaktion von INGENIEUR.de und VDI nachrichten im Gespräch mit prominenten Gästen.

INGENIEUR.de
TECHNIK - KARRIERE - NEWS

PROTOTYP

Karriere-Podcast

JETZT REINHÖREN UND KOSTENFREI ABONNIEREN:
WWW.INGENIEUR.DE/PODCAST

IN KOOPERATION MIT VDI NACHRICHTEN

Die Reihen der Fortschritt-Berichte VDI:

1 Konstruktionstechnik/Maschinenelemente
2 Fertigungstechnik
3 Verfahrenstechnik
4 Bauingenieurwesen
5 Grund- und Werkstoffe/Kunststoffe
6 Energietechnik
7 Strömungstechnik
8 Mess-, Steuerungs- und Regelungstechnik
9 Elektronik/Mikro- und Nanotechnik
10 Informatik/Kommunikation
11 Schwingungstechnik
12 Verkehrstechnik/Fahrzeugtechnik
13 Fördertechnik/Logistik
14 Landtechnik/Lebensmitteltechnik
15 Umwelttechnik
16 Technik und Wirtschaft
17 Biotechnik/Medizintechnik
18 Mechanik/Bruchmechanik
19 Wärmetechnik/Kältetechnik
20 Rechnerunterstützte Verfahren (CAD, CAM, CAE CAQ, CIM ...)
21 Elektrotechnik
22 Mensch-Maschine-Systeme
23 Technische Gebäudeausrüstung

ISBN 978-3-18-527108-3