Fakultät für Maschinenwesen
Lehrstuhl für Angewandte Mechanik

Beitrag zur Co–Simulation in der Gesamtsystementwicklung des Kraftfahrzeugs

Dipl.–Ing. Felix Christian Günther

Vollständiger Abdruck der von der Fakultät für Maschinenwesen der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktor–Ingenieurs (Dr.–Ing.)

genehmigten Dissertation.

Vorsitzender: Univ.–Prof. Dr.–Ing. Markus Lienkamp
Prüfer der Dissertation:
1. Univ.–Prof. Dr.–Ing. habil. Heinz Ulbrich i.R.
2. Prof. Dr.–Ing. Martin Otter, Fakultät EI

Die Dissertation wurde am 25.07.2016 bei der Technischen Universität München eingereicht und durch die Fakultät für Maschinenwesen am 08.02.2017 angenommen.

Das Ganze ist mehr als die Summe seiner Teile.

- *Aristoteles*

Kurzfassung

Zusammenfassung

In der Systementwicklung des Fahrzeugs müssen neben der Mechanik des Antriebsstrangs weitere relevante physikalische Domänen simuliert werden, was zu langen Rechenzeiten führt. Diese können durch Modellpartitionierung und Co-Simulation signifikant reduziert werden. Hierfür werden ein neues numerisches Verfahren mit Makroschrittweitensteuerung für beliebige Programme sowie eine modulare Modellbibliothek entwickelt. Anhand eingeführter Bewertungskriterien für gekoppelte Simulation wird die verbesserte Effizienz der Entwicklung gezeigt.

Abstract

In the automotive system development there is a need to investigate all relevant physical domains in addition to the mechanics of the powertrain. This severely increases simulation duration for the joint system. This duration can be reduced significantly by partitioning and co-simulating the overall model. Therefor a new numeric method with macro step size control for arbitrary tools is developed as well as a modular model library approach. Based on the introduction of validation criteria, the improved efficiency of the work is shown.

Vorwort

Die vorliegende Arbeit entstand aus meiner Tätigkeit im Doktorandenprogramm der Robert Bosch GmbH in Schwieberdingen bei Stuttgart und als Doktorand des Lehrstuhls für Angewandte Mechanik der TU München.

Meinem Doktorvater am Lehrstuhl, Herrn Univ.-Prof. Dr.-Ing. habil. Heinz Ulbrich (i.R.), möchte ich für die wissenschaftliche Betreuung der Arbeit und seine Unterstützung besonders danken. Bei Herrn Prof. Dr.-Ing. Martin Otter vom DLR bedanke ich mich für das Interesse an der Arbeit und die Erstellung des Zweitgutachtens. Für die Übernahme des Prüfungsvorsitzes danke ich Herrn Univ.-Prof. Dr.-Ing. Markus Lienkamp vom Lehrstuhl für Fahrzeugtechnik. Die Zusammenarbeit mit Dr.–Ing Markus Schneider am Lehrstuhl für Angewandte Mechanik möchte ich ebenfalls dankend hervorheben.

Für das Ermöglichen und den Anstoß zur vorliegenden Arbeit in der zentralen Forschung bei Bosch möchte ich mich bei Prof. Dr. sc. techn. Marcus Schulz und Prof Dr.–Ing. Lutz Rauchfuß bedanken. Die weitere Betreuung der Arbeit wurde von Dipl.–Ing. Georg Mallebrein übernommen, wofür ich ihm sehr dankbar bin. Das Arbeiten in der Forschungsabteilung für Antriebssysteme war einwandfrei und eine sehr wertvolle und schöne Erfahrung. Dankend hervorheben möchte ich die Kollegen Dipl.–Ing. Georg Heuer und Dr.–Ing. Martin Johannaber, für die anregende und angenehme Zusammenarbeit. Generell bin ich den vielen Kollegen und nicht zuletzt den Studenten, die mich dort begleitet haben, sehr dankbar.

Weiterhin möchte ich mich bei der Firma TLK-Thermo GmbH in Braunschweig für die freundliche Bereitstellung Ihrer Software in der Anfangsphase meiner Arbeit und aufschlussreiche Diskussionen und Anregungen bedanken.

Meinen Eltern möchte ich hier herzlich dafür danken, dass sie mir meinen bisherigen Werdegang ermöglicht haben und darüber hinaus meinem Vater für die orthographische Korrektur der vorliegenden Schrift. Der allergrößte Dank geht an dieser Stelle jedoch an meine Frau Natalie, die mich auf dem langen Weg zur Promotion immer unterstützt hat, auf einiges verzichten musste und mir trotzdem immer den nötigen Mut zugesprochen hat.

Felix Günther

Inhaltsverzeichnis

1. Einleitung

1.1. Einführung zu Kontext und Problemstellung

Innerhalb der letzten Jahrzehnte gestalten sich die Rahmenbedingungen für die Automobilindustrie zunehmend anspruchsvoller. Zum Einen wird dies durch die Ressourcenknappheit bei Rohstoffen und fossiler Energie sowie der Emissionsgesetzgebung beeinflusst. Zum Anderen stehen dem ein steigendes Verkehrsaufkommen, hohe Ansprüche der Kunden und der Kostendruck durch den weltweiten Wettbewerb gegenüber. Dies betrifft die Entwicklung bei Fahrzeugherstellern wie auch die der Zulieferindustrie.

Ein Schwerpunkt liegt besonders in der Reduktion des Verbrauchs bzw. des CO_2-Ausstoßes. Hierfür besteht eine Vielfalt an teilweise konkurrierenden Technologien. Neben der technischen Umsetzbarkeit ist deren Erfolg von wirtschaftlichen und politischen Einflüssen abhängig, und somit werden diese heutzutage häufig parallel entwickelt. Als alternative Antriebsformen bestehen sowohl ein rein elektrischer Antrieb, gespeist von Batterie oder Brennstoffzelle, als auch vielfältige Hybridtopologien. Weiterhin bedeutend bleiben aber Maßnahmen, die den Wirkungsgrad des konventionellen Antriebs mit Verbrennungsmotor steigern. Dies schließt innermotorische Technologien, wie Aufladung und Downsizing, Maßnahmen des Thermomanagements, wie Warmlauf oder Abwärmenutzung, und Betriebsstrategien für den Antriebsstrang, wie etwa Segelbetrieb mit ein. Zunehmend wichtig wird demzufolge das Zusammenspiel der technischen Teilbereiche und damit die Beherrschung des Gesamtsystems. Dies hat im Hause Bosch, als Anbieter einer breiten fahrzeugtechnischen Komponentenpalette, als Anbieter von Steuergeräten und als Systemlieferant, einen hohen Stellenwert.

Für eine stabile Funktionalität ist ein profundes Systemverständnis in der Produktentwicklung notwendig. Der Einsatz von Computersimulation trägt neben der Auslegung von Komponenten in entscheidendem Maße auch zur Systementwicklung bei. Dabei machen die o.g. Rahmenbedingungen die Systemsimulation aus Kosten- und Zeitgründen von der Vorentwicklung bis hin zur Applikation unverzichtbar. In den Phasen der Vorentwicklung werden Potentialabschätzungen im Hinblick auf CO_2-Ausstoß und Kosten-Nutzen-Verhältnis mittels Gesamtsystemsimulationen durchgeführt. Um das systemimmanente Verhalten zu simulieren, müssen alle relevanten Komponenten einbezogen werden. Beim Fahrzeug lag dafür lange Zeit der Fokus auf dem Antriebsstrang, vom Tank bis zum Rad (*Tank-to-Wheel*). Da zwei Drittel der Kraftstoffenergie über Wärmeverluste abgeführt werden, müssen für eine

ganzheitliche Betrachtung der Energieflüsse auch die thermischen Komponenten, Fluidkreisläufe und Abgasstrang, mit simuliert werden. Erweitert wird das System im Hinblick auf Elektrifizierung um die Komponenten des Bordnetzes und elektrische Antriebe. Für eine Vorwärtssimulation, die Betriebsstrategien und Fahrerverhalten mit einschließt, muss das Gesamtfahrzeugmodell mit Steuergerät und Fahrer ergänzt werden. Damit werden Fahrzyklen wie NEFZ[1] [34] oder WLTC[2] [140] simuliert.

Für die physikalische Modellierung findet der Sprachstandard *Modelica* [93] inzwischen weite Verbreitung in der Automobilindustrie. In der vorliegenden Arbeit findet die darauf basierende, kommerzielle Software *Dymola* [37] Anwendung, die eine gute numerische Behandlung der sich ergebenden hybriden DAE-Systeme (differential-algebraische Gleichungen mit kontinuierlichen und diskreten Zuständen) bereitstellt.

Jedoch führt die monolitische Simulation (geschlossene Lösung des Gesamtsystems) häufig zu hohem Zeitaufwand und kann Robustheitsprobleme verursachen. Verschiedene Faktoren beeinflussen dieses Verhalten: die große Zahl der Zustände im Gesamtfahrzeugsystem; deren stark unterschiedliche Dynamik; algebraische Gleichungen, die besonders bei der Modellierung der Medien in den Fluidkreisläufen notwendig sind; das Auftreten von Ereignissen im physikalischen Teil des Modells, vor Allem aber in der Kontrolllogik und im Fahrermodell. Die Herausforderung besteht darin, belastbare Simulationsergebnisse zu erhalten und gleichzeitig die Rechenzeiten für Fahrzyklen in Grenzen zu halten, die dem Systementwickler ein effizientes Arbeiten ermöglichen.

Zur Reduktion des Berechnungsaufwands nutzt die vorliegende Arbeit die Möglichkeit der modularen Simulation. Hierbei wird das Gesamtsystem in einer parallelen Co-Simulation gekoppelter Teilsysteme mit eigenem, angepassten Zeitintegrator berechnet. Neben dem Zeitvorteil birgt die modulare Herangehensweise die Möglichkeit, durch einheitliche Schnittstellen ein Baukastensystem mit mehreren Entwicklern bzw. verschiedenen Modellierungstools zu etablieren. Die Herausforderung besteht hierin in der Frage der Partitionierung und besonders in der Numerik der Schnittstellen sowie in der Kopplung der Teilsysteme. Haben monolithische Zeitintegrationsverfahren inzwischen einen hohen Entwicklungsstand erreicht, ist dagegen die Numerik der Co-Simulation auch aktuell Gegenstand verbreiteter Forschungsanstrengungen. Der Fokus dieser Arbeit liegt dabei auf Kopplungsalgorithmen im Kontext einer Modellbibliothek zur Zyklensimulation des Gesamtfahrzeugs.

[1] Neuer Europäischer Fahrzyklus
[2] Worldwide harmonized Light duty driving Test Cycle

1.2. Stand der Forschung und Technik mit Literaturüberblick

Die vorliegende Arbeit baut auf dem Kenntnisstand mehrerer Forschungsgebiete auf. Vom ingenieurwissenschaftlichen Bereich sind dies die Mechanik des Antriebsstrangs sowie die Thermo- und Fluiddynamik, jeweils im fahrzeugtechnischen Umfeld. Aus dem naturwissenschaftlichen Bereich werden die Numerik der Modellierung und Simulation behandelt, wobei der Schwerpunkt der Arbeit auf der numerischen Kopplung von Zeitintegrationsverfahren liegt. Der Stand in der Entwicklung verbrauchsarmer Fahrzeugsysteme, sowie der Stand der Simulationstechnik, können an dieser Stelle nur anhand von Beispielen und ohne den Anspruch auf Vollständigkeit dargestellt werden. Die folgende Übersicht gliedert sich sinnvollerweise in die einzelnen Themengebiete: Systemsimulation des Kraftfahrzeugs, numerische Lösungsverfahren, Co-Simulation allgemein und abschließend Co-Simulation des Kraftfahrzeugs.

Systemsimulation des Kraftfahrzeugs

Für die Automobilindustrie gelten teilweise von der Politik, teilweise selbst gesteckte Ziele, den CO_2-Ausstoß der jeweiligen Fahrzeugflotte unter immer geringeren Grenzwerten zu halten. Dafür gibt es bereits eine Vielzahl an verbrauchsreduzierenden Maßnahmen. Direkteinspritzung bei Diesel- und zunehmend auch Benzinmotoren, sowie Maßnahmen des Downsizing, wie Aufladung durch Kompressor oder Turbolader, schöpfen bereits einen Gutteil des Reduktionspotentials beim Verbrennungsmotor aus. Ebenso sind Maßnahmen wie Optimierung der Aerodynamik oder Leichtbau bereits länger Stand der Technik. Deutlich mehr Potential bieten alternative Antriebe, die den Verbrennungsmotor teilweise oder ganz ersetzen. In Verbindung mit CO_2-neutraler Stromerzeugung sind batterie-elektrische Fahrzeuge am günstigsten. Das Problem hierbei sind die teuren Batterien und trotzdem geringen Reichweiten. Beim elektrischen Antriebsstrang mit Brennstoffzelle stehen dem Vorteil größerer Reichweite der schlechtere Wirkungsgrad und fehlende Infrastruktur für Wasserstoff gegenüber. Die Entwicklung von alternativen Antrieben wird jedoch vorangetrieben, wie die Entwicklung spezieller Simulationsbibliotheken zeigt, Ungethüm [141]. Jedoch weist Beidl in [13] reinen Elektrofahrzeugen einen sehr langsam steigenden Marktanteil aus. Als Übergang stehen systemische Maßnahmen wie Hybridisierung, energieoptimierte Betriebsstrategien und verbessertes Thermomanagement im Vordergrund. Dabei ist die Kenntnis der Energieflüsse im Fahrzeug, durch Messung, wie in Rumbolz [112] und Simulation notwendig. Zur Vergleichbarkeit der Ergebnisse bestehen genormte Fahrzyklen, die jedoch im Moment noch regionale Gültigkeit besitzen, wie der NEFZ [34] für Europa. Derzeit wird an einem weltweit gültigen Fahrzyklus WLTC [140] entwickelt, dessen Richtlinien den Realver-

brauch (Zusatzverbraucher, wie Klimaanlage) und alternative Antriebe mit berücksichtigen soll. Somit lassen sich Betriebsstrategien, wie Start-Stopp oder Segeln bei konventionellen und Hybridantrieben vergleichen. Dies wird verstärkt auch bei Nutzfahrzeugen angewandt, wo Ennemoser [42] einen bedarfsgerechten Betrieb der Nebenaggregate vorschlägt. Zunehmend werden Systeme mit vorrausschauender Strategie etwa mit Kenntnis des Höhenprofils, Terwen [135], entwickelt. Ein weiterer Ansatz ist ein verbrauchsoptimiertes Thermomanagement. Jeweils in Gesamtfahrzeugsimulationen bewerten Mustafa [98] die Einsparung durch Motorraumkapselung bei verschiedenen Zyklen und Kaiser [75] optimierte Regelstrategien der Klimaanlage.

Je nach Zielsetzung kommen unterschiedliche Simulationswerkzeuge in der Fahrzeugentwicklung zum Einsatz. Ein Beispiel für die Optimierung eines Teilsystems ist die Schwingungsreduktion mittels MKS[3]-Simulation bei Filippi [46]. Häufig jedoch muss das Systemverhalten des Gesamtfahrzeugs berücksichtigt werden. Einen sehr aufwändigen Ansatz mit realem Fahrer und Fahrzeug in einer Echtzeitumgebung stellt Baumann [12] vor. Dies ist aus Kostengründen zumeist nicht möglich, und es kommen verschiedene Werkzeuge zur rechnergestützten Systementwicklung zum Einsatz. Rueckgauer stellt in [110] eine Umgebung im akademischen Umfeld zur Mechatroniksimulation des Gesamtsystems vor. Es besteht inzwischen jedoch auch eine Vielzahl kommerzieller Werkzeuge. Für den Antriebsstrang gibt Otter [102] eine Übersicht. Bei der Modellierung ist das richtige Maß an Detailtiefe zu anzuwenden, als Kompromiss aus Aussagegenauigkeit und Aufwand bei Parametrierung, Rechenzeit und Validierung. Diesen Kompromiss untersucht Wickord in seiner Dissertation [148] und Janz [73] stellt Anforderungen an eine modulare Modellbibliothek für Gesamtfahrzeuge vor. Eine kommerzielle Bibliothek ist beispielsweise *PowerTrain* [35] für *Modelica*.

Insbesondere, um alle verbrauchsrelevanten Energieflüsse in einer Gesamtfahrzeugsimulation darstellen zu können, müssen Modelle der verschiedenen physikalischen Domänen, wie Mechanik und Elektronik, gekoppelt werden. In einem FVV[4]-Bericht beschreibt Genger [54] die Kopplung eines eindimensionalen Modells des thermischen Netzwerks in *Flowmaster* [47] mit dem Modell restlichen Fahrzeugs in *Matlab/Simulink* [136] zu einem solchen Multi-Physik-Modell. Die Arbeit wird zur Entwicklung des Thermomanagements für Modelle unterschiedlicher Fahrzeuge von Stotz [128] und Stegmann in einem aktuellen FVV-Bericht [127] aufgegriffen. Die komplexen Fahrzeugmodelle werden in einer Rückwärtssimulation berechnet, was bedeutet, dass der Fahrzyklus direkt als Eingangsgröße der translatorischen Bewegung der Fahrzeugmasse eingeht und kein Fahrermodell benötigt wird. Ein auf die Abbildung der thermodynamischen Prozesse im Verbrenner selbst spezialisiertes Werkzeug ist *GT-Power* [133]. Von Heyden [64] nutzt es in einer Kopplung mit *Dymola/Modelica*, in dem die Mechanik des Antriebsstrangs modelliert ist. Ausschließlich mit

[3]Mehrkörpersystem
[4]Forschungsvereinigung Verbrennungskraftmaschinen e.V.

Modelica kommen die Multi-Physik-Bibliotheken zur Auslegung von Kühlkonzepten bei alternativen Antrieben aus, die Simic [125] beschreibt. Dort wird ebenfalls auf die Notwendigkeit der domänenübergreifenden Gesamtsystemsimulation hingewiesen. Eine weitere umfangreiche Fahrzeugbibliothek, die auch Modelle für Batterien und Brennstoffzellen enthält, wird von Hülsebusch [66] vorgestellt. Bei Kossel [84] wird ein Ansatz für eine Gesamtfahrzeugsimulation eines Omnibusses vorgestellt, der einheitlich in *Modelica* modelliert ist und detaillierte Teilmodelle für Fahrgastraum und Klimatisierung beinhaltet. Aufgrund stark unterschiedlicher Zeitkonstanten werden hier jedoch die Teilmodelle durch Kopplung und mehrerer Instanzen von *Dymola* parallel berechnet. In der vorliegenden Arbeit wird u.A. dieser Ansatz aufgegriffen und optimiert. Bei Guenther [58] wird die dafür entwickelte modulare Bibliothek als Baukasten für Modelle der unterschiedlichen Domänen vorgestellt.

Neben der Potentialabschätzung neuer Konzepte finden koppelbare Systemmodelle des Kraftfahrzeugs bei der Funktionsentwicklung und Applikation eine wichtige Anwendung. Bei der HiL[5]-Simulation werden Modelle der Regelstrecke (Gesamtfahrzeug oder Teilsysteme) mit einem real vorhandenen Steuergerät gekoppelt, VDI [143]. Boumans [17] stellt die HiL-Anwendung bei Bosch vor und erläutert Anforderungen an die Modelle, wobei die Echtzeitfähigkeit im Vordergrund stehen muss. Eine von BMW gewählte Vorgehensweise der physikalischen Modellierung mit *Dymola* und anschließender Einbindung via *Simulink* in die Echtzeitumgebung wird in Morawietz [96] gezeigt. Auch bei Daimler findet *Dymola* Anwendung, Brückmann [18]. Die Modelle werden jedoch mit der HiL-Umgebung *Silver* [109] eingebunden. Nägeli [99] erläutert den Einsatz von *Simulink*-Modellen für den HiL-Verband bei Volkswagen und wiest auf eine künftige Nutzung der Standard-Schnittstelle FMI[6] [94], die weiter unten in einem eigenen Abschnitt behandelt wird, hin. In Abel [1] werden Modelle eines Automatikgetriebes für den HiL-Einsatz in *Simulink* und dem *Modelica*-basierten Werkzeug *SimulationX* [70] verglichen. Schwingungseffekte beim Schaltvorgang werden dabei nur vom *Modelica*-Modell abgebildet, was die Notwendigkeit physikalischer Modellierung belegt. Howe [68] schlägt eine getrennte Berechnung des Motormodells und des restlichen Antriebsstranges mit asynchroner Kopplung zweier Prozessoren vor. Dabei wird das Motormodell mit einer Kurbelwellenwinkel-basierten Schrittweite integriert, die restliche Simulation mit festen Zeitschritten. Den Ansatz, mehrere Modellteile zu koppeln, greift auch Faure in [45] auf, der verschiedene Methoden vorstellt, komplexe Motormodelle Echtzeitfähig zu machen und dafür die Parallelisierbarkeit von HiL-Modellen untersucht. In seiner Umsetzung werden die einzelnen Zylinder parallel berechnet.

[5] Hardware in the Loop
[6] Functional Mockup Interface

Numerische Lösungsverfahren

Der zunehmenden Komplexität mechatronischer Systeme -wie dem Kraftfahrzeug- steht eine ständige Leistungszunahme der Computersysteme gegenüber, was auch die Entwicklung numerischer Simulationsmethoden vorangetrieben hat. Damit ist der Notwendigkeit nach simulationsgestützter Entwicklung, VDI [143], auch die Umsetzungsmöglichkeit gegeben. Den Überblick einiger Forschungsaktivitäten zu mechatronischen Systemen und deren Simulation gibt Ulbrich in [139].

Über die notwendigen numerischen Lösungsverfahren besteht eine Vielzahl an Werken. Beispielhaft seien hier das allgemeine Lehrbuch von Schwarz [121] und das viel zitierte Standardwerk Hairer [61] genannt, in dem ausführlich die unterschiedlichen Integrationsverfahren für Anfangswertprobleme von ODE[7] behandelt werden. Vorgestellt werden ebenfalls auf Fehlerschätzer (nach Richardson oder eingebetteten Verfahren) basierende variable Schrittweitenregelungen. Einen einfacheren Gegenentwurf für variable Schrittweiten von ODE, der auf der Änderungsrate der Lösung basiert, stellt Shampine [123] vor. Über die Anwendung der Modellierung und Simulation in der Ingenieurswissenschaft hat Gipser das Lehrbuch [55] verfasst. Als umfassendes Werk empfiehlt sich das Buch von Cellier [29]. Dort werden auch die Methoden für Echtzeitsimulation sowie die für diskrete Systeme und DAE[8]-Simulation behandelt.

Da die Modellbildung von Multi-Physik-Systemen durch algebraische Zwangsbedingungen oder die Formulierung von Stoffeigenschaften häufig DAE-Systeme erzeugt, bestehen auch zu dieser Problemstellung eine große Zahl an Lösungsalgorithmen. Oft wird versucht, das DAE-System durch Differentiation einiger Gleichungen in ein ODE-System zu überführen oder zumindest dessen differentiellen Index zu reduzieren. Dies wird bei Gear [52] angewandt und eine Prädiktor-Korrektor-Methode zur Lösung herangezogen. Standardwerke, die eine sehr gute Zusammenfassung der DAE-Methoden geben sind Brenan [21] und Hairer [62]. Brenan stellt schwerpunktmäßig den nordamerikanischen Erkenntnisstand und den sehr populären *DASSL*-Algorithmus vor, wohingegen Hairer europäisch geprägt ist und den ebenfalls weit verbreiteten *RADAU*-Algorithmus ausführlicher behandelt. In Verbindung mit symbolischer Vorverarbeitung von DAE-Systemen führt Elmqvist [38] die effiziente Technik *Inline Integration* ein, die Diskretisierungsformeln eines Lösers konstanter Schrittweite symbolisch in die Modellgleichungen integriert.

Sind sowohl Variablen mit kontinuierlichem Verlauf, als auch solche mit Unstetigkeiten (auch in der ersten Zeitableitung) im ODE- oder DAE-System des Modells vorhanden bzw. gibt es einen Modellteil mit diskreten Variablen, so spricht man von hybriden Systemen. Mosterman zählt in [97] die daraus resultierenden Phänomene wie Ereignistypen, Strukturumschaltung und Chattering auf und gibt eine Übersicht über deren Behandlung durch vorhan-

[7] Ordinary Differential Equations
[8] Differential Algebraic Equations

dene Simulationssoftware. Das Buch Abel [2] bespricht die Umsetzung hybrider Simulation beim Reglerentwurf mit physikalisch modellierter Strecke und Zustandsautomat. Einen aktuellen Übersichtsbeitrag zum Stand hybrider Simulation liefert der Artikel von Schwarz [122]. Er zeigt die Unterschiede der aus der Elektrotechnik stammenden Modellierungssprache *VHDL-AMS*[9], in der in Digital-und Analogteil getrennt wird, und dem Ansatz der *Modelica*-Löser. Deren kommerzielle Varianten, wie *Dymola* oder *SimulationX* verfügen über Löser mit Ereignisbehandlung für hybride Modelle. Eine effizientere, asynchrone Vorgehensweise für *Modelica*-Löser wird von Nikoukhah [100] vorgeschlagen.

Co-Simulation

Häufig müssen zur Simulation des Gesamtsystems Teilmodelle verschiedener physikalischer Domänen zusammengefügt werden. Diese sind oft mit unterschiedlichen, spezialisierten Modellierungswerkzeugen erstellt. Hessel [63] erläutert am Beispiel *VHDL-AMS* und *Modelica* die Möglichkeit der Modelltransformation - welche in beide Richtungen denkbar ist - mit anschließender monolithischer Simulation mit einem Löser. Da dies jedoch sehr aufwändig ist, schlägt er als Alternative die Kopplung der Modelle in einer Co-Simulation vor. Auch Tegethoff [134] vergleicht beide Alternativen, wobei er *Modelica* bereits als domänenübergreifenden Sprachstandard ausweist. Er schlägt jedoch die Co-Simulation vor, um stark unterschiedliche Zeitkonstanten (bei steifen Systemen) in den Teilmodellen getrennt integrieren zu können (*Multirate*-Ansatz).

Der Begriff „Co-Simulation“ findet in der Literatur mehrdeutige Verwendung. In Anlehnung an Dronka [36] schlägt Geimer in [53] den Weg für eine Begriffsvereinheitlichung vor. Als grundsätzliche Unterteilung sollen zwei Kriterien gelten: die Verwendung eines oder mehrerer Modellierungswerkzeuge sowie die Verwendung eines oder mehrerer Integratoren. In der vorliegenden Arbeit sei als „Co-Simulation“ die Verwendung zwei oder mehrerer Integrationsalgorithmen (*Mutimethod*) oder -parameter (*Multirate/-order*) definiert. Abbildung 1.1 zeigt die grundsätzliche Architektur einer Co-Simulation. Dabei können die Teilmodelle mit dem gleichen Modellierungswerkzeug erstellt sein. Sie zeichnen sich jedoch durch verteilte/modulare Simulation mit lokalen Integratoren aus. Gekoppelt werden sie über diskrete Ein- und Ausgangsgrößen ($\bar{\boldsymbol{u}}$ und $\bar{\boldsymbol{y}}$). Dies erfolgt mithilfe numerischer Algorithmen für die Co-Simulation, wobei die beteiligten Prozesse in eine Umgebung eingebunden sind. Dafür bestehen mehrere Realisierungsvarianten: bei einer Simulatorkopplung werden die Teilmodelle in eigenständigen Werkzeugen berechnet. Die Umgebung und die Co-Simulationsalgorithmen können in eines der Werkzeuge mit den Modellen eingebettet sein oder als eigene Software, wie *TISC* [137] bereitgestellt werden; bei einer reinen Multirate-Umsetzung vereint ein Werkzeug die Co-Simulationsumgebung, die -algorithmen sowie die Teilmodelle. Diese wer-

[9] Very (High Speed Integrated Circuit) Hardware Description Language - Analog Mixed Signal (Extension)

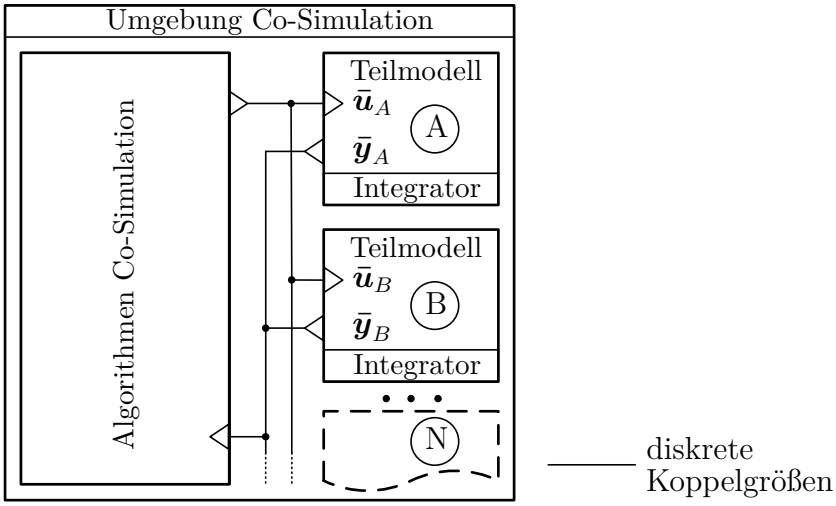


Abbildung 1.1.: Allgemeines Schema der Co-Simulation

den jedoch intern mit unterschiedlichen Schrittweiten simuliert. Im akademischen Umfeld werden seit über 50 Jahren solche Ansätze behandelt.

Eine umfangreiche Literaturübersicht zur Parallelisierung von ODE-Integratoren bis zum Anfang der 1990er Jahre bietet Jackson [72]. Besonders Anfangswertprobleme mit gekoppelten *Runge-Kutta*-Verfahren sind verbreitet. Wells vergleicht in [147] die sequentiellen Multirate-Varianten *Fastest-First* und *Slowest-First*, mit Interpolation bzw. Extrapolation der jeweiligen Ausgangsgrößen der vorab nach ihrer Dynamik getrennten Modellteile. Eine aktuellere Übersicht gibt Gómez [56], der in seiner Arbeit die Multirate-Stabilität bei unterschiedliche starker Kopplung zweier Testmodelle analysiert. Eine Stabilitätsanalyse beim Fastest-First-Ansatz mit Approximationen der Koppelgrößen unterschiedlicher Ordnung liefert Howe [69]. In den beiden Arbeiten Engstler [40] und [41] wird eine parallele Multi-*order*-Methode vorgestellt, die langsame und schnelle Modellteile mit unterschiedlicher Ordnung integriert, wobei die Partitionierung variabel und zur Laufzeit, je nach Dynamik der Zustände erfolgt. Allen Verfahren ist eine Effizienzsteigerung der Simulation gemein.

Für die Partitionierung steifer Systeme bestehen unterschiedliche Ansätze. Eine automatische Partitionierung für lineare ODE-Systeme zeigt Enright [43]. Kanth entwickelt in seiner Arbeit [76] einen Ansatz für Partitionierung anhand der Granularität. Diese wird durch Kopplungsstärke und Steifigkeit bestimmt. Zur Partitionierung von *Modelica*-Modellen bestehen einige wenige Arbeiten, die Nyström [101] zusammenfasst. Beispiele sind die Anwendung von *Transmission Line Modelling*, Johansson [74], wobei komponentenweise integriert wird oder die Anwendung der oben genannten *Inline Integration* [38], mithilfe derer Schiela [113] schnelle Zustände mit implizitem und langsame mit explizitem Verfahren löst. Elmqvist [39] schlägt ein Sprachkonstrukt in *Modelica* zur Partitionierung in *Subtasks* mit lokalen Integratoren vor, das jedoch in den Simulationsumgebungen nicht umgesetzt ist.

Gerade für die prinzipiell als steife Systeme interpretierbaren DAE-Systeme sind Multirate-

Methoden Gegenstand der Forschung. Eine Übersicht mit Einteilung solcher Methoden findet sich in Maten [88]. So wurde ein Stabilitätskriterium, die *Kontraktivitätsbedingung* für gekoppelte DAE eingeführt, auf deren Basis eine sequentiell-iterative Methode zur Stabilisierung durch Vorkonditionierung, Arnold [6, 7], entwickelt wurde. Dies findet Anwendung bei gekoppelten MKS. Häufig kommen Multirate-Ansätze für DAE-Systeme bei elektronischen Schaltungsnetzwerken zum Einsatz, wie bei den Slowest-First-Methoden mit variablen *Compound*-Schrittweiten des Gesamtnetzwerkes bei Striebel [129] oder Verhoeven [145].

Umfangreiche theoretische Untersuchungen zur Co-Simulation sind durch Kübler [78, 79, 80, 81] entstanden. Seine Arbeiten zu verschiedenen Koppelstrategien wie Filterung, Iteration und Vorschläge zur Steuerung der *Makroschrittweite* (Abstand zwischen zwei Kopplungszeitpunkten) der Co-Simulation finden häufig Beachtung. Sie kommen u.A. in der Nachfolgearbeit von Scholz [119, 120] zur Anwendung, wobei die Makroschrittweitensteuerung nicht ohne den Eingriff in die beteiligten Simulationswerkzeuge erfolgen kann. Vielfältige Veröffentlichungen auf dem Gebiet der Numerik zur Co-Simulation gehen auf Arnold[5, 6, 7, 8, 9, 10] bzw.Schierz [114, 115] zurück. Dabei bieten die Arbeiten [5, 10] von Arnold eine gute Übersicht. Schierz entwickelt einen Master-Slave-Ansatz zur Erfüllung der o.g. Kontraktivitätsbedingung. Untersuchungen zur Stabilität der Kopplung werden auch von Busch durchgeführt, [24, 25]. Diese finden anhand eines gekoppelten Zweimassenschwingers für verschiedene Makroschritte und Kopplungsparameter statt. Sehr ausführlich betrachtet auch Friedrich [49, 51] die Kopplung anhand eines Zweimassenschwingers. Dabei werden bei einer parallelen Master-Slave-Umsetzung verschiedene Approximationsmethoden des Koppelterms im Master zur Stabilisierung entwickelt. Ebenso werden in Friedrich [50] Koppelterme für andere Domänen vorgestellt und die Berechnung mit mehreren Rechnerkernen untersucht. Weitere Beispiele sind etwa Knorr [82] mit einer stetigen Approximationsmethode und dem Vorschlag für qualitative Indikatoren als Gütekriterium oder Petridis [105] mit einer asynchronen Kopplungsstrategie, wobei die Wahl von Makroschrittweiten entfällt. Speziell für die industrielle Anwendung, zur Wahrung von Geheimhaltung, sind Black-Box-Modelle interessant, für deren stabile Kopplung in Gu [60] eine Methode aus der Regelungstechnik vorgestellt wird. Eine Methode zur Steuerung der Makroschrittweite zur Minimierung der Iterationen für sequentiell-iterative Verfahren wurde von Benedikt [14] entwickelt.

Für an einer Simulatorkopplung beteiligte Werkzeuge stellt Clauss [31] Anforderungen auf und beschriebt ein Beispiel für eine eingebettete Umsetzung, wobei die Co-Simulation von einem der Werkzeuge aus über TCP/IP[10] gesteuert wird. So sind auch 3D-FEM[11]-Modelle in einer Simulatorkopplung mit einem 1D-Schaltungsnetzmodell möglich, Wünsche [149]. Hommel nutzt in [67] für die Kopplung physikalischer Simulation in *Dymola* mit den

[10] Transmission Control Protocol / Internet Protocol
[11] Finite-Elemente-Methode

signalflussorientierten Modellen in *Simulink* mit *Exite* [44] eine spezielle Software zur Simulatorkopplung.

Neben *Exite* repräsentieren weitere auch als „Middleware“ bezeichnete, eigenständige Softwarepakete zur Co-Simulation, wie *CosiMate* [30] oder *Silver* [109] den Stand der Technik. Letzteres dient dem Einsatz als HiL-Umgebung im Automobilbereich. Für Gesamtfahrzeugsimulationen mit Hilfe von Werkzeugen verschiedener Domänen bietet *ICOS* [11] ein iteratives Koppelverfahren. Die vorhandenen Programme dienen vorrangig dem Zweck, Teilsystemmodelle aus verschiedenen Werkzeugen weiter verwenden zu können, und sind nicht speziell für Multirate-Anwendungen ausgelegt. Eine parallele Kopplung ermöglicht *TISC* [137], Kossel [85], das Algorithmen für Extrapolation und Glättung der Koppelgrößen mitbringt. Das von Friedrich [51] beschriebene MDPCosim setzt eine parallele Kopplung im Master-Slave-Verbund um und beinhaltet Approximationsmethoden zur Kopplung verschiedener Domänen.

Die vorgenannten Pakete haben jeweils unterschiedliche Kommunikationsstandards und eine eigene Schnittstellenarchitektur. Auch haben die Simulationswerkzeuge jeweils unterschiedliche externe Schnittstellen. Im internationalen Forschungsprojekt Modelisar, Otter [103], wurde ein Standard zur Vereinheitlichung der Koppelschnittstellen zwischen Modellen entwickelt. Neben dem einfacheren Modellaustausch ist er vom Entwicklungsbedarf bei Co-Simulation motiviert, Arnold [8]. Der zunächst getrennt entwickelte Standard zum Modellaustausch Modelisar [95] sowie die Spezifikation für Modelle inklusive Integrator zur Co-Simulation Modelisar [94], Arnold [10] sind inzwischen im Standard FMI 2.0 in FMI [48] zusammengefasst. Er umfasst weder Definitionen zur Modellierung, noch beinhaltet er die Algorithmen zur Co-Simulation. Beispiele für die Umsetzung von FMI für Modellaustausch mit *Dymola* sind Sun [130] oder Brembeck [20]. Auch weitere Werkzeuge besitzen eine FMI-Schnittstelle für Modellaustausch, wobei FMI für Co-Simulation derzeit häufig noch nicht anwendbar ist. Die Umsetzung von Co-Simulationsalgorithmen für FMI sind ebenso Gegenstand der Entwicklung. Eine Möglichkeit ist der von Clauss [33] vorgestellte FMI-Master, der einige Kopplungsmethoden an ersten Testmodellen zeigt.

Co-Simulation des Kraftfahrzeugs

Abschließend soll der Stand der Forschung und Technik themenübergreifend zur Co-Simulation in der Fahrzeugentwicklung, neben der o.g. HiL-Anwendung, aufgezeigt werden. Die in Kübler [79] gezeigten Methoden wendet Rückgauer [110] für die Querdynamik an. In der Mechatroniksimulation eines automatischen Lenksystems nimmt er eine Partitionierung nach steifen und nicht-steifen Teilmodellen vor, wobei er als Fehlerkriterium der Co-Simulation die Fläche zwischen monolithischer und partitionierter Simulation verwendet. Zur Berechnung der Längsdynamik lässt sich beispielsweise ein dynamisches Motormodell mit kurbelwellen-

winkelsynchroner Schrittweite mit dem nichtlinearen Mehrmassenmodell des Chassis mit festen Zeitschritten asynchron koppeln, Howe [68].

Um die Wechselwirkungen mit der Umgebung zu untersuchen, koppelt Arnold [9] die MKS-Simulation zweier sich begegnender Schienenfahrzeuge mit einer CFD[12]-Aerodynamiksimulation. Für die echtzeitfähige Berechnung des Fahrwerks von Schienenfahrzeugen nutzt Dronka [36] eine Co-Simulation von MKS mit 1D-Hydraulikmodellen. In einem ähnlichen Ansatz für die Simulation des automobilen Fahrwerks etabliert Busch [22] die Kopplung des MKS-Werkzeuges *SIMPACK* [126] und *Dymola* für die Hydraulik einer Servolenkung, einschließlich unterschiedlicher Extrapolationsmethoden. Hommel [67] verwendet *Dymola* dagegen für das Modell eines Hybridantriebsstranges, das mit der in *Simulink* umgesetzten Steuerung co-simuliert wird.

Puntigam [108] behandelt die Kopplung von Motor, Antriebsstrang und Kühlkreislauf, um eine Gesamtfahrzeugsimulation zu erhalten, wobei explizite wie implizite Kopplungsalgorithmen Verwendung finden. Auch, um den Multirate-Vorteil ausnutzen zu können, setzt Kossel [84] *TISC* für die Co-Simulation aus einzelnen *Dymola*-Modellen zur Bewertung von Kühlkonzepten ein. Im Vergleich zur monolithischen Gesamtfahrzeugsimulation verkürzt sich die Rechendauer dadurch um ein Vielfaches. Das in der vorliegenden Arbeit weiterentwickelte MDPCosim verwendet Schneider [118] für die parallele Co-Simulation von fünf Teilmodellen. Auf diese Weise lässt sich das Systemverhalten eines variablen Ventilsteuertriebs mithilfe dreier spezialisierter Simulationswerkzeuge untersuchen.

1.3. Ziel und Gliederung der Arbeit

Der Ausgangspunkt der vorliegenden Arbeit ist die Notwendigkeit, die Energieflüsse im Fahrzeug zu verstehen. Das Ziel ist die gesamtheitliche Modellierung und Simulation des Fahrzeugs unter Verwendung von *Modelica*. Dabei sollen einerseits alle relevanten physikalischen Domänen integriert, andererseits eine robuste Simulation von Fahrzyklen mit handhabbarem Rechenaufwand durchführbar sein. Um die letztgenannte Anforderung im Zusammenhang mit dem sich ergebenden, umfangreichen, hybriden (s.o.) DAE-System zu erfüllen, wird die Umsetzung in einer Co-Simulations-Umgebung gewählt. Somit wird eine modulare Multirate-Simulation ermöglicht. Dafür soll eine numerische Methodik für die Co-Simulation in der Systemmodellierung mit Fahrzyklen entwickelt werden.

Zur Erreichung des Ziels sind demnach zwei Entwicklungspfade zu beschreiten und zusammenzuführen: Zum Einen ist dies die Modellbildung und Parametrierung des Gesamtfahrzeugs durch teilweise vorhandene Komponentenmodelle, zum Anderen die Entwicklung des Co-Simulationsverfahrens. Die Gesamtfahrzeugmodelle sollen in einer modularen Bibliothek

[12] Computational Fluid Dynamics

entstehen. Dazu ist die einheitliche Schnittstellenmodellierung für die mechanische und andere Domänen sowie für die Anbindung an die Co-Simulation notwendig. Die Rechenzeiten sollen durch den Einsatz der Co-Simulation signifikant reduziert werden, wobei der dadurch entstehende numerische Fehler begrenzt werden muss. Zu diesem Zweck werden Methoden der Approximation und eine neuartige Methode zur Steuerung der Makroschrittweiten entwickelt. Abschließend werden die Teil- und Gesamtmodelle aus der modularen Bibliothek mit den entwickelten Verfahren simuliert und bewertet.

Die vorliegende Dissertationsschrift legt zunächst in Kapitel 2 die Grundlagen zur Modellierung und Simulation von Multi-Physik-Systemen dar. Im ersten Abschnitt zur Modellierung mit hybriden DAE-Systemen wird ein Schwerpunkt auf physikalische, objektorientierte Modellbildung gesetzt. Der zweite Abschnitt behandelt die Simulation im Zeitbereich mit zugehörigen Grundlagen der numerischen Integrationsmethoden inklusive der Zeitschrittweitensteuerung. Ein weiterer Abschnitt behandelt die Gesamtsystemsimulation von Kraftfahrzeugen und den Zusammenbau von Multi-Physik Modellen. Er wird durch Betrachtungen zur Modellpartitionierung und zur verteilten Simulation als Motivation für Co-Simulation ergänzt.

Kapitel 3 gibt anschließend eine Einführung in den theoretischen Hintergrund der Co-Simulation mit einleitender Terminologie. In dieser Arbeit wurde ein Vorschlag zur Kategorisierung der numerischen Aspekte bei der Co-Simulation entwickelt, anhand derer die einzelnen Grundlagen abschnittsweise erläutert werden. Es folgt die Einführung umfassender Bewertungskriterien der Co-Simulation hinsichtlich Genauigkeit, Aufwand und Effizienz, die im Weiteren verwendet werden.

Auf Basis der Erkenntnisse aus Kapitel 2 und Kapitel 3 im Besonderen wurde das erweiterte Verfahren zur Co-Simulation entwickelt, das in Kapitel 4 vorgestellt wird. Zunächst werden die Randbedingungen für das Verfahren im Kontext der Systemsimulation und der angewendeten Modellierung erläutert. Als Einstieg wird die Schnittstelle beschrieben, die im Rahmen dieser Arbeit verwendet und weiterentwickelt wurde. Danach wird wieder abschnittsweise anhand der Kategorien aus Kapitel 3 die Umsetzung erarbeitet und anhand mechanischer Testmodelle demonstriert. Ausführlich wird auf die Entwicklung und Umsetzung zur Steuerung der Co-Simulations-Schrittweite eingegangen.

Mit Kapitel 5 folgt die Anwendung des Verfahrens in der Gesamtfahrzeugsimulation. Der Aufbau der dafür eingesetzten Fahrzeugmodelle, sowie die Entwicklung einer modularen Komponentenbibliothek wird beschrieben. Dabei werden die mechanischen Modellteile, wie auch die anderer Domänen, sowie deren Parametrierung beschrieben. Schließlich wird die Co-Simulation in dieser Anwendung nach den besprochenen Kriterien bewertet.

Abschließend fasst Kapitel 6 die wichtigsten Erkenntnisse der Arbeit zusammen und zeigt Möglichkeiten der Erweiterung auf.

2. Modellierung und Simulation von Multi-Physik-Systemen

Dieses Kapitel gibt eine Einführung in die Grundlagen der Modellierung und Simulation von Multi-Physik-Systemen. Dazu werden in Abschnitt 2.1 Prinzipien der Modellierung und die mathematische Beschreibungsform gezeigt. Abschnitt 2.2 stellt die objektorientierte Modellierung mit Modelica vor, die in dieser Arbeit zur Anwendung kommt. Die numerischen Verfahren zur Simulation im Zeitbereich werden in Abschnitt 2.3 gezeigt. Schließlich behandelt Abschnitt 2.4 die Modellierung und Simulation von Gesamtfahrzeugsystemen, und in Abschnitt 2.5 zur verteilten Simulation wird die Motivation zur Co-Simulation hergeleitet.

Die Notwendigkeit zur Simulation im Entwicklungsprozess mechatronischer Produkte wird u. A. in [143] dargestellt und anhand des V-Modells in Abbildung 2.1 verdeutlicht.

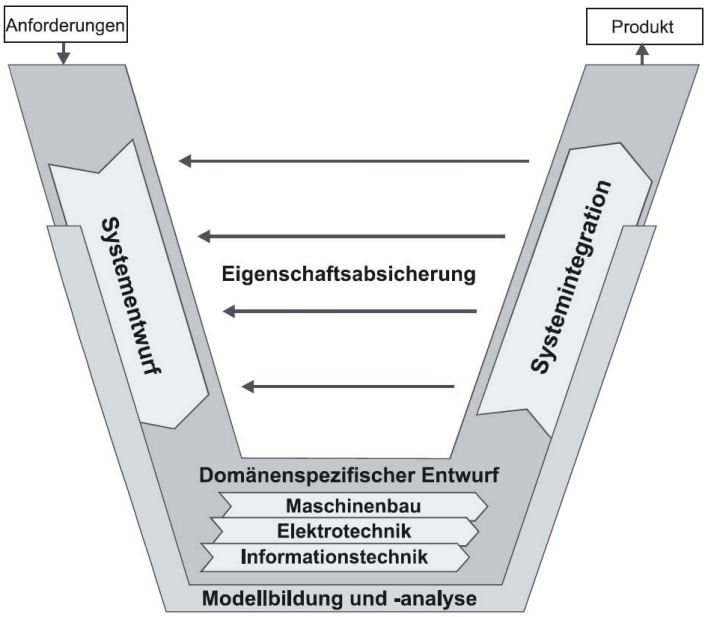


Abbildung 2.1.: V-Modell in der Entwicklung mechatronischer Systeme [143]

Demzufolge begleitet die „Modellbildung und -analyse“ den gesamten, iterativen Produktentstehungsprozess. Dies ist aus Kostengründen durch Einsparung bei Prüfständen und Versuchsträgern sowie aus Zeitgründen bei Variantenvergleich und Optimierung wichtig. Dabei liegt besonders im frühen Stadium ein Schwerpunkt auf der Systemsimulation.

Tabelle 2.1.: Beispiele für Werkzeuge zur Modellierung und Simulation

Modellierungsebene	**Werkzeuge**	**physikalische Domäne**
geometriebasiert	*Ansys* [3] (FEM)	Strukturmechanik
	Fluent [4] (CFD)	Fluiddynamik
	Simpack [126] (MKS)	Mehrkörpermechanik
Netzwerk / physikalisch	*Saber* [131]	Elektrotechnik
	Flowmaster [47]	Fluiddynamik
	Dymola [37] (Modelica)	Multi-Physik
	SimulationX [70] (Modelica)	Multi-Physik
	GT-Power [133]	Multi-Physik
Signalfluss kontinuierlich	*Simulink* [136]	
Signalfluss diskret	*Verilog-XL* [26] (SystemC)	

Es muss zwischen *Modellierung* und *Simulation* unterschieden werden. [29] definiert *Modellierung* als Extraktion des Wissens über ein physikalisches System und dessen eindeutige Darstellung in einem Modell. Dementsprechend ist die *Simulation* das Ausführen von Experimenten an diesem Modell. Die VDI-Richtlinie [142] definiert *Modellierung* noch allgemeiner als „[...] vereinfachte Nachbildung eines geplanten oder real existierenden Originalsystems [...] in einem anderen begrifflichen [...] System“; die *Simulation* als „[...] Vorbereiten, Durchführen und Auswerten gezielter Experimente mit einem Simulationsmodell [...]“. So kann auch die vorliegende Arbeit als zwei wesentliche Teile betrachtet werden: die Modellierung mit der Erstellung der Gesamtfahrzeug- und Schnittstellenmodelle sowie die Simulation mit der Entwicklung numerischer Methoden zur Co-Simulation.

Zumeist wird die Modellierung und die Simulation mit der selben Softwareumgebung durchgeführt. Es besteht eine Vielzahl an Umsetzungen im akademischen Umfeld und von kommerziellen Anbietern. U.A. aus Gründen des Pflegeaufwands werden für den industriellen Einsatz häufig kommerzielle Simulationswerkzeuge bevorzugt. Oft haben die Werkzeuge einen Schwerpunkt in einer physikalischen Domäne. In Tabelle 2.1 sind stellvertretend einige kommerzielle Beispiele aufgezählt und nach der Ebene der Modellierung eingeteilt. Letztere wird im folgenden Abschnitt und in Abb. 2.2 erläutert.

2.1. Modellierung

Ziel der Modellierung ist das Abbilden der Funktion und Physik eines realen Systems in einem mathematischen Ersatzmodell. Zu dessen Bildung sind verschiedene Abstraktionsebenen wählbar, die sich anhand von Abb. 2.2 einteilen lassen. Dabei nimmt mit abnehmendem Grad an Abstraktion vom realen System der Grad der Detaillierung und damit der des Aufwands zu. Sieht man von der Molekularsimulation ab, so wird der höchste Grad an Detaillierung bei

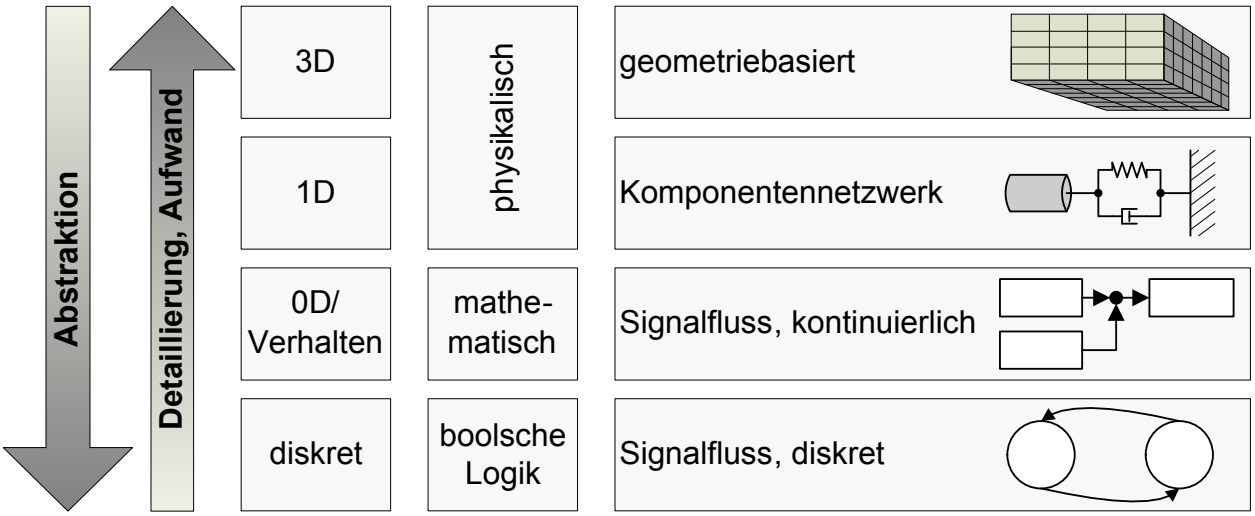


Abbildung 2.2.: Ebenen der Modellierung, ähnlich zu [76]

geometriebasierter, dreidimensionaler Modellierung erreicht. Dies findet bei FEM und CFD Anwendung. Abstrahiert in Systemkomponenten findet die MKS-Modellierung mit allen Freiheitsgraden auch im 3D-Bereich statt. Für größere, multidisziplinäre Systeme lassen sich die physikalischen Gleichungen auf den 1D-Bereich reduzieren, ein Netzwerk aus Komponenten und akausalen Bindungen erstellen und in der Weise z.B. ein vollständiges mechatronisches System erfassen. In den Blockschaltbildern signalflussbasierter Modellierung wird von den physikalischen Gleichungen abstrahiert und das Übertragungsverhalten (2.1) mit gerichteten (kausalen), kontinuierlichen Ein- und Ausgangssignalen ($u(t)$ und $y(t)$) dargestellt.

$$y(t) = f(u(t)) \tag{2.1}$$

Der Ursprung dieses Ansatzes liegt in der Darstellung von Regelstrecken und ist darüber hinaus noch sehr weit verbreitet. Den höchsten Grad an Abstraktion erreicht man in der Darstellung diskreter Algorithmen, etwa mittels boolscher Logik und Zustandsautomaten. Beispiele für die genannten Modellierungsebenen sind in Tabelle 2.1 zu finden.

Je nach Zielsetzung, Anforderungen und Ressourcen muss der entsprechende Grad an Detaillierung gemäß Abb. 2.2 oder eine Kombination verwendet werden. Dazu ist zum Einen das Verhältnis aus Zeitaufwand und Aussagegüte der Simulation wichtig, vgl. hierzu die Arbeiten [90, 91]. Zum Anderen ist die Verfügbarkeit und Genauigkeit der Daten wichtig, die zur Parametrierung der Modelle notwendig sind. Der Einfluss der Toleranzen in den Parameterdaten auf die Simulationsgenauigkeit wird in [148] unter probabilistischen Gesichtspunkten untersucht. Mit dem Ziel, das dynamische, funktionale Verhalten eines technischen Systems darzustellen, ist für die Systemsimulation -auch in dieser Arbeit- die Wahl physikalischer Modellierung im 1D-Bereich am besten geeignet.

Zustandsform ODE

Im Gegensatz zu partiellen Differentialgleichungen (PDE[1]) für FEM- und CDF-Modelle, mit Berücksichtigung von Zeit *und* Ort, wird hier nur Simulation im Zeitbereich behandelt. Das Modellieren mit Blockschaltbildern oder direkt mit physikalischen Gesetzen führt auf Modelle in Zustandsraumdarstellung. So lässt sich z.B. für lineare mechanische Systeme die Formulierung zweiter Ordnung aufstellen:

$$\boldsymbol{M}\ddot{\boldsymbol{x}} + \boldsymbol{D}\dot{\boldsymbol{x}} + \boldsymbol{C}\boldsymbol{x} = \boldsymbol{F} \tag{2.2}$$

mit Massenmatrix $\boldsymbol{M}$, Dämpfungsmatrix $\boldsymbol{D}$, Steifigkeitsmatrix $\boldsymbol{C}$ und aufgeprägten Lasten $\boldsymbol{F}$. Vereinfacht zur eindimensionalen, linearen Schwungmasse ergibt sich

$$m\ddot{x} + d\dot{x} + cx = F. \tag{2.3}$$

Die Umwandlung dieser Differentialgleichung zweiter Ordnung in ein System erster Ordnung erfolgt mit Hilfe des Zustandsvektors $\boldsymbol{z}$. Für den translatorischen bzw. rotatorischen Fall besitzt er je zwei Zustände (Position und Geschwindigkeit):

$$\boldsymbol{z} = \begin{pmatrix} x \\ v \end{pmatrix} = \begin{pmatrix} x \\ \dot{x} \end{pmatrix} \qquad \boldsymbol{z} = \begin{pmatrix} \varphi \\ \omega \end{pmatrix} = \begin{pmatrix} \varphi \\ \dot{\varphi} \end{pmatrix} \tag{2.4}$$

Allgemeiner lässt sich so ein lineares, zeitvariantes System (Systemmatrix $\boldsymbol{A}(t)$) mit Eingängen $\boldsymbol{u}(t)$ formulieren.

$$\dot{\boldsymbol{z}}(t) = \boldsymbol{A}(t)\boldsymbol{z}(t) + \boldsymbol{B}(t)\boldsymbol{u}(t) \tag{2.5}$$

In impliziter Darstellung und auch für nichtlineare Systeme ergibt sich somit die Zustandsform als System *gewöhnlicher Differentialgleichungen* (ODE):

$$0 = \boldsymbol{f}(\boldsymbol{z}(t), \dot{\boldsymbol{z}}(t), \boldsymbol{u}(t)) \qquad \text{kurz: } 0 = \boldsymbol{f}(t, \boldsymbol{z}, \dot{\boldsymbol{z}}, \boldsymbol{u}) \tag{2.6}$$

Damit ist das dynamische Verhalten allgemein modelliert. Soll nun ein bestimmtes reales System abgebildet werden, so müssen dessen Eigenschaften dimensionsbehaftet als Parameter $\boldsymbol{p}$ in das Modell einfließen,

$$0 = \boldsymbol{f}(t, \boldsymbol{z}, \dot{\boldsymbol{z}}, \boldsymbol{u}, \boldsymbol{p}), \tag{2.7}$$

wobei die Parameter konstant oder zeitvariant ($\boldsymbol{p} = \boldsymbol{p}(t)$) sein können. Parameter können als skalare Größen oder mehrdimensionale Kennfelder bereitgestellt werden. Die Parametrierung eines Modells ist neben der Verhaltensformulierung als zweiter wichtiger Teil häufig der aufwändigere. Sie muss durch Vermessen jeder benötigten Dimension einer Komponente

[1] Partial Differential Equations

oder durch Systemidentifikation erfolgen. Dazu sind das Vermessen das Systemverhaltens sowie iterative Simulationen des Modells notwendig, vgl. Abschnitt 2.3.

Damit lässt sich schließlich das vollständige Simulationsmodell als Anfangswertproblem (2.8), einschließlich der Ausgänge $\boldsymbol{y}$ und der Anfangszustände $\boldsymbol{z}_0$, formulieren.

$$\begin{aligned} 0 &= \boldsymbol{f}(t, \boldsymbol{z}, \dot{\boldsymbol{z}}, \boldsymbol{u}, \boldsymbol{p}) \\ \boldsymbol{y} &= \boldsymbol{g}(t, \boldsymbol{z}, \boldsymbol{u}, \boldsymbol{p}) \\ \boldsymbol{z}(t = t_0) &= \boldsymbol{z}_0 \end{aligned} \tag{2.8}$$

Eine für die Simulation relevante Eigenschaft ist die (numerische) Steifigkeit eines Systems. Sie wird in [55] wie folgt definiert:

$$\Lambda \geq 1000 \qquad \text{mit } \Lambda = \frac{|\lambda_{max}|}{|\lambda_{min}|} \qquad \text{und } \lambda_{min}, \lambda_{max}\text{: kleinster, größter Eigenwert} \tag{2.9}$$

In dieser Definition hängt sie ausschließlich von den Parametern der Systemmatrix ab. Allgemeiner lässt sich ein Gesamtsystem als steif bezeichnen, wenn die Dynamik seiner Komponenten bzw. Zustände stark unterschiedlich ist.

Deskriptorform DAE

In einem System können neben der freien Dynamik zusätzlich Zwangs- und Randbedingungen auftreten. Dies führt zu algebraischen Gleichungen im Modell. Auch die Einbeziehung mehrerer Domänen, z.B. die Formulierung von Medieneigenschaften, das Interpolieren der Werte aus Kennfeldern oder ein objektorientierter Modellierungsansatz führen auf ein gemischt *differential-algebraisches* System (DAE), und (2.7) wird zu

$$\begin{aligned} 0 &= \boldsymbol{f}_d(t, \boldsymbol{z}, \dot{\boldsymbol{z}}, \boldsymbol{s}, \boldsymbol{u}, \boldsymbol{p}) \\ 0 &= \boldsymbol{f}_a(t, \boldsymbol{s}, \boldsymbol{u}, \boldsymbol{p}) \end{aligned} \tag{2.10}$$

Es besteht aus in differentieller Form auftretenden Variablen $\boldsymbol{z}$ und aus solchen, deren Ableitung nicht auftritt, $\boldsymbol{s}$. Gemäß [62] können DAE- zwar in gewisser Hinsicht als eine Art steifer ODE-Systeme betrachtet werden, sie unterscheiden sich jedoch vor Allem unter Gesichtspunkten der Lösbarkeit und numerischen Behandlung signifikant von diesen. Die gebräuchlichste qualitative Klassifizierung von DAE-Systemen ist der *differentielle Index*. Dies ist die Anzahl an Zeitableitungen von (2.10), die notwendig sind, um es in ein explizites ODE-System zu überführen, [21]. Systeme mit höherem Index sind entweder nicht lösbar oder benötigen spezielle Lösungsverfahren. Aus diesem Grund wurden verschiedene Verfahren zur Indexreduktion entwickelt, die vor der eigentlichen Simulation zum Einsatz kommen, vgl. Abschnitte 2.2.1 und 2.3.

2.1.1. DAE mit Diskontinuitäten

Kommen zu den kontinuierlichen Variablen $\boldsymbol{z}, \boldsymbol{s}$ Diskontiniutäten in Form diskreter Variablen $\bar{\boldsymbol{z}}$ mit Komponenten aus $\boldsymbol{z}$ und $\boldsymbol{s}$ hinzu, so spricht man von *hybriden (DAE-)Modellen*. Gesamtmodelle mechatronischer Systeme haben häufig diesen Charakter. Zum Einen können Diskontinuitäten durch physikalische Effekte auftreten: Reibung, Stöße, Dioden,... , zum Anderen durch diskrete Modellteile der Steuerung und Regelung u.A. mit *boole*schen und *Integer*variablen. Abb. 2.3 a) zeigt beispielhaft die Modellierung der Reibung in einer Kupplung mit Sprüngen in der Reibungszahl μ beim Übergang vom Haften zum Gleiten sowie beim Richtungswechsel der relativen Rotationsgeschwindigkeit ω_{rel}. Umgesetzt wird dies etwa über

(a) diskontinuierliche Reibung (b) kontinuierliche Reibung

Abbildung 2.3.: Reibungsmodellierung am Beispiel Kupplung

if...else... -Abfragen im Modellcode. Dies führt zu *Ereignissen* während der Simulation hybrider Modelle. Auf die Einteilung und Behandlung von Ereignissen geht Abschnitt 2.3.2 ein. Die Ereignisbehandlung führt zu einem größeren Berechnungsaufwand und kann zu Robustheitsproblemen führen. Deshalb ist bei der Modellierung darauf zu achten, ob Ereignisse vermieden werden können. Am Beispiel der Kupplungsreibung ist ein Weg, statt der idealisierten Reibungskennlinie, Abb. 2.3 a), einen kontinuierlichen Verlauf, 2.3 b) vorzugeben. Zu beachten ist jedoch die numerische Steifigkeit, die durch den großen Gradienten zwischen den Extremwerten $\mu_{Peak,-}$ und $\mu_{Peak,+}$ ins System eingeführt wird.

2.2. Modellierungssprache Modelica

Nach wie vor ist in der industriellen Anwendung die Modellierung mit Signalflussplänen, vgl. Abbildung 2.2, weit verbreitet. Dieser aus der Darstellung von Regelsystemen entstandene Ansatz mit Blockschaltbildern stößt jedoch bei der Abbildung von realem physikalischen Verhalten an seine Grenzen. Um diesen Nachteil zu umgehen, werden seit den siebziger Jahren Sprachen zur *objektorientierten Modellbildung physikalischer Systeme* entwickelt. Durch seine Offenheit und die Weiterentwicklung innerhalb der nichtkommerziellen *Modelica Association* hat sich Modelica in den letzten Jahren als Sprachstandard [93] erfolgreich verbreitet.

Tabelle 2.2.: Vergleich: signalflussorientierte und physikalische Modellierung

	signalflussbasierte Modellierung	**physikalische, objekt-orientierte Modellierung**
Prinzip/ Wirkrichtung	kausal, gerichtete Signale, Zuweisungen: $y := f(u)$	akausal, bidirektionale Bindungen, (physikalische) Gleichungen: $0 = \boldsymbol{f}(t, \boldsymbol{z}, \dot{\boldsymbol{z}}, \bar{\boldsymbol{z}}, \boldsymbol{s}, \boldsymbol{u}, \boldsymbol{p})$
+ Vorteile	übersichtliche und domänenunabhängige Systemdarstellung	physikalische Gleichungen, modular, algebraische Schleifen auflösbar
- Nachteile	algebraische Schleifen nicht auflösbar [29], nicht modular, physikalische Struktur nicht erkennbar	Systemverhalten nicht direkt ablesbar
Beispiel	*Simulink* [136]	*Dymola* (Modelica) [37]

Es ist eine Weiterentwicklung der energieflussorientierten Systemformulierung mittels *Bond-Graphen*, [19]. Tabelle 2.2 fasst die wesentlichen Unterschiede zwischen signalflussorientierter und physikalischer Modellierung zusammen und zeigt einige Vor- und Nachteile auf. Für den Sprachstandard Modelica existieren mehrere Simulationswerkzeuge (freie, wie kommerzielle [37, 70]), welche als Modelica-*Editor* und -*Compiler* betrachtet werden können, die zusätzlich Integrationsalgorithmen zur Ausführung der Modelle bereitstellen. Da objektorientierte Multi-Physik-Modelle als hybride DAE-Systeme formuliert werden, müssen im *Preprocessing* der Werkzeuge, der Vorverarbeitung vor der eigentlichen Zeitintegration, verschiedene Algorithmen zur symbolischen Gleichungsmanipulation eingesetzt werden. Darauf wird in den Abschnitten 2.2.1 und 2.2.2 näher eingegangen.

2.2.1. Gleichungsbasierter Ansatz

Modelica ist eine gleichungsbasierte, objektorientierte Sprache zur Modellformulierung und wird in einer öffentlichen Spezifikation [93] definiert. Ein Modell setzt sich modular aus einer oder mehreren hierarchisierten Komponenten zusammen. Dies sind Objekte, die von Modellklassen *instantiiert*, d.h. abgeleitet und parametriert werden. Das physikalische Verhalten ist im Gleichungsteil einer Klasse beschreibbar. Im Deklarationsteil können Variablen und Parameter sowie Objekte anderer Klassen eingebunden werden. Die spezifischen physikalischen Eigenschaften werden dann über Parameterwerte bedatet, wie im Beispiel unten dargestellt.

Zur Interaktion zwischen den Objekten werden Schnittstellen in *Konnektorklassen* definiert. Sie enthalten die Definition von *Potential-* (x_i) und *Flussvariablen* (f_i). Bei Verbindung von N entsprechenden Konnektoren entsteht eine akausale Bindung durch automatisches erzeu-

gen von Gleichungen, (2.11). (2.11a) stellt die Erfüllung der Randbedingung N identischer Potentialwerte im Verbindungspunkt sicher und durch (2.11b) werden die Erhaltungsgleichungen, wie das Energiegleichgewicht erfüllt (*1. Kirchhoff'sches Gesetz*).

$$x_1 = x_2 = ... = x_N \tag{2.11a}$$

$$\sum_{i=1}^{N} f_i = 0 \tag{2.11b}$$

Dies ermöglicht eine Netzwerkmodellierung mit beliebigen verzweigten oder zyklischen Strukturen. Darüber hinaus lassen sich in Modelica auch gerichtete Signalschnittstellen für kausale Modellierung darstellen. Somit sind auch Regelstrukturen mit den bekannten Blockschaltbildern realisierbar. Es lassen sich weiterhin Bussysteme aus Signalen zusammenfassen, die in der Fahrzeugsystemsimulation Verwendung finden.

Der Modellaufbau soll an der textuellen und graphischen Repräsentation einer rotierenden Schwungmasse verdeutlicht werden. Auf oberster Ebene werden im Deklarationsteil die Komponenten für festes Ende, Feder-Dämpfer und Masse als Objekte ihrer entsprechenden Klassen mit den Parameterwerten (für c, d, θ), sowie einem Anfangswert für die Auslenkung $\varphi_{T,0}$ eingebunden. Im Gleichungsteil werden die Konnektoren dieser Ebene verbunden und es können weitere Gleichungen hinzugefügt werden. Für die grafische Darstellung im Objektdiagramm besitzen Objekte und Verbindungen *Annotationen*, im Beispiel zur Übersicht als **a** zusammengefasst. Diese definieren das Objektdiagramm, Abbildung 2.4. Somit kann sowohl grafisch, als auch textbasiert modelliert werden:

```
model Schwungmasse
  Fixed  Fix(phi0=0) a;
  SpringDamper  FederDaempfer (c=100, d=0.5) a;
  Inertia  Traegheit (J=1, phi(start=1.5707963267949,fixed=true)) a;
  Real  energie_Traegheit;
  Real  energie_FederDaempfer;
  Real  energie_Verlust;
equation
  connect(Fix.flange,  FederDaempfer. flange_a) a;
  connect(FederDaempfer.flange_b,  Traegheit.flange_a) a;
   energie_Traegheit = 0.5*Traegheit.J*Traegheit.w ^2;
   energie_FederDaempfer = 0.5*FederDaempfer.c*FederDaempfer.phi_rel ^2;
   energie_Verlust = FederDaempfer.tau_d*FederDaempfer.phi_rel;
  a;
end Schwungmasse;
```

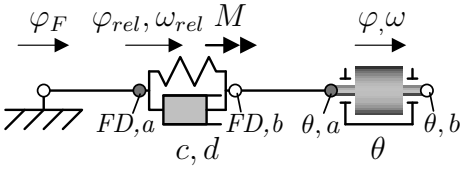


Abbildung 2.4.: Graphische Repräsentation in Modelica: Beispielmodell Drehschwingung

Die eingebundenen Komponenten für festes Ende, Feder-Dämpfer und Trägheit enthalten die Gleichungen ihrer Modellklassen, Teilsystem (2.12). Mit $x_i = \varphi_i$ und $f_i = M_i$ entsteht

aus (2.11) das Teilsystem (2.13), worin die letzte Gleichung in Modelica automatisch für die Flussvariablen freier Enden eingefügt wird. Ergänzt wird das Gleichungssystem des Modells durch die Gleichungen in oberster Modellhierarchieebene, (2.14).

festes Ende	Feder-Dämpfer	Trägheit
$\varphi_F = \varphi_{F0}$	$\varphi_{rel} = \varphi_{FD,b} - \varphi_{FD,a}$	$\varphi = \varphi_{\theta,b}$
	$\omega_{rel} = \dot{\varphi}_{rel}$	$\varphi = \varphi_{\theta,a}$
	$M_{FD,b} = M$	$\omega = \dot{\varphi}$
	$M_{FD,a} = -M$	$\dot{\omega} = \ddot{\varphi}$
	$M_c = c\varphi_{rel}$	$\theta\dot{\omega} = M_{\theta,a} + M_{\theta,b}$
	$M_d = d\omega_{rel}$	
	$M = M_c + M_d$	

(2.12)

$$\begin{array}{cc} \varphi_F = \varphi_{FD,a} & \varphi_{FD,b} = \varphi_{\theta,a} \\ M_F + M_{FD,a} = 0 & M_{FD,b} + M_{\theta,a} = 0 \\ & M_{\theta,b} = 0 \end{array} \tag{2.13}$$

$$E_{kin} = \frac{1}{2}\theta\omega^2 \qquad E_{pot} = \frac{1}{2}c\varphi_{rel}^2 \qquad E_{Wärme} = M_d\varphi_{rel} \tag{2.14}$$

Neben der Sprachdefinition existiert die frei zugängliche Modelica-*Standard-Bibliothek*. Ihr sind u. A. die Modellklassen der Einzelkomponenten im obigen Beispiel entnommen. Aufgrund des objektorientierten Ansatzes, ist es möglich in Modelica alle physikalischen Domänen abzudecken. Somit existiert eine Vielzahl an definierten Komponenten und Konnektoren. Tabelle 2.3 gibt einen Überblick der Potential- und Flussvariablen der Konnektoren für die in dieser Arbeit relevanten Domänen. In der Standard-Bibliothek sind weitere Domänen, wie Magnetismus, vorhanden. Darüber hinaus sind weitere, teilweise kommerzielle Modellbibliotheken erhältlich, die oft in einer Domäne spezialisiert sind und darin auch komplexere, vorgefertigte Modelle aus mehreren Komponenten enthalten.

Die letzte Spalte in Tabelle 2.3 zeigt die Umsetzung hydraulischer Strömungsschnittstellen in Modelica. Hierfür wurden zusätzlich *Streamvariablen* eingeführt, deren Behandlung vom Vorzeichen der Flussvariablen ($\dot{m}$) abhängt. Somit ist eine komponentennahe bidirektionale Modellierung möglich, bei der die Strömungsrichtung nicht a priori bekannt sein muss. Komponenten der Domäne Fluid werden mit Modellen für das entsprechende Medium parametriert. Diese enthalten Funktionen für die thermodynamische Zustandsänderung, sowie einen Parametersatz für die spezifischen Medieneigenschaften. Dies führt häufig zu algebraischen Nichtlinearitäten im Gesamtsystem.

Tabelle 2.3.: Schnittgrößen der physikalischen Domänen

physikalische Domäne	**Potential-Variablen**	**Fluss-Variablen**	**Stream-Variablen**
3D mechanisch	Ortsvektor $\boldsymbol{r}$ Drehmatrix $\boldsymbol{R}$	Schnittkräfte $\boldsymbol{F}$ Schnittmomente $\boldsymbol{M}$	
1D mech. translatorisch	Weg s	Schnittkraft F	
1D mech. rotatorisch	Winkel φ	Schnittmoment M	
thermisch	Temperatur T	Wärmefluss $\dot{Q}$	
fluidisch (Strömung, kalorisch)	Druck p	Massenstrom $\dot{m}$	spez. Enthalpie h Massenbrüche $\boldsymbol{x}$ Medieneigensch. $\boldsymbol{c}$
elektrisch	Potential V	Strom i	
...	...	...	

Vorverarbeitung der Modellgleichungen

Vom Modelica-Compiler wird aus einem hierarchischen Modell ein im allgemeinen Fall hybrides DAE-System, vgl 2.1.1, erstellt. Gemäß [93] werden zunächst alle Modellgleichungen eingebundener und vererbter Objekte zusammengefasst (*Flattening*). Anschließend werden alle Konnektorgleichungen (2.11) eingefügt, sowie zusätzlich alle Zuweisungen aus den in Modelica möglichen *Algorithmus*-Abschnitten. Schließlich wird sortiert und auf ein System folgender Form gebracht:

$$0 := \boldsymbol{f}_z(\boldsymbol{b}, t, \boldsymbol{z}, \dot{\boldsymbol{z}}, \boldsymbol{s}, \bar{\boldsymbol{z}}, \boldsymbol{u}, \boldsymbol{p}) \tag{2.15a}$$

$$\boldsymbol{b} := \boldsymbol{f}_{bed}(t, \boldsymbol{z}, \dot{\boldsymbol{z}}, \boldsymbol{s}, \bar{\boldsymbol{z}}, \boldsymbol{u}, \boldsymbol{p}) \tag{2.15b}$$

$$\bar{\boldsymbol{z}} := \boldsymbol{f}_{\bar{z}}(\boldsymbol{b}, t, \boldsymbol{z}, \dot{\boldsymbol{z}}, \boldsymbol{s}, \bar{\boldsymbol{z}}, \boldsymbol{u}, \boldsymbol{p}) \tag{2.15c}$$

Darin ist (2.15a) das abschnittsweise kontinuierliche und numerisch zu lösende DAE-System. (2.15b) fasst alle Bedingungen (z.B. in *if...else...*) zusammen, zu denen eine diskrete Änderung einer Variablen auftritt. Tritt ein solches *Ereignis* auf und eine Komponente in $\boldsymbol{b}$ ändert sich, so wird das gesamte System, inklusive der diskreten Modellteile (2.15c) zum Zeitpunkt des Ereignisses einmalig gelöst. Anschließend wird wieder nur (2.15a) numerisch integriert.

Die modulare Modellierung mit Objekt- und Bindungsgleichungen führt zu Deskriptorsystemen (2.15a) mit hohem Anteil algebraischer Gleichungen. Je nach Domäne führt dies zusätzlich auf singuläre Systeme mit Index 2 oder mehr. Eine direkte numerische Lösung ([21, 62]) ist aus diesem Grund nicht möglich bzw. wegen des hohen Aufwands nicht zielführend. Folglich kommen weitere Algorithmen zur Vorverarbeitung von (2.15) zum Einsatz. Das jeweilige Vorgehen kann hier nicht im Detail behandelt werden. Eine ausführliche Darstellung ist u. A. in [29, 102] zu finden. Zusammenfassend gilt: Durch symbolische Transformation wird

das System in explizite und implizite Variablen sortiert, und alle redundanten Gleichungen werden eliminiert. Dafür ist zu unterscheiden, ob die Jacobi-Matrix von (2.15a) im Bezug auf die Ableitung des Zustandsvektors $\boldsymbol{z}$ und der algebraischen Variablen $\boldsymbol{s}$

$$\boldsymbol{J} = \left[\frac{\partial \boldsymbol{f}_z}{\partial \dot{\boldsymbol{z}}} : \frac{\partial \boldsymbol{f}_z}{\partial \boldsymbol{s}}\right] \tag{2.16}$$

regulär oder singulär ist. Bei regulärer Matrix $\boldsymbol{J}$ und linearem $\boldsymbol{f}_z$ wird die Adjazenzmatrix, die das Auftreten der Komponenten von $\boldsymbol{z}$ und $\boldsymbol{s}$ in $\boldsymbol{f}_z$ zuordnet, auf Basis des *Tarjan*-Algorithmus ([132]) durch Permutation auf *BLT*-Form[2] gebracht. Im nichtlinearen Fall kann die *Tearing*-Methode, [86], angewandt werden, bei der Variablen substituiert werden, um die Systemordnung zu reduzieren.

Ist $\boldsymbol{J}$ dagegen singulär, so muss der DAE-Index von (2.15a) reduziert werden. Dazu kommt der auf den Tarjan-Algorithmus aufbauende Algorithmus nach Pantelides [104] zum Einsatz, der versucht, in (2.15a) den Minimalsatz an Gleichungen zu detektieren, die zu differenzieren sind, um den Index zu reduzieren. Er findet so zum Einen konsistente Anfangswerte für die Lösung und liefert zum Anderen den Gleichungssatz für die Methode der *Dummy Derivatives* nach [89]. Sie behandelt die durch die Ableitung entstehenden zusätzlichen Unabhängigen als algebraische Variablen, die explizit bestimmbar sind und nicht als Zustände deklariert werden. Das so reduzierte DAE-System ist dann regulär, und im Weiteren können BLT-Transformation oder Tearing verwendet werden.

Nach beendeter Vorverarbeitung wird (2.15) mit Hilfe eines der in Abschnitt 2.3 diskutierten Solver in der jeweiligen Modelica-Umgebung gelöst. Zur Initialisierung der Anfangswerte des Zustandsvektors gibt es in Modelica die Möglichkeit zusätzliche *initial equations* zu definieren. Für die Übrigen werden konsistente Werte nach Möglichkeit automatisch bestimmt. Insgesamt besteht neben den Solveralgorithmen vor allem in der Umsetzung der Gleichungsvorverarbeitung die Expertise der Modelica-Simulatoren. Aus diesem Grund ist die Umsetzung bei den kommerziellen Vertretern nicht transparent. Als Ergebnis der Vorverarbeitung des obigen Beispiels verbleiben aus den 18 Gleichungen in (2.12) und (2.13) fünf nicht-triviale Gleichungen:

$$\begin{aligned}
\dot{\varphi}_{rel} &:= \omega_{rel} \\
M_c &:= c\varphi_{rel} \\
M_d &:= d\omega_{rel} \\
M_F &:= M_c + M_d \\
\dot{\omega}_{rel} &:= \frac{-M_F}{\theta}
\end{aligned} \tag{2.17}$$

[2]Block Lower Triangular Form: untere Dreiecksmatrix

mit den zwei Zuständen

$$\boldsymbol{z} = \begin{pmatrix} \varphi_{rel} \\ \omega_{rel} \end{pmatrix}. \tag{2.18}$$

Da die Berechnung der Energien (2.14) auf die Modellzustände keinen Einfluss hat, werden diese in einen gesonderten Satz Gleichungen für Modellausgänge $\boldsymbol{y} = [\varphi \quad E_{kin} \quad E_{pot} \quad E_{W\ddot{a}rme}]^T$ sortiert, der nur bei Bedarf berechnet werden muss. Darin ist ω durch ω_{rel} ersetzt es kommt $\varphi := \varphi_{rel} - \varphi_{F0}$ hinzu. Der Parametervektor im Beispiel hat die vier Werte: $\boldsymbol{p} = [\varphi_{F0} \quad c \quad d \quad \theta]^T$.

2.2.2. Objektorientierte Modellierung

Modelica bedient sich als objektorientierte Sprache der gleichen Paradigmen wie die Sprachen Java oder C++. Dies sind die Abstraktion in Klassen und deren abgeleitete Objekte, Datenkapselung und Vererbung. Alle Objekte einer Klasse haben die in der Klasse definierten Attribute (Deklarationsteil mit Variablen, Parametern und Objekten), sowie die darauf anwendbaren Methoden (Gleichungs-/Algorithmusteil) gemein. Dabei lässt sich der Zugriff von außen über Schlüsselwörter steuern und so die Daten lokal in einem Objekt kapseln. So nutzt Modelica etwa die Typ-Klassen, um physikalische Größen vorzudefinieren, die als Attribut ihre SI-Einheit haben. Durch die Möglichkeit, Objekte in Klassen zu integrieren, wird eine übersichtliche Modellierung auf verschiedenen Hierarchieebenen und in Objektdiagrammen ermöglicht. Die Vererbung ermöglicht es, eine Klasse von einer Basisklasse abzuleiten, von der sie Attribute und Methoden erbt. Diese können überschrieben und erweitert werden. Im Beispiel unter 2.2.1 erbt die Klasse `SpringDamper` durch `extends` von der Basisklasse `PartialCompliantWithRelativeStates` die beiden Konnektoren (vgl. Abb. 2.4) sowie die ersten vier Gleichungen im Mittelteil von (2.12). Diese werden auch in anderen rotatorischen Bindungsmodellen verwendet.

```
model SpringDamper
  [...]extends PartialCompliantWithRelativeStates;[...]
protected
  Torque tau_c;[...]
equation[...]
end SpringDamper;
```

In einem Modell lassen sich einzelne Objekte als `replaceable` deklarieren. Diese sind dann per `redeclare`-Befehl durch andere Objekte ersetzbar, welche dieselbe Basisklasse besitzen. Somit wird modulares Modellieren durch die Objektorientiertheit ermöglicht. Auf diese Weise lassen sich bei der Fluid-Modellierung in ein bestehendes Leitungsmodell unterschiedliche Medienmodelle einsetzen. Das Konzept wird in Kapitel 4 und 5 für die Trennung von Daten und Modell sowie die Konfiguration von Modellen und für die Co-Simulation verwendet.

2.3. Simulation

Gemäß obiger Definition ist Simulation das Durchführen eines Experimentes am Simulationsmodell. Bei Simulation im Zeitbereich bedeutet dies eine Erstellung von Ergebnistrajektorien

der Modellzustände $\boldsymbol{z}(t)$ (durch Lösen von (2.15)). Zunächst sollte dies der *Validierung* des Modells dienen. Das Vorgehen zur Parametrierung und Validierung ist Abb. 2.5 zu entnehmen. Sollte das Vermessen des Gesamtsystems nicht möglich sein, so sind die Subsysteme

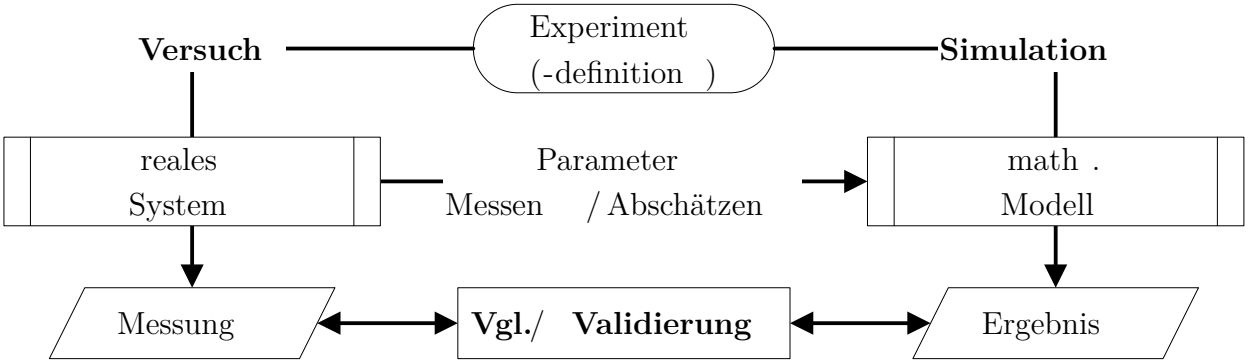


Abbildung 2.5.: Parametrierung und Validierung des Simulationsmodells

einzeln zu validieren. Ein deterministisches, validiertes Modell erlaubt es, belastbare Aussagen zum realen Systemverhalten zu treffen. Dies sollte dann auch gelten, wenn die Modellparameter $\boldsymbol{p}(t)$, die Eingänge $\boldsymbol{u}(t)$ oder die Konfiguration des Modells in sinnvollen Grenzen variiert werden. Bleibt ein Modell bei dieser Variation valide, so ist es als *robust* zu bezeichnen. Im allgemeinen Fall ist eine analytische Lösung des Modellgleichungssystems für $\boldsymbol{z}(t)$ nicht möglich. Stattdessen wird ein numerisches Integrationsverfahren gewählt. Diese werden im folgenden Abschnitt vorgestellt. Zumeist sind sie in ihrer Umsetzung in einen Algorithmus eingebunden, der darüber hinaus konsistente Anfangswerte finden, die Integrationsschrittweite oder -Ordnung variieren oder Ereignisse handhaben kann. Solche Algorithmen werden als Solver bezeichnet und in Abschnitt 2.3.2 besprochen.

2.3.1. Numerische Integrationsverfahren

Zur Lösung eines Anfangswertproblems (2.8) oder Systemen der Form (2.10) mit geringem Index werden Verfahren herangezogen, die die Lösung $\boldsymbol{z}(t)$ durch numerische Integration approximieren, $\boldsymbol{z}_k \approx \boldsymbol{z}(t_k)$. Dies geschieht durch Diskretisierung der Simulationsintervalls von t_0 bis zur Endzeit t_S in S Schritte Δt_k.

$$\Delta t_k = t_{k+1} - t_k = h_k, \quad k = 0, ..., S-1, \quad t \in [t_0, t_S] \tag{2.19}$$

Für konstante Zeitschritte $h_k = h$ folgt daraus $t_k = t_0 + k * h$. Es können jedoch im Allgemeinen variable Werte $h_k \neq h_{k+1}$ gewählt werden. Ausführliche Beschreibungen der vielfältigen Integrationsverfahren für $\boldsymbol{z}_k$ finden sich u.A. in [21, 29, 55, 61, 62, 121]. Die in der Praxis Eingesetzten lassen sich gemäß Abb. 2.6 einteilen. Auf oberster Ebene lässt sich die allgemeine

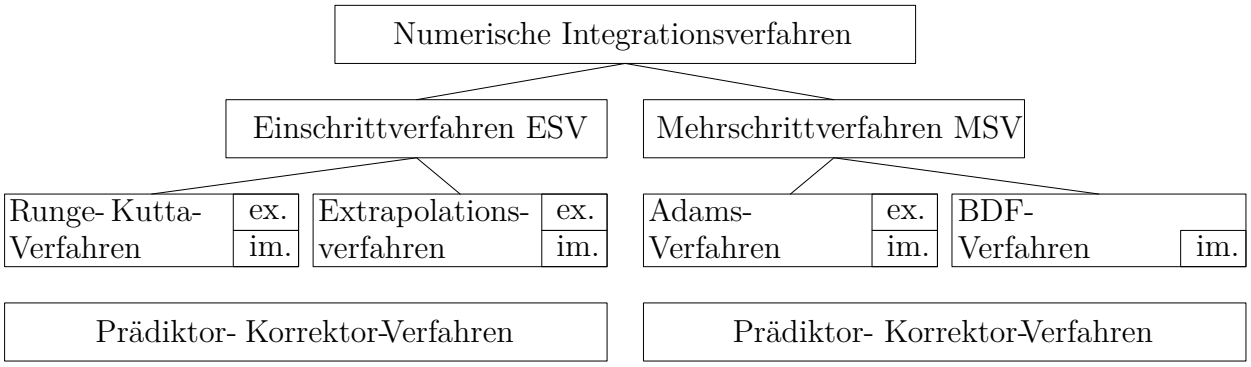


Abbildung 2.6.: Kategorien numerischer Zeitintegrationsverfahren

Integrationsvorschrift mit der Verfahrensfunktion $\boldsymbol{\phi}$ formulieren:

$$\sum_{i=0}^{n} \alpha_i \boldsymbol{z}_{k+1-i} = h_k \boldsymbol{\phi}(t_{k+1}, \boldsymbol{z}_{k+1}, \boldsymbol{u}_{k+1}, ..., t_{k+1-m}, \boldsymbol{z}_{k+1-m}, \boldsymbol{u}_{k+1-m}, \boldsymbol{p}) \tag{2.20}$$

Die *Einschrittverfahren* (ESV) verwenden lediglich die Werte des k-ten Schritts zur Berechnung des $k+1$ten. Sie haben in (2.20) die Werte $n = 1$, $\alpha_0 = 1$ und $\alpha_1 = -1$ und lassen sich weiter unterteilen. Runge-Kutta-Verfahren (RK) basieren auf Stützstellen c_j innerhalb h_k mit $\boldsymbol{\phi} = \sum_{j=0}^{r} \beta_j \boldsymbol{f}_j(t_k + c_j h_k, \boldsymbol{z}_k + h_k \sum_{l=0}^{r} a_{lj} \boldsymbol{f}_l)$. Die Extrapolationsverfahren berechnen mehrfach $\boldsymbol{z}_{k+1,j}$ in q_j kleineren Partialschritten $\frac{h_k}{q_j}$ und extrapolieren dann $\boldsymbol{z}_{k+1}$ für $q_j \to \infty$. ESV können jeweils explizit oder implizit formuliert sein. Im impliziten Fall steht $\boldsymbol{f}$ zum neuen Zeitpunkt t_{k+1} auf der rechten Seite und es muss ein Newton-Raphson-Iterationsverfahren für $\boldsymbol{z}_{k+1}$ herangezogen werden, [55]. Für die *Mehrschrittverfahren* (MSV) läßt sich die meist verwendete, lineare Verfahrensfunktion:

$$\boldsymbol{\phi} = \sum_{j=0}^{m} \beta_j \boldsymbol{f}(t_{k+1-j}, \boldsymbol{z}_{k+1-j}, \boldsymbol{u}_{k+1-j}, \boldsymbol{p}) \tag{2.21}$$

formulieren. Die Verfahren vom Adams-Typ haben in (2.20) $n = 1$, in (2.21) $m \geq 1$ und es kommen somit Werte der Systemfunktion mehrerer vergangener Schritte zum Einsatz. Man unterscheidet weiter in die expliziten Adams-Bashfoth-Verfahren mit $\beta_0 = 0$ und die impliziten Adams-Moulton-Verfahren mit $\beta_0 \neq 0$. Als vierter Verfahrenstyp sind die BDF[3]-Verfahren zu nennen, welche auf Werte des Zustandsvektors mehrerer vergangener Schritte zugreifen, hier gilt somit $n \geq 1$ und $m = 0$. Mit $\beta_0 = 1$ sind diese stets implizit. Beim Start einer Simulation verwenden MSV zunächst die Werte eines ESV und werden dann schrittweise auf das entsprechende m- bzw. n- Schrittverfahren erhöht.

Der Rechenaufwand pro Zeitschritt hängt bei expliziten Verfahren hauptsächlich von der Anzahl der $\boldsymbol{f}$-Auswertungen innerhalb $\boldsymbol{\phi}$ ab und ist damit begrenzt. Die numerisch stabileren impliziten Verfahren haben im Allgemeinen einen höheren Aufwand aufgrund der

[3]Backward Differentiation Formulas

zusätzlich notwendigen Iterationen und der dazu benötigten Ermittlung der Jacobi-Matrix der Systemfunktion $\boldsymbol{f}$. Um Vorteile beider Ansätze zu kombinieren, werden die Prädiktor-Korrektor-Verfahren eingesetzt. Diese verwenden zunächst ein explizites Verfahren im Prädiktorschritt für $\boldsymbol{z}_{k+1}^{(P)}$, welcher als Ausgangspunkt für einen oder mehrere Korrektorschritte $\boldsymbol{z}_{k+1}^{(C)}$ mit einem impliziten Verfahren verwendet wird.

Als einfachstes Beispiel für Umsetzungen von ESV wird das Eulerverfahren eingesetzt. Es kann als RK-Verfahren explizit ($r = 0,\ \beta_0 = 1,\ a_{lj} = 0$) oder implizit formuliert sein ($r = 1,\ \beta_0 = 0,\ \beta_1 = 1,\ a_{01} = 1,\ a_{11} = 0$). Meist müssen jedoch genauere ESV-Verfahren angewandt werden, wie RK-Verfahren höherer Ordnung, die Verfahren der Dormand-Prince-Klasse oder RADAU-Verfahren, [61]. Vertreter der MSV sind die auf Umsetzungen des Adams-Ansatzes basierenden Algorithmen DEABM oder LSODE, [62]. Der in [106] erstmals vorgestellte DASSL-Algorithmus, hat sich besonders für den Einsatz mit DAE-Modellen mechatronischer Systeme bewährt [21], und kommt auch in der vorliegenden Arbeit zum Einsatz. Es handelt sich um ein Prädiktor-Korrektor-Verfahren mit Prädiktorpolynom und BDF-Korrektor. Der Code besitzt sowohl eine Steuerung der Schrittweiten h_k als auch eine Steuerung der Verfahrensordnung, siehe auch Abschnitt 2.3.2. Generell können zur Lösung von DAE-Systemen nur die impliziten Verfahren herangezogen werden, sofern das System nicht durch Vorverarbeitung, vgl. 2.2.1 in ein ODE-System überführt werden kann.

Numerische Eigenschaften

Die numerische Mathemathik erlaubt die Analyse der vorgestellten Verfahren hin auf ihre Genauigkeit und Ordnung. So lassen sich mit der *Konvergenzordnung* Aussagen über die Qualität eines bestimmten Verfahrens und mit dem *Stabilitätsbegriff* über die Kombination mit dem Modellsystem treffen.

Bei der Untersuchung auf *Konsistenz* eines Verfahrens wird der *lokale Fehler*, auch *Diskretisierungsfehler*,

$$\bar{\boldsymbol{e}} = \boldsymbol{z}_{k+1} - \boldsymbol{z}(t_{k+1}) \quad \text{mit } \boldsymbol{z}_k := \boldsymbol{z}(t_k), \tag{2.22}$$

der durch die Approximation der exakten Lösung in *einem* diskreten Schritt entsteht, abgeschätzt. Dies geschieht meist durch Vergleich der Taylorreihenentwicklung beider Lösungen mit Schrittweite h. Ein Verfahren besitzt die Konsistenzordnung p, wenn gilt:

$$\boldsymbol{z}_{k+1} - \boldsymbol{z}(t_{k+1}) = \mathcal{O}(h_k^{p+1}). \tag{2.23}$$

Aussagen über das Verhalten bei verschwindender Schrittweite, $h_k \to 0$, liefert die *Nullstabilität*: Liegen die Nullstellen der charakteristischen Gleichung der linken Seite von (2.20) in der komplexen Ebene innerhalb (bei einfachen Nullstellen auch auf) dem Einheitskreis, so ist das Verfahren nullstabil, [62]. Dies ist bei ESV durch $n = 1$ immer gegeben. Zur Analyse

der *Konvergenz* wird nun der Fehler nach k Schritten, der *globale Fehler*, der sich aus der Fortpflanzung der lokalen Fehler ergibt, betrachtet. Verschwindet er für $h_k \to 0$, so liegt ein konvergentes Verfahren vor. Gilt für den globalen Fehler

$$\boldsymbol{z}_k - \boldsymbol{z}(t_k) = \mathcal{O}(h_k^p), \tag{2.24}$$

so hat es die Konvergenzordnung p. Gemäß dem Satz von Lax [87] besitzt ein nullstabiles Verfahren mit Konsistenzordnung p auch die Konvergenzordnung p.

Die Konvergenz garantiert numerische Stabilität jedoch nur für sehr kleine Schrittweiten. Im Hinblick auf den Rechenaufwand sind in der Anwendung möglichst große Werte für h_k anzustreben. Da die größtmögliche stabile Schrittweite h_{krit} auch vom Modell und den Anfangswerten abhängt, wird zur Analyse der *absoluten Stabilität* eines Verfahrens das lineare *Dahlquist*-Testproblem $\dot{z} = \lambda z$ mit $z_0 = 1$ und $\lambda \in \mathbb{C}$ herangezogen. Mit dessen Hilfe kann das *Stabilitätsgebiet* S eines Verfahrens in der komplexen Ebene dargestellt werden, wie etwa in [61, 121]. Mit Kenntnis aller Eigenwerte λ_i eines Modells läßt sich dann h_{krit} so bestimmen, dass alle $h\lambda_i$ noch innerhalb von S liegen.

In der praktischen Anwendung ist neben Stabilität und Genauigkeit bei der Auswahl eines Integrationsverfahrens für ein bestimmtes Modell die Rechenzeit t_{CPU} von Interesse. Der Berechnungsaufwand eines Solvers hängt vom Modell, der Schrittweite, der Zahl der Auswertungen der rechten Seite, der Jacobiberechnungen und dem Auftreten von Ereignissen ab. Erst abhängig vom verwendeten Computersystem ergibt sich daraus die Rechenzeit. Ein Maß entsteht aus dem Vergleich mit der Simulationszeit $t_{Sim} = t_S - t_0$, dem Echtzeitfaktor

$$\theta_E = \frac{t_{CPU}}{t_{Sim}}. \tag{2.25}$$

2.3.2. Schrittweitensteuerung und Ereignisse

Neben der Wahl eines Integrationsverfahrens ist eine problemangepasste Wahl der Diskretisierungsschrittweite h ebenso notwendig. Sie stellt einen Kompromiss aus Genauigkeit und Aufwand dar. Zu beachten ist hierbei, dass sich der Rundungsfehler proportional zu $\frac{1}{h}$ und der Diskretisierungsfehler $\bar{\boldsymbol{e}}$ proportional zu h^p verhält. Der Rundungsfehler kann somit bei zu klein gewählten h den Genauigkeitsgewinn aufheben und es ergibt sich somit eine Untergrenze für h. Als Obergrenze sollte das noch stabile h_{krit} gelten. Gegenüber einer festen Schrittweite h_{fix} bietet deren Steuerung Vorteile: zuverlässigere Lösung durch Entfall der h_{fix}-Wahl durch den Nutzer; stabilisierte Integration; Anpassung an dynamischen Lösungsverlauf. So veranschaulicht [121], dass die Steuerung von h_k gegenüber h_{fix} einerseits bei gleichem Aufwand die Genauigkeit erhöht und andererseits bei gleichem globalen Fehler den Aufwand reduziert. Abbildung 2.7 zeigt eine Einteilung der vorhandenen Verfahren zur Schrittweitensteuerung. Ihnen gemein ist eine jeweilige Strategie ν zur Bestimmung der neuen

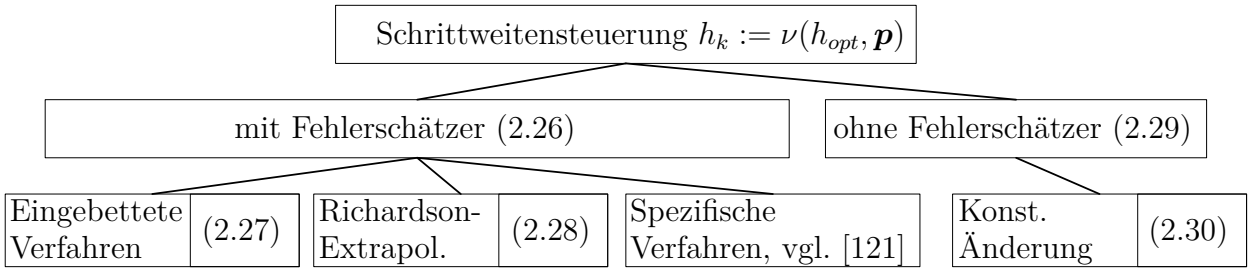


Abbildung 2.7.: Kategorien der Verfahren zur Schrittweitensteuerung

Schrittweite h_k aus einem Steuerergebnis h_{opt} und Parametern $\boldsymbol{p}$, die u.A. die Änderung von h einschränken. Neben heuristischen Strategien werden auch Methoden aus der Kontrolltheorie herangezogen, [144]. Nahezu alle gängigen Verfahren bedienen sich einer Fehlerschätzung $\boldsymbol{est}$, die mit vom Nutzer vorgegebenen Fehlertoleranzen $\boldsymbol{tol}$ verglichen wird:

$$h_{opt} = h_{opt}(\boldsymbol{z}_k, \boldsymbol{est}, \boldsymbol{tol}) \tag{2.26}$$

Da die Abschätzung des globalen Fehlers schwierig ist, wird der Betrag des lokalen Fehlers (2.22) abgeschätzt, $\boldsymbol{est} \approx |\bar{\boldsymbol{e}}|$. Nach der Art der Fehlerschätzung lässt sich weiterhin unterteilen. Bei den Fehlberg- bzw. eingebetteten Verfahren wird parallel mit unterschiedlicher Verfahrensordnung, meist $\hat{p} = p \pm 1$, gerechnet und verglichen,

$$\boldsymbol{est} = \left| \boldsymbol{z}_{k+1}(\phi_p(h_{k-1})) - \hat{\boldsymbol{z}}_{k+1}(\phi_{\hat{p}}(h_{k-1})) \right| . \tag{2.27}$$

Bei der Richardson-Extrapolation wird der Fehler durch Parallelrechnung mit unterschiedlichen Schrittweiten, $q \geq 2$, bestimmt,

$$\boldsymbol{est} = \left| \frac{\boldsymbol{z}_{k+q}(h_{k-1}) - \boldsymbol{z}_{k+1}(q h_{k-1})}{q^p - 1} \right| . \tag{2.28}$$

Weiterhin gibt es Verfahrensabhängige Fehlerschätzer, die häufig bei MSV zum Einsatz kommen und etwa, wie beim DASSL, den Vergleich des Prädiktors mit Korrektortermen heranziehen. Da die Fehlerschätzung zunächst einen Mehraufwand mitbringt, wurden auch Verfahren ohne Fehlerschätzer untersucht. Dazu müssen andere Indikatoren $\boldsymbol{ind}$, als $|\bar{\boldsymbol{e}}|$ herangezogen werden:

$$h_{opt} = h_{opt}(\boldsymbol{z}_k, \boldsymbol{ind}, \boldsymbol{tol}). \tag{2.29}$$

So untersucht beispielsweise [123] die Methode konstanter Änderungsraten, bei der sich die Schrittweite nach den Gradienten der Lösungskomponenten richtet,

$$\boldsymbol{ind} = \dot{\boldsymbol{z}}, \tag{2.30}$$

und deren Wirksamkeit an verschiedenen Testfällen nachgewiesen wird. zur Bestimmung einer skalaren h_{opt} aus den vektorwertigen Größen in (2.26) und (2.29) für n Lösungskomponenten in $\boldsymbol{z}$, kommen unterschiedliche Normen zum Einsatz. Die Gängigste [61] skaliert die Komponenten mit absoluter $Atol_i$ beziehungsweise relativer Toleranz $Rtol_i$ zu einem Fehlerindikator

$$err = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(\frac{est_i}{Atol_i + Rtol_i|z_i|}\right)^2}. \tag{2.31}$$

Bei den MSV müssen aufgrund geänderter Schrittweiten durch die Steuerung und damit zusätzlich benötigter Werte besondere Maßnahmen getroffen werden, die u.A. in [121] besprochen werden. Weiterhin kommen bei Solvern mit MSV zur Steuerung der Genauigkeit und Effizienz Verfahren variabler *Ordnung* zum Einsatz, wie es beim DASSL der Fall ist.

Ereignisse

Die oben beschriebenen Integrationsverfahren setzen einen ausreichend oft differenzierbaren Lösungsverlauf aller Systemkomponenten voraus. Jedoch kann das Modell, wie unter 2.1.1 beschrieben, durch ein strukturvariantes System bedingte Unstetigkeiten enthalten. Zum Zeitpunkt der Unstetigkeit wird dann während der Simulation ein Ereignis ausgelöst, die Berechnung unterbrochen und (2.10) neu initialisiert. Anschließend wird bis zum nächsten Ereignis fortgefahren, vgl. auch Abschnitt 2.2.1 zur Behandlung hybrider DAE in Modelica. Die Behandlung von Ereignissen ist bereits länger Gegenstand der Forschung und häufig implementiert [2, 97, 107, 122].

Gemäß der meist englischen Notation lassen sich drei Typen von Ereignissen unterscheiden: ein *state event* tritt auf, wenn eine Variable einen definierten Grenzwert übertritt. Hierbei entsteht der Zusatzaufwand, dass der genaue Zeitpunkt häufig iteriert werden muss (Detektion Nulldurchgang), bevor das System neu initialisiert werden kann. Dagegen ist ein *time event* durch vorherige Kenntnis des Ereigniszeitpunkts definiert, und der Solver kann h_k entsprechend wählen. Ähnlich werden Solver mit dynamischer Zustandswahl diese nur zu vollendeten Schritten, also an *step events* wechseln.

Jedoch kann die Umschaltung bei Ereignissen auch zu hohen Rechenzeiten und Robustheitsproblemen führen. So werden beim Initialisieren die Schrittweiten minimiert. Beim *chattering* [97, 150] springt der Solver in immer kürzer werdenden Abständen zwischen zwei gegeneinander strebenden Gleichungsstrukturen hin und her, was auch als Festrechnen bezeichnet wird. In Modelica besteht zur Vermeidung dessen ein `noEvent()`-Operator, der das Ereignis unterdrückt, wodurch jedoch bei großen h_k signifikante Fehler auftreten können. Besser ist die Verwendung des `smooth()`-Operators, der jedoch Stetigkeit vorraussetzt und demzufolge nur beschränkt einsetzbar ist. Die höher entwickelten Solveralgorithmen (vgl. 2.3.1) zeichnen sich durch eine Kombination aus Implementierung eines numerischen Integrationsverfahrens, Steuerung der Schrittweite/Ordnung sowie Ereignisbehandlung aus. Häufig sind

sie zusätzlich mit effizienzsteigernden Methoden wie der Ausnutzung dünnbesetzter Jacobimatrizen (*sparse matrix*) ausgestattet.

2.4. Gesamtfahrzeugsimulation

Wie in Kapitel 1 erläutert, behandelt die vorliegende Arbeit die Ermittlung der Energieflüsse im Fahrzeug zur Reduzierung des Kraftstoffverbrauchs. Dazu notwendig ist die Darstellung aller verbrauchsrelevanten Domänen in einem physikalischen Gesamtfahrzeugmodell [13, 125, 141]. Vor dem Kontext von Aussagegenauigkeit verglichen mit Rechenzeit und Parametrieraufwand [148], geschieht dies auf 1D- und 0D-Systemebene, vgl. Abschn.2.1. Zur Ermittlung von vergleichbarem Verbrauch und Emissionen werden die Modelle mit Fahrzyklen simuliert. Dies können offizielle Zertifizierungszyklen wie NEFZ [34] (Abb. 5.1) oder WLTC [140] sein oder individuelle Fahrprofile zum Vergleich mit Messfahrten. Es handelt sich somit um Simulationen mit Eingangssignalen $\boldsymbol{u}(t)$, in Form von Vorgaben für Geschwindigkeit $\dot{x}(t)$, Gangwahl oder Steigung $\alpha_{st}(x)$. Auch können weitere Komponenten in $\boldsymbol{u}(t)$ hinzukommen, etwa bei HiL-Simulation.

Grundlegend müsste zur Zyklussimulation lediglich der Antriebsstrang als Fahrzeugmodell verwendet werden. Zur Verbrauchsermittlung genügt dann ein 1D-Einspurmodell für die Längsdynamik. Man ermittelt aus den Fahrwiderständen an den Rädern F_R, durch Steigung F_{st}, Luftwiderstand F_L und Beschleunigung F_a den Zugkraftbedarf

$$F_{Z,B} = \underbrace{m_F g f_R \cos\alpha_{st}(x)}_{F_R} + \underbrace{m_F g \sin\alpha_{st}(x)}_{F_{st}} + \underbrace{\frac{1}{2}\rho c_W A_F \dot{x}^2}_{F_L} + \underbrace{(m_F + \frac{\theta_{red,i}}{r_{dyn}^2})\ddot{x}}_{F_a}, \tag{2.32}$$

des Fahrzeugs mit Masse m_F, Rollwiderstandsbeiwert f_R, c_W-Wert und Stirnfläche A_F, (gangabhängig) reduzierten Trägheiten $\theta_{red,i}$ und Radhalbmesser r_{dyn}. Mit Übersetzung i_A und Wirkungsgrad η_{ges} des gesamten Triebsstrangs ergibt sich der Motormomentenbedarf zu

$$M_{mot,B} = \frac{F_{Z,B} r_{dyn}}{i_A \eta_{ges}}. \tag{2.33}$$

Es muss zwischen *Vor-* und *Rückwärtssimulation* gewählt werden. Die zugehörigen Komponenten zeigt Abb. 2.8. Eine detaillierte Beschreibung der Komponenten ist in Kapitel 5 zu finden. Bei *Rückwärtssimulation* (a)) prägt die Zyklusgeschwindigkeit $\dot{x}(t)$ direkt über r_{dyn} die Raddrehzahl auf und $F_{Z,B}$ wird direkt aus $\dot{x}(t)$ berechnet. Über das Triebsstrangmodell mit Differential, Getriebe und Anfahrelementen ergeben sich daraus das erforderliche effektive Motormoment und die Drehzahl. Mit Kenntnis dieses Betriebspunkts lassen sich, in aller Regel über Kennfelder, Emissionen und der momentane Kraftstoffverbrauch ermitteln. Hierbei müssen Limitierungen des fahrbaren Bereichs nachträglich abgefangen werden. Rea-

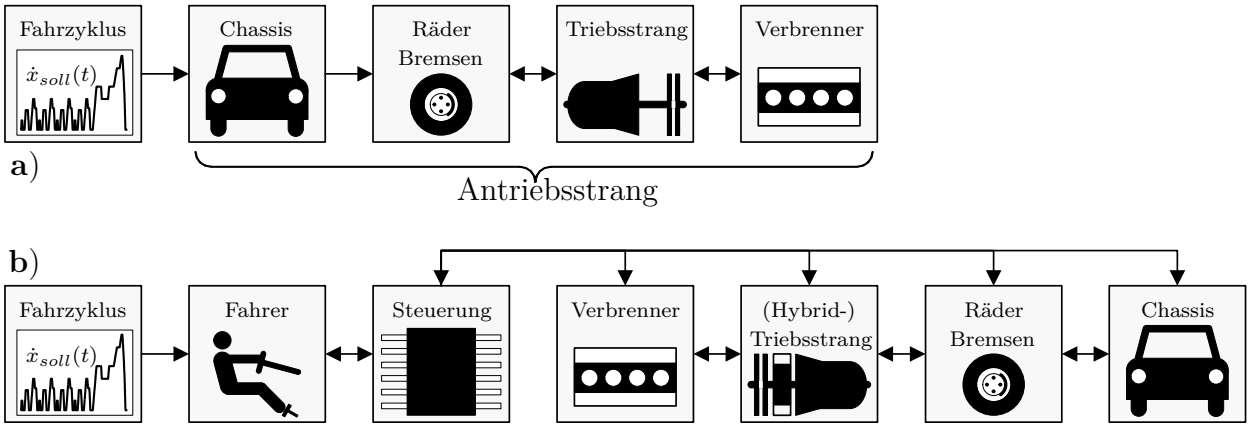


Abbildung 2.8.: Antriebsstrang: a) Rückwärts- und b) Vorwärtssimulation

litätsnäher ist die *Vorwärtssimulation* (b)): hier dient die Zyklusvorgabe $\dot{x}_{soll}(t)$ als Eingang eines Fahrermodells, das Signale für Pedalstellungen und evtl. Gangwahl als Ausgänge hat. Somit entsteht die Möglichkeit, Modelle der Motor/-Fahrzeugsteuerung einzusetzen und den Antriebsstrang hinsichtlich Elektrifizierung und Betriebsstrategien zu erweitern. Auch ist es so möglich, komplexere (Motor-)Modelle, die transiente Vorgänge abbilden, einzusetzen. Größter Vorteil der Rückwärtssimulation ist ihre kurze Rechendauer durch die quasistationäre Modellierung. Demgegenüber stehen die Einschränkungen, die bei der flexibleren Vorwärtssimulation wegfallen, wie für Entwicklung eines Energiemanagements gefordert.

Da die Triebstrangmodelle auch bei $\omega \to 0$ verwendet werden, werden Verluste über Schleppmomente $M_{V,j}$ direkt, anstelle von Wirkungsgraden η_j, modelliert, so dass sich $\eta_{ges} = 1 - (\sum P_{V,j}/P_{mot})$ ergibt. Zur Darstellung aller Energieflüsse muss die dabei entstehende Wärme mit bilanziert werden. Für eine Gesamtfahrzeugsimulation müssen also die Domänen Mechanik und Steuersignale aus Abb. 2.8 um thermische, elektrische und strömungsmechanische Modelle erweitert werden. Der allgemeine Aufbau der in dieser Arbeit erstellten Modelle ist in Abb. 2.9 dargestellt. Zu den grundlegenden Komponenten aus der Vorwärtssimulation sind die explizite Formulierung der Nebenaggregate, des thermischen Systems mit Verbrennungsmotor, Öl- und Kühlkreisläufen, des Abgasstrangs sowie der elektrischen Komponenten im Bordnetz hinzugekommen. Für die detaillierte Beschreibung der Komponenten sei wieder auf Kapitel 5 verwiesen. Vormals wurden bei Vorwärtssimulation zumeist nicht alle Domänen berücksichtigt und ein Schwerpunkt in detaillierter Modellierung eines Teils gelegt, vgl. [57, 64, 67, 148]. Alternativ wurde aufgrund der zu hohen Komplexität beim Gesamtmodell auf Rückwärtssimulation zurückgegriffen, wie zur Darstellung der Energieflüsse mit thermischen Modellen bei [54, 127, 128]. In dieser Arbeit werden alle energieflussrelevanten Domänen in Vorwärtssimulation verwendet und verschiedene Subsysteme oder Betriebsstrategien umgesetzt. Aufgrund der Voraussetzung von Echtzeitfähigkeit ($\theta_E \overset{!}{<} 1$) kommen in HiL-Simulationen bisher vereinfachte Modelle zum Einsatz [17, 96, 99]. Komplexe Gesamtfahrzeugmodelle, wie in Abb. 2.9, resultieren in hybriden DAE mit vielen Zuständen und

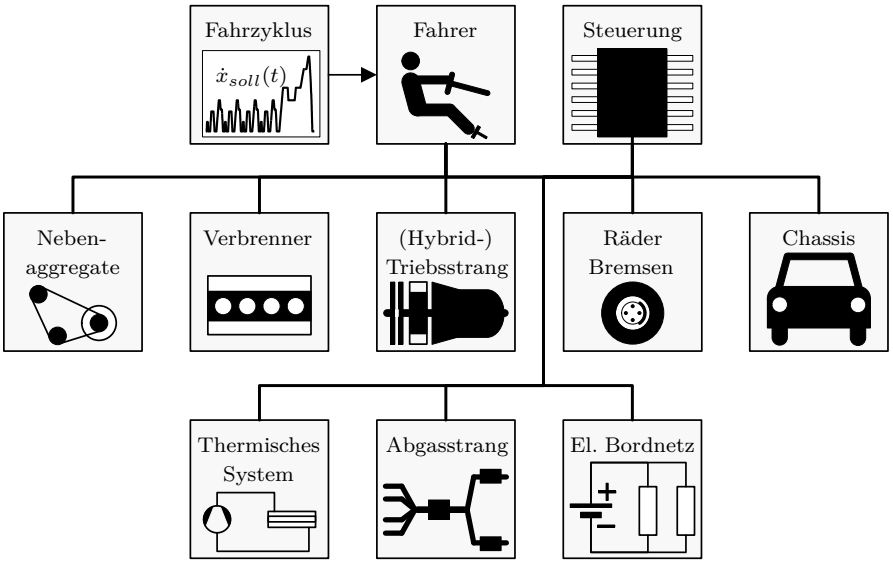


Abbildung 2.9.: Multi-Domänen-Gesamtfahrzeugsimulation

sind bisher nicht effizient genug ($\theta_E > 1$).

2.5. Verteilte Simulation

Im Gegensatz zu einer Rückwärtssimulation ohne Multi-Physik-Modell, die sehr effizient mit einem Eulerintegrator fester Schrittweite gerechnet werden kann, müssen bei den hybriden DAE implizite Solver mit variabler Schrittweite höherer Ordnung eingesetzt werden. Im Zusammenhang mit der angesprochenen Komplexität der Modelle aus unterschiedlichen physikalischen Domänen mit >>100 Zuständen, algebraischen und diskreten Variablen, führt dies zu sehr langen Simulationszeiten mit $\theta_E > 1$. Dies bedeutet für die Fahrzyklussimulation mit t_{Sim} im Bereich von 10^3 s Rechenzeiten t_{CPU} im Stundenbereich. Um den Arbeitsfluss aufrecht erhalten zu können, ist eine Rechenzeitverkürzung notwendig, wobei jede Beschleunigung der Simulation, auch schon um geringe Faktoren (<10), zielführend ist. Abgesehen von Trivialitäten, wie der Umstellung auf aktuellere Simulationswerkzeuge bzw. leistungsfähigere Rechnersysteme, gibt es folgende *Methoden zur Reduktion der Rechenzeit*:

- **Modellierungsspezifisch**: Reduktion des Ausgangsvektors $\boldsymbol{y}(t)$ um alle nicht zwingend benötigten Größen; (abschnittsweise) Linearisierung des Modells; Vermeidung von Ereignissen (s. Abschn 2.3.2); Vermeidung algebraischer Schleifen (durch geänderte Struktur oder Einführung von Trägheiten); Dynamisierung durch PT1-Verhalten beim Auslesen von Kennfeldern; generell Methoden zur *Modellvereinfachung*, darunter: Modellordnungsreduktion bei PDE-Modellen; Abstraktion von 3D zu 2D [65]; Vereinfachung der Kontaktmodellierung (z.B. bei Reibung, Verzicht auf Reifenschlupf, Bremsen nur eines Rades, etc.).

- **Simulationsspezifisch**: Wahl eines modellangepassten Solvers; Verbesserung der numerischen Methoden (z.B. Steuerung von h und p, *Block-Gauss-* und *sparse-matrix-*Methoden, *atol* und *rtol* vergrößern/an Parametergenauigkeit anpassen, etc.); vorab Kompilieren von Modell inklusive Solver (z.B. erstellt Dymola `dymosim.exe`); Verzicht auf Diagnose- und Debuggingfunktionen; verteilte Simulation.

Eine Methode, die beiden Kategorien entspricht, ist die *Inline Integration* [38], die jedoch die Einschränkung fester Schrittweite besitzt. Die Aufstellung allgemeingütiger Formalismen zur Modellvereinfachung und Verminderung der Detailtiefe ist bei Modellen aus unterschiedlichen Domänen kaum möglich.

Der Zusammenbau der Submodelle aus den unterschiedlichen Domänen führt zu einem Overhead im Berechnungsaufwand. Er lässt sich für ein Gesamtmodell aus N Submodellen mit dem Faktor

$$\theta_A = \frac{t_{CPU,ges}}{\sum_{j=1}^{N} t_{CPU,j}} \tag{2.34}$$

quantifizieren. Da die Submodelle zunächst getrennt entwickelt werden, wobei die Kopplung an das Gesamtsystem über $\boldsymbol{u}(t)$ aus Tabellen simuliert wird, lässt sich θ_A bestimmen. So ergibt sich in einem Beispiel für $N = 3$ (j = Antriebsstrang, Kühlkreislauf, Abgasstrang) ein θ_A von 8,5. Ein weiteres Beispiel (vgl. 5.2) mit nur $N = 2$, jedoch mit komplexerem thermischen System inkl. Ölkreislauf (j = Antriebsstrang, therm. System) führt zu $\theta_A = 12,1$. Den größten Anteil an der Erzeugung des Overheads hat das Entstehen eines steifen Gesamtsystems durch den Zusammenbau, so haben auch die enstprechenden Vergleiche der Funktionsauswertungen $\theta_{F,j}$ bzw. Ereignissen $\theta_{Ev,j}$

$$\theta_F = \frac{n_{F,ges}}{\sum_{j=1}^{N} n_{F,,j}} \qquad \theta_{Ev} = \frac{n_{Ev,ges}}{\sum_{j=1}^{N} n_{Ev,,j}} \tag{2.35}$$

geringere Werte, vgl. Tabelle 4.1.

Beispielhaft sind das dynamische M_{Mot} und das sehr träge $T_{\text{Ö}lsumpf}$ zwei Komponenten von $\boldsymbol{z}(t)$, deren Lösung jeweils nur sehr kurze (M_{Mot}) bzw. längere ($T_{\text{Ö}lsumpf}$) Schrittweiten erlauben würde. Da der Solver die Lösung für alle Komponenten in $\boldsymbol{z}(t)$ berechnen muss, passt sich die Schrittweite dem momentan größten Eigenwert im System an, und für alle anderen Komponenten werden zu kleine Schritte gewählt. Dies fällt insbesondere beim Lösen der algebraischen Gleichungen bei Medienmodellen der Fluide ins Gewicht. Zusätzlich besteht die Gefahr von Auslöschungsfehlern in den trägen Teilsystemen. Weiterhin löst jedes Ereignis in einem Teilsystem das Initialisieren des gesamten Modells aus.

Die hohen Werte von θ_A weisen somit Potential durch modulare, d.h. verteilte Simulation aus. Zum Einen lässt sich dieses Potential durch Aufteilen in Teilsysteme unterschiedlicher Dynamik und Integrieren mit unterschiedlich gewählter Schrittweite oder Ordnung, *Multirate* bzw. Integrationsverfahren, *Multimethod*, schöpfen [40, 41, 56, 147]. Zum Anderen bewirkt

die Möglichkeit, diese Teilsysteme -etwa auf mehreren Rechnerkernen, *Multicore-* parallel berechnen zu können, eine weitere Effizienzsteigerung [45, 49, 92], siehe auch Abschnitte 3.2.1 bzw. 3.3. Aufgrund ihres Potentials, vgl. Kap. 4, und der allgemeinen Anwendbarkeit wird die modulare Simulation in dieser Arbeit als Mittel zur Rechenzeitverkürzung gewählt.

2.5.1. Modellpartitionierung

Ausgehend von einem vorliegenden Gesamtsystemmodell muss für die verteilte Simulation eine zielführende Partitionierung in N Teilmodelle gewählt werden. Dies kann nach verschiedenen Gesichtspunkten geschehen:

- *numerisch*: Anhand der Dynamik der Komponenten Einteilung in schnell/langsam bzw. steifes/nicht-steifes System; anhand der Kopplungsstärke zwischen Modellzuständen.
- *informationstechnisch*: Anhand der Granularität, d.h. dem Verhältnis von Berechnungs- zu Kommunikationsaufwand bei Software-Architektur, wobei grobgranulare Probleme gut parallelisierbar sind; anhand möglichst paritätischer Auslastung der Prozessoren.
- *heuristisch*: Anhand des physikalischen Hintergrunds bzw. der technischen Domänen; Anhand von Erfahrungswerten.

Je nach Gesichtspunkt der Partitionierung gibt es unterschiedliche Möglichkeiten zu deren Umsetzung. Es ist häufig schwierig, die Steifigkeit eines Systems vorab zu bestimmen. Jedoch gibt es bereits länger Umsetzungen zur automatischen Detektion und entsprechender, teilweise auch mit der Zeit variierender Partitionierung [40, 41, 43, 111]. Für eine Umsetzung variierender Partitionierung muss jedoch direkt in den Solveralgorithmus eingegriffen werden können. [113] schlägt eine Umsetzung mit Einteilung in ein schnelles und langsames Teilsystem anhand eines Grenzwerts für $Im(\lambda)$ vor, die in Modelica mithilfe der *Inline Integration* für verschiedene Schrittweiten gelöst werden. Eine weitere Möglichkeit der Partitionierung im Modelica-Kontext wäre die TLM [4] -Methode [74], wobei jede Komponente mit eigenem Integrator gerechnet würde und somit der Vorteil der Gleichungsvorverarbeitung hinfällig wird. Neben den beiden Vorgenannten gibt es für Modelica noch keine bestehenden Umsetzungen [101]. Da die Modellteile mechatronischer Gesamtsysteme häufig bereits nach Domänen getrennt entwickelt werden, kann sich eine Partitionierung auch aus solchen praktischen Gesichtspunkten ergeben. Weiterhin lässt sich eine Auswahl experimentell anhand des θ_A-Faktors treffen, was auch automatisiert erfolgen kann. Sobald die Schnittstellen der Partitionierung bestimmt sind, müssen die Teilmodelle dem jeweiligen Integrator zugeordnet werden. Ist kein Simulationswerkzeug vorhanden, welches dies unterstützt, so muss ein eigener Algorithmus entwickelt werden. Für Modelica gibt es den Ansatz, den Sprachstandard für eine Partitionierung in *Subtasks* zu erweitern [39], jedoch nur mit sehr einfachen Integratoren. Eine weitere Möglichkeit, ist das Lösen der partitionierten Modelle über Co-Simulation.

[4] Transmission Line Modelling

2.5.2. Steifigkeit und Kopplung

Im Falle einer automatischen Modellpartitionierung sind die numerischen Kriterien Steifigkeit und Kopplung notwendig. Auch zur Einschätzung einer manuell getroffenen Teilung sind sie wichtig. Je steifer ein System, desto mehr Effizienz ist von verteilter Simulation zu erwarten. Je stärker die Kopplung zwischen den Teilsystemen, desto größer der Fehler an den Koppelgrößen, die zur Abbildung des Gesamtsystemverhaltens bei Partitionierung ausgetauscht werden müssen. Für den Begriff der *Steifigkeit* ist in der Literatur jedoch keine eindeutige Definition zu finden. So ist laut [76] eine Definition wie in (2.9) zwar ein starker Hinweis, jedoch zu kurz greifend, da Steifigkeit zusätzlich von den Anfangswerten und der Diskretisierung der numerischen Verfahren abhängt [62, 111]. In dieser Folge schlägt [76] eine Quantifizierung von Steifigkeit in Abhängigkeit des Stabilitätsgebiets bei explizitem Eulerverfahren vor. Die Stärke der *Kopplung* zwischen zwei Teilsystemen A und B lässt sich mit

$$K_{(AB)} = K_{(BA)} = |\boldsymbol{\kappa}_{AB}|\,|\boldsymbol{\kappa}_{BA}| = \sum_i \frac{\partial \boldsymbol{f}_B}{\partial z_{A,i}} \frac{dz_{A,i}}{dt} \sum_i \frac{\partial \boldsymbol{f}_A}{\partial z_{B,i}} \frac{dz_{B,i}}{dt} \tag{2.36}$$

ausdrücken, wobei über $\boldsymbol{\kappa}_{AB}$ und $\boldsymbol{\kappa}_{BA}$ Dynamik und Richtung deutlich werden und $K_{(AB)}$ vergleichenden Charakter der Rückkopplungen besitzt. Jedoch ist im Allgemeinen die Jacobimatrix nicht bekannt. [27, 76] verwenden zur Bestimmung von Kopplung deswegen eine Sensitivitätsanalyse vorher bestimmter Teilsysteme. Um die Stabilität zu erhöhen und den Kommunikationsaufwand niedrig halten zu können, ist eine schwache Kopplung anzustreben. In [28] wird ein Vorschlag für die Kennzeichnung (`weak`) schwacher Koppelvariablen in Modelica für eine Partitionierung vorgeschlagen. Liegt ein steifes Systemverhalten vor, so ist folglich eine numerisch grobgranulare Partitionierung mit schwacher Kopplung zielführend.

3. Co-Simulation von Multi-Physik-Modellen

Kapitel 3 bereitet die Basis für die in dieser Arbeit angewandten und entwickelten Methoden. Zunächst wird in Abschnitt 3.1 der Begriff der Co-Simulation abgegrenzt und in 3.2 werden die numerischen Aspekte der Co-Simulation zusammengetragen sowie die zugehörige Nomenklatur eingeführt. Abschnitt 3.3 führt Bewertungskriterien ein, die im Folgenden Anwendung finden.

Es sind sehr unterschiedliche Hintergünde, die zur Entwicklung von Verfahren zur Co-Simulation und deren Anwendung führen: Häufig werden Komponenten mechatronischer Systeme in jeweils domänenspezialisierten Werkzeugen simuliert. Will man nun das Systemverhalten untersuchen, so müssen die unterschiedlichen Werkzeuge gekoppelt werden [85]. Für eine Kopplung von DAE- mit PDE-Modellen ist dies ebenso der einzige Weg. Werden Teilmodelle nicht nur in unterschiedlichen Werkzeugen, sondern von Experten unterschiedlicher Firmen entwickelt, so können verschlüsselte Modelle unter Wahrung von Geheimhaltung ausgetauscht und in einer Co-Simulation berechnet werden [60]. Bei MiL, SiL und HiL [1] werden die Modelle der Regelstrecke durch Co-Simulation mit dem Regler(-Modell) verbunden [1, 17, 18]. Gemäß Abschnitt 2.5 kann Co-Simulation auch zur Verkürzung der Rechendauer eines Gesamtmodells durch Partitionierung angewandt werden, zum Einen durch Trennung von Zeitkonstanten [40, 69, 147], zum Anderen durch Parallelisierung [50].

3.1. Terminologie

Aufgrund der unterschiedlichen Motivationen gibt es nach wie vor keine einheitliche Begrifflichkeit zur Kopplung von Simulationsmodellen, und es werden gleiche Begriffe für unterschiedliche Methoden oder mehrere Begriffe für gleiche Methoden verwendet. Zur Abgrenzung der Terminologie soll hier der in [36] vorgestellte und von [53] zu diesem Zweck aufgegriffene Vorschlag herangezogen werden. Dessen Einteilung ist in angepasster Form in Tabelle 3.1 dargestellt und definiert vier Quadranten: **Ia** bezeichnet die gebräuchlichste Variante, *monolithische Simulation*. Sie kann als Standardanwendung verstanden werden und

[1] Model/ Software/ Hardware in the Loop

Tabelle 3.1.: Matrix der Simulationsvarianten in Anlehnung an [36]

		Anzahl der Integratoren	
		$= 1$ geschlossene Simulation	> 1 verteilte Simulation
Anzahl der Modellierungswerkzeuge	$=1$ geschlossene Modellierung	**Ia** monolithische Simulation	**IIa** partitionierte Simulation *Co-Simulation*
	>1 verteilte Modellierung	**Ib** Modellkopplung	**IIb** Werkzeugkopplung *Co-Simulation*

stellt das Gegenstück zur gekoppelten Simulation dar, da das gesamte System sowohl geschlossen modelliert als auch geschlossen simuliert wird. Werden dagegen Subsystemmodelle getrennt erstellt und für eine geschlossene Simulation zusammengeführt, geschieht dies durch *Modellkopplung* (**Ib**). Wird hierfür ein Modell in die Syntax des anderen überführt, spricht man auch von *Modelltransformation*.

Bei Variante **IIa**, *partitionierte Simulation* (häufig: *modulare* oder *verteilte Simulation*), wird das Gesamtsystem zwar in einer Umgebung erstellt, soll jedoch getrennt simuliert werden, vgl. Abschnitt 2.5. Dies geschieht im klassischen Fall von *Multirate, -order* und *-method* oder *parallelisierter Simulation* durch *Modellpartitionierung*. Werden die Subsystemmodelle bereits getrennt erstellt und sollen aus diesem Grund mit den Integratoren ihrer jeweiligen Werkzeuge simuliert werden, so müssen letztere gekoppelt werden: *Werkzeugkopplung* **IIb**. Auch die Begriffe *Tool-, Simulator-* und *Integratorkopplung* oder *hybride Simulation* sind für Variante **IIb** gebräuchlich.

Vor dem Hintergrund der Entwicklung hin zur Multi-Physik-Modellierung mit Sprachstandardisierung und monolithischer Simulation besteht vermehrt die Konkurrenz zwischen den Varianten **Ia** und **IIb** [63, 134]. Dem gegenüber steht jedoch die zunehmende Öffnung der Simulationswerkzeuge für externe Modelle, die auch in der verstärkten Entwicklung einheitlicher Schnittstellen, wie FMI [48] mündet. Dies erleichtert die Umsetzung von **IIa** und **IIb**.

Der Begriff *Co-Simulation* wird zumeist für Variante **IIb** verwendet. Jedoch sind die notwendigen numerischen Koppelmethoden der getrennten Integration geschuldet und damit bei **IIa** und **b** gleichermaßen anzuwenden. Somit wird in dieser Arbeit unter *Co-Simulation* die verteilte Simulation im Allgemeinen verstanden. Die weiterführende Terminologie wird im folgenden Abschnitt beschrieben.

3.2. Numerische Aspekte der Co-Simulation

Eine Co-Simulation (CS) sei durch die Kopplung von $N \geq 2$ Subsystemen -bestehend aus Teilmodell mit eigenem Integrator- durch je mindestens eine Koppelgröße bestimmt. Wie in Abb. 3.1 dargestellt, übernimmt dabei eine übergeordnete Software die Funktion der CS-Umgebung, die die Subsysteme verbindet, steuert und zusätzliche Koppelalgorithmen beinhalten kann. Die Subsysteme besitzen diskrete Ein- und Ausgänge, $\bar{\boldsymbol{u}}_j \in \mathbb{R}^{n_u}$, $\bar{\boldsymbol{y}}_j \in \mathbb{R}^{n_y}$, deren Werte

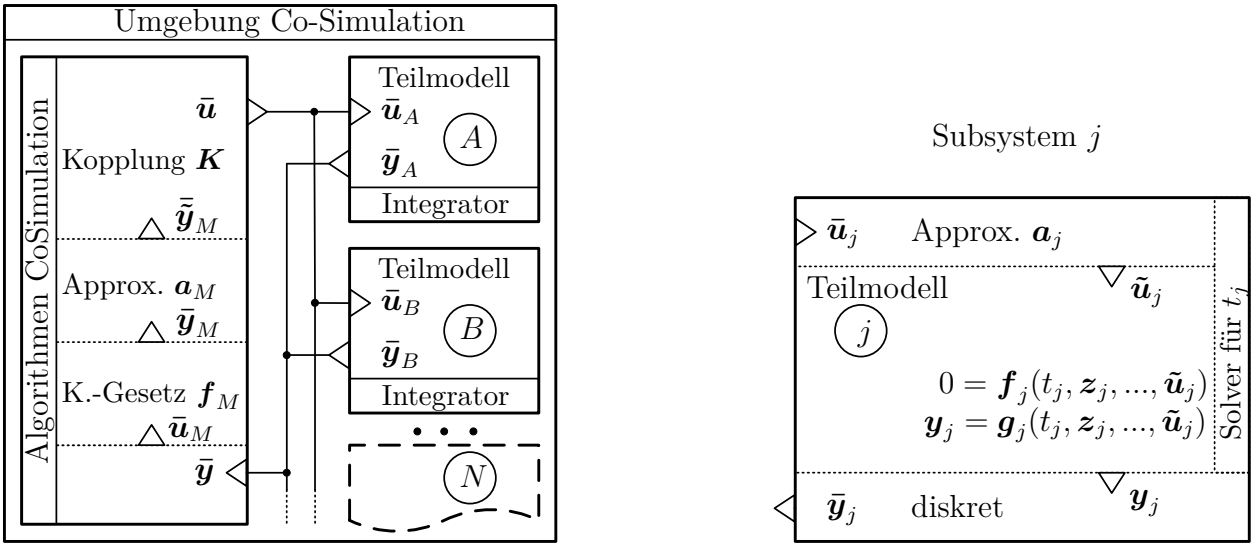


Abbildung 3.1.: Aufbau der Co-Simulation mit Koppelgrößen und Teilsystemaufbau

sich nur zu Koppelzeitpunkten T_l ändern können. Sie lassen sich zu den Gesamtvektoren

$$\bar{\boldsymbol{y}} = \bar{\boldsymbol{y}}_{ges} = \begin{pmatrix} \bar{\boldsymbol{y}}_A \\ \cdots \\ \bar{\boldsymbol{y}}_N \end{pmatrix} \in \mathbb{R}^{m_y} \qquad \bar{\boldsymbol{u}} = \bar{\boldsymbol{u}}_{ges} = \begin{pmatrix} \bar{\boldsymbol{u}}_A \\ \cdots \\ \bar{\boldsymbol{u}}_N \end{pmatrix} \in \mathbb{R}^{m_u} \tag{3.1}$$

der Dimensionen

$$m_y = \sum_{j=A}^{N} n_{y,j} \qquad m_u = \sum_{j=A}^{N} n_{u,j} \tag{3.2}$$

zusammenfassen. Dabei sollte $m_u \geq m_y$ gelten. Gemäß des häufig verwendeten Master-Slave-Schemas kann der CS-Algorithmus (Index M) als Subsystem ohne eigene Zustände und $\bar{\boldsymbol{u}}_M = \bar{\boldsymbol{y}}$ betrachtet werden, [49]. In Abhängigkeit der Kausalität von $\bar{\boldsymbol{u}}$ und $\bar{\boldsymbol{y}}$ können Koppelgesetze $\boldsymbol{f}_M$ angewandt werden, deren linke Seite $\bar{\boldsymbol{y}}_M$ mit der Vorschrift $\boldsymbol{a}_M$ durch Iteration oder Extrapolation zu $\bar{\tilde{\boldsymbol{y}}}_M$ approximiert werden kann. Die Verknüpfung der Subsysteme erfolgt schließlich über die *Koppel-* oder *Inzidenzmatrix* $\boldsymbol{K}^{m_u \times m_y}$

$$\bar{\boldsymbol{u}} = \boldsymbol{K}\bar{\tilde{\boldsymbol{y}}}_M; \quad \boldsymbol{K} \in \{0, 1\} \tag{3.3}$$

Je nach Synchronisierung der Subsysteme ist $\boldsymbol{K}$ konstant oder es gilt $\boldsymbol{K} = \boldsymbol{K}(t_A, ..., t_N)$. Der Aufbau der Subsysteme kann eine zeitkontinuierlichen Approximation $\boldsymbol{a}_j$ der diskreten $\bar{\boldsymbol{u}}_j$ zu $\tilde{\boldsymbol{u}}_j$ beinhalten, die innerhalb des (hybriden) DAE- oder ODE-Modells $\boldsymbol{f}_j$ berücksichtigt wird. Die kontinuierlichen Ausgänge $\boldsymbol{y}_j$ werden zu den diskreten Koppelgrößen $\bar{\boldsymbol{y}}_j$. Insgesamt ergibt sich daraus das Gesamtsystem, bestehend aus den N $\boldsymbol{f}_j, \boldsymbol{a}_j$ und $\boldsymbol{g}_j$, sowie $\boldsymbol{f}_M$ und $\boldsymbol{K}\boldsymbol{a}_M(\bar{\bar{\boldsymbol{y}}}_M)$.

Der Austausch der Koppelgrößen $\bar{\boldsymbol{u}}, \bar{\boldsymbol{y}}$ erfolgt zu diskreten Koppelzeiten. Dies geschieht durch Diskretisierung des Simulationsintervalls von T_0 bis zur Endzeit T_M in M Schritte ΔT_l

$$\Delta T_l = T_{l+1} - T_l = H_l, \quad l = 0, ..., M-1, \quad T_l \in [T_0, T_M] \tag{3.4}$$

H_l wird als *Makroschrittweite* bezeichnet. Für Start- und Endzeitpunkt der Co-Simulation müssen $T_0 = t_{j,0} = t_0$ bzw. $T_M = t_{j,S} = t_S$ gelten. Es soll eine Einteilung aller numerischen Aspekte in diese Kategorien erfolgen:

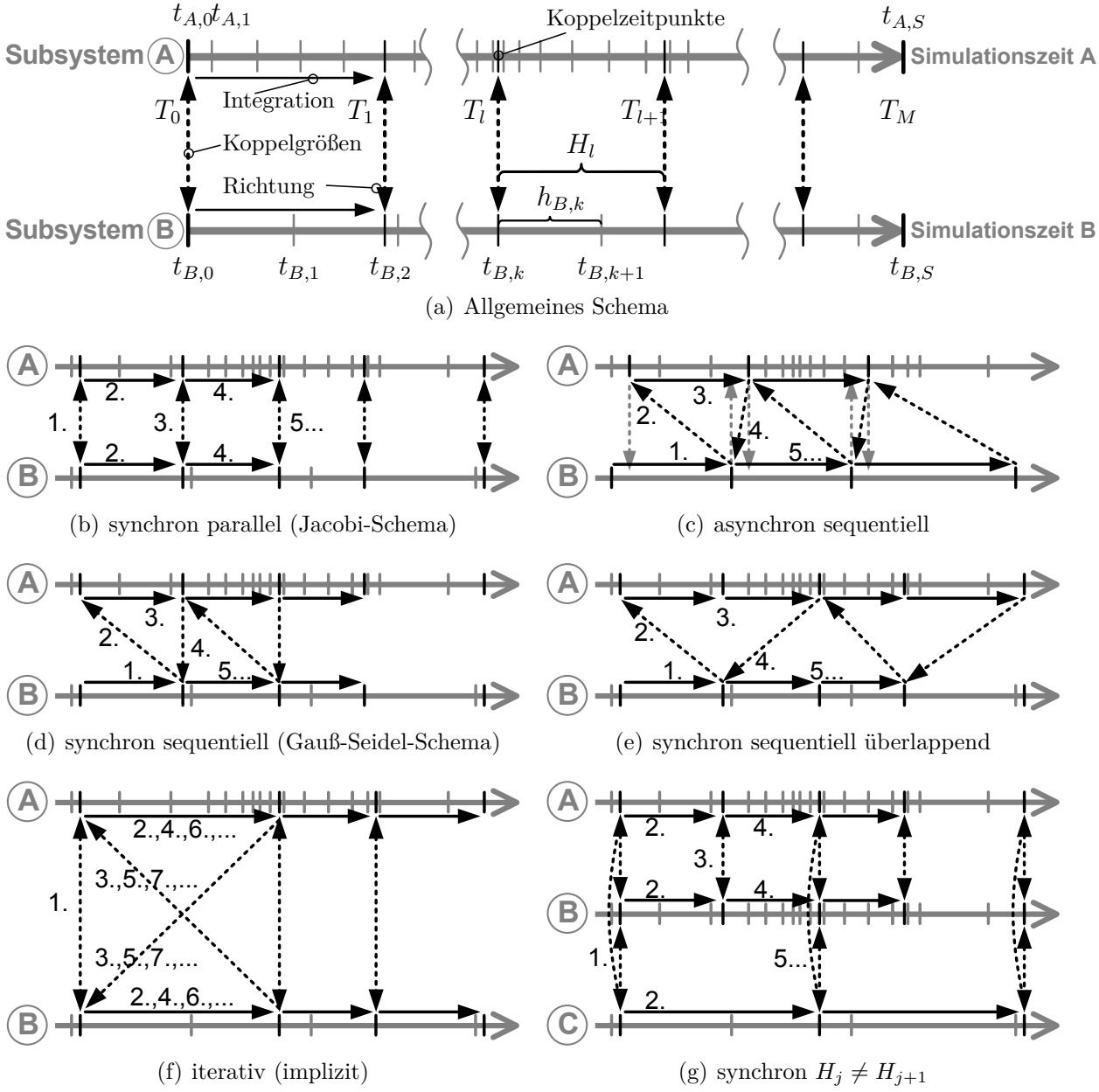
- **Synchronisierung**: Kommunikationsstrategie, die den Ablauf der CS steuert und Zusammenhang zwischen T und t_j herstellt.
- **Kausalität**: Umsetzung physikalischer Schnittstellen bei der CS. Der Vorteil der gleichungsbasierten Modellierung geht durch das Einführen gerichteter $\boldsymbol{u}$ und $\boldsymbol{y}$ verloren.
- **Approximation**: Zusätzlicher Fehler durch die Diskretisierung in ΔT_l. Approximation des Verlaufs in den Subsystemen durch verschiedene Methoden, $\boldsymbol{a}_M, \boldsymbol{a}_j$.
- **Makroschrittweiten**: Wahl der Werte für H. Wie bei den lokalen $h_{k,j}$ muss die globale Makroschrittweite als Kompromiss aus Genauigkeit und Effizienz gewählt werden.
- **Behandlung von Unstetigkeiten**: Analog zur monolithischen Simulation können innerhalb der Subsysteme oder in $\boldsymbol{y}_j$ Unstetigkeiten auftreten.

Die in den folgenden Abschnitten beschriebenen Umsetzungsvarianten der einzelnen Kategorien stellen eine erweiterte Zusammenstellung als Basis für den in Kapitel 4 vorgestellten Beitrag zu deren Erweiterung dar. Die Varianten sind zwar im Gesamtkontext nicht beliebig kombinierbar, jedoch können die Kategorien gesondert behandelt werden.

3.2.1. Synchronisierung

Die CS-Umgebung (vgl. Abb. 3.1), die auch innerhalb eines der beteiligten Simulationswerkzeuge implementiert sein kann, übernimmt das Aufrufen und Beenden der Subsystemprozesse. Weiterhin sorgt sie für deren Initialisierung, sowie Verbindung und Ablaufsteuerung. Für die Art dieser Synchronisierung gibt es unterschiedliche Strategien. So kann abhängig vom Gesamtsystem eine sequentielle Abarbeitung gefordert oder eine Parallele möglich sein. Stets müssen jedoch die Subsysteme an den Koppelzeitpunkten T_l zur Ermittlung von $\bar{\boldsymbol{y}}_j$

angesprochen werden. In Abb. 3.2 sind die Varianten der Synchronisation dargestellt, wobei **a)** das allgemeine Schema im Gesamtintervall $[T_0, T_M]$ anhand zweier Subsysteme A und B zeigt. Auch wird deutlich, dass für das Erreichen der T_l teilweise zusätzliche Mikroschritte $h_{j,k}$ notwendig werden, was mit Zusatzaufwand verbunden ist. Variante **b)** (u.A.



(a) Allgemeines Schema

(b) synchron parallel (Jacobi-Schema)

(c) asynchron sequentiell

(d) synchron sequentiell (Gauß-Seidel-Schema)

(e) synchron sequentiell überlappend

(f) iterativ (implizit)

(g) synchron $H_j \neq H_{j+1}$

Abbildung 3.2.: Strategien der Synchronisierung

[25, 51]) zeigt das Schema *synchroner, paralleler* Kopplung, auch Jacobi-Schema [25]. Hierbei gilt an den T_l jeweils $T = t_A = t_B$ und $\boldsymbol{K} = konst.$, da stets alle Größen ausgetauscht werden, bevor alle Subsysteme parallel weiter integriert werden. Die Eingänge $\tilde{\boldsymbol{u}}_j$ können somit nur durch Extrapolation approximiert werden. Hingegen ist bei Variante **c)** (u.A. [105]) die Kopplung *asynchron* und *sequentiell*, d.h. die Makroschritte der Subsysteme werden nacheinander ausgeführt. Hierbei entfällt die Wahl synchroner Koppelzeitpunkte und

zusätzlicher $h_{j,k}$. Stattdessen wird der letzte vom Integrator gewählte Wert $t_{j,k}$ propagiert, um den nächsten Makroschritt $H_{j+1,l}$ zu begrenzen. $\tilde{\boldsymbol{u}}_j$ wird jeweils zunächst interpoliert, dann extrapoliert. Ebenfalls *sequentiell*, jedoch *synchron* ist Variante **d)** (u.A. [14, 45]), auch Gauß-Seidel-Schema [25]. Da hierbei im zuerst ausgeführten Subsystem $\tilde{\boldsymbol{u}}_B$ extrapoliert werden muss, dessen Ausgänge $\bar{\boldsymbol{y}}_B$ dann in den restlichen $\tilde{\boldsymbol{u}}_j$ interpoliert werden können, ist die Wahl der Reihenfolge über $\boldsymbol{K} = \boldsymbol{K}_j$ wichtig [7, 147]. Diese Wahl entfällt bei Variante **e)**, bei der *sequentiell überlappend* jeweils über zwei *synchrone H* gerechnet wird. Als iterative Ausprägung von **e)** ist auch das wiederholte Berechnen eines Makroschritts in einem der Subsysteme möglich [32]. Werden die Makroschritte in allen Subsystemen wiederholt, bis ein Konvergenzkriterium erreicht wird, so handelt es sich um *iterative* bzw. *implizite* Kopplung -häufig auch *waveform relaxation-* , wie in Variante **f)** (u.A. [7, 71, 79]) dargestellt. Variante **g)** (u.A. [83, 137]) stellt eine synchrone Kopplung mit $N > 2$ Subsystemen dar, wobei $H_j \neq H_{j+1}$ gelten kann, sequentiell wie parallel.

Der Vorteil paralleler Schemata ist ihre höhere Effizienz und die Möglichkeit, auf mehreren Rechnern zu arbeiten. Dem gegenüber stehen die geringere Stabilität und Genauigkeit bei den Extrapolationsmethoden. Bei sequentiellen Schemata kommen dagegen auch die genaueren Interpolationsmethoden zum Einsatz, jedoch bei langsamerem Rechenfortschritt. Die größte Stabilität lässt sich beim iterativen Schema erreichen, was jedoch durch einen erheblich höheren Berechnungsaufwand erkauft werden muss. Zusätzlich zu den dargestellten Varianten seien parallel überlappende Schemata [115] oder ereignisbasierte Ablaufsteuerungen [68] genannt. Bei Umsetzung einer Master-Slave-Strategie wird der CS-Algorithmus auch bei paralleler Berechnung der N Subsysteme im Wechsel mit diesen, also sequentiell, aufgerufen.

3.2.2. Kausalität

Gemäß Abbildung 3.1 sind die Teilmodelle über gerichtete Ein- und Ausgangsgrößen ($\bar{\boldsymbol{u}}_j$, $\bar{\boldsymbol{y}}_j$) verbunden. Dies entspricht einer signalflussbasierten, *kausalen* Modellierung und die Subsysteme können als Übertragungsblöcke (vgl. (2.1)), die Koppelgrößen als eine Art diskreter Signale betrachtet werden. Dies ist bei Co-Simulation einer Steuereinheit mit zugehöriger Strecke durch Sensor- bzw. Aktorsignale auch entsprechend gegeben. Der Vorteil gleichungsbasierter, physikalischer Modellierung jedoch ist der *akausale*, bidirektionale Charakter ihrer Verbindungen. So können in Modelica die Konnektorgleichungen (2.11) gebildet, die Modellgleichungen vorverarbeitet und die Lösung des Systems geschlossen bestimmt werden (vgl. Abschnitt 2.2.1). Dieser Vorteil geht bei Partitionierung eines physikalischen Modells verloren: die Konnektoren müssen an den Teilmodellgrenzen aufgebrochen werden, um Kausalität zu erzeugen. Wichtig ist hierbei die Wahl der gerichteten Schnittgrößen ($\boldsymbol{u}_j$, $\boldsymbol{y}_j$), da sie Einfluss auf Stabilität und Genauigkeit der CS hat.

Allgemein in Fluss- und Potentialgrößen formuliert, ergeben sich die drei in Abb. 3.3 anhand

zweier Teilmodelle dargestellten Möglichkeiten, physikalische Schnittstellen zu kausalisieren.

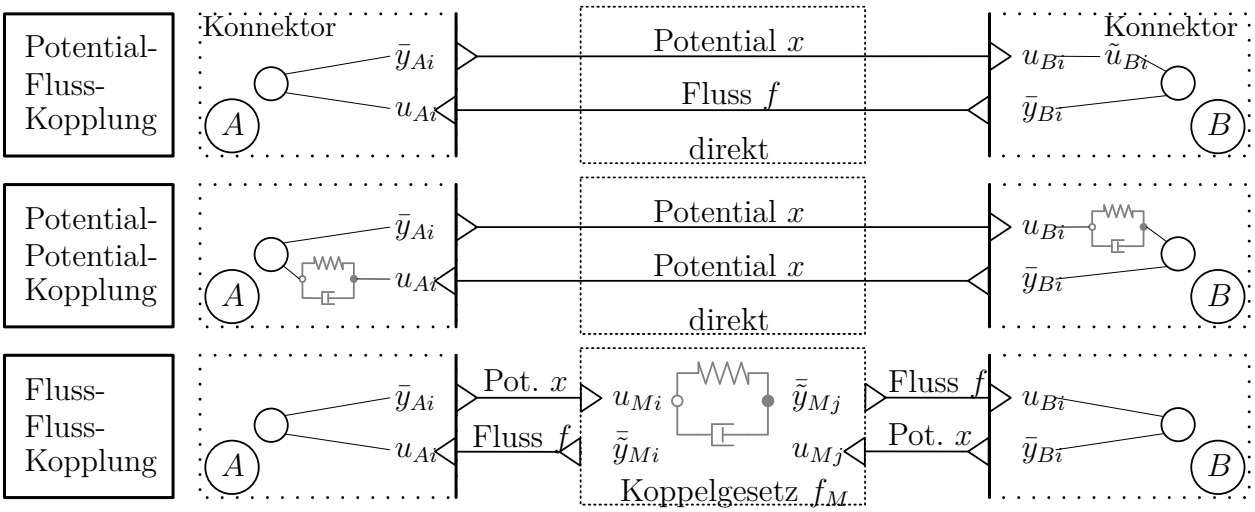


Abbildung 3.3.: Möglichkeiten physikalischer Schnittgrößen

- **Fluss-Potential- /Potential-Fluss- Kopplung** $(x-f)$: eines der Subsysteme sendet nur die Potentialgröße x_A, das andere die Flussgröße f_B. Es handelt sich um eine direkte Signalverbindung ohne zusätzliches Koppelgesetz. Eine solche Kopplung findet bisher die häufigste Anwendung, u.A. [24, 28, 83, 85, 115, 130, 134].
- **Potential-Potential- Kopplung** $(x-x)$: beide Subsysteme senden direkt Signale der Potentialgröße x_A, x_B. Hierbei muss die Flussgröße über ein jeweils identisches Potential-Speicher-Glied (im Beispiel ein Feder-Dämpfer-Modell) berechnet werden, u.A. [25, 49, 91].
- **Fluss-Fluss- Kopplung** $(f-f)$: wieder senden die Subsysteme die Potentialgrößen x_A, x_B, jedoch nicht direkt. Über ein zusätzliches Koppelgesetz f_M in einem CS-Master werden daraus die Flussgrößen $f = f_A = -f_B$ ermittelt, evtl. extrapoliert und weitergesendet. Somit kann das Potential-Speicher-Glied in den Teilmodellen entfallen, [51].

Bei Fluss-Potential-Kopplung hat die Richtung, d.h. x_A mit f_B oder x_B mit f_A großen Einfluss auf die Stabilität der CS [28]. Werden etwa im Fall mechanischer Kopplung zwei Massen $m_A >> m_B$ gekoppelt, so sollte A die Potentialgröße, die den Zustand von m_A darstellt, vorgeben, da m_B nur geringen Einfluss hat.

Bei der Wahl der Koppelsignale ist darauf zu achten, ob eine *algebraische Schleife* gebildet wird. Dies ist der Fall wenn die Ausgänge zweier gekoppelter Subsysteme jeweils explizit voneinander abhängen, d.h. $y_A = g_A(u_A = y_B) \wedge y_B = g_B(u_B = y_A)$. Eine algebraische Schleife kann in einer CS getrennt nicht gelöst werden, Null-Stabilität ($H \longrightarrow 0$) ist nicht mehr gegeben, und es müssen bestimmte Maßnahmen ergriffen werden, [80]. Im Falle einer

Potential-Fluss-Kopplung kann zum Einen ein iterativer Ansatz gemäß Abb. 3.2(f) gewählt werden, wobei die g_j oder der ganze Makroschritt H_l iteriert werden [31, 79, 110]. Zum Anderen kann einer der Eingänge gefiltert werden. Durch Wahl einer Potential-Potential- oder Fluss-Fluss- Kopplung kann die algebraische Schleife durch das eingeführte Koppelglied vermieden werden [49].

Die Wahl der Kopplung muss für jede physikalische Domäne gesondert betrachtet werden. So lassen sich in jeder Domäne gemäß Tabelle 2.3 Potential- und Flussvariablen definieren. In Abhängigkeit von Domäne und angebundener Komponente können jedoch nicht immer alle Größen bestimmt werden [49, 50]. Oder es werden weitere Größen benötigt, wie die Stream-Variablen, Tabelle 2.3. Zur Kopplung kann auch der differentielle Zustand als Potentialgröße herangezogen werden, vgl. Abschnitt 4.3.

3.2.3. Approximation

Bei der Co-Simulation wird in Makroschritte diskretisiert. Für diese gilt immer $H_l \geq h_{j,k}$, d.h. es wird ein zusätzlicher Diskretisierungsfehler $\bar{\boldsymbol{\tau}}$ eingebracht, für den ähnlich dem lokalen Fehler (2.22) am Ende eines Makroschrittes je Komponente in j gilt:

$$\bar{\boldsymbol{\tau}}_{\boldsymbol{j}} = \bar{\boldsymbol{y}}_{j,l} - \boldsymbol{y}_j(T_{l+1}) \quad \text{mit } \bar{\boldsymbol{y}}_{j,l} := \boldsymbol{y}_j(T_l). \tag{3.5}$$

Analog zur Approximation der exakten Lösung durch ein Integrationsverfahren, lässt sich der außerhalb des Subsystems j unbekannte, kontinuierliche Verlauf von $y_j(T)$ in $T_l \mapsto T_{l+1}$ für die anderen Subsysteme approximieren. Im Falle einer direkten Kopplung gilt dann für die Eingänge:

$$\tilde{\boldsymbol{u}} \approx \boldsymbol{K}\boldsymbol{y}. \tag{3.6}$$

Darin sind alle kontinuierlichen Aus- und Eingänge innerhalb der Subsysteme zusammengefasst zu

$$\boldsymbol{y} = \boldsymbol{y}_{ges} = \begin{pmatrix} \boldsymbol{y}_A \\ \cdots \\ \boldsymbol{y}_N \end{pmatrix} \in \mathbb{R}^{m_y} \qquad \tilde{\boldsymbol{u}} = \tilde{\boldsymbol{u}}_{ges} = \begin{pmatrix} \tilde{\boldsymbol{u}}_A \\ \cdots \\ \tilde{\boldsymbol{u}}_N \end{pmatrix} \in \mathbb{R}^{m_u}. \tag{3.7}$$

Von der Synchronisierung abhängig sind verschiedene Approximationsmethoden einsetzbar: Sind, wie bei sequentieller Kopplung, Werte anderer Subsysteme zum Zeitpunkt T_{l+1} bekannt, so können die Eingänge *interpoliert* werden

$$\tilde{\boldsymbol{u}}_{\boldsymbol{j}} = \boldsymbol{a}_j(t_j, \bar{\boldsymbol{u}}_{j,l}, \bar{\boldsymbol{u}}_{j,l+1}, \boldsymbol{p}, ...) \tag{3.8}$$

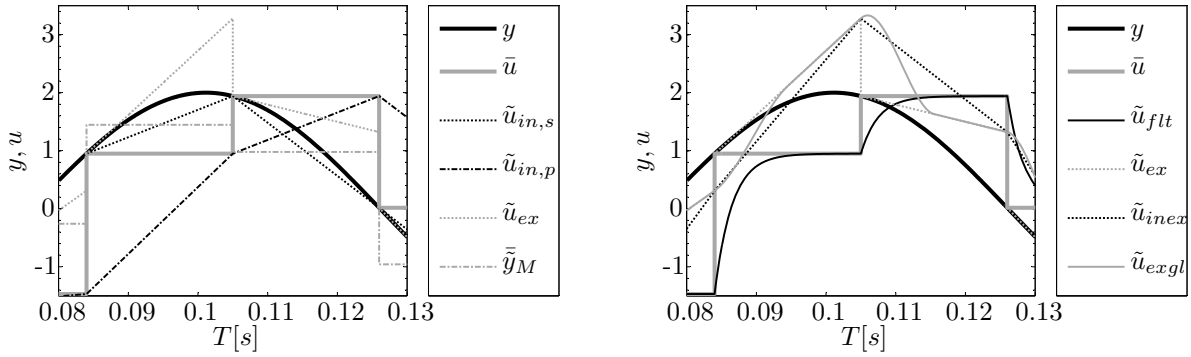
Sind alle Werte bei T_{l+1}, wie im Fall paralleler Kopplung, unbekannt, so muss der Verlauf *extrapoliert* werden:

$$\tilde{\boldsymbol{u}}_{\boldsymbol{j}} = \boldsymbol{a}_j(t_j, \bar{\boldsymbol{u}}_{j,l}, \boldsymbol{p}, ...). \tag{3.9}$$

Im Gegensatz zu den unter 2.3.1 erwähnten Extrapolationsverfahren, wird hierbei nicht ein einzelner lokaler Wert, sondern die für alle $h_{j,k}$ während H_l bestimmt. Im trivialen Fall einer (konstanten) Extrapolation 0.Ordnnug gilt $\tilde{\boldsymbol{u}}_{\boldsymbol{j}} = \bar{\boldsymbol{u}}_{j,l}$. In Abbildung 3.4(a) sind der kontinuierliche Verlauf einer direkten, skalaren Koppelvariablen y mit deren diskretem Koppelwert $\bar{u} = \bar{y}$ dargestellt. Verglichen werden eine Interpolation erster Ordnung im sequentiellen Fall $\tilde{u}_{in,s}$, im parallelen Fall $\tilde{u}_{in,p}$ und eine Extrapolation nullter $\tilde{u}_{ex,0} = \bar{u}$ bzw. erster Ordnung $\tilde{u}_{ex}$. Neben der *Art* der Approximation kann analog zur Kausalität auch nach dem *Ort* unterschieden werden: im Subsystem kann ein kontinuierlicher Teilmodelleingang durch $\boldsymbol{a}_j$ approximiert werden. Daneben kann für den Ausgang des Masters ein konstanter Mittelwert über H_l

$$\bar{\tilde{\boldsymbol{y}}}_M = \boldsymbol{a}_M(\bar{\boldsymbol{y}}_M, T, \boldsymbol{p}, ...) \tag{3.10}$$

bestimmt werden [51], siehe Abb. 3.4. Für die Approximationen $\boldsymbol{a}_j$ und $\boldsymbol{a}_M$ können unter-



(a) Interpolation und Extrapolation

(b) kontinuierliche Approximation

Abbildung 3.4.: Vergleich unterschiedlicher Approximationsansätze

schiedliche Verfahren gewählt werden. Sie unterscheiden sich in der Anzahl L der verwendeten Schrittwerte $(\bar{\boldsymbol{u}}_{l-L+1}, ..., \bar{\boldsymbol{u}}_l)$, der Verwendung von Ableitungen $(\dot{\bar{\boldsymbol{u}}}_l, \ddot{\bar{\boldsymbol{u}}}_l, ...)$, sowie der Verfahrensvorschrift $\boldsymbol{a}_j, \boldsymbol{a}_M$ selbst und ihren Parameterwerten $\boldsymbol{p}$. Sehr häufig kommen Polynome zur Anwendung, u. A. [5, 25, 65, 69, 82, 115]. Bei den Synchronisationsvarianten in Abb. 3.2 c) und e) kann zunächst jeweils interpoliert werden, bei f) muss einmalig extrapoliert werden, bevor interpoliert wird, ähnlich zu einem Prädiktor-Korrektor-Verfahren.

Ein Problem der Extrapolation sind die Sprünge zu den Koppelzeitpunkten T_l, die sich negativ auf die Subsystemsolver auswirken. Um die Unstetigkeit zu umgehen, besteht die Möglichkeit, einen kontinuierlichen Verlauf zu erzwingen. Gemäß Abb. 3.4(b) kann dies z.B. durch *Filterung*, $\tilde{u}_{flt}$ erfolgen [60, 80, 110]. Bei der *interpolierten Extrapolation* [82], $\tilde{u}_{inex}$,

wird eine lineare Verbindung zwischen den Extrapolationswerten am Schrittende $\tilde{u}_{ex}(T_l)$ gezogen. Eine Weiterentwicklung dessen ist die Methode der *Glättung* [83], $\tilde{u}_{exgl}$. Dabei wird ein stetig differenzierbarer Übergang zwischen dem letzten Wert von $\tilde{u}_{ex}$ in H_{l-1} und einem bestimmten Punkt im Verlauf der Extrapolation a_j in H_l, etwa durch Splines, gebildet. Weiterhin ist im Zusammenhang der Approximation der regelungstechnische Ansatz der *Fehlerkompensation* zu nennen, welcher in [15] patentiert ist. Hierbei wird der Fehler der Approximation ermittelt und im folgenden Makroschritt kompensiert.

Die Wahl der Approximationsmethode ist in Abhängigkeit von Synchronisierung, Subsystem inkl. gewähltem Solver, sowie der Kausalität der Schnittstellen für jede Lösungskomponente gesondert zu treffen. Wie bei den Integrationsmethoden lassen sich Approximationen bei CS nach ihrer Genauigkeit und Ordnung charakterisieren. Im Zusammenhang mit dem Subsystem lassen sich Aussagen zu Stabilität treffen. Dies wird an dieser Stelle nicht näher behandelt. Eine ausführliche Darstellung der gewählten Methoden wird in Kapitel 4.4 vorgenommen. Mit der Analyse von Extrapolationsverfahren in der CS befasst sich eine große Zahl an Arbeiten, u. A. [9, 22, 25, 51, 69, 82, 83]. So weist [5] eine Stabilisierung der CS durch Extrapolation nach. Zumeist erweist sich eine Extrapolation bis maximal zweiter Ordnung als stabil und genauigkeitssteigernd [6, 65, 114]. Nur in Einzelfällen kommt eine Polynomextrapolation höherer Ordnung erfolgreich zum Einsatz [119].

3.2.4. Makroschrittweiten

Wie die Integrationsschrittweite h, so muss auch die Diskretisierung in Makroschritte H, als Kompromiss zwischen Genauigkeit und Aufwand, sorgfältig an das Problem angepasst werden. Zum Einen bedeuten große H weniger Kommunikationsaufwand und in den Subsystemen weniger *time events*, die für die Synchronisierung notwendig sind und ein Reinitialisieren zu jedem T_l erfordern. Jedoch sinkt die Genauigkeit mit wachsendem H bis hin zur Instabilität. So ergibt sich H_{max} aus der Grenze stabilen Verhaltens, praktisch jedoch aus dem Bedarf, das dynamische Verhalten aller Lösungskomponenten in $\boldsymbol{y}$ in ausreichender Auflösung abzubilden, d.h. $H_{max} \leq 1/f_{max}$, wobei f_{max} die höchste Koppelgrößenfrequenz ist [65]. Zum Anderen bedeuten kleinere H kleinere Fehler $\bar{\boldsymbol{\tau}}$, [5], wobei die Schrittanzahl und damit der Aufwand steigen. Der kleinste mögliche Wert ergibt sich zu: $H_{min} := \min_j(h_j)$

Die Wahl des H-Wertes ist abhängig von der Synchronisierung. So entfällt bei asynchroner Kopplung die Notwendigkeit der Wahl. Bei sequentieller Kopplung ist etwa eine Wahl zu $H := h_A$ (mit A: erstes ausgeführtes Subsystem) möglich. Generell kann ein Subsystem j als Master deklariert sein, so dass $H := h_j$. Dies ist entspricht dem Vorgehen bei Multirate-Verfahren, wo $H := h_j = rh_{i \neq j}$ mittels Faktor $r \in \mathbb{N}$ festgelegt wird [40, 147]. In Abhängigkeit von r und der Approximationsordnung von $\tilde{\boldsymbol{u}}$ lässt sich eine kritische Schrittweite H_{max} ermitteln, die jedoch auch vom lokalen Integrator in j abhängt, [79].

Je nach Anwendungsgebiet finden sich in bestehenden Umsetzungen sehr unterschiedliche H-Werte: So gibt [65] $1 \cdot 10^{-6}s$ als typisch für MKS-Simulationen des verbrennungsmotorischen Kettentriebs an. [49] verwendet hierfür $1 \cdot 10^{-6}s$ bis $8 \cdot 10^{-6}s$. In den Arbeiten zur Stabilitätsbestimmung der CS [25, 51, 115] anhand zweier gekoppelter 1D-Schwungmassen wird der Bereich $1 \cdot 10^{-6}s$ bis $10 \cdot 10^{-3}s$ betrachtet. Allgemein für die MKS-Simulation zu verwenden sind laut [9, 79] Werte von $0,1 \cdot 10^{-3}s$ bis $10 \cdot 10^{-3}s$. Betrachtet man die Systemsimulation von Kraftfahrzeugen, so werden größere Schritte verwendet. Mit $H = 10 \cdot 10^{-3}s$ und physikalischem Triebstrangmodell läuft die Simulation bei [67] extrem langsam, $\theta_E >> 1$. In den Bereich von $\theta_E \approx 1$ kommt die Kopplung nach Synchronisationsschema g) (Abb. 3.2(g)), wo mehrere Modelle zu einem Gesamtfahrzeugmodell mit $0,1s$ bzw. $0,5s$ gekoppelt werden, [83]. Ähnliche Rechenzeiten erzielt [54] bei der CS eines thermischen Fahrzeugmodells mit $H = 1s$. Für die Ankopplung des Batteriemodells eines Hybridfahrzeugs verwendet [14] gar Schrittweiten von $1s$ bis $40s$. Werden sehr lange Simulationszeiten t_{Sim} betrachtet, so werden Makroschritte bis in den Bereich mehrerer Minuten gewählt, wie in der Gebäudetechnik [138].

Setzt man nun die Makroschrittweite in Relation zur Simulationszeit und ermittelt über

$$H_l^* = \frac{H_l}{t_{Sim}}; \quad \text{wenn } H_l := H \Rightarrow M = \frac{t_{Sim}}{H} = \frac{1}{H^*} \tag{3.11}$$

eine normierte Makroschrittweite H_l^* bzw. die Anzahl an Schritten M, so verringern sich die Unterschiede. Spannen die üblichen Absolutwerte einen Bereich von $H = 1 \cdot 10^{-6}s$ bis $1 \cdot 10^{2}s$ über die Größenordnung 10^8 auf, so reduziert sich dies auf Größenordnung 10^3 bei H^* bzw. M. Dies liegt an den Unterschieden der Simulationszeiten t_{Sim}, dem entsprechend angepassten Detaillierungsgrad und der daraus folgenden Auflösung der Diskretisierung. So werden bei MKS-Simulationen meist wenige Sekunden, in der Systemsimulation mehrere tausend Sekunden simuliert.

Variable Makroschrittweiten

Bisher wurde in den meisten Arbeiten eine angepasste, aber konstante Makroschrittweite $H_l := H$ gewählt. Erst in jüngster Zeit kommt man in der Forschung dem Bedarf [8, 23, 114, 146] nach variablen Schritten H_l nach. Prinzipiell sind davon die gleichen Vorteile, wie bei lokaler (Integrations-)Schrittweitensteuerung (vgl. Kapitel 2.3.2) zu erwarten: Entfall der Wahl von H, dynamische Anpassung an den Lösungsverlauf, d. h. Verringerung des Fehlers oder des Aufwands. Naheliegend ist auch der Ansatz, die bekannten Verfahren zur lokalen Schrittweitesteuerung auf die CS zu übertragen. Insbesondere die etablierten Verfahren mit Fehlerschätzer (vgl Abb. 2.7 und Vorschriften (2.27), (2.28)). Hieraus ergeben sich besondere *Anforderungen* an die Subsysteme und die beteiligten Simulationswerkzeuge:

- Eingriffsmöglichkeit in lokale Solver (mind. Vorgabe von H_l bei T_l).

- Möglichkeit, Makroschritte in Subsystemen zu verwerfen / wiederholen.
- Speichern aller (alten) Zustandswerte $\boldsymbol{z}_j$ und Ausgänge $\boldsymbol{y}_j$ bei T_l

Entsprechend sieht z.B. der in der Entwicklung befindliche FMI-Standard [48] bereits eine Schnittstellendefinition für variable Makroschritte vor, die die Anforderungen erfüllt. Hier werden jedoch lediglich die Schnittstellen definiert, innerhalb des Standards werden keine CS-Algorithmen entwickelt. Doch es finden sich inzwischen in der Literatur einige Ansätze für eine variable Makroschrittweite. So überträgt [79] erstmals die Richardson-Extrapolation auf die Co-Simulation. Dies wird zunächst von [119] aufgegriffen und um eingebettete Verfahren erweitert. Die Methoden werden in [115] weiter untersucht und von [23] um ein Prädiktor-Korrektor-Verfahren erweitert. Ein spezielles sequentielles Verfahren, das den Makroschritt bei Erreichen einer bestimmten Fehlergrenze abbricht wird in [149]. Es handelt sich jeweils um Arbeiten im akademischen Umfeld mit Testmodellen. Im industriellen Kontext und in Verbindung mit komplexeren Systemen steht eine Umsetzung noch aus. Kapitel 4.5 behandelt ausführlich den Beitrag innerhalb dieser Arbeit zu diesem Thema.

3.2.5. Behandlung von Unstetigkeiten

Treten während einer Co-Simulation Unstetigkeiten $\bar{\boldsymbol{z}}_j$ innerhalb eines Teilmodells auf, so muss dies die Ereignisbehandlung des lokalen Solvers abdecken. Ist jedoch eine Komponente in $\boldsymbol{y}_j$ nicht stetig, muss dies gesondert betrachtet werden. Einerseits sind $\bar{\boldsymbol{y}}_{j,l}$ und $\bar{\boldsymbol{u}}_{j,l}$ diskrete Variablen, und an den bekannten Synchronisationszeitpunkten T_{l+1} wird demzufolge ohnehin ein *time event* (vgl. Kapitel 2.3.2) ausgelöst, an dem sie ihre Werte ändern. Andererseits wird der Zeitpunkt einer Unstetigkeit in $\boldsymbol{y}_j$ nicht durch den Makroschritt H_l berücksichtigt. Weiterhin kann auch das Überschreiten eines Grenzwerts $u_{m,i}^{lim}$ durch $y_{j,i}$ ein *state event* in Teilsystem $m \neq j$ erforderlich machen, evtl. verbunden mit einer Strukturumschaltung. Dies hängt von der Extrapolationsordnung von $\tilde{u}_{m,i}$ ab, die ein zu frühes oder zu spätes Überschreiten von $u_{m,i}^{lim}$ bedeuten oder ein zweimaliges Überschreiten nicht erkennen könnte, falls H_l groß genug ist. Hierdurch können unplausible Zustandswerte entstehen. Beispiele hierfür sind Stoßvorgänge mit $\tilde{u}_{m,i} = x \geq 0$ oder die Drehzahl eines Verbrennungsmotors mit $\tilde{u}_{m,i} = n_{mot} \in (750, 5000]$. Aufgrund der fehlenden Eingriffsmöglichkeiten in lokale Solver existieren bisher keine Algorithmen zur CS-Ereignisbehandlung. Als positiven Effekt ergibt sich daraus lediglich, dass *chattering* (vgl. 2.3.2) an Subsystemschnittstellen nicht auftreten kann. Zur Berücksichtigung von Unstetigkeiten bei CS bestehen zwei Möglichkeiten: Reaktive oder aktive Ereignisbehandlung. Zum Einen kann auf den während H_l durch ein Ereignis entstandenen Fehler mit einer Fehlerkompensationsmethodik reagiert werden. So ließe sich zumindest ein Fehler in der Energiebilanz im Schritt H_{l+1} wieder ausgleichen, [15]. Zum Anderen entspräche eine aktive Ereignisbehandlung dem Vorgehen bei monolithischer Simulation. Notwendig ist hierfür eine variable Schrittweite H_l und das Zurückspringen (*revoke*)

in der Zeit t_j in den Subsystemen mit zugehörig gespeicherten Zuständen $\boldsymbol{z}_{j,l}$.

Dies könnte wie folgt ablaufen: **1.)** zu T_{l+1}: Anzeigen des Ereignisses in H_l durchs Subsystem, Angabe von T_{event}; **2.)** Zurücksetzen der CS auf T_l, Auslesen aller $\boldsymbol{z}_{j,l}$; **3.)** Ausführen eines Makroschrittes $H_l^{neu} := T_{event} - T_l$, $T_{l+1}^{neu} := T_{event}$; **4.)** Reinitialisieren bei T_{l+1}^{neu}, iterativ durch $H = 0s$; **5.)** Fortfahren der CS mit neuem $H_{l+1} > 0s$.

Der FMI-Standard [48] liefert bereits den Rahmen für ein so geartetes Vorgehen mit Eventindikator und der Möglichkeit zur Eventiteration ($H = 0s$). Jedoch wird hier nur die Schnittstelle, nicht ein Algorithmus umgesetzt. Dies wird möglich, sobald die Simulationswerkzeuge und Solver entsprechende Anpassungen (s. o.) zulassen.

3.3. Bewertungskriterien für Co-Simulationen

Analog zur monolithischen Simulation (vgl. Kapitel 2.3.1) lassen sich die numerischen Eigenschaften der Co-Simulation in theoretischen Untersuchungen analysieren. Konsistenz, Konvergenz und Stabilität hängen sehr stark von der Kombination aus Synchronisierung, Kausalität, Approximation und der Makroschrittweite ab. So gibt es v. A. Arbeiten mit Analysen der CS für jeweils bestimmte Kombinationen u. A. in [6, 7, 23, 49, 69, 79, 115]. Im Hinblick auf die in der vorliegenden Arbeit im Zusammenhang der Gesamtfahrzeugsimulation *angewandte* CS werden an dieser Stelle Kriterien vorgestellt, die die Bewertung anhand numerischer Experimente zulassen. Sie kommen in den Kapiteln 4 und 5 als Entwicklungswerkzeuge zur Anwendung. [82] schlägt Indikatoren zur qualitativen Bewertung der CS vor, zum Einen die relative Differenz der Gradienten zweier aufeinanderfolgender Koppelsignale, zum Anderen die relative Anzahl an Vorzeichenwechseln dieser Gradienten. Hier sollen jedoch quantitative Kriterien gezeigt werden, die Aussagen über Genauigkeit und Aufwand, sowie in deren Kombination über die Effizienz der CS erlauben.

Genauigkeit

Die Bewertung der Genauigkeit erfolgt über ein Fehlermaß τ. Im Gegensatz zur Fehlerschätzung während der Simulation ist dies der dabei tatsächlich entstandene Fehler. Um gezielt die CS zu untersuchen, wird nicht ein analytisch exakter Lösungsverlauf (welcher für komplexe Modelle häufig nicht bestimmbar ist), sondern das Ergebnis einer monolithischen Simulation $\hat{\boldsymbol{y}}$ als Referenz gewählt. Dabei gilt die Annahme, dass $H >> h$ und $\hat{h}$ der Referenz sowie h_j der Subsysteme so klein gewählt sind, dass $\hat{\boldsymbol{y}}$ als ausreichend genau und $\hat{t}$ sowie t_j als näherungsweise kontinuierlich betrachtet werden können. Der Fehler soll der Übersichtlichkeit halber an einer skalaren Koppelgröße y bzw. u diskutiert werden, siehe Abb. 3.5(a). Dabei soll die k_y-te Ausgangsgröße

$$y := y_{k_y}; \quad \text{aus } \boldsymbol{y} \tag{3.12}$$

mit (3.3) und

$$\bar{\tilde{\boldsymbol{y}}}_M \approx \bar{\boldsymbol{y}}_M = \bar{\boldsymbol{u}}_M = \bar{\boldsymbol{y}} \tag{3.13}$$

der k_u-ten Eingangsgröße

$$\bar{u} := \bar{u}_{k_u} = \bar{\tilde{y}}_{M,k_y} \tag{3.14}$$

entsprechen. Durch den Austausch zu diskreten T_l entsteht zunächst der *Diskretisierungsfehler* der CS

$$\bar{\tau} = \bar{y} - y. \tag{3.15}$$

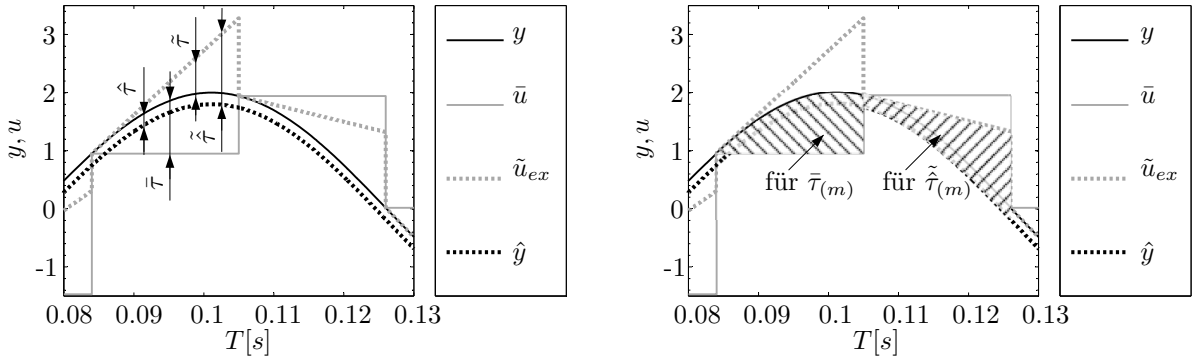
Dessen Einfluss wird anschliessend durch die Approximationen $\bar{\tilde{y}}_M$ bzw. $\tilde{u}$ verändert. Am Ende eines Makroschrittes ergibt sich daraus analog zu (2.22) der *lokale Fehler* der CS,

$$\tilde{\tau} = \tilde{u} - y. \tag{3.16}$$

Während des Makroschrittes lässt sich allgemein dessen Verlauf $\tilde{\tau}(T)$ angeben,

$$\tilde{\tau}(T) = \tilde{u}(t_i) - y(t_j) \quad \text{für } t_i = t_j = T. \tag{3.17}$$

Da sich der lokale Fehler in den Subsystemen fortpflanzt, weicht auch die Ausgangsgröße y



(a) Verlauf $\hat{\tau}$, $\bar{\tau}$, $\tilde{\tau}$ und $\hat{\tilde{\tau}}$

(b) Flächen $(T_{l+1} - T_l) \cdot \bar{\tau}_{(m)}(T_{l+1} - T_l)$ und $(T_{l+2} - T_{l+1}) \cdot \hat{\tilde{\tau}}_{(m)}(T_{l+2} - T_{l+1})$, vgl. (3.21)

Abbildung 3.5.: Fehlerermittlung bei der Co-Simulation

von der entsprechenden Referenzgröße $\hat{y}$ ab. Diese lässt sich durch die Ausgangsgleichung

$$\hat{\boldsymbol{y}} = \boldsymbol{g}_{CS}(t, \hat{\boldsymbol{z}}, ...); \quad \text{mit } \boldsymbol{g}_{CS} = \left[\boldsymbol{g}_A \cdots \boldsymbol{g}_j \cdots \boldsymbol{g}_N\right]^T \tag{3.18}$$

aus der monolithischen Simulation auslesen. So können auch innere Zustände $\boldsymbol{z}_j$ der Subsysteme, die keine Koppelgrößen sind, mit der Referenz $\hat{\boldsymbol{z}}$ verglichen werden, indem $\boldsymbol{g}_j$ entsprechend erweitert wird. Es ergibt sich der *Fehler durch Rückkopplung* bei CS zu

$$\hat{\tau}(T) = y(t_j) - \hat{y}\left(\hat{t}\right) \quad \text{für } t_j = \hat{t} = T. \tag{3.19}$$

Da die Genauigkeit der CS jedoch maßgeblich von der Approximation der Koppelgrößen und deren Makroschrittweite abhängt, ergibt sich analog zu (2.24) der *globale Fehler* der CS zu

$$\hat{\tilde{\tau}}(T) = \tilde{u}(t_i) - \hat{y}(\hat{t}) \quad \text{für } t_i = \hat{t} = T. \tag{3.20}$$

Aufgrund der Dynamik innerhalb eines Makroschritts und während $T_0 \rightarrow T_M$ kann $\tau(T)$ stark schwanken. Um die Genauigkeit eines Zeitintervalls zu bewerten, soll der mittlere Fehler betrachtet werden. Er entspricht dem Integralmittelwert, d. h. den Flächen gemäß Abb. 3.5 zwischen den entsprechenden Verläufen bezogen auf die Intervalldauer. Für den *mittleren globalen Fehler* ergibt sich somit

$$\hat{\bar{\tau}}_{(m)}(T) = \frac{1}{T - T_0} \int_{t=T_0}^{T} |\tilde{u}(t) - \hat{y}(t)|\, dt \quad \text{mit } t_i = \hat{t} = t. \tag{3.21}$$

Analog lassen sich der *mittlere Diskretisierungsfehler* $\bar{\tau}_{(m)}(T)$ und der *mittlere lokale Fehler* $\tilde{\tau}_{(m)}(T)$ bestimmen. Die Berechnung des Integrals in (3.21) muss numerisch erfolgen: da hier der Einfluss der CS betrachtet werden soll, kann mit obiger Annahme $H >> h$ davon ausgegangen werden, dass die Fläche unter Verwendung der Trapezregel hinreichend genau angenähert wird. Hierfür muss die Verwendung gleicher Stützstellen $t_k = t_{i,k} = \hat{t}_k \forall k$ sichergestellt sein.

Zur besseren Vergleichbarkeit wird der relative Fehler herangezogen. Als Bezugswert soll jedoch nicht $\hat{y}$ bzw. y direkt angewandt werden, um Singularitäten bei Nulldurchgängen zu vermeiden. Stattdessen wird der Mittelwert

$$\hat{y}_{(m)}(T) = \frac{1}{T - T_0} \int_{t=T_0}^{T} |\hat{y}(t)|\, dt \tag{3.22}$$

herangezogen und der *mittlere relative globale Fehler* zum Zeitpunkt T ergibt sich zu:

$$\hat{\bar{\tau}}^*_{(m)}(T) = \frac{\hat{\bar{\tau}}_{(m)}(T)}{\hat{y}_{(m)}(T)}. \tag{3.23}$$

Analog stellt sich der *mittlere relative lokale Fehler* $\tilde{\tau}^*_{(m)}(T)$ bezogen auf $y_{(m)}(T)$ dar. Um die Genauigkeit unterschiedlicher Co-Simulationsexperimente im Gesamtintervall t_{Sim} zu vergleichen, stellt der Vergleich von $\hat{\bar{\tau}}^*_{(m)} = \hat{\bar{\tau}}^*_{(m)}(T_M)$ schließlich einen allgemeinen Ansatz dar, wobei das Auftreten einzelner Spitzen $\max_T(\hat{\tilde{\tau}}(T))$ beachtet werden muss. Die Notwendigkeit des Betrags in (3.21) wird deutlich, wenn der durch die CS ins System eingebrachte Fehler in der Energieerhaltung betrachtet wird. So kann der Effekt, dass der Ansatz der CS der Einführung eines Energiespeichers ins System entspricht [15], für Bilanzbetrachtungen ausgeblendet werden, vgl. 4.3.1.

Aufwand

Die Umsetzung der CS erzeugt zunächst im Berechnungsaufwand des Gesamtsystems einen Overhead im Vergleich zur monolithischen Simulation: zum Einen durch den sequentiell auszuführenden CS-Algorithmus, vgl. Abb. 3.1, zum Anderen durch das Anhalten der Subsysteme an den Zeitpunkten T_l , wodurch jeweils ein Ereignis auftritt. Kausalität und Approximation haben bei Verwendung lokaler Schrittweitensteuerung über $h_{j,k}$ einen Einfluss auf den Aufwand. Die größere, direktere Auswirkung besitzen jedoch Synchronisierung und Makroschrittweite. So ist der Overhead bei parallel bzw. sequentiell synchroner Kopplung näherungsweise proportional zu Schrittzahl M bzw. $\frac{1}{H^*}$. Der Aufwand lässt sich durch Vergleich der Rechenzeiten

$$S_{CS} = \frac{\hat{t}_{CPU}}{t_{CPU,CS}} \quad \text{mit } \hat{t}_{CPU} \text{... Rechenzeit der Referenz} \tag{3.24}$$

bemessen. So bedeutet $S_{CS} > 1$ eine Beschleunigung der Simulation (*Speed-up*), wohingegen bei $S_{CS} < 1$ der Overhead überwiegt. Im Kontext dieser Arbeit ist das Ziel $S_{CS} > 1$, ab dem eine Partitionierung Vorteile bringt, durch möglichst geringes M zu erreichen wie es in Abb. 4.2 eingezeichnet ist. Dies kann als Hinweis für die Wahl eines festen H^* bzw für Algorithmen zur Makroschrittweitensteuerung herangezogen werden. Bei Letzterem ergibt sich die mittlere Schrittweite aus

$$H_{(m)} = \frac{t_{Sim}}{M}; \quad \text{bzw. } H^*_{(m)} = \frac{1}{M} \tag{3.25}$$

Über den Einfluss der verwendeten Synchronisierung auf die Gesamtrechenzeit der Co-Simulation $t_{CPU,CS}$ lässt sich der erreichbare Speed-up S_{CS} vergleichen. So setzt sich der Aufwand

$$t_{CPU,CS} = \sum_{l=1}^{M} t_{CPU,CSA,l} + \begin{cases} \sum_{l=1}^{M} \sum_{j=A}^{N} t_{CPU,j,l}; & \text{sequentiell} \\ \sum_{l=1}^{M} \max_{j}(t_{CPU,j,l}); & \text{parallel} \end{cases} \tag{3.26}$$

aus der Summe des nicht parallelisierbaren Aufwands des CS-Algorithmus in jedem Schritt $t_{CPU,CSA,l}$ und der Summe des Aufwands für die Subsysteme zusammen. Im sequentiellen Fall folgt daraus mit $t_{CPU,j} = \sum_{l=1}^{M} t_{CPU,j,l}$

$$t_{CPU,CS} > \sum_{j=A}^{N} t_{CPU,j} \tag{3.27}$$

Durch Verwendung einer monolithischen Gesamtsystemsimulation als Referenz, $\hat{t}_{CPU} = t_{CPU,ges}$, und mit der vereinfachenden Annahme, dass der Aufwand der Subsysteme im monolithisch gerechneten Fall (vg. Abschn. 2.5) dem im Kontext der CS entspricht, $\hat{t}_{CPU,j} \approx t_{CPU,j}$,

folgt daraus mit (2.34) für (3.24)

$$S_{CS}^{(s)} < \theta_A \tag{3.28}$$

im *sequentiellen* Fall. Dies entspricht dem Umstand, dass der durch Zusammenbau von Submodellen zu einem Gesamtmodell entstehende Mehraufwand θ_A durch Zusammenbau mithilfe CS bis zu einem gewissen Grad wieder aufgehoben werden kann (Multirate). Bei paralleler Kopplung kann durch Verteilung auf mehrere CPUs ein weiterer Zeitvorteil entstehen. Dieser ist durch das die meisten Ressourcen belegende Subsystem begrenzt und aus (3.26) und (3.24) folgt

$$S_{CS} < \frac{\hat{t}_{CPU}}{\sum_{l=1}^{M} \max\limits_{j}(t_{CPU,j,l})} \approx \frac{\hat{t}_{CPU}}{\max\limits_{j}(t_{CPU,j})} \tag{3.29}$$

im *parallelen* Fall. Dem liegt die Annahme zugrunde, dass stets das gleiche Subsystem den größten Aufwand verursacht, d. h. $\max\limits_{j}(t_{CPU,j}) = \max\limits_{j}(\sum_{l=1}^{M} t_{CPU,j,l}) = \sum_{l=1}^{M} \max\limits_{j}(t_{CPU,j,l})$. Dies ist in der Praxis zwar häufig der Fall, jedoch kann sich aufgrund der sich ändernden Dynamik in den Subsystemen der Rechenbedarf in manchen Schritten ändern, so dass allgemein gilt: $\max\limits_{j}(t_{CPU,j}) \leq \sum_{l=1}^{M} \max\limits_{j}(t_{CPU,j,l})$, was den Speed-up weiter einschränkt. Beobachtet man den Rechenfortschritt der Subsysteme innerhalb eines Makroschrittes, ist offensichtlich, dass die Subsysteme für die Synchronisation auf das Langsamste warten müssen. Diese Wartezeit fließt bei Verwendung nur einer CPU in die benötigte Zeit pro Schritt $t_{CPU,j,l}$ mit ein, so dass die Verwendung paralleler Kopplung mit nur einer CPU der Begrenzung (3.28) unterliegt. Werden mehre CPUs verwendet, so ist auch physikalisch eine parallele Rechnung möglich. Dies kann mithilfe des *Thread-Profiling* gemessen werden, wie in [49] verwendet. Die theoretischen Grenzen für den Speed-up durch Parallelisierung werden hierbei durch die mathematisch äquivalenten Gesetze von *Amdahl* bzw. *Gustafson* [124] beschrieben, worin wieder der nicht-parallelisierbare Anteil begrenzend eingeht. Im Grenzfall $t_{CPU,CSA} \to 0$ geht aus beiden Gesetzen hervor, dass $S_{CS}^{(p)} \leq N_{CS}$, wobei $N_{CS} = \min(N, N_{CPU})$, falls N Subsysteme auf N_{CPU} CPUs gerechnet werden können. Aufgrund der eher geringen N_{CS} im vorliegenden Fall lässt sich für den insgesamt durch Parallelisierung ($S_{CS}^{(p)}$) und Multirate-Umsetzung ($S_{CS}^{(s)}$) erreichbaren Speed-up festhalten: $S_{CS} < \theta_A N_{CS}$. Dies ist über N_{CPU} abhängig von der verwendeten Hard- und Software. Davon unabhängige Bewertungskriterien sind die Verhältnisse an Funktionsauswertungen $\theta_{F,CS}$ bzw. Ereignissen $\theta_{Ev,CS}$

$$\theta_F = \theta_{F,CS} = \frac{\hat{n}_F}{\sum_{j=1}^{N} n_{F,CS,j}} \qquad \theta_{Ev} = \theta_{Ev,CS} = \frac{\hat{n}_{Ev}}{\sum_{j=1}^{N} n_{Ev,CS,j}} \tag{3.30}$$

mit $n_{F/Ev}$ als Summe der Funktionsauswertungen (F) bzw. Ereignisse(Ev), analog zu (2.35).

Effizienz und Effizienzindikatoren

Die Effizienz der CS stellt das Verhältnis von Kosten und Nutzen dar. Dies entspricht dem Kompromiss aus Genauigkeit und Aufwand, wobei der bessere Kompromiss eine höhere Effizienz bedeutet. Neben dem Vorteil des Einsparens von Entwicklungsaufwand bei getrennt modellierten Subsystemen ist der Nutzen der CS hier in der Reduktion des Rechenaufwands zu sehen, so dass der Wert S_{CS} als quantitatives Kriterium verwendet werden kann. Als Kostenkriterium kann der Verlust an Genauigkeit verwendet werden, d. h. der durch die CS eingebrachte Fehler τ^*.

Um die Effizienz zu berechnen, muss der skalare Wert S_{CS} ins Verhältnis zu einem skalaren Wert für τ^* gesetzt werden, d.h. einem ganzheitlichen Fehlerkriterium. Gemäß (3.15) bis (3.23) lassen sich die Fehler für alle Koppelgrößen getrennt berechnen woraus sich jeweils ein Fehlervektor $\boldsymbol{\tau}^*$ ergibt. Es gibt verschiedene Wege, diesen zu einem Skalarwert zu normieren: die Wahl des arithmetischen Mittels $\tau^* := \tau^*_{(m,arithm)}$, die Maximumsnorm $\tau^* := \|\boldsymbol{\tau}^*\|_\infty$, die Euklidische Norm $\tau^* := \|\boldsymbol{\tau}^*\|_2 = |\boldsymbol{\tau}^*|$ oder deren abgewandelte Form

$$\tau^* := \|\boldsymbol{\tau}^*\|_h = \sqrt{\frac{1}{m_y}\sum_{k_y=1}^{m_y} \tau^{*2}_{k_y}}. \tag{3.31}$$

Dafür kann über alle m_y Koppelgrößen der N Subsysteme oder jeweils für die $n_{y,j}$ Größen der Systeme einzeln gemittelt werden. Die Normierungen werden Kapitel 4.5 verglichen. Für die Effizienzberechnung wird $\|\boldsymbol{\tau}^*\|_h$ für m_y berechnet. Als Fehlerkriterium $\tau^*_{k_y}$ soll $\hat{\tilde{\tau}}^*_{(m)}(T_M)$ aus (3.23) verwendet werden, da es sich um den relativen auf das gesamte Simulationsintervall bezogenen Fehler der CS handelt. Somit ergibt sich zur Bewertung der Effizienz einer CS der Indikator

$$CSI = \frac{S_{CS}-1}{100\tau^*} = \frac{S_{CS}-1}{100\left\|\hat{\tilde{\boldsymbol{\tau}}}^*_{(m)}(T_M)\right\|_h}. \tag{3.32}$$

Der Wert von CSI ist als Indikator zu verstehen, anhand dessen sich verschiedene CS-Verfahren vergleichen lassen. Abbildung 3.6 zeigt am Beispiel $CSI = 0.5$, was z.B. einer Zielsetzung: „halbe Rechenzeit bei 2 % Fehler“ entspricht, die Möglichkeit der Klassifizierung verschiedener Verfahren. Hierbei bedeuten negative CSI-Werte: nicht zielführend für die Partitionierung. Mithilfe des CS-Indikators lassen sich verschiedene Methoden der Synchronisierung, Kausalität, Approximation und Makroschrittweitensteuerung auch einzeln vergleichen. Soll bei der Untersuchung verschiedener Makroschrittweiten beispielsweise der Einfluss der Approximation ausgeblendet werden, so lässt sich auch ein Indikator für den Diskretisierungsfehler definieren:

$$\overline{CSI} = \frac{S_{CS}-1}{100\left\|\bar{\boldsymbol{\tau}}^*_{(m)}(T_M)\right\|_h}. \tag{3.33}$$

Analog dazu lässt sich ein Indikator $\widetilde{CSI}$ definieren, der den lokalen Fehler $\tilde{\boldsymbol{\tau}}^*_{(m)}(T_M)$ her-

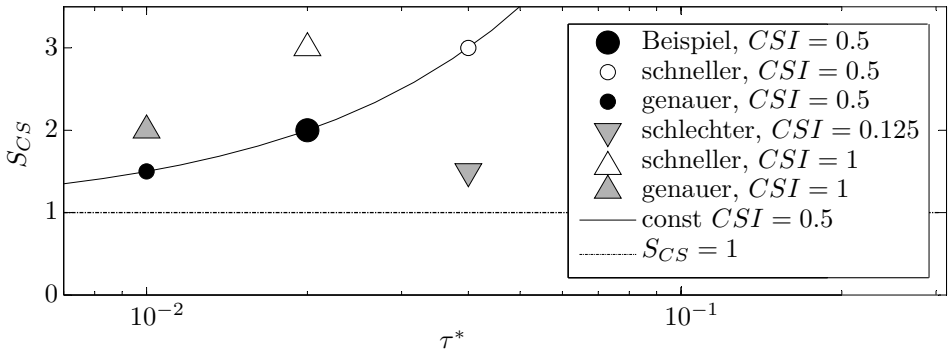


Abbildung 3.6.: CSI-Bewertung

anzieht. Soll die Effizienz der CS unabhängig der ausführenden Systeme verglichen werden, so lässt sich unter Verwendung von (3.30) der Indikator

$$CSI_{\theta_F} = \frac{\theta_F}{100 \left\| \hat{\tilde{\boldsymbol{\tau}}}^*_{(m)}(T_M) \right\|_h} \tag{3.34}$$

ausdrücken. Wieder können entsprechend $\overline{CSI}_{\theta_F}$ bzw. $\widetilde{CSI}_{\theta_F}$ verwendet werden. Einen einfachen Direktvergleich von Makroschrittweitensteuerungen mit H_{fix} unter Verwendung gleicher Schrittanzahl erlaubt, mit (3.25),

$$CSI_M = \frac{H_{(m)}}{100 \left\| \hat{\tilde{\boldsymbol{\tau}}}^*_{(m)}(T_M) \right\|_h}. \tag{3.35}$$

Wieder können entsprechend $\overline{CSI}_M$ bzw. $\widetilde{CSI}_M$ verwendet werden.

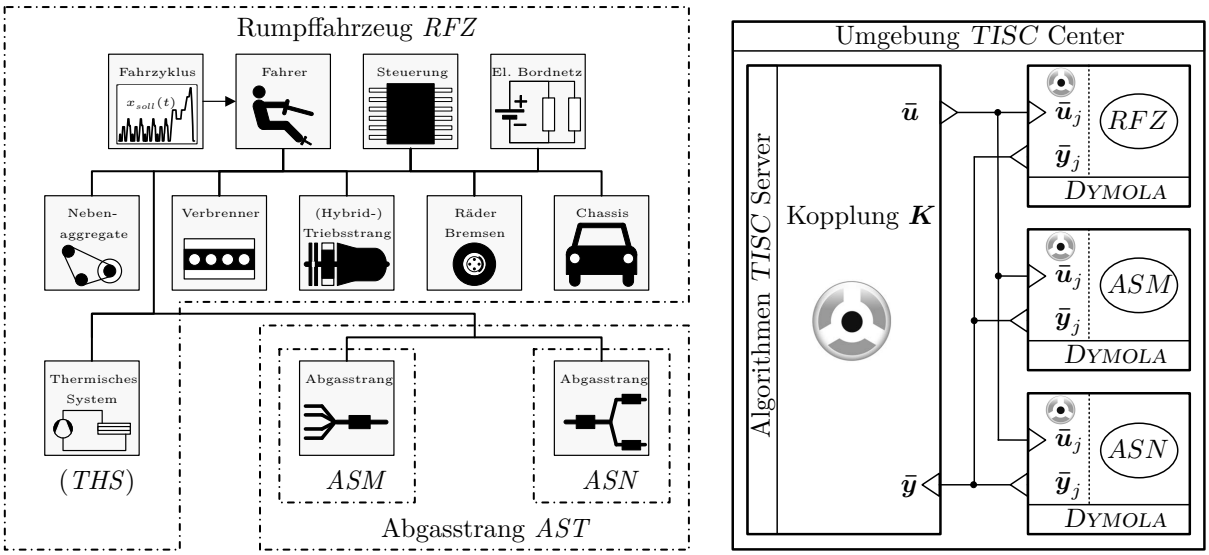
4. Erweitertes Verfahren zur Co-Simulation

Dieses Kapitel stellt das in dieser Arbeit entwickelte Verfahren vor. Nach einem Potentialnachweis für Co-Simulation gliedert es sich in die Abschnitte 4.1 zu den Randbedingungen, die auf das angewandte Konzept führen und 4.2, worin die Co-Simulations-Umgebung und die Teilsystemschnittstellen erläutert werden. Im Weiteren schließen sich drei Abschnitte über die Details an: zur Kausalität der physikalischen Schnittstellen, 4.3, zur Analyse der Approximationsmethoden, 4.4, sowie zur Entwicklung der Makroschrittweitensteuerung in Abschnitt 4.5.

Potentialnachweis

Gemäß dem in Kapitel 2.5 hergeleiteten Ansatz kann die modulare Simulation eines Gesamtsystems zu einer Reduktion der mit monolithischer Simulation benötigten Rechenzeit führen. Dazu wird das Gesamtmodell partitioniert und in einer Co-Simulation berechnet. Grundsätzliche Bedingung für die Verfolgung des Ansatzes und die Entwicklung eines erweiterten Verfahrens ist somit das Erreichen von $S_{CS} \overset{!}{>} 1$. Dies muss in der Zielanwendung dieser Arbeit, der Gesamtfahrzeugsimulation, Gültigkeit haben. Um das Speed-up-Potential zu überprüfen, soll ein Beispielmodell in *Dymola* [37] in monolithischer Simulation sowie in Co-Simulation berechnet und verglichen werden. Zur Kopplung wird die kommerzielle Schnittstellensoftware *TISC* [137] eingesetzt.

Verwendet wird das Modell eines Pkw mit aufgeladenem 2-Liter-Dieselmotor für Vorwärtssimulation, vgl. Kapitel 2.4. Für eine detailierte Beschreibung sei auf das Kapitel 5.1 verwiesen, wo die in dieser Arbeit eingesetzten Subsystemmodelle der einzelnen Fahrzeugdomänen sowie deren Parametrierung mit Messungen beschrieben sind. Abbildung 4.1 a) zeigt den schematischen Aufbau: Das hier als Rumpffahrzeug (RFZ) bezeichnete Modell besteht aus den Komponenten des Antriebsstranges, des Chassis' sowie des Bordnetzes und der Fahrzeugregelung. Es wird um ein Modell des thermischen Systems (THS) und des Abgasstranges (AST) erweitert. Das Modell des Abgasstranges bildet den motornahen Luftpfad mit Turbolader und geregelter Abgasrückführung sowie die Abgasnachbehandlung ab. Es beinhaltet Medienmodelle des Abgas-Luft-Gemisches in 1D-Strömungsmodellierung. Die einzelnen Modelle werden getrennt entwickelt und an den Schnittstellen zu den anderen Domänen jeweils



(a) Gesamtfahrzeugmodell mit Partitionierungen (b) Subsystemkopplung mit *TISC*

Abbildung 4.1.: Co-Simulation mit TISC zum Potentialnachweis

mit Eingängen $\boldsymbol{u}(t)$ aus Messungen oder vereinfachten Simulationen bedient. Dadurch lässt sich der Overhead θ_A aus (2.34) messen, der beim Zusammenbau der Subsystemmodelle entsteht. Aufgrund des in diesem Beispiel verwendeten, durch Reduktion auf wenige thermische Massen vereinfachten Modells des thermischen Systems, entsteht durch Zusammenbau RFZ & THS kein Overhead ($\theta_A \approx 1$). Dagegen führt die Integration des Abgasstranges (AST) ins Fahrzeugmodell (RFZ) fast zu einer Vervierfachung der Rechenzeit, $\theta_A = 3,95$, siehe Tabelle 4.1. Dies ist besonders auf das große Gleichungssystem, das durch die Medienmodelle im AST entsteht, zurückzuführen. Dies führt schon beim einzeln berechneten Abgasstrang zu langen Rechenzeiten. Hier lässt sich durch Partitionierung in den durch Ereignisbehandlung in den Reglermodellen betroffenen motornahen Teil (ASM) und den passiven Teil der Abgasnachbehandlung (ASN) mit Katalysator und Partikelfilter weiteres Potential ermitteln, $\theta_A = 2,37$. Insgesamt ergibt sich für die drei Submodelle (RFZ, ASM und ASN) ein Wert von $\theta_A = 7,37$, so dass die Co-Simulation mit diesen drei Subsystemen umgesetzt wird. Gemäß Abbildung 4.1 b) werden diese mit *TISC*-Modell-Schnittstellen versehen und jeweils in einer eigenen *DYMOLA*-Instanz aus Modell und Solver berechnet.

Verbunden und gesteuert werden die Subsysteme über *TISC*. Dabei handelt es sich um eine sog. Middleware, die Implementierung der CS-Umgebung und CS-Algorithmus vereint [83, 137]. Letzterer setzt eine *synchron parallele* Kopplung um. Er beinhaltet keine Koppelgesetze $\boldsymbol{f}_M$, so dass die *Kausalität* in den Subsystemschnittstellen erzeugt wird. Dabei kommen bei der Verbindung von RFZ und AST lediglich signalflussbasierte Schnittstellen ohne direkte Rückkopplung zum Einsatz. Bei der Kopplung ASM - ASN müssen jedoch physikalische Schnittstellen durch Potential-Fluss-Kopplung umgesetzt werden. Dies ist sowohl in der thermischen als auch der fluidischen Domäne der Fall und wird in Abschnitt

4.3 beschrieben, wobei in den fluidischen Schnittstellen kein Abgasrückfluss zugelassen wird, diese somit nur für positive Fließrichtung umgesetzt sind. Die *TISC*-Modell-Schnittstellen sehen zur *Approximation* verschiedene Extrapolationsverfahren vor. Für den Potentialnachweis wird jedoch Extrapolation nullter Ordnung, $\tilde{\boldsymbol{u}}_j = \bar{\boldsymbol{u}}_j$, gewählt. Dies ist auch durch die gewählten, großen *Makroschrittweiten* H bedingt. Der Parameter H wird zunächst von $H = 0.2s$ bis $H = 5s$ variiert (bei Berechnung des NEFZ mit $t_{Sim} = T_M = 1180s$: normiert $H^* = 0,17 \cdot 10^{-3}$ bis $H^* = 4,24 \cdot 10^{-3}$), vgl. Abbildung 4.2 a). In Anhang A ist die konstante Inzidenzmatrix $\boldsymbol{K}$ für die Kopplung von RFZ, ASM und ASN dargestellt, die die Dimension $m_u \times m_y = 24 \times 20$ besitzt.

Tabelle 4.1 vergleicht den Berechnungsaufwand der NEFZ-Simulation, wobei alle Konfigurationen zu gleichen Randbedingungen, DASSL-Solver ($Atol = Rtol = 1 \cdot 10^{-4}$) berechnet werden. Monolithische sowie *TISC*-Co-Simulation werden auf einem System mit *einer* CPU ausgeführt, wodurch direkt der Multirate-Vorteil gezeigt werden kann. Es sind drei Parti-

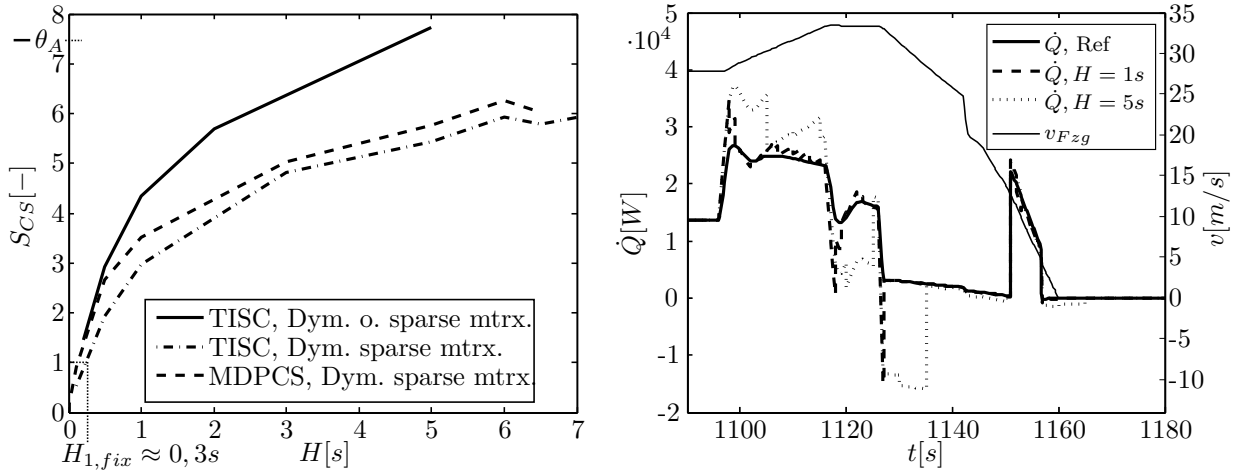
Tabelle 4.1.: Verhältnisse des Berechnungsaufwands der Subsysteme, monolithisch und in Co-Simulation (CS); **Gesamt**: Gesamtfahrzeug; **RFZ**: Rumpffahrzeug; **AST**: Abgasstrang; **ASM** AST Motornah; **ASN**: AST Nachbehandlung

$\sum j$	j	θ_A	θ_F	θ_{Ev}	$S_{CS}^{(H=5s)}$	$\theta_{F,CS}^{(H=5s)}$	$\theta_{Ev,CS}^{(H=5s)}$
Gesamt	**RFZ**	3,95	1,62	1,08	**4,44**	1,74	1,09
	AST						
AST	**ASM**	2,37	1,31	0,95	**2,24**	1,58	1,00
	ASN						
Gesamt	**RFZ**	7,37	1,78	1,07	**7,73**	1,85	1,01
	ASM						
	ASN						

tionierungen dargestellt: die Kopplung von Rumpffahrzeug und Abgasstrangmodell in der ersten Zeile. Darunter ist separat die Partitionierung des Abgasstranges gezeigt, gefolgt von der Kopplung des Gesamtfahrzeugmodells aus drei Subsystemen. In allen drei Fällen ergibt der Vergleich des Zusammenbaus der Subsysteme im monolithischen Fall, dass $\theta_A > \theta_F$ und $\theta_A > \theta_{Ev}$. D.h. die benötigte Rechenzeit steigt überproportional zur Anzahl an Funktionsauswertungen und Ereignissen, da letztere im Zusammenbau stets alle $\boldsymbol{z}$ betreffen. Somit ist der Multirate-Ansatz gerechtfertigt. Dies lässt sich am erreichten Speed-up $S_{CS} > 1$ bei der Co-Simulation nachweisen, der mit $H = 5s$ in den Bereich von θ_A kommt. In der Tabelle sind jeweils auch die Werte $\theta_{F,CS} \sim \theta_F$ und $\theta_{Ev,CS} \sim \theta_{Ev}$, da durch die große Schrittweite kaum Overhead entsteht. Vielmehr ist zweimal $S_{CS} > \theta_A$, was sich mit der unzulässig stark verringerten Dynamik an den Koppelstellen: $\tilde{\boldsymbol{u}} = \bar{\boldsymbol{u}} = konst.$ für je $5s$ begründen lässt.

Die Variation der festen Schrittweite H in Abbildung 4.2(a) zeigt eine Zunahme des Overheads bei kleineren Makroschritten und entsprechend kleineres S_{CS}. Zusätzlich zum Ergebnis

aus der Tabelle sind zwei weitere Konfigurationen aufgetragen: die Verwendung einer verbesserten Version von *Dymola* mit beim DASSL integrierten *sparse-matrix*-Verfahren, welche θ_A und damit den erreichbaren S_{CS} etwas senken; sowie die Verwendung, der unten im Detail beschriebenen CS-Umgebung MDPCoSim, bei gleicher CS-Konfiguration, wie mit *TISC*. Es lässt sich die theoretische Grenze $H_{1,fix} = H(S_{CS} = 1)$ ablesen, ab der sich im vorliegenden Kontext der Ansatz mit CS lohnt, wobei größere H-Werte das Potential weiter ausschöpfen. Dies muss im Zusammenhang mit den Genauigkeitseinbußen erwogen werden. Betrachtet man den aufintegrierten Kraftstoffmassenstrom, so gilt $\hat{\hat{\tau}}^*_{(m)}(T_M) < 1\% \forall H$. Werden jedoch Energieflüsse betrachtet, so zeigt sich exemplarisch bei der Kühlleistung $\dot{Q}$ das unplausible Verhalten in Abb. 4.2(b), das sich mit $H \uparrow$ verschlechtert. Zur Zielerreichung ist somit eine Weiterentwicklung der CS-Algorithmen notwendig, vgl. Abschnitte 4.2 bis 4.5.



(a) S_{CS} für verschiedene Makroschrittweiten H (b) Fehler bei Kühlleistungswert durch Co-Simulation

Abbildung 4.2.: Partitionierung RFZ-ASM-ASN

4.1. Randbedingungen an das Verfahren

Kommerzielle Simulationsumgebungen

Im Gegensatz zu Open-Source Anwendungen haben kommerzielle Simulationswerkzeuge proprietäre Implementierungen, insbesondere ihrer Solveralgorithmen, die somit weder einsehnoch beeinflussbar sind. Subsysteme in solchen Werkzeugen stellen für den CS-Algorithmus eine Black Box dar und im Gegensatz zu anderen Arbeiten [79, 119] ist das enthaltene Modell unbekannt und kann hybride DAE-Systeme enthalten. Das Simulationswerkzeug muss eine API zur Anbindung der CS bieten und gibt damit deren Umsetzung vor. Dies ermöglicht *Dymola* über *External Objects* für C-Code, über die Simulationen unterbrochen und Größen ausgetauscht werden können. Im Allgemeinen bieten kommerzielle Werkzeuge keine

Möglichkeit für interne Multirate-Umsetzungen, jedoch sind die hier vorgestellten Verfahren auch für solche Umsetzungen denkbar.

Generell bilden sich die Randbedingungen durch die Verwendung kommerzieller Werkzeuge auf die Kategorien aus 3.2 ab: ohne Eingriffsmöglichkeit in die lokalen Solver ist (nur) *implizite Synchronisierung* ausgeschlossen. Im Rahmen der Modellierungsmöglichkeiten im Werkzeug können sowohl *Kausalität* als auch die *Approximationen* $\boldsymbol{a}_j$ umgesetzt werden. Die *Makroschrittweite* H_l muss an die Subsysteme gesendet werden, um T_{l+1} setzen zu können. Da die lokalen Solver nicht in t_j zurückgesetzt werden können, sind für variable H keine Schritte verwerfbar. Dabei wäre die Möglichkeit, stattdessen jeweils $t_{j,Sim} = H_l \; \forall l$ zu setzen, bei größeren Modellen aufgrund des Speicher- und Initialisierungsaufwands nicht zielführend. Somit sind die Anforderungen aus 3.2.4 nicht erfüllbar, und es muss ein Verfahren für H_l gemäß (2.29) gewählt werden. Analog kann für die *Behandlung von Unstetigkeiten* in $\boldsymbol{y}$ kein iterativer Ansatz gewählt werden. In diesem Fall sind bei der Partitionierung nicht-stetige Komponenten in $\boldsymbol{y}_j$ zu vermeiden.

Gesamtsystem- und Zyklensimulation

Zum Einen wird in dieser Arbeit das Gesamtfahrzeug mit Subsystemen aller Domänen (vgl. Kapitel 2.4) modelliert, zum Anderen damit Verbrauchszyklen (vgl. Abbildung 5.1) simuliert, die ein transientes Systemverhalten für $t_{sim} > 10^3 s$ vorgeben. Die Kombination führt zu langen t_{CPU} und der Zielvorgabe $S_{CS} \overset{!}{>} 1$, was sich auf die Kategorien aus 3.2 abbildet:

Um große Werte S_{CS} zu erreichen, wird die *parallele Synchronisierung* gewählt, die auch eine Verteilung auf mehrere CPUs ermöglicht. Bei der *Kausalität* sollte auf schwache Kopplung, $K_{(AB)} \downarrow$ in (2.36), geachtet werden, um große H zu ermöglichen. Dazu müssen entsprechend der Partitionierungsgrenzen Domänen-spezifische Schnittstellen umgesetzt werden. Aufgrund der Synchronisierung können für die *Approximationen* Extrapolationen der Ordnung ≥ 0 und Glättung eingesetzt werden, die für variable H_l umgesetzt sein müssen. Die Synchronisierung macht ebenfalls eine Wahl der *Makroschrittweiten* durch das CS-Verfahren notwendig. Zur Zielerreichung sind dies Werte im Bereich $H_{(m)} \approx 0,1s...1s$, wobei zyklenbedingt $H_{l,max} \leq 10s$ gewählt werden kann. Aufgrund der transienten Fahrzyklen ergibt sich eine über t_{Sim} variable Dynamik, die eine variable Schrittweite nahelegt. Hierfür kann die Vorgabe durch den Zyklus als Indikator für die Makroschrittweitensteuerung dienen. Das Auftreten von *Unstetigkeiten* etwa durch die Fahrzeugsteuerung ist ebenfalls stark zyklenabhängig. Insofern kann die Zyklusvorgabe für ein vorrausschauendes Setzen von T_l helfen, jedoch mit aufgrund der Vorwärtssimulation eingeschränkter Genauigkeit.

4.1.1. Konzept der Verfahrensumsetzung

Mit allen Umsetzungsvarianten aus Kapitel 3.2 lässt sich ein morphologischer Kasten erstellen, mithilfe dessen sich unter Berücksichtigung obiger Randbedingungen das methodische Konzept für das CS-Verfahren in Tabelle 4.2 darstellen lässt. Es ist allgemein für Subsysteme werkzeugunabhängig einsetzbar.

Tabelle 4.2.: Umsetzung der einzelnen Kategorien; (*offline*: vor bzw. *online*: während der CS)

Synchronisierung	synchron parallel $T_l : T = tj$ (Master-Slave-Umsetzung: Subsysteme parallel, Master sequentiell)
Kausalität	Potential-Fluss-Kopplung (Filterung) / (Fluss-Fluss-Kopplung möglich ($\boldsymbol{f}_M$ im Master))
Approximation	$\boldsymbol{a}_j$ Extrapolation ≥ 0. Ordnung & Glättungspolynome / (Kombination aus $\boldsymbol{a}_M$ & $\boldsymbol{a}_j$ möglich)
Makroschrittweiten	fix / Vorgabe $H_l = f(t)$ gemäß Zyklenvorgabe *(offline)*/ variable H_l gemäß (2.29) *(online)*
Ereignisbehandlung	Setzen von T_l gemäß Zyklenvorgabe *(offline)*
Umgebung Subsysteme *(lokal)*	*Dymola* Schnittstellen in Modelica / C
CS-Umgebung *(global)*	*MDPCosim* in C++

4.2. Co-Simulationsumgebung *MDPCosim*

Als Co-Simulationsumgebung für das Verfahren kommt das Programmpaket *MDPCosim* (*M*ulti-*D*isciplinary *P*arallel *Cosim*ulation) zum Einsatz, das im Gegensatz zur oben verwendeten Middleware (*TISC*) nicht kommerziell ist. Es wird in [49] ausführlich vorgestellt und bereits u. A. von [51, 117, 118] angewandt. Im Rahmen der vorliegenden Arbeit wird es erweitert und soll demzufolge an dieser Stelle beschrieben werden.

Es ist in der Sprache C++ verfasst, die wie Modelica eine objektorientierte Programmierung ermöglicht, vergleiche Kapitel 2.2.2. Für die Kommunikation, IPC[1], zwischen den an der CS beteiligten Programmen bietet es verschiedene Möglichkeiten, die in [49] verglichen werden. Sie sind je nach verwendeter Plattform unterschiedlich und bieten jeweils Vor-und Nachteile. In dieser Arbeit kommt als IPC *Shared-Memory* zum Einsatz, da es die effektivste Möglichkeit auf einem Rechner mit mehreren CPUs darstellt. Eine Verteilung von Subsystemen auf mehrere Rechner ist ebenfalls möglich, jedoch mit den erheblich langsameren IPC-Varianten *Message-Passing* oder *Sockets* (etwa für TCP/IP) und kommt hier auch aufgrund der geringen Anzahl an Subsystemen nicht zum Einsatz. Die Umsetzung der Shared-Memory-Kommunikation erfolgt mit Hilfe von *Semaphoren*, die in Abschnitt 4.2.1 beschrieben sind.

[1] Inter Process Communication

Wie bei monolithischer Simulation lässt sich der Ablauf der Co-Simulation in drei Phasen aufteilen: Preprocessing, Lösen der Modellgleichungen und Postprocessing.
1.) Das *Preprocessing* schließt alle konfigurierenden Schritte der CS ein, wie das Anlegen der Subsystemprozesse und das Verbinden anhand der Inzidenzmatrix aus Gleichung (3.3). Das Startprogramm von *MDPCosim* ist in 4.2.3 dargestellt. Im weiteren Sinne sind zu 1.) die (automatische) Partitionierung sowie die Umsetzung der Kausalität und Approximationen zu zählen. **2.)** Die eigentliche Co-*Simulation* $T_0 \rightarrow T_M$ ist bei *MDPCosim* als *Master-Slave*-Architektur umgesetzt. Der Master sowie die Subsysteme (*Slaves*) sind jeweils eigenständige Prozesse und werden in 4.2.1 bzw. 4.2.2 erläutert. **3.)** Dem *Postprocessing* sind die Erstellung von Diagrammen und Statistiken / numerischen Analysen zuzuordnen. Zu Letzteren gehört die Auswertung der in 3.3 eingeführten Kriterien, wie $\bar{\tau}^*_{(m)}$. Dies wird teilweise in *MDPCosim* umgesetzt, wogegen das Visualisieren der Simulationsergebnisse in Diagrammen über die Subsystemprogramme umgesetzt wird.

Im Rahmen dieser Arbeit wird *MDPCosim* in Zusammenarbeit mit dem Lehrstuhl für eine breitere industrielle Anwendung auf Windows portiert und um eine Schnittstelle für die Software *Dymola* erweitert. Der Schwerpunkt des Beitrags liegt dagegen in der Umsetzung neuer numerischer Funktionen, wie der Makroschrittweitensteuerung, erweiterten Schnittstellen sowie Postprocessing oder der Batchfunktionalität des Programmpakets.

4.2.1. Master und Synchronisierung

Stellt *MDPCosim* die CS-*Umgebung* insgesamt dar, so beinhaltet das Masterprogramm den CS-*Algorithmus*. Seine Funktionsweise lässt sich entlang des Ablaufplans in Abb. 4.3 erläutern. Der linke Part beinhaltet Teile des Preprocessings, im rechten befindet sich die

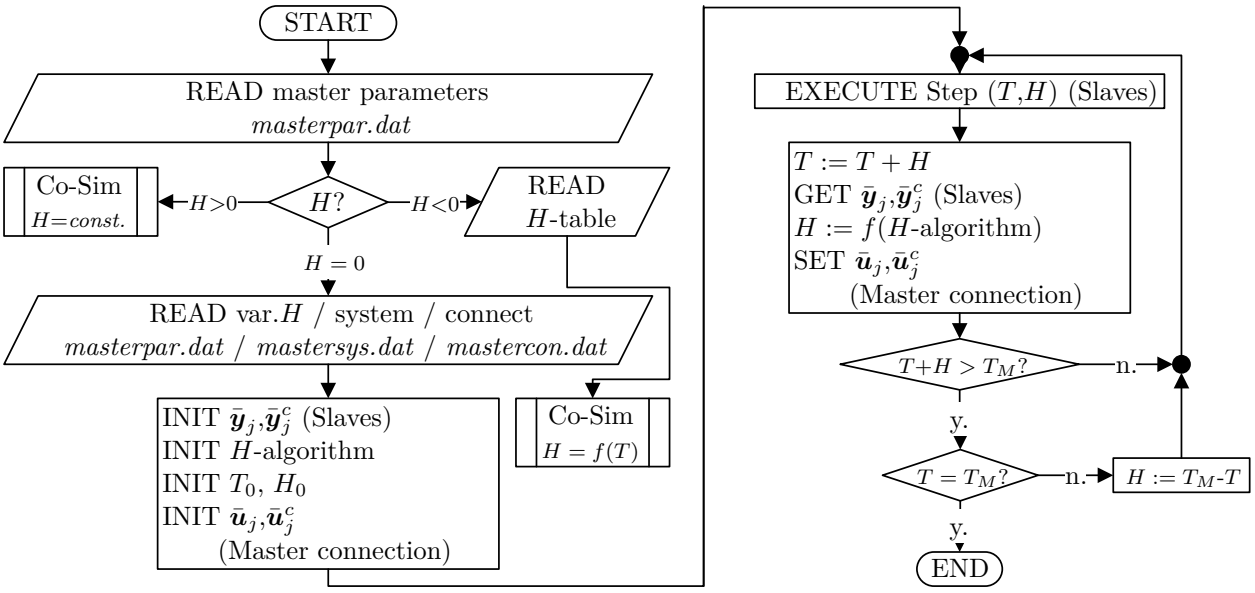


Abbildung 4.3.: Programmablaufplan des Master-Algorithmus

eigentliche Co-Simulationsschleife. Die Konfiguration des Masters erfolgt mittels Parameterdateien, die im Anhang C erläutert werden. Über *masterpar.dat* werden globale Werte, wie T_0, T_M, H bzw. variable Makroschritte $H_l = f(T_l)$ oder die Methode für die Makroschrittweitensteuerung gewählt, wie im Bild dargestellt, und zugehörige Parameter definiert, vergleiche Abschnitt 4.5. Die Schnittstellen für die Slaveprogramme werden über *mastersys.dat* definiert und für die N Subsysteme je ein Objekt einer Kommunikatorklasse angelegt. An dieser Stelle werden die Dimensionen $n_{y,j}$ und $n_{u,j}$ sowie deren zugehörige Kontrollbus-Bandbreite (s.u.) festgelegt. Mit *mastercon.dat* werden die Koppelgesetze $\boldsymbol{f}_M$, die zugehörigen Master-Approximationen $\boldsymbol{a}_M$ sowie die Inzidenzmatrix $\boldsymbol{K}$ konfiguriert. Dabei wird entsprechend der Kausalität für jede Art der Verbindung ein Objekt einer Konnektorklasse angelegt, siehe [49] für Details.

Nach einer Initialisierungsphase, in der erstmalig die Initialwerte $\bar{\boldsymbol{y}}(T_0)$ aus den Modellen abgefragt, sowie H_0 und $\bar{\boldsymbol{u}}(T_0) = \boldsymbol{K}\boldsymbol{a}_M(\boldsymbol{f}_M(\bar{\boldsymbol{y}}(T_0)))$ berechnet werden, beginnt mit dem ersten Makroschritt die CS-Schleife. Sie wird in dieser Arbeit um die Algorithmen für Schrittweitensteuerung erweitert, vgl. 4.5. Bis zum genauen Erreichen von T_M wird sie $M-$fach durchlaufen.

Innerhalb der Schleife wird vom Master die Synchronisierung der Slaves übernommen. Dies geschieht in den *GET* $\boldsymbol{y}$ bzw. *SET* $\boldsymbol{u}$-Methoden und wird über *Semaphoren* gesteuert. Diese haben die Funktionsweise einer Ampel, die ein Weiterlaufen der Schleife vor Beendigung des Makroschritts in den Subsystemen verhindert. Auf diese Wiese wäre jede Art der Synchronisierung aus 3.2.1 möglich. Bei der umgesetzten, parallelen synchronen Variante wartet *GET* $\boldsymbol{y}$, bis alle $\bar{\boldsymbol{y}}_j$ im Shared Memory stehen.

4.2.2. Subsystemaufbau

Slave-Schnittstellen in MDPCosim

Der Aufbau sowie die Ankopplung der Subsysteme werden hier am Beispiel der Modelica-Slaves gezeigt. Sie sind für andere Subsysteme jedoch analog, siehe [49, 117], und der Aufbau der Schnittstellen ist identisch. Abbildung 4.4 zeigt die Architektur der Verbindung zwischen

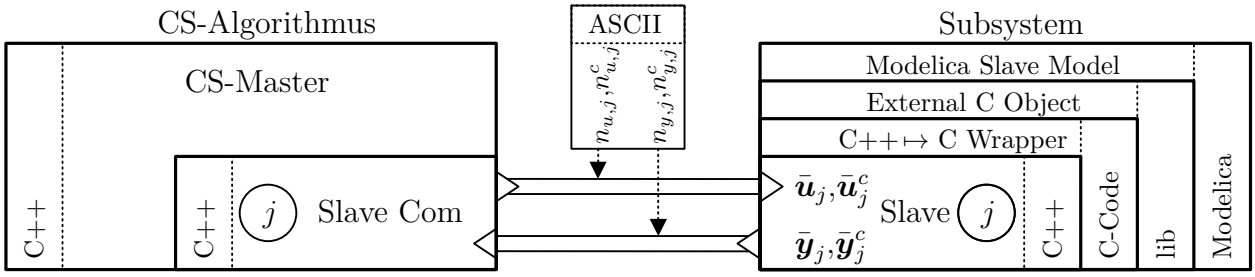
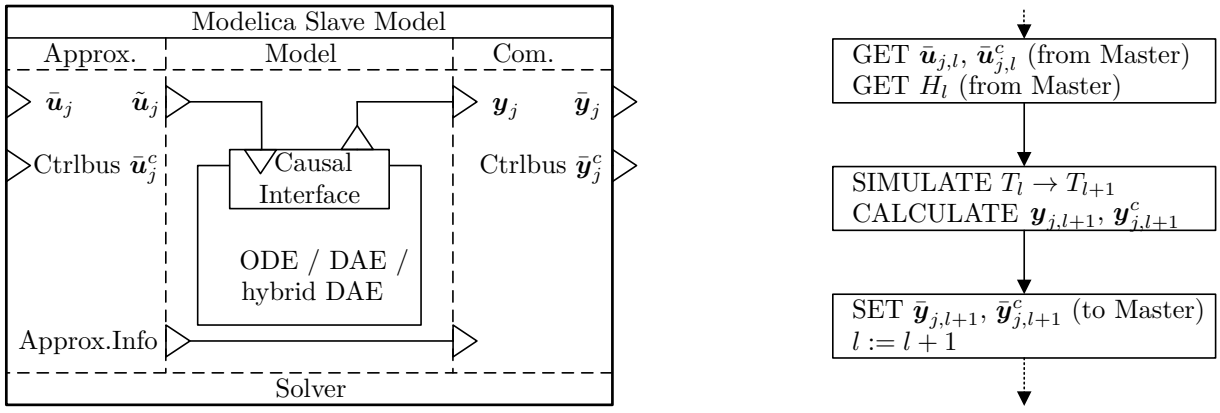


Abbildung 4.4.: Schnittstellenaufbau MDPCosim/Modelica

Master- und Slaveprogramm. Jedem der N Subsysteme entspricht ein *Slave* j, dem im Master ein Objekt j der Kommunikatorklasse *Slave Com* zugeordnet wird, hier für *shared memory*. Beide greifen auf den gleichen Speicherplatz zu und tauschen so ihre Signale aus. Deren Anzahl wird über je ein ASCII-File *Slave<Name>.dat* pro Subsystem konfiguriert, siehe unten bzw. Anhang C. Im rechten Teil der Abbildung ist die Umsetzung für die Anbindung in Modelica dargestellt, wobei der Slave umgebungsunabhängig in C++ realisiert ist. Als API besitzt Modelica die Möglichkeit, External Objects [16], in C umgesetzte Bibliotheken, einzubinden. Über eine zusätzliche Wrapper-Ebene ist der Slave so in den Modelica-Quelltext der Teilmodelle eingebunden.

Modelica-Slave

Die Struktur eines Modelicamodells als CS-Subsystem ist in Abbildung 4.5(a) dargestellt. So wird das bestehende Modell um einen Approximations- sowie Kommunikationsteil erweitert und die physikalischen Konnektoren in den Kausalschnittstellen freigeschnitten, siehe Abschnitt 4.3. Kausalsignale werden direkt weitergegeben. Im Approximationsblock werden die diskreten Eingänge $\bar{\boldsymbol{u}}_j$ über $\boldsymbol{a}_j$ in kontinuierliche Signale gewandelt, was in Abschnitt 4.4 näher untersucht wird. Im Kommunikationsteil ist der Slave-Code eingebunden und die Kontrollbussignale werden ermittelt. Dabei können jeder Koppelgröße $y_{j,i}$ $n^c_{y,j}$ numerische



(a) Struktur des Subsystems (b) Ablauf

Abbildung 4.5.: Modelica-Slaves

Signale, wie deren Ableitungen oder Fehlergrößen $\tau_{j,i}$, zugeordnet werden. Über den Kontrollbus sind weiterhin die Ordnung der $a_{j,i}$ oder $n^i_{y,j}$ zusätzliche Informationen etwa über während des Makroschrittes aufgetretene Ereignisse austauschbar. Entsprechend sind jeder Eingangsgröße $u_{j,i}$ $n^c_{u,j}$ Kontrollsignale zugeordnet, wie z.B. deren Ableitung. Damit ergeben sich die Signaldimensionen $n^*_{y,j}$ bzw. $n^*_{u,j}$ aus der Anzahl der Aus- und Eingangsgrößen sowie der Kontrollbus-Bandbreiten $n^c_{y,j}$ bzw. $n^c_{u,j}$, die den Komponenten in $\bar{\boldsymbol{y}}_j$ bzw. $\bar{\boldsymbol{u}}_j$ zugeordnet

sind, mit $n^i_{y,j}$ Zusätzen zu

$$n^*_{y,j} = (1 + n^c_{y,j})n_{y,j} + n_{u,j} + n^i_{y,j} \qquad n^*_{u,j} = (1 + n^c_{u,j})n_{u,j}. \tag{4.1}$$

Abbildung 4.5(b) zeigt den Ablauf eines Makroschrittes H_l nach Initialisierung in Modelica. Vom Zeitpunkt T_l bis zum Erreichen von T_{l+1} wird das Modell durch den gewählten Solver mit $\tilde{\boldsymbol{u}}_j$ simuliert und die kontinuierlichen Ausgänge $\boldsymbol{y}_j$ berechnet. Sobald T_{l+1} erreicht ist, werden die Kontrollbussignale $\bar{\boldsymbol{y}}^c_j$ berechnet und gemeinsam mit den Ausgangsgrößen $\bar{\boldsymbol{y}}_j$ an den Kommunikator im Master gesendet. Mit Hilfe von Semaphoren wird so lange gewartet, bis die Eingangsgrößen $\bar{\boldsymbol{u}}_j, \bar{\boldsymbol{u}}^c_j$ einschließlich der neuen Schrittweite H_l+1 empfangen werden. Danach wird der nächste Schritt ausgeführt ($l := l + 1$), sofern nicht $t_{j,s}$ erreicht wurde.

4.2.3. Batchverfahren

Als CS-Umgebung initiiert *MDPCosim* die Prozesse des Masters sowie aller Slaves. Somit dient es auch als Rahmen für das Batch-CS-Verfahren, welches automatisiertes Durchführen einer Reihe von Simulationen ermöglicht. Dies erlaubt einerseits besseres Handling etwa bei Parametervariationen im Master zur Entwicklung der CS-Algorithmen, und automatisierte Parametersuche in den Subsystemen andererseits. Weiterhin ist so die Möglichkeit gegeben, effektiver zu simulieren, indem parallel mehrere Simulationen (auf mehreren Rechnern) durchführbar sind, oder Optimierungsprobleme zu bearbeiten.

Für die Batch-Umsetzung wird das Pre-und Postprocessing der Umgebung erweitert, was zum Ablauf gemäß Abb. 4.6 führt. Die Konfiguration erfolgt über die Datei *batchpar.dat*, vgl. Anhang C und beinhaltet vier Optionen: die einfache Co-Simulation und drei Batchvarianten: Parametervariation mit Vorgabewerten; Parametervariation für definierten Wertebereich; Variantenrechnung - dies können verschiedene Parametersätze sein oder die Variation ganzer Subsystemmodelle (via *mastersys.dat*, s.o.) etwa bei Vergleich unterschiedlicher Versionen eines Teilsystems. Die Steuerung der Batch-Läufe erfolgt über Dateien. Die Sequentialisierung erfolgt mittels Semaphoren für das Masterprogramm, welches nach der Initialisierung durch die Umgebung die Synchronisierung sowie das Beenden der Slaveprozesse nach $T = T_M$ übernimmt. Das Postprocessing erfolgt ebenfalls über eine Dateiausgabe. Zur weiteren Auswertung aller Kriterien aus Kapitel 3.3 und Visualisierung können diese Dateien in *Matlab* eingelesen werden. *MDPCosim* ist damit analog zu den meisten monolithischen Umgebungen für Parameterstudien ausgestattet. Als Erweiterung der gesteuerten Batch-CS wäre eine geregelte Batch-CS durch Auslesen einer Zielgröße aus $\bar{\boldsymbol{y}}(T_M)$ umsetzbar, als Lösung für ein Optimierungsproblem oder zur Umsetzung der Synchronisierungsvariante f) in Abb.3.2f) mit kommerziellen Simulationswerkzeugen.

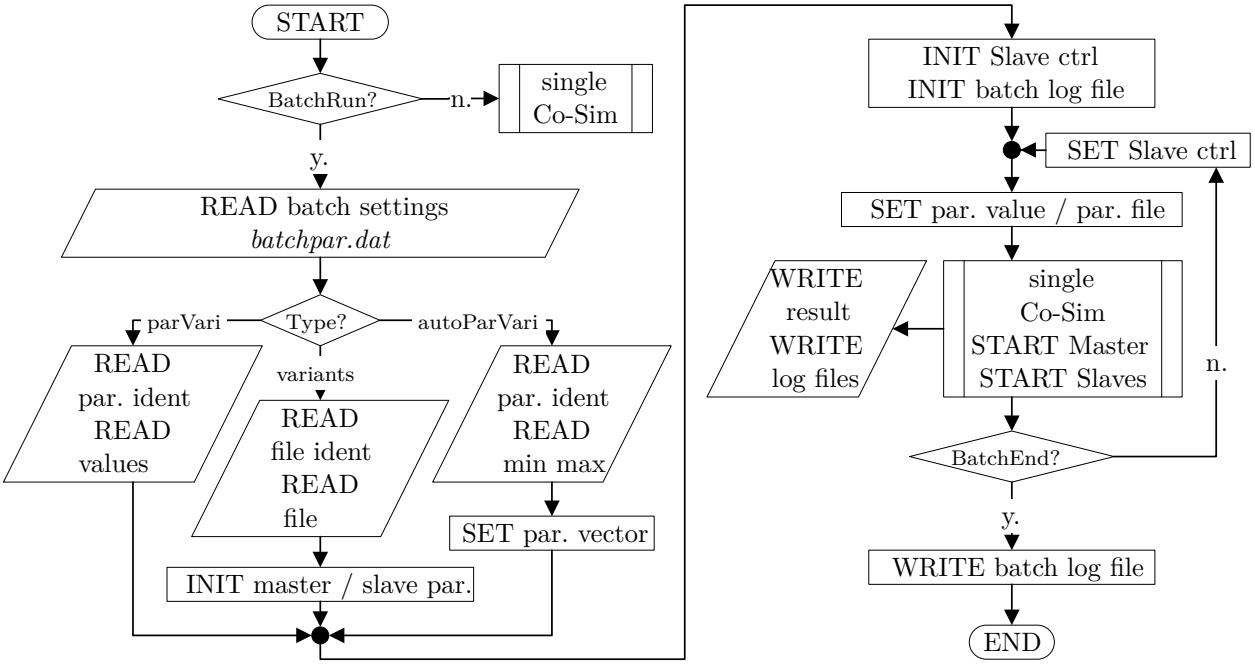


Abbildung 4.6.: Programmablaufplan des Batchverfahrens mit MDPCosim

4.3. Kausalität

Die Slaves sind in der Co-Simulation als Übertragungsblöcke gemäß (2.1) mit den Koppelgrößen als Ein- und Ausgänge zu betrachten. In Kapitel 3.2.2 wurden die notwendigen Lösungsansätze bei physikalischer Modellierung der Teilmodelle, wie vorliegend mit Modelica, erläutert. Die Konnektorgleichungen (2.11) müssen an den Partitionierungsgrenzen aufgebrochen und die Potentialgrößen $\boldsymbol{x}$ und Flussgrößen $\boldsymbol{f}$ kausalisiert werden. Zusätzlich handelt es sich durch die CS um diskrete Größen $\bar{\boldsymbol{u}}$ und $\bar{\boldsymbol{y}}$, d.h. $x = u = \bar{u}_{k_u} \wedge f = u = \bar{u}_{l_u}$, die Eingänge sind somit nicht differenzierbar. Hinzu kommt die Problematik algebraischer Schleifen, vgl. 3.2.2. Für das CS-Verfahren werden Kausalschnittstellen eingeführt, die diesen Anforderungen gerecht werden.

Kausalschnittstellen

Der allgemeine Aufbau der Kausalschnittstellen ist domänenunabhängig in Abbildung 4.7 dargestellt. Grundsätzlich gibt es zwei Arten dieser Schnittstellen: Potentialsender mit dem Potential als Ausgang ($\rightarrow x = y \wedge f = \bar{u}$) und Flusssender mit dem Fluss als Ausgang ($\rightarrow f = y \wedge x = \bar{u}$). Damit möglich sind alle *Potential-Fluss*-Kopplungen sowie *Fluss-Fluss*-Kopplungen, vgl. Abbildung 3.3. Bei Letzteren wird in beiden Teilmodellen ein Potentialsender verwendet und mit einer Kopplung im Master $\boldsymbol{f}_M$ aus der entsprechenden Domäne kombiniert. Für die in dieser Arbeit entwickelte Gesamtfahrzeugsimulation kommt jedoch ausschließlich *Potential-Fluss*-Kopplung zum Einsatz, da auf diese Weise Teilmodelle modular ausgetauscht werden können. Dem Vorteil einer *Potential-Potential*-Kopplung, nur eine

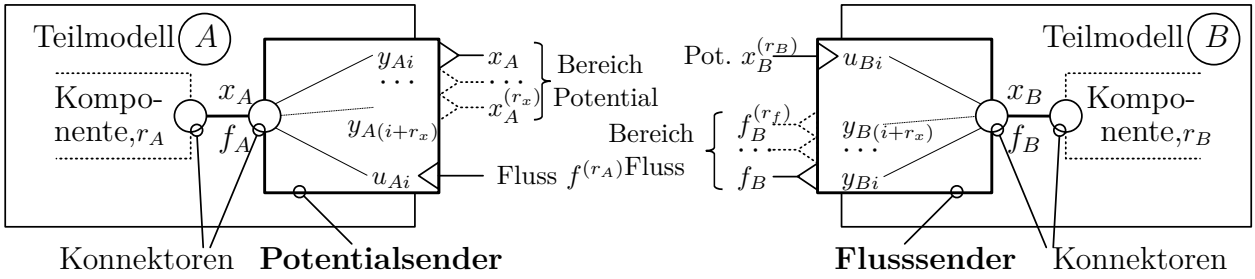


Abbildung 4.7.: Allgemeindarstellung beider Kausalschnittstellen

Art an Kausalschnittstellen zu benötigen, steht das Problem gegenüber, die Teilmodelle jeweils mit identischen Koppelgliedern zu erweitern, vgl. 3.2.2. Dies wäre im Zusammenhang modularer Modelle nicht möglich. Auch bei *Fluss-Fluss*-Kopplung, ebenfalls mit dem Vorteil nur einer Art an Schnittstellen, wäre Modularität aufgrund der benötigten Koppelglieder $\boldsymbol{f}_M$ nicht möglich.

Die Ausgänge der Kausalschnittstellen in Abb. 4.7 decken jeweils die nullte bis r-te Zeitableitung der gemäß Tabelle 2.3 definierten Potential- und Flussgrößen ab, d.h. die Bereiche $[0, r_x]$ bzw. $[0, r_f]$. Aus Sicht der Modellierung mit Konnektoren und in Modelica fest definierten x und f je Domäne, ergäbe sich kein Bereich ($\rightarrow r \equiv 0$), da in den Konnektorgleichungen (2.11) die Größen nur in der nullten Ableitung vorkommen. Jedoch entfallen durch das Preprocessing die trivialen Konnektorgrößen $\boldsymbol{x}$ und $\boldsymbol{f}$, vgl. Kapitel 2.2.1. Des Weiteren gilt

$$x_B = x_A \rightarrow \dot{x}_B = \dot{x}_A \rightarrow \cdots \qquad \sum_{j=1}^{N} f_j = 0 \rightarrow \sum_{j=1}^{N} \dot{f}_j = 0 \rightarrow \cdots \tag{4.2}$$

Somit kann für die Simulation direkt $x_B^{(r_B)}$ in der angeschlossenen Komponente k zur Bestimmung der (differentiellen) Zustandsgröße(n) verwendet werden:

$$\boldsymbol{z}_B = \boldsymbol{f}_k(x_B^{(r_B)} = x_A^{(r_B)} = y_{A(i+r_x)}) \tag{4.3}$$

Da die Junktion (4.2) umgekehrt keine Gültigkeit besitzt, $\neg(\dot{x}_B = \dot{x}_A \rightarrow x_B = x_A)$, muss bei $r > 0$ für konsistente Werte $x_B := x_A$ gesorgt werden. Das Empfangen des gesamten Bereichs $[0, r_x]$ ist nicht möglich, da durch die Diskretisierung *nur* zu T_l gilt: $\bar{y}_{A(i+r_x)}(T_l) = \bar{y}_{Ai}^{(r_x)}(T_l)$. Während des Makroschritts H_l sind diese Signale nicht konsistent, $\bar{y}_{A(i+r_x)}(T \neq T_l) \neq \bar{y}_{Ai}^{(r_x)}(T \neq T_l)\ \forall\ \bar{y}_{A(i+r_x)} \neq 0$. Der Fluss-Sender wird demnach gemäß der in der Komponente benötigten Ableitung r_B gewählt. Die gleiche Betrachtung gilt ebenso für den Potentialsender, und es gibt somit $r_x + 1$ verschiedene Flusssender und $r_f + 1$ verschiedene Potentialsender für jede Domäne, wobei r_f und r_x von dieser abhängen, siehe Abschnitt 4.3.2.

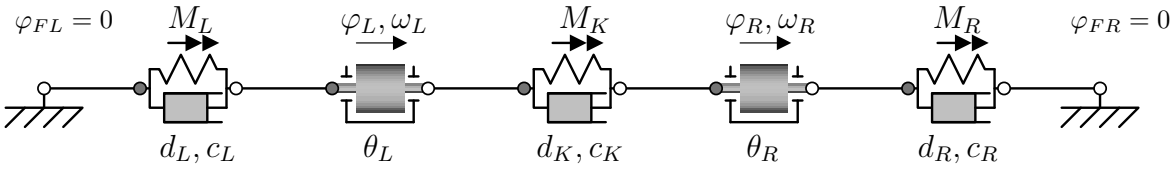
Der Eingang der Flusssender wird mit einem Filter versehen, der zum Einen u_{Bi} glättet, zum Anderen aber notwendig ist, um durch Partitionierung entstandene algebraische Schleifen im

Teilmodell zu lösen, wofür im Preprocessing der DAE-Index reduziert werden muss. Dies wird durch den Filter als Übertragungsglied mit zusätzlichem Zustand erreicht, was ähnlich dem bei $f - f$- und $x - x$ -Kopplung notwendigen Koppelglied die Modellphysik ändert. Aus diesem Grund muss eine problemabhängige Applikation der Filterparameter erfolgen, s. u.

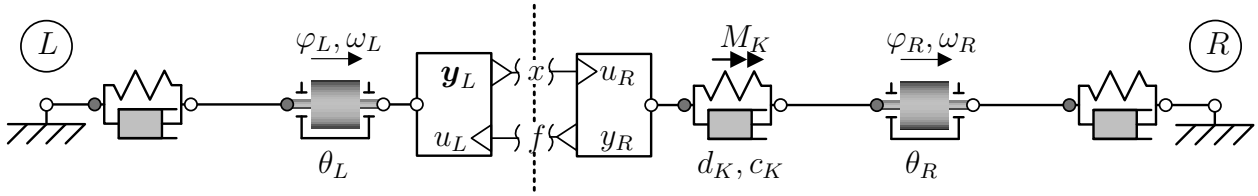
Ein Nachteil der Potential-Fluss-Kopplung sind die zwei Arten an Schnittstellen: sie bilden eine Hierarchie und machen somit die Wahl der Kommunikationsrichtung $A \longrightarrow B$ notwendig. Gemäß Kapitel 3.2.2 hat diese einen Einfluss auf die CS-Genauigkeit, was am folgenden Beispiel gezeigt wird.

4.3.1. Kausalisierung beim Zwei-Massen-Schwinger

Ein Zwei-Massen-Schwinger stellt eine Verkopplung von Potential-bzw. Flussspeichergliedern aus zwei einzeln schwingfähigen Systemen dar und dient somit als Anschauungsbeispiel für numerische Betrachtungen [23, 49, 77]. Wird die mechanische Dämpfung zu null gesetzt, bleibt die Energie im System, (2.14), $E_{mech} = E_{kin} + E_{pot} = konst.$ erhalten, und der Fehler durch numerische Dämpfung wird mit $\Delta E_{mech}(t)$ sichtbar. Abbildung 4.8(a) zeigt das



(a) monolithisches, akausales Referenzsystem



(b) Teilmodelle mit Kausalschnittstellen

Abbildung 4.8.: Zweimassenschwinger und Partitionierung

Gesamtsystem, das aus zwei rotatorischen Schwungmassen, L, R, gemäß (2.17) mit je festem Ende besteht. Im ungekoppelten Fall ($c_K = d_K = 0$) gälte für beide Schwungmassen die Bewegungsgleichung

$$\theta_j \dot{\omega}_j = -c_j \varphi_j - d_j \omega_j \quad \text{mit } j = L, R \tag{4.4}$$

und dem Zustandsvektor $\boldsymbol{z} = [\varphi_j \quad \omega_j]^T$. Die Eigenwerte $\lambda_{j1,2}$ lassen sich mit

$$\lambda_{j1,2} = -D\Omega_0 \pm i\Omega_0\sqrt{1 - D^2} \qquad D = \frac{d_j}{2\Omega_0\theta_j} \qquad \Omega_0 = \sqrt{\frac{c_j}{\theta_j}}. \tag{4.5}$$

bestimmen. Bei Vernachlässigung der Dämpfung, $d_j = 0$, ergibt sich mit Gleichung (2.9)

$$\Lambda = \frac{|\lambda_{j,max}|}{|\lambda_{j,min}|} = \frac{\max\limits_j(\sqrt{\frac{c_j}{\theta_j}})}{\min\limits_j(\sqrt{\frac{c_j}{\theta_j}})} \tag{4.6}$$

als Maß für die *Steifigkeit* des Gesamtsystems beider Massen. Somit tragen stark unterschiedliche Federsteifigkeiten sowie Trägheiten in gleichem Maße zu einem steifen System bei. Die Kopplung wird durch die Verbindung mit einem weiteren Feder-Dämpferglied, c_K, d_K, hergestellt und beeinflusst. Es entsteht das Reaktionsmoment M_K aus Verdrehwinkel $(\varphi_R - \varphi_L)$ bzw. Geschwindigkeitsdifferenz $(\omega_R - \omega_L)$, das in (4.4) zusätzlich eingeht: bei $L: +M_K$, bei $R: -M_K$. Es ergibt sich eine lineares ODE-Modell der Form (2.5):

$$\begin{pmatrix} \dot{\varphi}_L \\ \dot{\omega}_L \\ \dot{\varphi}_R \\ \dot{\omega}_R \end{pmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ \frac{-c_L - c_K}{\theta_L} & \frac{-d_L - d_K}{\theta_L} & \frac{c_K}{\theta_L} & \frac{d_K}{\theta_L} \\ 0 & 0 & 0 & 1 \\ \frac{c_K}{\theta_R} & \frac{d_K}{\theta_R} & \frac{-c_R - c_K}{\theta_R} & \frac{-d_R - d_K}{\theta_R} \end{bmatrix} \begin{pmatrix} \varphi_L \\ \omega_L \\ \varphi_R \\ \omega_R \end{pmatrix} \quad \text{mit } \boldsymbol{z}_0 = \begin{pmatrix} \varphi_{L,0} \\ \omega_{L,0} \\ \varphi_{R,0} \\ \omega_{R,0} \end{pmatrix}, \tag{4.7}$$

ohne Eingänge von außen: $\boldsymbol{u} = \boldsymbol{0}$.

Partitionierung

Wird der Zwei-Massen-Schwinger, wie in Abb. 4.8(b) dargestellt, in einen linken und rechten Teil partitioniert, entstehen daraus die Modelle $A = L$ und $B = R$, die mit den Ein- und Ausgängen der Koppelgrößen erweitert und kausalisiert sind. Der linke Schwinger wird direkt mit dem Potentialsender verbunden, der rechte wird links der Verbindungsfeder freigeschnitten mit dem Flusssender versehen, da dort M_K berechnet wird. Eine umgekehrte Verwendung der Kausalschnittstellen ist nicht möglich, da der Potentialsender dem Potentialspeicherglied, der Masse, zugeordnet sein muss. Im vorliegenden Fall eindimensionaler mechanischer Modellierung gilt für die Schnittstellen $r_x = 2$ sowie $r_f = 0$.

Für *Teilmodell L* wird die rechte Seite von (4.4) um das aufgeprägte Moment M_K als Eingangsgröße $\bar{u}_L = \bar{y}_R = -M_K$ erweitert. Zur besseren Darstellung wird hier auf die Notation der im nächsten Abschnitt behandelten Approximation $\boldsymbol{a}_j$ verzichtet, die in der Eingangsmatrix $\boldsymbol{B}_j$ stünde. L läßt sich nun mit $\boldsymbol{u}_L (= \tilde{u}_L)$ und $\boldsymbol{y}_L$ in der Zustandsform (2.5) darstellen:

$$\begin{pmatrix} \dot{\varphi}_L \\ \dot{\omega}_L \end{pmatrix} = \begin{bmatrix} 0 & 1 \\ \frac{-c_L}{\theta_L} & \frac{-d_L}{\theta_L} \end{bmatrix} \begin{pmatrix} \varphi_L \\ \omega_L \end{pmatrix} + \begin{pmatrix} 0 \\ \frac{-1}{\theta_L} \end{pmatrix} \tilde{u}_L \quad \text{mit } \boldsymbol{z}_0 = \begin{pmatrix} \varphi_{L,0} \\ \omega_{L,0} \end{pmatrix} \quad \text{und } \boldsymbol{y}_L = \begin{pmatrix} \varphi_L \\ \omega_L \\ \dot{\omega}_L \end{pmatrix} \tag{4.8}$$

Das doppelte Vorzeichen bei M_K ist aufgrund der Schnittuferkonvention notwendig, damit der Potentialsender universell einsetzbar ist. $\boldsymbol{y}_L$ ist jedoch über $\dot{\omega}_L = g(\tilde{u}_L)$ vom Eingang

abhängig und würde eine algebraische Schleife darstellen. Da es sich um eine Integration handelt, kann diese durch Verwendung von $\dot{\omega}_L(t_{L,k-1})$ mit $t_{L,k} = T_l$ umgangen werden.

Bei *Teilmodell R* sind aufgrund von $r_x = 2$ prinzipiell drei verschiedene Flussender einsetzbar. Die angrenzende Komponente ist das verbindende Feder-Dämpferglied, wo jedoch zwei Zustände ($\varphi_L \rightarrow r_R = 0$ und $\omega_L \rightarrow r_R = 1$) benötigt werden. Da das Empfangen des Bereichs $[0, r_x]$ nicht möglich ist, vgl. oben, werden im Folgenden die Fälle $r_R = 0$ und $r_R = 1$ betrachtet:

- $r_R = 0; \bar{u}_R = \varphi_L$: das Modell erhält die kausalisierte Form:

$$\begin{pmatrix} \dot{\varphi}_R \\ \dot{\omega}_R \end{pmatrix} = \begin{bmatrix} 0 & 1 \\ \frac{-c_R - c_K}{\theta_R} & \frac{-d_R - d_K}{\theta_R} \end{bmatrix} \begin{pmatrix} \varphi_R \\ \omega_R \end{pmatrix} + \begin{bmatrix} 0 & 1 \\ \frac{c_K}{\theta_R} & \frac{d_K}{\theta_R} \end{bmatrix} \begin{pmatrix} \tilde{u}_R \\ \dot{\tilde{u}}_R \end{pmatrix} \quad \text{mit } \boldsymbol{z}_0 = \begin{pmatrix} \varphi_{R,0} \\ \omega_{R,0} \end{pmatrix} \tag{4.9a}$$

$$y_R = -M_K = -[c_K(\varphi_R - \tilde{u}_R) + d_K(\omega_R - \dot{\tilde{u}}_R)]. \tag{4.9b}$$

  Die Berechnung von $\dot{\omega}_R$ ist nur dann möglich, wenn der Eingang $\tilde{u}_R$ vom Preprocessing des verwendeten Werkzeugs zur Reduzierung des DAE-Index symbolisch zu $\dot{\tilde{u}}_R$ abgeleitet werden kann. Für diesen Fall wäre eine entsprechende Approximation $\boldsymbol{a}_R$ notwendig. Jedoch tritt zusätzlich eine *algebraische Schleife* auf, da im Ggs. zu oben direkt gilt: $y_R = g(\tilde{u}_R)$.

- $r_R = 1; \bar{u}_R = \omega_L$: unter Einführung eines neuen Zustandes φ'_L ergibt sich:

$$\begin{pmatrix} \dot{\varphi}_R \\ \dot{\omega}_R \\ \dot{\varphi}'_L \end{pmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ \frac{-c_R - c_K}{\theta_R} & \frac{-d_R - d_K}{\theta_R} & \frac{c_K}{\theta_R} \\ 0 & 0 & 0 \end{bmatrix} \begin{pmatrix} \varphi_R \\ \omega_R \\ \varphi'_L \end{pmatrix} + \begin{pmatrix} 0 \\ \frac{d_K}{\theta_R} \\ 1 \end{pmatrix} \tilde{u}_R \quad \text{mit } \boldsymbol{z}_0 = \begin{pmatrix} \varphi_{R,0} \\ \omega_{R,0} \\ \varphi'_{L,0} \end{pmatrix} \tag{4.10a}$$

$$y_R = -M_K = -[c_K(\varphi_R - \varphi'_L) + d_K(\omega_R - \tilde{u}_R)]. \tag{4.10b}$$

  Auch hier entsteht wieder eine *algebraische Schleife*, da $y_R = g(\tilde{u}_R)$ in (4.10b).

Der Fall $r_R = 2; \bar{u}_R = \dot{\omega}_L$ wird nicht betrachtet, da in keiner Komponente dieser Zustand verwendet wird.

Lösen der algebraischen Schleife

Neben der Verwendung von Potential-Potential- bzw. Fluss-Fluss-Kopplung besteht bei Potential-Fluss-Kopplung die Möglichkeit der Iteration und der Filterung der Eingangssignale, vgl. Kapitel 3.2.2. Das Ergebnis der Anwendung eines Tiefpassfilters 1.Ordnung (PT_1) mit dem Verhalten $T\dot{z} + z = k\tilde{u}_R$ auf die Eingänge der beiden obigen Fälle ist in der unteren Zeile der Tabelle 4.3 dargestellt. Mit dem Filter wird jeweils ein weiterer Zustand eingeführt. Jedoch entseht bei $r_R = 0$ wieder eine algebraische Schleife mit $y_R = g(\tilde{u}_R)$. Da für $r_R = 1$ insgesamt zwei neue Zustände, φ'_L und ω'_L , hinzukommen, genügt hier die Filterung 1.Ordnung, um die Schleife zu lösen, so dass gilt: $y_R \neq g(\tilde{u}_R)$. Im Allgemeinen ist r_B

Tabelle 4.3.: Fallunterscheidung der Flussender von *Teilmodell R*

Fall	$r_R = 0; \bar{u}_R = \varphi_L$	$r_R = 1; \bar{u}_R = \omega_L$
$\tilde{u}_R$	$\boldsymbol{z}_R = [\varphi_R \quad \omega_R]^T \in \mathbb{R}^2$ $y_R = -[c_K(\varphi_R - \tilde{u}_R) + d_K(\omega_R - \dot{\tilde{u}}_R)]$	$\boldsymbol{z}_R = [\varphi_R \quad \omega_R \quad \varphi'_L]^T \in \mathbb{R}^3$ $\dot{\varphi}'_L = \tilde{u}_R$ $y_R = -[c_K(\varphi_R - \varphi'_L) + d_K(\omega_R - \tilde{u}_R)]$
$\tilde{u}_{R,PT_1}$	$\boldsymbol{z}_R = [\varphi_R \quad \omega_R \quad \varphi'_L]^T \in \mathbb{R}^3$ $\dot{\varphi}'_L = \frac{k\tilde{u}_R - \varphi'_L}{T}$ $y_R = -[c_K(\varphi_R - \varphi'_L) + d_K(\omega_R - \frac{k\tilde{u}_R - \dot{\varphi}'_L}{T})]$	$\boldsymbol{z}_R = [\varphi_R \quad \omega_R \quad \varphi'_L \quad \omega'_L]^T \in \mathbb{R}^4$ $\dot{\varphi}'_L = \omega'_L; \quad \dot{\omega}'_L = \frac{k\tilde{u}_R - \omega'_L}{T}$ $y_R = -[c_K(\varphi_R - \varphi'_L) + d_K(\omega_R - \omega'_L]$

in Abhängigkeit der Domäne und Komponente nicht frei wählbar. So wird für die numerischen Experimente am Zwei-Massenschwinger $d_{j,K} = 0$ gesetzt, was die Verwendung des Verdrehwinkels φ nahelegt, d.h. $r_R = 0$. Dies kann mit einem Filter 2.Ordnung geschehen, so dass ebenfalls zwei Zustände hinzukommen und $y_R \neq g(\tilde{u}_R)$ gilt. Für die Gesamtfahrzeugsimulation wird der Flusssender mit $r_j = 1$ und Filterung 1.Ordnung verwendet, da dort die Drehzahlen anstelle der -winkel im Antriebsstrang relevant sind. Die Grenzfrequenz des Filters muss entsprechend an $\omega'_{0,j}$ H und $\boldsymbol{a}_j$ angepasst werden.

Kopplung und Richtung

Im geschlossenen Gesamtsystem (4.7) der Form $\dot{\boldsymbol{z}} = \boldsymbol{f}(\boldsymbol{z}) = \boldsymbol{A}\boldsymbol{z}$ entspricht die Systemmatrix der Jacobimatrix, $\boldsymbol{J}_f = \boldsymbol{A}$. Nach [77] geben deren Einträge $J_{f,ij;i\neq j}$ direkt die Stärke und Richtung der *Kopplung* der einzelnen Zustände an. Setzt man die Einträge von $\boldsymbol{J}_f$ in (2.36) ein, gilt für die *dynamische* Kopplungsrichtung $\boldsymbol{\kappa}_{LR,RL}$ der jeweiligen Teilschwinger sowie für die Kopplungsstärke $K_{(LR)}$ des Gesamtsystems aus L und R im ungedämpften Fall

$$\boldsymbol{\kappa}_{LR} = \sum_i \frac{\partial \boldsymbol{f}_R}{\partial z_{L,i}} \frac{dz_{L,i}}{dt} = \frac{1}{\theta_R} \begin{pmatrix} 0 \\ c_K\dot{\varphi}_L + d_K\dot{\omega}_L \end{pmatrix}; \quad \boldsymbol{\kappa}_{RL} = \frac{1}{\theta_L} \begin{pmatrix} 0 \\ c_K\dot{\varphi}_R + d_K\dot{\omega}_R \end{pmatrix} \tag{4.11}$$

$$K_{(LR)}(d_K = 0) = |\boldsymbol{\kappa}_{LR}|\,|\boldsymbol{\kappa}_{RL}| = \frac{c_K^2\dot{\varphi}_L\dot{\varphi}_R}{\theta_L\theta_R}. \tag{4.12}$$

Man sieht, dass die dynamische Kopplung von der Steifigkeit Λ durch resultierende Unterschiede der $\dot{\varphi}_j$ beeinflusst wird. Jedoch haben v. A. die Federsteifigkeit und die Trägheiten Einfluss auf $K_{(LR)}$. Es lässt sich die vom Verhalten der Teilmodelle unabhängige *statische* Kopplungsrichtung

$$K_{LR} = \left|\sum_i \frac{\partial \boldsymbol{f}_R}{\partial z_{L,i}}\right| \left|\sum_i \frac{\partial \boldsymbol{f}_L}{\partial z_{R,i}}\right|^{-1} = K_{RL}^{-1} \tag{4.13}$$

definieren, die hier über $K_{LR} = \frac{\theta_L}{\theta_R}$ nur vom Verhältnis der Trägheiten abhängt. Der Einfluss der Kopplungsrichtung auf die CS soll anhand des Fehlers $\hat{\tilde{\tau}}^*_{(m)} = \hat{\tilde{\tau}}^*_{(m)}(T_M)$ aus (3.23) für $\theta_L << \theta_R$, $K_{LR} = 0,01$ und die umgekehrte Richtung $K_{LR} = 100$ dargestellt werden.

Die beiden Fälle werden jeweils ohne Approximation ($\tilde{u}_j = \bar{u}_j$) und bei unterschiedlichem Λ, r_R und H^* betrachtet. Da der ungedämpfte Fall ($d_L = d_K = d_R = 0$) gewählt ist, wird der CS-Fehler kumuliert durch eine Abweichung von der Energieerhaltung $\Delta E_{mech}(t)$ sowohl für Phase als auch Amplitude sichtbar. Gemäß (2.14) fließen alle Modellzustände gleichermaßen in die Berechnung ein ($E \sim z_{k_j}^2$), was alle betrachteten Kopplungen anhand einer Größe, $\hat{\bar{\tau}}^*_{(m)}$ von E_{mech}, vergleichbar macht. Die Ergebnisse zeigt Abbildung 4.9. φ_L und

Λ	r_R	H^*	K_{LR}
100	0	$5 \cdot 10^{-5}$	$0,01$
			100
1	0	$5 \cdot 10^{-5}$	$0,01$
			100
		$1 \cdot 10^{-4}$	$0,01$
			100
	1	$5 \cdot 10^{-5}$	$0,01$
			100
		$1 \cdot 10^{-4}$	$0,01$
			100

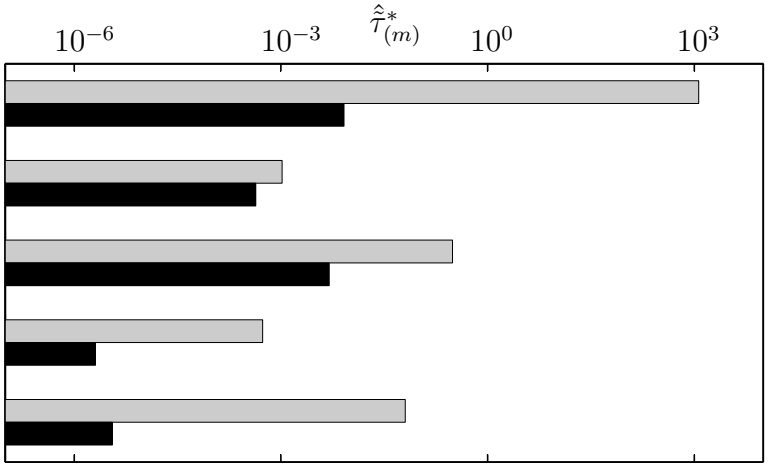


Abbildung 4.9.: Einfluss der Steifigkeit Λ und Kopplungsparameter r_R, H^* und K_{LR} beim Zweimassenschwinger

ω_L schwingen mit derselben Frequenz, so dass für $r_R = 0$ und $r_R = 1$ im Beispiel die gleiche Filtergrenzfrequenz verwendet wird mit $\Omega_{krit} = 100\pi/s$. Die Initialisierungswerte sind jeweils $\boldsymbol{z}_0^T = [\boldsymbol{z}_{0,L}^T \quad \boldsymbol{z}_{0,R}^T] = [\varphi_0 \quad 0 \quad \varphi_0 \quad 0 \quad \varphi_0] \forall r_R = 0$ bzw. $\boldsymbol{z}_0^T = [\varphi_0 \quad 0 \quad \varphi_0 \quad 0 \quad \varphi_0 \quad 0] \forall r_R = 1$. Ungeachtet der Richtung besitzen alle Modelle den gleichen Wert $\frac{K_{(LR)}}{\dot{\varphi_L}\dot{\varphi_R}} = 1$, d. h. statisch die gleiche Gesamtkopplungsstärke.

Erstens ist in Abb 4.9 der Einfluss der Steifigkeit auf den Fehler zu sehen, der bei $K_{LR} = 0,01$ sechs Größenordnungen beträgt, bei $K_{LR} = 100$ jedoch auf eine Größenordnung abnimmt. Zweitens zeigt sich entgegen obiger Annahme, dass trotz Vernachlässigung der Dämpfung die Verwendung von $r_R = 1$ bessere Ergebnisse liefert, was auf die Filterung 2.Ordnung bei $r_R = 0$ zurückzuführen ist. Dies gilt drittens auch bei unterschiedlicher Makroschrittweite, deren Einfluss erwartungsgemäß sichtbar wird, jedoch bei $r_R = 1$ und $K_{LR} = 100$ am geringsten ausfällt. Viertens zeigt die Untersuchung, dass in allen Fällen, die Richtung $K_{LR} = 100$ die weitaus genaueren Ergebnisse liefert. Dies begründet die Modellierungsvorschrift, den Potentialsender stets im Teilmodell mit der größten Masse zu verwenden.

Nicht im Bild dargestellt ist der globale Fehler $\hat{\bar{\tau}}^* = \hat{\bar{\tau}}^*(T_M)$, der sich prinzipiell analog zum dargestellten *mittleren* globalen Fehler $\hat{\bar{\tau}}^*_{(m)}$ verhält. Bei $r_R = 1$ oszilliert $\hat{\bar{\tau}}^*(T)$ jedoch mit der mehrfacher Modellfrequenz, was der Betrachtungsweise der CS als Energiespeicher entspricht, vgl. Kapitel 3.3, und den Vergleich über $\hat{\bar{\tau}}^*_{(m)}$ nötig macht.

Grenzfrequenz

In obigen Simulationen wurde jeweils die gleiche Grenzfrequenz parametriert. Folgend soll deren Einfluss auf den Fehler $\hat{\bar{\tau}}^*_{(m)}$ für beide Richtungen K_{LR} untersucht werden. Dem gegenüber wird der reine Einfluss der gleichen Filterung bei Einsatz der Kausalschnittstellen in einer monolithischen Simulation gestellt. Abbildung 4.10 zeigt $\hat{\bar{\tau}}^*_{(m)}$ über der Grenzfrequenz

$$\Omega^*_{krit} = \frac{\Omega_{krit}}{\Omega_{0,L}}, \tag{4.14}$$

bezogen auf die bei allen konstante Frequenz des linken Schwingers. Es gelten jeweils $\Lambda = 1$ und $r_R = 1$. Ω_{krit} ist für $1,1\Omega_{0,L}; 10\Omega_{0,L}; 100\pi/s; 1,1H$ und $10H$ dargestellt. Anhand des

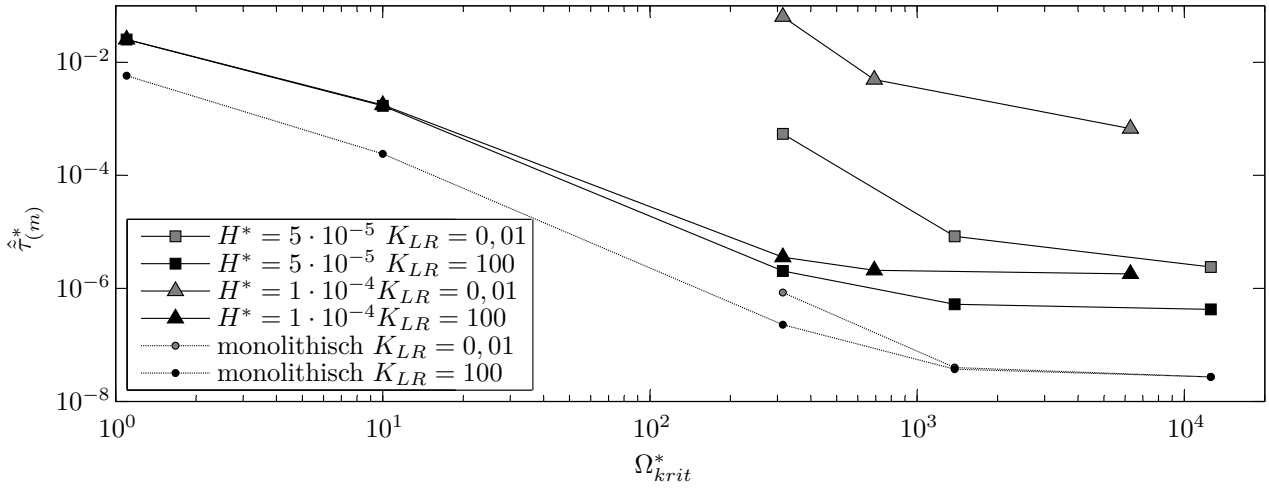


Abbildung 4.10.: $\hat{\bar{\tau}}^*_{(m)}$ in Abhängigkeit der Filterfrequenz Ω^*_{krit}

Fehlers der monolithischen Rechnungen ist sichtbar, dass durch Einführen der Filterung in die Modellphysik eingegriffen wird. Mit zunehmender Entfernung der Durchlassfrequenz von jener der Schwingung der linken Masse nimmt er jedoch ab. Ist diese kleiner als die rechte, $K_{LR} = 0,01$, ist der Fehler entsprechend größer und führt bei $\Omega^*_{krit} < 10^2$ zu instabilem Verhalten (nicht mehr dargestellt). Der Unterschied in der Richtung verschwindet bei hinreichend großer Frequenz $\Omega^*_{krit} > 10^3$ ebenso wie der Fehler aufgrund der Filterung. Infolge des Diskretisierungsfehlers bei den CS-Ergebnissen liegt das Niveau von $\hat{\bar{\tau}}^*_{(m)}$ mit steigendem H^* höher, wobei der Disketisierungsfehler im rechten Bildteil überwiegt. Die Abhängigkeit von der Filterfrequenz bleibt jedoch sichtbar. Da der Aufwand mit Ω^* zunimmt und der Fehler nicht weiter gesenkt wird, sind für die CS sind somit Werte von $\Omega^*_{krit} \approx 10^4$ zu wählen.

4.3.2. Umsetzung der Schnittstellen

Folgend werden die Kausalschnittstellen in Modelica im Kontext der CS-Schnittstelle aus Abbildung 4.5 dargestellt. Alle für die Gesamtfahrzeugsimulation notwendigen Domänen

sind als Bibliothek *CausalInterfaces*, im linken Teil der Abbildung 4.11, modelliert. Hier gilt

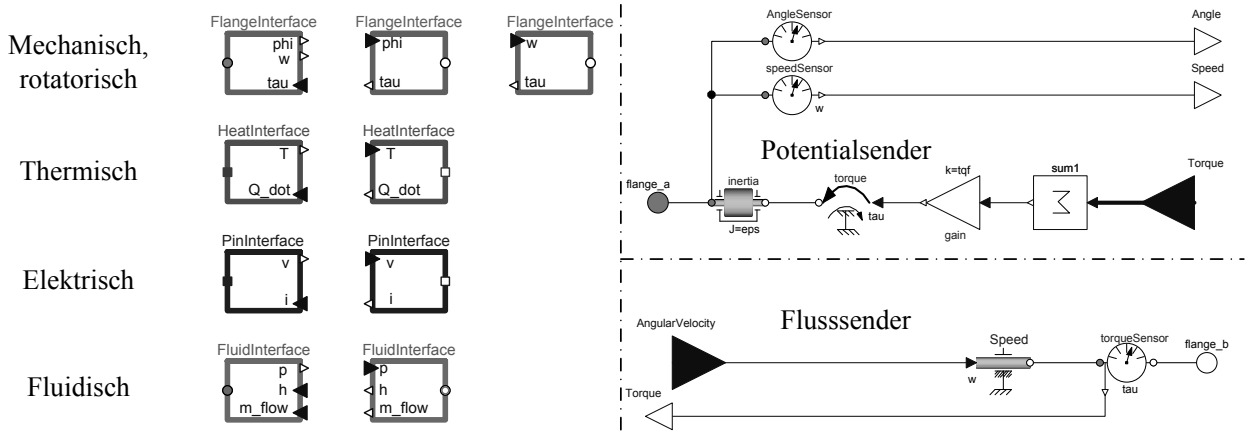


Abbildung 4.11.: Umsetzung der Kausalschnittstellen in Modelica

überall $r_f = 0$, so dass ganz links die Potentialsender, daneben die Flusssender und von oben nach unten die verschiedenen Domänen mit den Variablen aus Tabelle 2.3 als Ein-und Ausgänge zu sehen sind. In der Fluidik sind auch kompressible Medien zulässig sowie kalorische Größen mitgeführt, so dass zusätzlich die Streamvariablen ausgetauscht werden müssen. Da sich die Anwendung bisher auf die Partitionierung im Abgasstrang beschränkt, ist das Konzept der Streamvariablen für die CS nur für eine Fließrichtung umgesetzt. Weiterhin sind die Modelle der Fluidschnittstellen erst mit Parametrierung der Medieneigenschaften, die sie von den Subsystemen übertragen bekommen, vollständig formuliert.

Der rechte Teil der Abbildung stellt exemplarisch die oben diskutierten Schnittstellen für den 1D-rotatorischen Fall vor, welcher im Antriebsstrang verwendet wird. Oben ist der Potentialsender, darunter der Flussender für $r_x = 1$ gezeigt. Die *Potentialsender* beinhalten jeweils eine dynamische Speicherkomponente K (Im Bsp. die Masse θ_K). Diese ermöglicht volle Modularität mit der Modellierungsfreiheit, ein Subsystem ohne Zustand in der entsprechenden Domäne anzuschließen. Im *Default*fall sind sie mit dem kleinsten darstellbaren Wert ($\theta_K = \epsilon$) vorbedatet, sodass K erstens vernachlässigbar klein wird ($\theta_K << \theta_j$) und zweitens, da vom Preprocessing direkt der Nachbarkomponente j zugeschlagen ($\theta_j := \theta_K + \theta_j$), keine zusätzlichen Zustände entstehen. Der *gain*-Block, welcher die Vorzeichenkonvention sicherstellt, ermöglicht eine parametrierbare An-/Abkopplung ganzer Subsysteme. So können etwa Nebenaggregate mechanisch von der Kurbelwelle entkoppelt werden, ohne das Modell zu ändern. Die Filterung der Potentialgröße im *Flusssender* ist in der Modelica-Textebene umgesetzt und so über eine *if-else*-Verzweigung schaltbar. Dadurch können die Flusssender gleichermaßen für CS- wie monolithische Anwendung eingesetzt werden, wo durch den Wegfall von $\bar{u}$ und $\bar{y}$ nicht gefiltert werden muss.

Analog zur Möglichkeit, Konnektoren von $N > 2$ Komponenten zu verbinden, ist es möglich, mehrere Flusssender mit einem Potentialsender zu verbinden. Dazu ist dessen Eingang vek-

torwertig modelliert, so dass zunächst die Summe der eingehenden Flussvariablen gebildet wird: $-f_j = \sum f_{i \neq j} = \sum \tilde{u}_{ji}$, wobei x_j an alle $N-1$ Flusssender geht.

Mit den Kausalschnittstellen ist die Partitionierung innerhalb der physikalischen Domänen nahezu beliebig möglich. Jedoch erfordert die Verbindung zweier Flussspeicherglieder -etwa Massen, die einen Potentialsender erfordern- eine dieser zwei Maßnahmen: entweder wird ein Koppelglied auf Seiten des kleineren Speichergliedes eingeführt, siehe oben, oder die Massen werden in einem Subsystem zusammengefasst. Eine Partitionierung an der Grenze zweier Domänen ist nicht möglich, da die Interaktion zwischen ihnen innerhalb einer Komponente, wie etwa einer Pumpe, modelliert ist. Hier ist es möglich, an der Pumpennabe zu schneiden und eine mechanische Kausalschnittstelle zu verwenden. Die Masse des Pumpenrads kann nun dessen im anderen Subsystem befindlicher Welle zugeordnet werden.

Die algebraische Schleife, wie sie etwa in (4.8) auftritt, lässt sich unter der Annahme, dass $H_{l-1} >> h_{j,k-1}$ und somit die kontinuierliche lokale Änderung $\Delta \boldsymbol{y}_j(h_{j,k-1}) << \bar{\boldsymbol{\tau}}(H_{l-1})$ ist, generell auflösen. Hierfür wird im Kommunikationsteil des Slave-Modells gemäß Abb. 4.5 der Ausgangsvektor $\bar{\boldsymbol{y}}_j$ mit dem letzten gültigen Mikroschrittwert beschrieben:

$$\bar{\boldsymbol{y}}_j(T_l) = \boldsymbol{y}_j(t_{j,k-1}) \quad \text{mit } t_{j,k} = T_l. \tag{4.15}$$

4.4. Approximation

Die diskreten Subsystemeingänge $\bar{\boldsymbol{u}}$ werden zu kontinuierlichen Verläufen $\tilde{\boldsymbol{u}}$ gewandelt, welche die Verläufe in $\boldsymbol{y}$ approximieren sollen. Dazu gibt es Verfahren zur Interpolation oder Extrapolation und Glättung. Dem Konzept paralleler Kopplung in 4.1.1 und Bild 4.5 entsprechend, werden hier Verfahren gemäß Gleichung (3.9) verwendet. Entsprechend (3.5) gilt die Schreibweise $\bar{\boldsymbol{u}}_{j,l}$ für die Koppelwerte $\boldsymbol{u}_j(T_l)$. Nach Kapitel 3.2.3 lassen sich die Verfahren in $\boldsymbol{a}_j$ nach dem Wert r der höchsten verwendeten Ableitung $\bar{\boldsymbol{u}}_{j,l}^{(r)}$, nach der Anzahl L verwendeter Makroschrittstützstellen und den Verfahrensparametern einteilen. Dabei ist $L = 1 \wedge n = 0$ der Trivialfall $\tilde{\boldsymbol{u}}_j(t_j) = \bar{\boldsymbol{u}}_{j,l}$. Für $L = 1 \wedge r \geq 0$ kommen *Taylor*polynome n-ten Grades zum Einsatz:

$$\tilde{\boldsymbol{u}}_j(t_j) = \sum_{r=0}^{n} \frac{\bar{\boldsymbol{u}}_{j,l}^{(r)}}{r!}(t_j - T_l)^r \tag{4.16}$$

Für $L \geq 1 \wedge r = 0$ wird aus (3.9) unter Verwendung von *Lagnrange*polynomen:

$$\tilde{\boldsymbol{u}}_j(t_j) = \sum_{i=0}^{L-1} \bar{\boldsymbol{u}}_{j,l-i} \ell_{l-i}(t_j) \tag{4.17a}$$

$$\text{mit } \ell_{l-i}(t_j) = \prod_{\substack{p=0 \\ p \neq i}}^{L-1} \frac{t_j - T_{l-p}}{T_{l-i} - T_{l-p}} \tag{4.17b}$$

Sie haben den Grad $n = L - 1$ und für das Basispolynom ℓ_l gilt im Trivialfall:

$$n = 0 \rightarrow \ell(L = 1) \equiv 1 \tag{4.18}$$

Für $L \geq 1 \wedge r > 0$ werden *Hermite*polynome verwendet, die für $r = 1$ lauten:

$$\tilde{\boldsymbol{u}}_j(t_j) = \sum_{i=0}^{L-1} [\bar{\boldsymbol{u}}_{j,l-i} \tilde{H}_{l-i}(t_j) + \bar{\dot{\boldsymbol{u}}}_{j,l-i} \tilde{h}_{l-i}(t_j)] \tag{4.19a}$$

$$\text{mit } \tilde{H}_{l-i}(t_j) = [1 - 2\frac{d}{dt_j}\ell_{l-i}(t_j)(t_j - T_{l-i})]\ell^2_{l-i}(t_j) \tag{4.19b}$$

$$\text{und } \tilde{h}_{l-i}(t_j) = (t_j - T_{l-i})\ell^2_{l-i}(t_j), \tag{4.19c}$$

sowie ℓ_{l-i} aus (4.17b), welche den Grad $n = 2(L-1)+1$ besitzen. In den drei vorgenannten Verfahren sind die Polynomkoeffizienten $\boldsymbol{p}$ jedem n entsprechend vorgegeben. Ein allgemeinerer Ansatz wäre, diese Parameter frei wählbar zu lassen und zu optimieren. Ein solches Vorgehen soll an dieser Stelle nicht näher vertieft werden. Es sei jedoch auf die Arbeiten [49, 51] verwiesen, wo es für die Approximation in $\boldsymbol{a}_M$ und $L \geq 1 \wedge r = 1$ ausführlich beschrieben ist. Die Hermiteextrapolation (4.19) mit $r = 1$ entspricht bei Verwendung von nur einer Stützstelle, $L = 1$ mit (4.17b) in (4.19b) und (4.19c) über

$$\begin{aligned} n = 1 \rightarrow \ell(L = 1) \equiv 1 &\rightarrow \tilde{H}_{l-i}(t_j) = 1 \\ &\rightarrow \tilde{h}_{l-i}(t_j) = t_j - T_{l-i} \end{aligned} \tag{4.20}$$

genau dem Taylorpolynom ersten Grades. Dieses wiederum entspricht dem expliziten Euler-Integrationsverfahren, vergleiche Kapitel 2.3.1.

Ein Nachteil ist die nicht stetige Änderung von der kontinuierlichen Extrapolation $\tilde{\boldsymbol{u}}_{j,H_{l-1}}(t_j)$ im alten Zeitschritt H_{l-1} zur kontinuierlichen Extrapolation $\tilde{\boldsymbol{u}}_{j,H_l}(t_j)$ im neuen Zeitschritt H_l zum Zeitpunkt T_l. Um dies zu vermeiden, kann ein stetiger Verlauf zwischen den beiden Funktionen erzwungen werden. Dies stellt die Änderung von y besser dar und erleichtert die Reinitialisierung der lokalen Solver, was zu einer Steigerung von S_{CS} führt. Dafür wird ein über den Parameter

$$T^*_G = \frac{T_G - T_l}{T_{l+1} - T_l} \tag{4.21}$$

normierter Übergangspunkt T_G definiert, ab dem $\tilde{\boldsymbol{u}}_{j,H_l}(t_j)$ genutzt wird. Es wird weiter eine normierte lokale Zeit

$$t^*_j = \frac{t_j - T_l}{T_G - T_l} \tag{4.22}$$

definiert, innerhalb der nun eine Interpolation zwischen den Werten $\tilde{\boldsymbol{u}}_{j,H_{l-1},l}$ und $\tilde{\boldsymbol{u}}_{j,H_l,G}$ möglich ist. Dafür kommen wieder Hermitepolynome als *Splines* $\gamma(t^*_j)$ zum Einsatz. Mit

diesem Ansatz lautet die Vorschrift der Approximation für den Makroschritt H_l wie folgt:

$$\tilde{\boldsymbol{u}}_j(t_j) = \begin{cases} \gamma(t_j, \tilde{\boldsymbol{u}}^{(r)}_{j,H_{l-1},l}, \tilde{\boldsymbol{u}}^{(r)}_{j,H_l,G}) & \text{für } T_l \leq t_j < T_G \quad \text{und } r = 0, ..., n \\ \tilde{\boldsymbol{u}}_{j,H_l}(t_j) & \text{für } T_G \leq t_j < T_{l+1} \end{cases} \tag{4.23}$$

Soll nur ein stetiger Verlauf erzwungen werden, lautet der Spline ersten Grades

$$\gamma(t_j^*(t_j)) = (1 - t_j^*)\tilde{\boldsymbol{u}}_{j,H_{l-1},l} + t_j^*\tilde{\boldsymbol{u}}_{j,H_l,G}. \tag{4.24}$$

Für den Grenzfall $T_G^* = 1$ und der Verwendung von Lagrangepolynomen mit $L = 2$ für $\tilde{\boldsymbol{u}}_{j,H_l,G}$ entspricht dies der *interpolierten Extrapolation* aus [82]. Soll der gesamte Verlauf $\tilde{\boldsymbol{u}}_j(t_j)$ stetig differenzierbar sein, so muss ein kubischer hermitescher Spline

$$\begin{aligned} \gamma(t_j^*(t_j)) =& (2t_j^{*3} - 3t_j^{*2} + 1)\tilde{\boldsymbol{u}}_{j,H_{l-1},l} + (-2t_j^{*3} + 3t_j^{*2})\tilde{\boldsymbol{u}}_{j,H_l,G} \\ &+ (t_j^{*3} - 2t_j^{*2} + t_j^*)\tilde{\dot{\boldsymbol{u}}}_{j,H_{l-1},l}(T_G - T_l) + (t_j^{*3} - t_j^{*2})\tilde{\dot{\boldsymbol{u}}}_{j,H_l,G}(T_G - T_l) \end{aligned} \tag{4.25}$$

verwendet werden. Dieser benötigt zusätzlich die Werte der Ableitung $\tilde{\dot{\boldsymbol{u}}}_j$ der Extrapolation. Der von [83] vorgestellte Ansatz der Glättung schaltet kontinuierlich zwischen den Verläufen $\tilde{\boldsymbol{u}}_{j,H_{l-1}}(t_j)$ und $\tilde{\boldsymbol{u}}_{j,H_l}(t_j)$ um. Er ist damit im Vgl. zum Spline-Ansatz (4.23) um die Ordnung der verwendeten Extrapolationsfunktion genauer. Gemäß [83] lautet die Vorschrift der Approximation für den Makroschritt H_l wie folgt:

$$\tilde{\boldsymbol{u}}_j(t_j) = \begin{cases} \gamma(t_j)\tilde{\boldsymbol{u}}_{j,H_{l-1}}(t_j) + (1 - \gamma(t_j))\tilde{\boldsymbol{u}}_{j,H_l}(t_j) & \text{für } T_l \leq t_j < T_G \\ \tilde{\boldsymbol{u}}_{j,H_l}(t_j) & \text{für } T_G \leq t_j < T_{l+1} \end{cases} \tag{4.26}$$

Die Übergangsfunktion $\gamma(t_j^*)$ besteht wiederum aus Polynomen, die sich aus den Randbedingungen ergeben. Ist nur ein stetiger Verlauf gefordert, ergibt sich

$$\gamma(t_j^*(t_j)) = 1 - t_j^*. \tag{4.27}$$

Soll $\tilde{\boldsymbol{u}}_j(t_j)$ zusätzlich stetig differenzierbar sein

$$\gamma(t_j^*(t_j)) = 2t_j^{*3} - 3t_j^{*2} + 1. \tag{4.28}$$

Aus der Bedingung zweifach stetiger Differenzierbarkeit ergibt sich schließlich das Polynom fünften Grades:

$$\gamma(t_j^*(t_j)) = -6t_j^{*5} + 15t_j^{*4} - 10t_j^{*3} + 1. \tag{4.29}$$

4.4.1. Analyse der Approximationsverfahren

Extrapolation

Ausführliche Konvergenz- und Stabilitätsanalysen anhand exakter Lösungen werden bereits u. A. in [6, 23, 114] für $\boldsymbol{a}_j$ sowie in [49, 51] für $\boldsymbol{a}_M$ behandelt. Im Hinblick auf die Anwendung ist eine Analyse der Verfahren im Gesamtkontext der CS mit Beeinflussung des lokalen Solvers sinnvoll und wird demnach hier mittels numerischer Experimente durchgeführt. Als Testmodell dient wieder der Zwei-Massen-Schwinger aus Abbildung 4.8(b). Im Gegensatz zur Literatur, in der für alle Komponenten in $\tilde{\boldsymbol{u}}_j$ das gleiche Approximationsverfahren verwendet wird, sollen hier alle Kombinationsmöglichkeiten ausgewählter Verfahren betrachtet und somit für $N = 2$ und $m_u = 2$ eine Matrix aufgespannt werden. Tabelle 4.4 zeigt die implementierten Vertreter aus den in 4.4 vorgestellten Extrapolationen. Als Referenz wird die

Tabelle 4.4.: Untersuchte Extrapolationsverfahren für $\boldsymbol{a}_L$ und $\boldsymbol{a}_R$

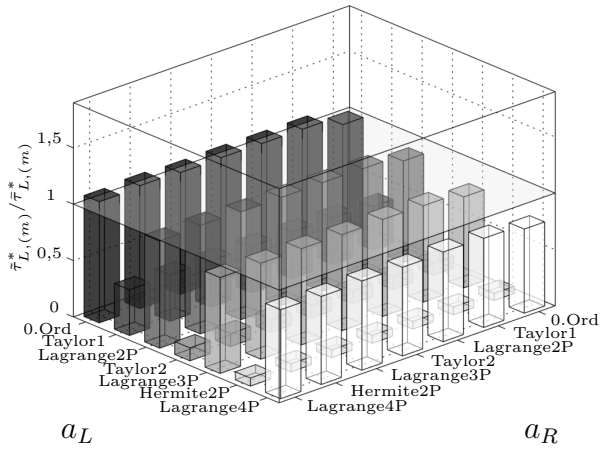
Grad n		0		1				2				3			
L	r	1	0	1	1	2	0	1	2	3	0	2	1	4	0
Verfahren		0.Ordng.		Taylor1		Lagrange2P		Taylor2		Lagrange3P		Hermite2P		Lagrange4P	

Extrapolation nullter Ordnung verwendet, für die der Diskretisierungsfehler der CS aus (3.15) dem lokalen Fehler aus (3.16) entspricht, so dass $\bar{\tau}^*_{(m)}(L = 1, r = 0) = \tilde{\tau}^*_{(m)}(L = 1, r = 0)$. Somit wird über die erste Zeile und Spalte der Kombinationsmatrix der Einfluss jedes Verfahrens auch einzeln, wie in Abbildung 4.12 sichtbar, gezeigt. Es werden Verfahren 1. bis 3. Grades verglichen, jeweils für r>0 sowie für r=0, wobei Letztere im Falle nicht verfügbarer lokaler Ableitung eingesetzt werden können.

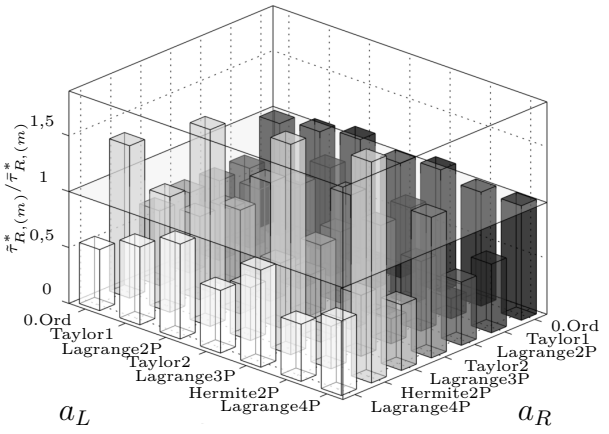
Den Erkenntnissen aus 4.3.1 entsprechend werden die Koppelparameter gewählt: $r_R = 1$ ($\rightarrow \bar{u}_L = \bar{y}_R = -M_K$ und $\bar{u}_R = \bar{y}_L = \omega_L$); $\Omega^*_{krit} = 10^4$ (Auch bei Extrapolation höherer Ordnung ist Filtern nötig, da $\tilde{\boldsymbol{u}}_j$ nur in den Abschnitten $T_l \mapsto T_{l-1}$ stetig differenzierbar ist.); $\frac{K_{(LR)}}{\dot{\varphi}_L \dot{\varphi}_R} = 1$, die Richtung $K_{LR} = 100$ (d. h. im linken Teilmodell die größere Masse); $\Lambda = 100$ (Anwendungsbezogen im Hinblick auf die Wahl der Approximation bei Entkopplung steifer Systeme); $d_L = d_K = d_R = 0$ (vollständig ungedämpftes System).

Außer im Fall von $\bar{\bar{u}}_R = \bar{\bar{y}}_L = \dot{\omega}_L$, das aufgrund der Zustandsgröße ω_L in den Modellgleichungen vorhanden ist, müssen im Allgemeinen für die Ermittlung der (höheren) Ableitungen $max(r)$ zusätzliche Hilfszustände z_r eingeführt werden, um diese $\bar{y}^{(r)}_{k_y}$ näherungsweise zu berechnen. Dies geschieht über die Einführung von DT_1-Gliedern. Sie sind mit dem Verhalten $y^{(r)}_{k_y} = kz_r$ und $\dot{z}_r = \frac{y^{(r-1)}_{k_y} - z_r}{T_D}$ beschrieben. Die Wahl von T_D erfolgt wieder anhand $\Omega^*_{krit_D}$ nach (4.14), für welches sich Werte der Größenordnung $\sim 10^4$ als geeignet erweisen. Im Testmodell ist jeweils $T_D = 0,001s$ gewählt. Hängt $y^{(r-1)}_{k_y}$ direkt von einem Eingang $\tilde{u}_{k_u}$ ab, was bei $\dot{\omega}_L$ der Fall ist, (4.8), so entstehen unabhängig von a_L Sprünge an den Koppelzeitpunkten,

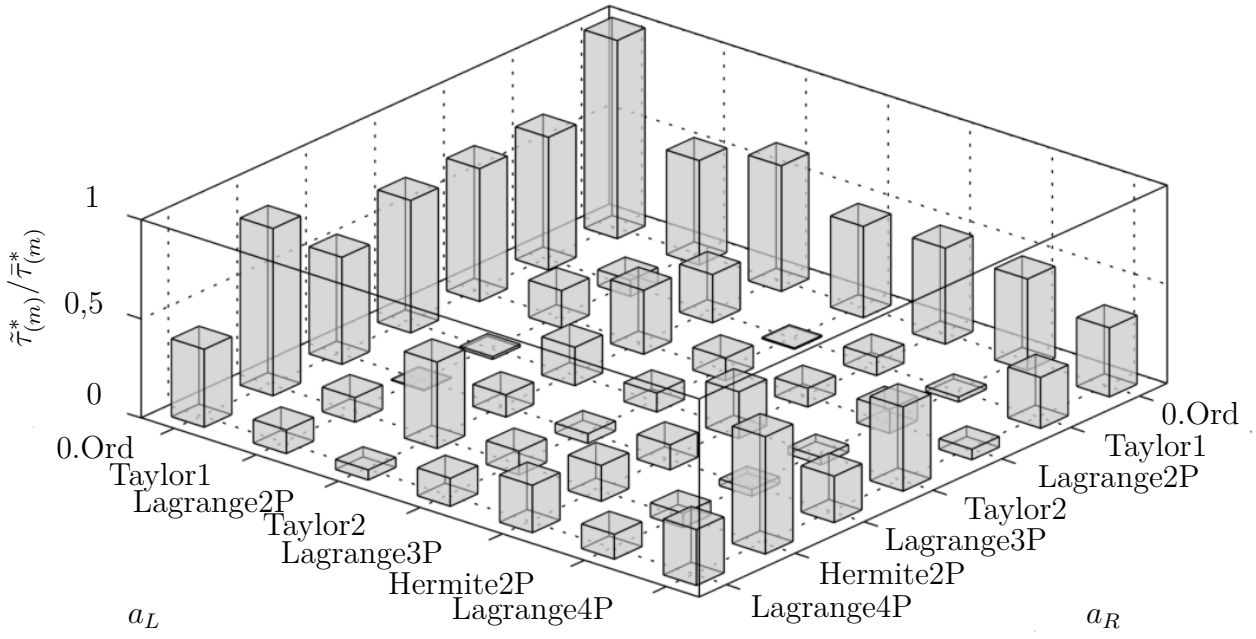
sofern keine Glättung verwendet wird ($T_G^* = 0$). Dies hat bei den notwendig kleinen Werten für T_D zu hohe Werte für $y_{k_y}^{(r)}$ zur Folge. Dieser Fehler kann vermieden werden, wenn durch Einführung von Glättung mit $T_G^* > 0$ die diskreten Sprünge zu kontinuierlichen Übergängen werden. Auf diese Weise wird auch umgangen, dass die Annahme, die zur Verwendung von (4.15) getroffen wird, im Falle $\tilde{u}_L = \bar{u}_L$ für $\dot{\omega}_L$ keine Gültigkeit besäße und $\bar{\dot{y}}_L$ um H_{l-1} phasenverschoben wäre.



(a) Beeinflussung des lokalen Fehlers $\tilde{\tau}^*_{L,(m)}$ von $\tilde{u}_L$



(b) Beeinflussung des lokalen Fehlers $\tilde{\tau}^*_{R,(m)}$ von $\tilde{u}_R$



(c) Beeinflussung des globalen Fehlers $\hat{\tilde{\tau}}^*_{(m)}$ von E_{mech}

Abbildung 4.12.: Kombinationsmatrix der Approximationsverfahren beim Zwei-Massen-Schwinger: Fehlerreduktion für $H^* = 10^{-3}$

Um den Einfluss der Approximation messen zu können, wird H^* so gewählt, dass die höherfrequente Schwingung φ_R ($\Omega_{0,R} = \Lambda\Omega_{0,L}$) aufgelöst wird und ohne Approximation stabil bleibt, aber die Änderung während eines Makroschrittes $\Delta\boldsymbol{u}_j(H_l)$ signifikant wird. Abbil-

dung 4.12 zeigt entsprechend die Ergebnisse für konstantes $H^* = 10^{-3}$, was einer Auflösung von $\frac{T_{0,R}}{H} = 20$ entspricht. Um den reinen Einfluss der Verfahren aus Tabelle 4.4 zu sehen, wird hier $T_G^* = 0$ gewählt. $\boldsymbol{z}_0$ ist analog zu Abschnitt 4.3.1 definiert.

Die Balken in 4.12(a) und 4.12(b) zeigen mit dem auf den Diskretisierungsfehler $\bar{\tau}_{j,(m)}^*$ bezogenen lokalen Fehler $\tilde{\tau}_{j,(m)}^*$ den Einfluss der Approximation für alle Kombinationen aus Tabelle 4.4. Im Teil (c) ist der globale Energiefehler $\hat{\tilde{\tau}}_{E,(m)}^*$, bezogen auf $\hat{\bar{\tau}}_{E,(m)}^*$ ohne Approximation, dargestellt.

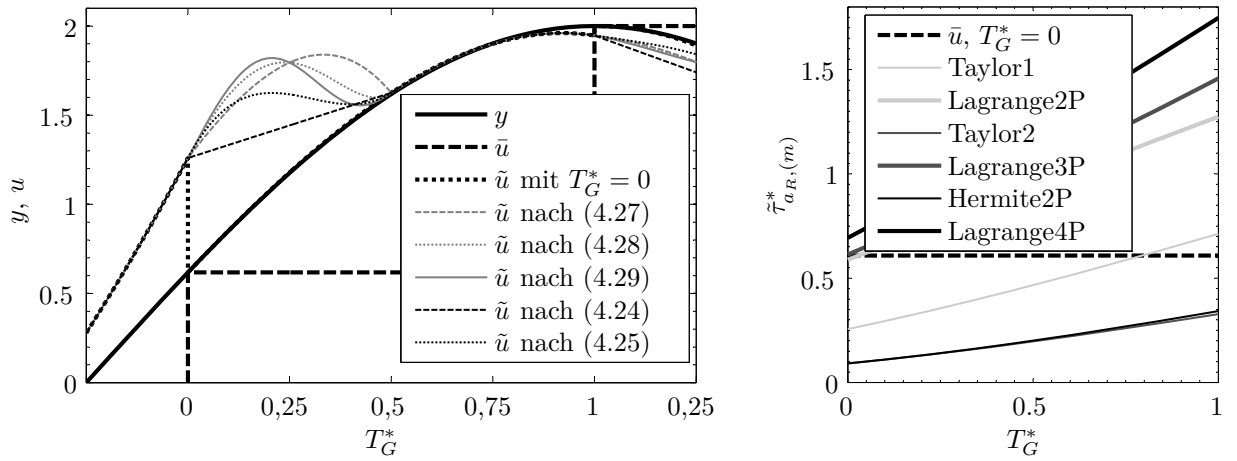
Man sieht, dass $\tilde{u}_L$ wegen $K_{LR} = 100$ von a_R kaum beeinflusst wird und a_L mehr Einfluss auf $\tilde{u}_R$ ausübt. $\tilde{\tau}_{L,(m)}^*$ erscheint in Kombination mit a_R: $0.Ordnung$ etwas geringer als mit größerer Ordnung, was auf einen größeren Gesamtfehler bei a_R: $0.Ordnung$ und daraus resultierendem größerem Mittelwert $\omega_{L,(m)}$ zurückgeht. Mit dem Grad von a_L nimmt die Genauigkeit zu, so dass Hermite2P die beste Approximation darstellt. Jedoch sind die Ergebnisse mit $r = 0$ (Lagrange) hier aufgrund des groß gewählten H^* schlechter.

Dieser Einfluss wird in Teil (b) insbesondere bei $r > 0$ sichtbar, da hierfür $\dot{\omega}_L = f(\tilde{u}_L) = f(a_L)$ verwendet wird. Somit bildet im Beispiel a_R: $Taylor1$ in Kombination mit a_L: $Hermite2P$ die größte Genauigkeit. Durch aufgrund fehlender Glättung von $\tilde{u}_L$ zu großem $\ddot{\omega}_L =$ (s.o.) fällt Taylor2 hier schlechter aus.

Im Gesamtergebnis, Teil (c), ist jedoch nicht die Kombination aus den lokal optimalen Approximationen, sondern a_L: $Taylor1$ mit a_R: $Lagrange3P$ am genauesten mit einer Reduktion des Fehlers um 99,6%. Im steifen Gesamtsystem mit $\Lambda = 100 \rightarrow \theta_L >> \theta_R \wedge c_L << c_R$ haben beide Seiten Einfluss auf den Energiefehler, so dass die Effekte aus beiden lokalen Fehlern sichtbar werden. Die Wahl der Approximation hängt somit von H^*, aber auch von der Glättung ab, die im Folgenden bewertet wird.

Glättung

Um Verlauf und Einfluss der Glättung sichtbar zu machen, werden diese zunächst am Ein-Massen-Schwinger ohne Rückkopplung gezeigt. Zur besseren Darstellung wird hierfür eine geringe Auflösung von $\frac{T_{0,L}}{H} = 5$ gewählt. Abbildung 4.13(a) zeigt für a_R: $Hermite2P$ und $T_G^* = 0.5$ die Splineglättungen nach (4.23) sowie die mit Umschaltung nach [83] ,(4.26). Demnach stellt die Splineglättung dritten Grades den besten Kompromiss aus Differenzierbarkeit und zusätzlichem Fehler dar. In Teil 4.13(b) ist der Fehler $\tilde{\tau}_{a_R,(m)}^*$ der Approximationen mit Splineglättung dritten Grades in Abhängigkeit von T_G^* aufgetragen. Dabei entsprechen die Werte bei $T_G^* = 0$ denen der reinen Extrapolationsverfahren. $T_G^* = 1$ hingegen entspricht *interpolierter Extrapolation* mit dem jeweiligen Verfahren sowie Interpolation mit Splines dritten Grades. Insgesamt entsteht durch die gewählte Auflösung ein großer Fehler, der bei den Methoden mit $r = 0$ entstprechend größer als der Diskretisierungsfehler ausfällt. Man sieht jedoch, dass jeweils $\tilde{\tau}_{a_R,(m)}^*$ ansteigt, da die Schwingung mit T_G^* zunehmend pha-



(a) Glättung mit $T_G^* = 0.5$ bei Hermite2P: Splines und Umschaltung (b) Einfluss der Wahl von T_G^* auf $\tilde{\tau}^*_{a_R,(m)}$

Abbildung 4.13.: Einfluss der Glättung auf die Approximation einer einfachen Schwingung

senverschoben wird. Auf den Gesamtenergiefehler hat T_G^* entsprechend kleineren Einfluss, was im Folgenden am Zwei-Massen-Schwinger betrachtet wird.

Nach obiger Erwähnung kann der Fehler von $\bar{y}_L$, der durch die Sprünge an den T_l entsteht, durch Glättung bei a_L verringert werden, wodurch a_R genauer wird. Mit steigendem T_G^* vergrößert sich jedoch der Fehler von $\tilde{u}_L$, und es entsteht ein Zielkonflikt zwischen a_L und a_R. Tabelle 4.5 zeigt zusätzlich zu den beiden lokalen Fehlern den globalen Energiefehler für $T_G^* = 0$ bzw. $T_G^* = 0.5$, bezogen auf den Diskretisierungsfehler. Gleichen sich bei der

Tabelle 4.5.: Zielkonflikt zwischen a_L und a_R bei $T_G^* > 0$

	T_G^*	$\tilde{\tau}^*_{a_L,(m)}/\bar{\tau}^*_{a_L,(m)}$	$\tilde{\tau}^*_{a_R,(m)}/\bar{\tau}^*_{a_R,(m)}$	$\hat{\tilde{\tau}}^*_{E,(m)}/\hat{\bar{\tau}}^*_{E,(m)}$
a_L:$Lagrange4P$; a_R:$Taylor2$	0	$0,787$	$1,257$	$0,426$
	$0,5$	$1,394$ ↑	$0,888$ ↓	$0,426$ →
a_L:$Taylor1$; a_R:$Lagrange3P$	0	$0,409$	$0,744$	$0,004$
	0.5	$0,753$ ↗	$0,697$ ↘	$0,084$ ↑

oberen Kombination der Tabelle die starken Einflüsse auf die lokalen Fehler beim phasenunabhängigen Energiefehler aus, so hat die Glättung bei der sonst besten Kombination für den Energiefehler (siehe Abb. 4.12(c)) negativen Einfluss darauf. Für $T_G^* = 0.5$ wäre a_L: $Taylor2$; a_R: $Taylor1$ mit $\hat{\tilde{\tau}}^*_{E,(m)}/\hat{\bar{\tau}}^*_{E,(m)} = 0,003$ am energieneutralsten. Stärker als die Glättung beeinflusst jedoch H^* den CS-Fehler, siehe unten.

Durch die Glättung werden Sprünge an T_l vermieden, und der Verlauf des mit $T_G^* = 0.2$ geglätteten Signals ähnelt der Filterung mit $\Omega^*_{krit} = 10^4$. Durch die rein algebraische Glättung wird jedoch kein zusätzlicher Zustand eingeführt, so dass weiterhin gefiltert werden muss. Umgekehrt ist bei Verwendung der Kausalschnittstelle mit Filterung die Glättung überflüssig.

4.4.2. Implementierung der Approximation

Als Verfahren nach Gleichung (3.9) sind alle Vorgenannten gemäß Kapitel 4.2.2 innerhalb des Modelica-Slave-Modells, Abbildung 4.14 links, implementiert. Im Signalfluss, rechts, kommen die Approximationsverfahren *Apx*, hinter dem *Com*-Block, in dem Koppelgrößen und Kontrollbussignale ausgetauscht werden. Sie sind - im Gegensatz zu den Kausalschnittstellen - in die Slave-Schnittstelle intergiert, da sie domänenunabhängig sind. Die an eine Makro-

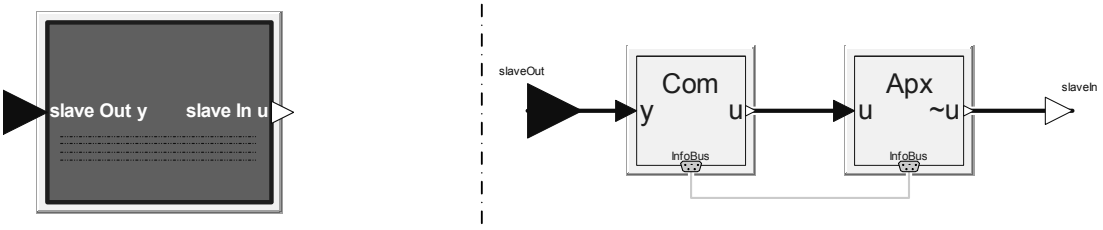


Abbildung 4.14.: Approximation *Apx* im Modelica-Slave-Modell

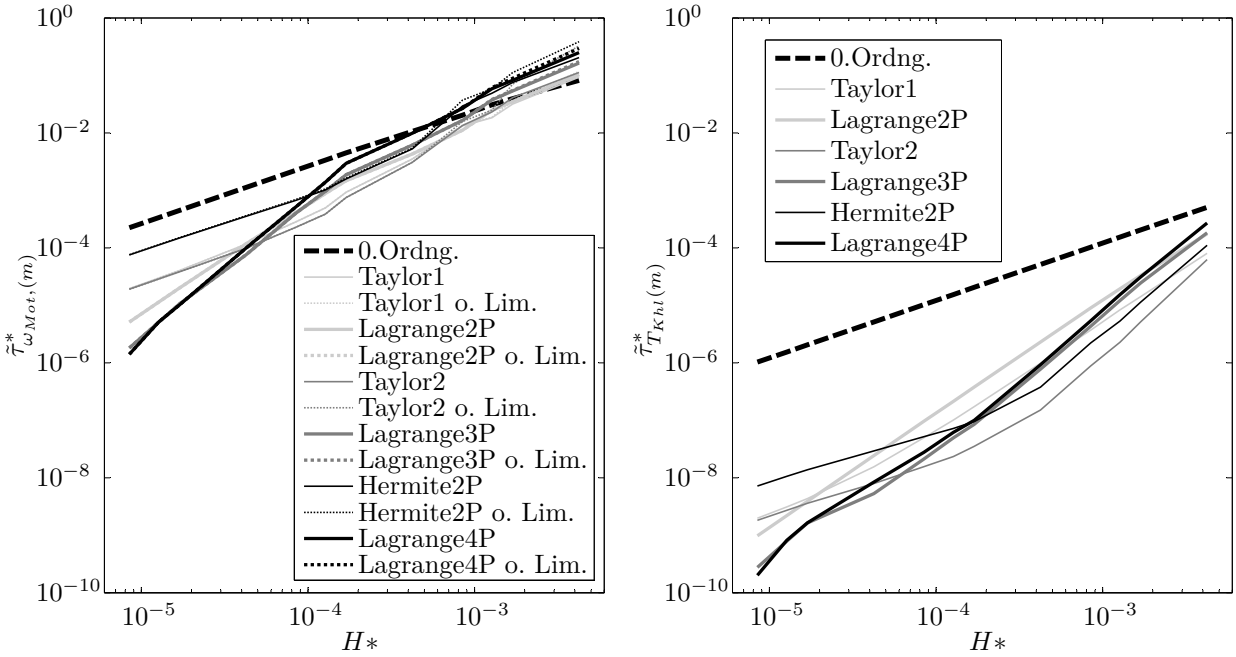
schrittweitensteuerung angepassten Extrapolations- und Glättungsverfahren sind in einer Bibliothek abgelegt. Sie können beliebig im *Apx*-Block komponentenweise eingesetzt werden. Die Verfahren mit $r > 0$ erhalten die benötigten $\bar{u}_{k_u}^{(r)}$ vom entsprechenden Subsystem über den Kontrollbus durch den *Com*-Block, in dem auch alle $\bar{y}_{k_y}^{(r)}$ der Ausgänge ermittelt werden. Bei den Lagrangeverfahren ($L > 1$) wird, bis die nötige Anzahl $L = l+1$ Stützwerte T_l erreicht ist, mit dem höchstmöglichen Grad l initialisiert. Im *Apx*-Block sind Grenzen für jede Komponente über die Vektoren $\tilde{\boldsymbol{u}}_{j,max}$ / $\tilde{\boldsymbol{u}}_{j,min}$ applizierbar, die die Approximation nicht verletzen darf, mit Ausnahme der Koppelsignale selbst, so dass für die obere Grenze

$$\tilde{\boldsymbol{u}}_{j,maxLim} := \min[\max(\bar{\boldsymbol{u}}_j, \tilde{\boldsymbol{u}}_{j,max}), \tilde{\boldsymbol{u}}_j], \tag{4.30}$$

und für die Gesamtlimitierung

$$\tilde{\boldsymbol{u}}_j = \tilde{\boldsymbol{u}}_{j,Lim} := \max[\min(\bar{\boldsymbol{u}}_j, \tilde{\boldsymbol{u}}_{j,min}), \tilde{\boldsymbol{u}}_{j,maxLim}] \tag{4.31}$$

gilt. Diese zulässigen Grenzen sind für die Anwendung der CS, etwa zur Vermeidung der Approximation eines Rückwärtsdrehens des Verbrennungsmotors $\omega_{Mot} \overset{!}{\geq} 0\frac{rad}{s}$, von Bedeutung. Für den Anwendungsfall wird die Genauigkeit der Approximationen mit $T_G^* = 0$ in Abhängigkeit der Makroschrittweite H^* anhand zweier Koppelgrößen des Fahrzeugmodells untersucht. Dabei werden mehrere NEFZ-Zyklen mit konstanter Schrittweite simuliert und der lokale Fehler $\tilde{\tau}_{j,(m)}^*$ im Zyklus betrachtet. Abbildung 4.15(a) zeigt die Genauigkeit der Motordrehzahl ω_{Mot} als dynamischere Größe, wohingegen 4.15(b) das trägere Verhalten der Kühlmitteltemperatur T_{Khl} zugrunde liegt. So werden $\tilde{u}_{\omega_{Mot},max}$ und $\tilde{u}_{\omega_{Mot},min}$ bei gößeren H^* und vor allem von den Verfahren mit $r > 0$ verletzt, siehe oben rechts im Teil (a). Die Temperaturgrenzen werden von T_{Khl} hingegen nie verletzt (nicht im Bild (b) notwendig). In



(a) Fehler der Drehzahl $\tilde{\tau}^*_{\omega_{Mot},(m)}$ (b) Fehler der Kühlmitteltemperatur $\tilde{\tau}^*_{T_{Khl}(m)}$

Abbildung 4.15.: lokaler Fehler $\tilde{\tau}^*_{j,(m)}$ der Approximationen mit und ohne Begrenzung in Abhängigkeit fester Makroschrittweiten H^*

Anhang B finden sich die entsprechenden Ergebnisse für $T^*_G > 0$, wobei die Genauigkeitsverläufe nahezu parallel oberhalb derer in Abb. 4.15 liegen. In allen Fällen bleibt die Reihenfolge der Genauigkeit der Verfahren über H^* nicht konstant. Um sowohl eine kürzere Rechenzeit $S_{CS} \overset{!}{\geq} 1$, als auch verwendbare Ergebnisse zu bekommen, sind Werte von $H^* = 1 \cdot 10^{-4}$ bis $1 \cdot 10^{-3}$ anzustreben, wobei die Rolle der Genauigkeitsunterschiede mit H^* zunimmt. D. h. für die Approximation von ω_{Mot} sind Verfahren mit $r > 0$ vorteilhaft, wobei L klein gewählt werden muss. Auch wenn $Taylor2$ im relevanten Bereich zunächst am genauesten ist, ist die Wahl von $Taylor1$ insgesamt besser. Für T_{Khl} hingegen ist die Wahl von $Taylor2$ außer bei sehr kleinen Makroschritten die beste. Beiden Koppelgrößen gemein sind die Vorteile von *Lagrange* für $H^* < 5 \cdot 10^{-5}$. Insgesamt muss jedoch auch bei variablen H_l, die in den Bereich kommen können, die Relevanz im vorgenannten Bereich betrachtet werden.

4.5. Algorithmen zur Makroschrittweitensteuerung

Mit dem Ziel, den Kompromiss aus Genauigkeit und Aufwand zu optimieren, d.h. höhere Werte der CS-Indikatoren (CSI) aus (3.32) bis (3.35) zu erreichen, wird *MDPCosim* um Algorithmen zur Makroschrittweitensteueung erweitert. Ihr Grundaufbau lehnt an den der lokalen Verfahren in Kapitel 2.3.2 an, wie bereits in Kapitel 3.2.4 angedeutet. Die Anwendung

der Vorschriften aus (2.26) bzw. (2.29) auf die Steuerung der Makroschrittweite lautet dann

$$H_{opt} = H_{opt}(\boldsymbol{y}, \boldsymbol{y}_c, \boldsymbol{EST}, \boldsymbol{TOL}) \tag{4.32}$$

$$H_{opt} = H_{opt}(\boldsymbol{y}, \boldsymbol{y}_c, \boldsymbol{IND}, \boldsymbol{TOL}). \tag{4.33}$$

So gibt es bereits erste Verfahren mit Fehlerschätzer nach (4.32) und (2.27) bzw. (2.28) für die Steuerung von H [79, 115], die jedoch mit dem hier vorliegenden Ziel $S_{CS} \overset{!}{\geq} 1$ nicht vereinbar sind aufgrund des Speicheraufwands und der Schrittwiederholungen. [23] stellt ein explizites Verfahren mit Fehlerschätzer durch Prädiktor-Korrektor-Terme auf Basis von a_j und eines a'_j abweichender Ordnung vor, das jedoch nur für $H \downarrow\downarrow$ Gültigkeit besitzt und somit wieder $S_{CS} \overset{!}{\geq} 1$ entgegensteht. In der vorliegenden Arbeit dagegen wird - den Randbedingungen aus 4.1 folgend - ein Verfahrensansatz nach (4.33) gewählt. Mit diesem ist sowohl die Anforderung einer beschleunigten Gesamtsimulation als auch die Verwendung proprietärer Subsystemmodelle mit Komponenten unterschiedlicher physikalischer Domänen erfüllbar. Es handelt sich um heuristische, explizite Verfahren, da nur auf bekannte Werte zurückgegriffen werden kann. Zur Entwicklung der Algorithmen ist eine Einteilung der einzelnen Gesichtspunkte hilfreich:

- **offline/online**: Bestimmung aller H_l im Preprocessing / Bestimmung während der CS
- **mit/ohne Fehlerschätzer**: Schrittweitenkriterium nach (4.32) / nach (4.33)
- **Anzahl relevanter Ausgänge/Slaves**: Abhängigkeit der H_l von einer Lösungskomponente / allen Lösungskomponenten bzw. eines Subsystems / aller Subsysteme
- **Steuergrößen aus den Subsystemen**: Für das Schrittweitenkriterium verwendet werden können: $\boldsymbol{y}_j, \bar{\boldsymbol{\tau}}_j, h_j, \boldsymbol{a}_j$, Eventinformationen, etc.
- **Anzahl nötiger Eingabeparameter durch Benutzer**: Eingabeparameter $\boldsymbol{p}$, die zur Ausführung der CS in Kenntnis des Modell- und CS-Verhaltens zusätzlich eingegeben werden müssen, etwa $j, ATOL, RTOL$, etc.

Die Wahl der Gesichtspunkte ist abhängig von der Umsetzungsmöglichkeit und der Anwendung, was in Teilen unten verglichen wird. Sollen die H_l an mehrere Lösungskomponenten k_y angepasst sein, so muss ein skalarer Indikator dafür bestimmt werden. Für Verfahren nach (4.32) wird aus dem vektorwertigen $\boldsymbol{EST}$ durch Skalierung mittels Toleranzangaben und der Normierung nach (2.31) der skalare Fehlerindikator

$$ERR = \sqrt{\frac{1}{m_y} \sum_{k_y=1}^{m_y} \left(\frac{EST_{k_y}}{ATOL_{k_y} + RTOL_{k_y} y_{k_y,(R)}} \right)^2} \tag{4.34}$$

$$\text{mit} \quad y_{k_y,(R)} = \max_{k_y}(|y_{k_y,(l)}|, |y_{k_y,(l+1)}|) \tag{4.35}$$

nach ([61]). Für Verfahren nach (4.33) muss aus dem vektorwertigen $\boldsymbol{IND}$ durch Skalierung mittels Toleranzangaben der skalare Indikator

$$IDC = IDC(\boldsymbol{y}, \boldsymbol{ATOL}, \boldsymbol{RTOL}, \boldsymbol{IND}) \tag{4.36}$$

bestimmt werden, s.u. Mithilfe der skalaren Indikatoren wird innerhalb einer geeigneten Steuerung H_{opt} ermittelt, aus dem gemäß der Strategie ν der folgende Makroschritt

$$H_l := \nu(H_{l-1}, H_{opt}, \boldsymbol{p}) \tag{4.37}$$

bestimmt wird, vgl. Abb. 2.7.

Variable Schrittweite im Preprocessing (offline)

Ein naheliegender simpler offline-Ansatz für die Fahrzyklensimulation ist die Bestimmung der H_l aus der Zyklusvorgabe $\dot{x}_{soll}(t)$ im Preprocessing. Aus der Geschwindigkeit $v_Z = \dot{x}_{soll}$, in gegebener Auflösung $[\boldsymbol{t}_Z \;\; \boldsymbol{v}_Z]$, ist die Kinematik des Fahrzeugs auf Systemebene grob vorhersagbar. Ohne Kenntnis des Antriebsstranges kann die Beschleunigung $a_Z = \ddot{x}_{soll}$ als skalares Kriterium verwendet werden, da auch deren Grenzen im Zyklus a Priori bekannt sind. Dabei ist die betragsmäßig kleinste Beschleunigung i.d.R. $a_{min} = 0m/s^2$. Die Vorschrift der Steuerstrategie für H_l laute damit:

$$H_l := H_{opt} = H_{min} + (1 - a_l^*)^{p_0}(H_{max} - H_{min}) \tag{4.38a}$$

$$\text{mit } a_l^* = \frac{|a_l| - a_{min}}{a_{max} - a_{min}} \quad (\text{häufig } = \frac{|a_l|}{a_{max}}) \tag{4.38b}$$

Es entspricht also $\boldsymbol{IND} := |a_l|$ und $IDC := a_l^*$ und es ergeben sich bei größeren Beschleunigungen die kleineren Makroschrittweiten. Diese werden mit IDC über den *ALGORITHMUS 4.1* mithilfe der Parameterwerte $\boldsymbol{p} = [H_0 \quad H_{min} \quad H_{max} \quad p_0]^T$ vorausberechnet:

---

berechne $[\boldsymbol{t}_Z \;\; \boldsymbol{a}_Z]$ aus Zyklusvorgabe $[\boldsymbol{t}_Z \;\; \boldsymbol{v}_Z]$
Grenzen $a_{min} := \min_k(|a_{Z,k}|)$; $a_{max} := \max_k(|a_{Z,k}|)$
dimensionslos $\boldsymbol{a}_Z^* := \boldsymbol{a}_Z^*(a_{min}, a_{max})$
Startweite $T_1 := T_0 + H_0$
While $T_l < T_M$ **do**
  interpoliere a_l^* aus $[\boldsymbol{t}_Z \;\; \boldsymbol{a}_Z^*]$
  berechne H_l nach (4.38)
  $T_l := T_l + H_l$
  $l++$
end
Zusammenfassen konstantbleibender H_l: $[\boldsymbol{T}_l \;\; \boldsymbol{H}_l] \Rightarrow [\boldsymbol{T}_Z \;\; \boldsymbol{M}_Z]\boldsymbol{M}_Z \in \mathbb{N}^J$

Abbildung 4.16.: *ALGORITHMUS 4.1*: Offlinebsetimmung variabler Schrittweiten H_l

Dabei gilt $\sum_{j=1}^{J} M_{Z,j} = M$. Das Verfahren ist nur bei vorab bekannter Gesamtmodelldynamik möglich, und macht eine nichttriviale Wahl von $\boldsymbol{p}$ notwendig.

4.5.1. Heuristische Verfahren

Eine Makroschrittweitensteuerung im eigentlichen Sinne erfolgt *online* während der Co-Simulation. Sie wird im vorliegenden Fall vom CS-Master übernommen und ist in dessen Ablauf eingebunden, wie in Abbildung 4.3 dargestellt. Die folgenden Verfahren sind explizite Verfahren, die ohne Iteration von Schritten auskommen und streng monoton über der Zeit T arbeiten. Da sie anstatt einer Vorabschätzung des kommenden Fehlers auf bekannten Ist-Werten beruhen, sind sie als *heuristische* Verfahren zu bezeichnen. Sie können jedoch mit einer Vorausschau bei Zyklensimulation verfeinert werden, s.u.

Gradientenbasierte Methode (online)

Die konsequente Weiterführung des Ansatzes aus (4.38) führt auf eine Steuerung der Makroschrittweite, die von den relativen Gradienten der Lösungskomponenten ausgeht. So wird die Verwendung von $\boldsymbol{IND} := |a_l|$ verallgemeinert zu $\boldsymbol{IND} = |\dot{\boldsymbol{y}}_l|$ von jeder beliebigen Koppelgröße. Deren Gradient ist über den Kontrollbus, $\bar{\dot{\boldsymbol{y}}} \in \bar{\boldsymbol{y}}^c$, im Master bekannt. In diesem erfolgt zu jedem Zeitpunkt T_l die Ermittlung der optimierten Makroschrittweite H_{opt} unter Verwendung von Parametern wieder über den skalaren Indikator IDC:

$$H_{opt} = H_{min} + (1 - IDC)^{p_0}(H_{max} - H_{min}) \tag{4.39a}$$

$$\text{mit} \quad IDC = \min(1, \|\dot{\boldsymbol{y}}^*\|) \tag{4.39b}$$

$$\text{und} \quad \dot{y}^*_{k_y} = \frac{|\dot{y}_{k_y,l}|}{p_1 \dot{y}_{k_y,l,(M)}} \tag{4.39c}$$

Im Gegensatz zur *offline*-Steuerung, in der die Grenzen der auftretenden Gradienten bekannt sein müssen (a_{min}, a_{max}, s.o.), ist dies hier, während der CS, nicht mehr der Fall. So ist zur Ermittlung der dimensionslosen Indikatoren $\dot{y}^*_{k_y}$ ein anderes Referenzband nötig. Zum Einen wird eine Untergrenze von $\dot{\boldsymbol{y}}_{min} \stackrel{!}{=} 0$ unterstellt, was bei Erreichen eines Stationärzustandes oder bei Oszillationen zutrifft und andernfalls einer konservativen Herangehensweise entspricht. Zum Anderen wird die Obergrenze über einen gleitenden Mittelwert vom Master erlernt. Dazu wird der Integralmittelwert aus (3.22),

$$\dot{y}_{k_y,l,(M)} := \dot{y}_{k_y,(m)}(T_l) \tag{4.40}$$

$$\text{oder} \quad \dot{y}_{k_y,l,(M)} := \max_l(\dot{y}_{k_y,(m)}(T)) \tag{4.41}$$

verwendet, was einer Speicherung des bisher erreichten Maximalwertes entspricht. Mithilfe des Steuerparameters p_1 und (4.39c) werden daraus die relativen Gradienten berechnet.

Über (4.39b) wird schließlich der skalare Wert IDC bestimmt. Soll *eine* Koppelgröße eines bestimmten Slaves als Schrittweitenführungsgröße dienen, etwa da sie vom Potentialsender mit der stärksten Kopplung kommt (siehe Abschnitt 4.3), so folgt einfach:

$$\|\dot{\boldsymbol{y}}^*\| = \dot{y}^*_{k_y} = \dot{y}^*_{j,i} \tag{4.42}$$

$$\text{und} \quad \boldsymbol{p} = [(H_0) \quad j \quad i \quad H_{min} \quad H_{max} \quad p_0 \quad p_1]^T \tag{4.43}$$

sind die zu bestimmenden Parameter. Davon wird aus dem Subsystemindex j und dem zugehörigen Komponentenindex i der Koppelgrößenindex k_y ermittelt. Im Allgemeinen jedoch ist nicht bekannt, welche Koppelgröße den größten Einfluss auf die Genauigkeit der CS hat, und es sind zu unterschiedlichen Simulationszeiten unterschiedliche Koppelgrößen maßgeblich am Fehler beteiligt. Dann sind alle $n_{y,j}$ Ausgänge eines Subsystems bzw. alle m_y unterschiedlichen Koppelgrößen zu berücksichtigen. Für IDC wird dazu eine der folgenden Normierungen gewählt:

$$\|\dot{\boldsymbol{y}}^*\| = \|\dot{\boldsymbol{y}}^*\|_\infty = \max_{k_y}(\dot{\boldsymbol{y}}^*) \tag{4.44}$$

$$\text{oder} \quad \|\dot{\boldsymbol{y}}^*\| = \|\dot{\boldsymbol{y}}^*\|_2 = \sqrt{\sum_{k_y=1}^{m_y} (\dot{y}^*_{k_y})^2} \tag{4.45}$$

$$\text{oder} \quad \|\dot{\boldsymbol{y}}^*\| = \|\dot{\boldsymbol{y}}^*\|_H = \sqrt{\frac{1}{m_y} \sum_{k_y=1}^{m_y} (\dot{y}^*_{k_y})^2}. \tag{4.46}$$

$$\boldsymbol{p} = [(H_0) \quad H_{min} \quad H_{max} \quad p_0 \quad p_1]^T \tag{4.47}$$

kommt dann mit weniger zu wählenden Elementen aus. Entsprechend 4.2.1 kann der Startwert H_0 beim Initialisieren der Makroschrittweitensteuerung gesetzt oder in $\boldsymbol{p}$ frei gewählt werden. Die gradientenbasierte Methode und die möglichen Normierungen werden in Abschnitt 4.5.2 bewertet.

Methode konstanter Änderungsraten (online)

Ein weiterer Ansatz zur Steuerung der Makroschrittweiten nach (4.33) ist die Übertragung der Methode (2.30) auf die Co-Simulation. Hierbei werden in jedem Zeitschritt konstante Änderungsraten für $\boldsymbol{y}$ angestrebt, und es wird $\boldsymbol{IND} = |\dot{\boldsymbol{y}}_l|$ verwendet. Die Makroschritte werden dafür an zulässige Toleranzgrenzen für den Diskretisierungsfehler, (3.15), angepasst. Dieser in [123] bei monolithischer Simulation nachgewiesene Ansatz führt zur Formulierung der Steuerung durch

$$H_{opt} = \alpha \left(\frac{1}{IDC}\right)^q. \tag{4.48}$$

Gemäß [61] dient q der Anpassung an die Approximation bzw. α als Sicherheitsfaktor und soll keinen Teil von $\boldsymbol{p}$ darstellen. Soll, wie bei (4.42), *eine* definierte Koppelgröße die Schrittweiten dominieren, so lauten IDC und $\boldsymbol{p}$:

$$IDC = \frac{|\dot{y}_{k_y,l}|}{\sqrt{ATOL^2 + (RTOL y_{k_y,l,(R)})^2}}, \tag{4.49}$$

$$\boldsymbol{p} = [(H_0) \quad j \quad i \quad ATOL \quad RTOL]^T. \tag{4.50}$$

Darin ist wieder der Index der Koppelgrößen über j, i vorzugeben, die Startschrittweite H_0 ist optional. Hinzu kommen lediglich die Angaben der absoluten Toleranz für den Diskretisierungsfehler $ATOL$, beziehungsweise die relative zulässige Änderung $RTOL$. Über (4.49) wird ein konservativer Fehlertest angewandt. Allgemein sollten alle m_y Lösungskomponenten herangezogen werden, so dass wieder eine Normierung für IDC benötigt wird. Hierfür sollen

$$IDC = \max_{k_y} \left(\frac{|\dot{y}_{k_y,l}|}{\sqrt{ATOL_{k_y}^2 + (RTOL_{k_y} y_{k_y,l,(R)})^2}} \right) \tag{4.51}$$

$$\text{und} \quad IDC = \frac{\sqrt{\sum_{k_y=1}^{m_y} (\dot{y}_{k_y,l})^2}}{\max\limits_{k_y} \left(\sqrt{ATOL_{k_y}^2 + (RTOL_{k_y} y_{k_y,l,(R)})^2} \right)} \tag{4.52}$$

$$\text{und} \quad IDC = \sqrt{\frac{1}{m_y} \sum_{k_y=1}^{m_y} \left(\frac{\dot{y}_{k_y,l}}{ATOL_{k_y} + RTOL_{k_y} |y_{k_y,l,(R)}|} \right)^2} \tag{4.53}$$

$$\text{und} \quad IDC = \sqrt{\frac{1}{m_y} \sum_{k_y=1}^{m_y} \frac{\dot{y}_{k_y,l}^2}{ATOL_{k_y}^2 + (RTOL_{k_y} y_{k_y,l,(R)})^2}}, \tag{4.54}$$

$$\text{je mit} \quad \boldsymbol{p} = [(H_0) \quad \boldsymbol{ATOL} \quad \boldsymbol{RTOL} \quad]^T, \tag{4.55}$$

untereinander verglichen werden. Darin wird $\boldsymbol{y}_{(R)}$ benötigt, um die zulässige Änderung zu quantifizieren, die bei großen Werten über $RTOL$ begrenzt wird. Im Gegensatz zu $\boldsymbol{y}_{(M)}$ aus (4.39c), wo ein längerfristiger Grenzwert ermittelt wird, ist hierfür ein Wert für den künftigen Makroschritt gesucht. Aufgrund des verwendeten heuristischen Ansatzes ohne Fehlerschätzer sind weder H_l noch $\boldsymbol{y}_{l+1}$ bekannt, und die Anwendung von (4.35) ist nicht möglich. Stattdessen soll über

$$\boldsymbol{y}_{l,(R)} := \boldsymbol{y}_l \tag{4.56}$$

$$\text{oder} \quad \boldsymbol{y}_{l,(R)} := \boldsymbol{y}_l + \frac{H_{l-1}}{2} \dot{\boldsymbol{y}}_l \tag{4.57}$$

$$\text{oder} \quad \boldsymbol{y}_{l,(R)} := \boldsymbol{y}_{(m)}(T_l) \tag{4.58}$$

eine Extrapolation oder ein Integralmittelwert der vergangenen Werte angewendet werden. Die Verwendung von (4.51) entspricht der konservativsten Umsetzung durch Minimierung

der H_{opt} über die Maximumnorm. Der an [123] angelehnte Ansatz (4.52) besteht aus einer gemischten Norm. (4.53) ist der etablierte Ansatz aus (4.35), und (4.54) eine konservative Modifikation dessen (vgl. (4.49)).

Die Parametrierung $\boldsymbol{p}$ der Steuerung erfolgt, wie dem Benutzer von der Einstellung monolithischer Solver bekannt, nur über die Toleranzangaben. Es bleibt jedoch offen, durch gute Kenntnis der Subsystemmodelle, die Toleranzen in Vektoren $\boldsymbol{ATOL}, \boldsymbol{RTOL}$ der Größe m_y anzugeben und so unterschiedlichen Domänen mit Werten, die um Größenordnungen abweichen (etwa Drücke und Kraftstoffmassenströme) gerechter zu werden und eine effizientere Co-Simulation zu erzielen. Bei der vorliegenden Steuerung handelt es sich um einen I-Regler für H_{opt}. [62] zeigt die Ansätze zur Erweiterung der Schrittweitensteuerung als PI- oder PID-Regler. Deren Anwendung würde hier jedoch einer weiterführenden mathematischen Analyse bedürfen. Die Methode wird wegen der großen angestrebten H_l und der Komplexität der Zielmodelle über numerische Applikation verbessert, welche in 4.5.2 dargestellt wird.

Aufbau der Methoden-Algorithmen

Die oben genannten Methoden werden mit einer Strategie ν nach (4.37) zu den Verfahren der Makroschrittweitensteuerung kombiniert, die im Master eingesetzt werden können, vgl. Abbildung 4.3. H_{opt} stellt hierbei einen der Methode entsprechend optimierten Wert dar, der zusätzlichen Randbedingungen genügen muss, um ein numerisch stabiles Verhalten zu erzeugen. Dies ist zum Einen eine relative Begrenzung bezüglich der Änderung zwischen H_{l-1} und H_l:

$$H_{rel} := \min\left\{(\alpha_+ H_{l-1}), \max[(\alpha_- H_{l-1}), H_{opt}]\right\}. \tag{4.59}$$

Dadurch wird ein zu schnelles Anwachsen oder ein Oszillieren der Schrittweiten verhindert. Ergänzt wird zum Anderen um die Gesamtlimitierung

$$H_l := \min\left\{H_{max}, \max[H_{min}, H_{rel}]\right\}, \tag{4.60}$$

die besonders bei Simulationen mit Ereignissen oder endlichen Stationärphasen wichtig ist.

Die heuristischen Verfahren sind gemäß *ALGORITHMUS 4.2* implementiert, Abb. 4.17, der nach jedem Makroschritt H_l ausgeführt wird.

$H_{l-1} := H_l$
lese Werte $\bar{\boldsymbol{y}}_l, \bar{\boldsymbol{y}}_l^c$
berechne $\dot{\boldsymbol{y}}_{l,(M)}$ nach (4.40)$\wedge$(4.41)$\rightarrow\dot{\boldsymbol{y}}_l^*$ nach (4.39c)
bzw. $\boldsymbol{y}_{l,(R)}$ nach (4.56)$\wedge\ldots\wedge$(4.58)
berechne IDC nach (4.42)$\wedge$(4.44)$\wedge\ldots\wedge$(4.46)
bzw. IDC nach (4.49)$\wedge$(4.51)$\wedge\ldots\wedge$(4.54)
berechne H_{opt} nach (4.39a)
bzw. H_{opt} nach (4.48)
berechne $H_{rel}(H_{l-1}, H_{opt}, \alpha_+, \alpha_-)$ nach (4.59)
begrenze H_l nach (4.60)
if Zyklussteuerung **then**
$[H_l, \alpha_+, \alpha_-, \ldots]$:=ZyklusSteuerAlgorithmus, *ALGORITHMUS 4.3*, Abb.4.19
end

Abbildung 4.17.: *ALGORITHMUS 4.2*: Aufbau der Onlinealgorithmen zur Bestimmung von H_l

Darin erfolgt die Abfrage, ob eine überlagerte Zyklussteuerung aktiv ist, die Einfluss auf H_l und einige Verfahrensparameter hat. Diese wird unten im *ALGORITHMUS 4.3* dargestellt.

Die Verfahren müssen über die Applikationsparameter $H_{min}, H_{max}, \alpha_+, \alpha_-$ sorgfältig eingestellt werden. Teilweise gibt es in der Literatur Erfahrungswerte, die als Richtungsvorgabe der Applikationsaufgabe verstanden werden können, vgl. auch Kapitel 3.2.4:

- monolithische Verfahren: [61, 62] geben einen Wertebereich für die positiven und negativen Wachstumsraten $\alpha_+ = 1,5...5$ bzw. $\alpha_- = 0,2...0,5$ an. [123] verwendet neben dem Sicherheitsfaktor $\alpha = 0,8$ große Werte $\alpha_+ = 5, \alpha_- = 0,5$, die zu weniger Aufwand führen. Eine Untergrenze h_{min} ergibt sich am Minimum aus Rundungsfehler und lokalem Fehler.
- Co-Simulation: in [115] wird die Verwendung der o.g. Wertebereiche für α_+, α_- auch für die Co-Simulation vorgeschlagen. [23] verwendet $H_{min} = 1 \cdot 10^{-6} s$ für die Mindestmakroschrittweite, was dort $H_{min}^* = 1 \cdot 10^{-6}$ entspricht.

Da in dieser Arbeit heuristische Verfahren angewendet werden, muss die Steuerung schneller auf die Modelldynamik reagieren können als bei den Verfahren nach (4.32). D.h. es können große Werte für α_+ eingesetzt werden, dafür werden sehr kleine Werte für α_- benötigt. Mit dem Ziel $S_{CS} \overset{!}{\geq} 1$ in der Gesamtfahrzeugsimulation, insbesondere in Kombination mit *ALGORITHMUS 4.3*, kann H_{max} relativ groß gewählt werden, so dass $H_{max}^* = 1 \cdot 10^{-2}$ möglich ist. Allgemein sollte jedoch $H_{max} \leq 1/f_{max}$ gelten. Eine für parallele Synchronisation sinnvolle Untergrenze ergibt sich aus $H_{min} := \max_{j,k}(h_{j,k})$. Stellen H_{min} und H_{max} die Sicherheitslimitierung der Strategie ν dar, so können sie modellunabhängig gewählt werden. Die Wahl erfolgt via H_{min}^*, H_{max}^*. Stellen H_{min} und H_{max} dagegen direkte Methodenparameter dar, wie in (4.38) oder (4.39a), so sind sie modellabhängig und enger zu wählen, vgl. hierzu die numerischen Untersuchungen in 4.5.2.

Algorithmus zur Steuerung bei Fahrzyklussimulationen

Ein nutzbarer Vorteil bei Fahrzyklensimulationen ist die einheitliche und fahrzeugunabhängige Vorgabe der Geschwindigkeitsprofile. Dies kann, wie bei der offline-Steuerung aus Abschnitt 4.5, durch Anpassung von H_l an das vorab bekannte Profil für eine effizientere CS genutzt werden. Jedoch berücksichtigt die offline-Steuerung nur *einen* Zustand und kann somit nie der gesamten Modelldynamik mit Zyklus-unabhängigem Verhalten wie der Schaltstrategie, Thermomanagement etc. Rechnung tragen. Dafür ist eine online-Strategie für alle Größen notwendig. Es ist aber möglich, beides zu kombinieren, indem der online-Makroschrittweitensteuerung eine zyklusabhängige Steuerung überlagert wird. Deren Funktionsweise wird anhand Abbildung 4.18 am Beispiel des NEFZ beschrieben.

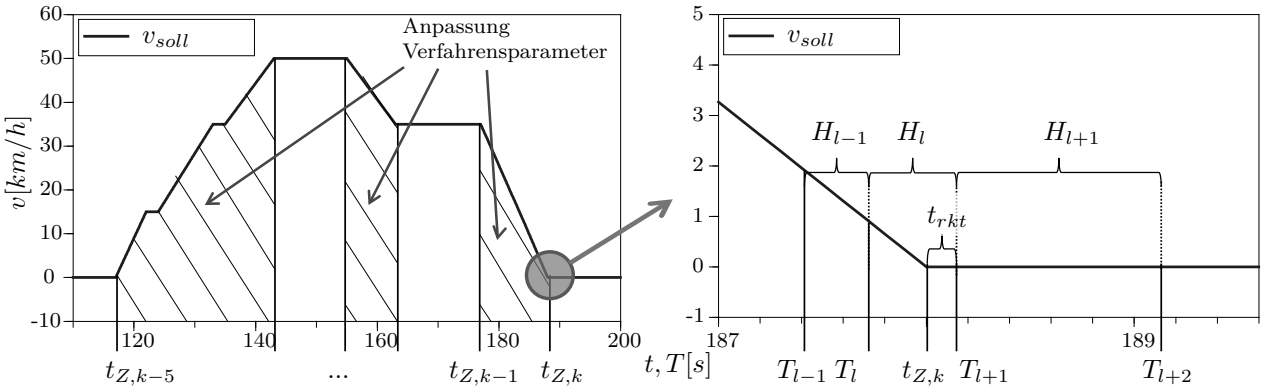


Abbildung 4.18.: Ausschnitt eines Fahrzyklus mit Stützstellen der Vorsteuerung

Es zeigt die Sollgeschwindigkeit eines Abschnitts zwischen zwei Stopp-Phasen, im rechten Teil einen vergrößerten Ausschnitt. Weder eine Steuerung mit Fehlerschätzer, (4.32), noch ohne, (4.33), ist in der Lage, den Beginn und das Ende einer Standphase oder einer Konstantfahrphase abzusehen. Bei monolithischen Solvern würde hier eine Ereignissteuerung greifen. Die Zeitpunkte $t_{Z,k}$ dieser Beschleunigungssprünge können jedoch dem CS-Master als *time events*, vgl. Kapitel 2.3.2 vorgegeben werden.

Zusätzlich zu dieser simplen Ereignissteuerung sollen in Zeiträumen mit größerer betragsmäßiger Beschleunigung bei gleichem IDC kleinere H_l gewählt werden. Dies erfolgt über eine Anpassung der Verfahrensparameter der H_l-Steuerung: $H_{min}, H_{max}, \alpha_+, \alpha_-, \alpha$, wobei weitere möglich sind. Die Werte werden über die Faktoren $\alpha_{Z,k}$ variiert, die gemeinsam mit den Zeitpunkten in einer Wertetabelle $[\boldsymbol{t}_Z\ \boldsymbol{\alpha}_Z]$ vorgegeben werden. Dabei kann $\alpha_{Z,k} = 0$ gewählt werden für Phasen, in denen das Fahrzeug stehen soll. Da die Co-Simulation in der Vorwärtssimulation mit Fahrermodell, vgl. Kapitel 2.4, eingesetzt wird, wird eine Reaktionszeit vorgehalten und die Schritte an den time events entsprechend

$$H_l := t_{Z,k} - T_l + t_{rkt} \tag{4.61}$$

gesetzt. Die so erfolgende H_l-Korrektur, sowie die Justierung der Parameterwerte wird dem *ALGORITHMUS 4.3* entsprechend ausgeführt.

---

if $l = 0$ **then** $k := 1$; $\alpha_{+,0} := \alpha_+$; $\alpha_{-,0} := \alpha_-$; ... **end**
lese Werte $t_{Z,k}, \alpha_{Z,k}$ aus Vorgabe $[\boldsymbol{t}_Z \ \boldsymbol{\alpha}_Z]$
if $(T_l + H_l) > t_{Z,k}$ **then**
  korrigiere H_l nach (4.61)
  if $\alpha_{Z,k} = 0$ **then**
    $H_{max} := H_{max,fix} \cdot \alpha_Z^0$
    $\alpha_- := min(0,99, \ \alpha_{-,0} \cdot \alpha_Z^0)$; justiere weitere Verfahrensparameter ...
  else
    $H_{max} := H_{max,fix} \cdot \alpha_{Z,k} \cdot \alpha_Z^{H_{lim}}$
    $\alpha_- := min(0,99, \ \alpha_{-,0} \cdot \alpha_{Z,k})$; justiere weitere Verfahrensparameter ...
  end
  $k++$
end

Abbildung 4.19.: *ALGORITHMUS 4.3*: Überlagerte Vorsteuerung von H_l bei Zyklussimulation

Hierbei werden die Applikationsparameter α_Z^0 und $\alpha_Z^{H_{lim}}$ an die Methode der Steuerung angepasst. Dieser Algorithmus zur überlagerten Zyklen(vor-)steuerung ist prinzipiell bei allen Makroschrittweitensteuerungen einsetzbar und wird im nächsten Abschnitt untersucht.

4.5.2. Numerische Experimente

Die o. g. heuristischen Verfahren zur Makroschrittweitensteuerung zielen insbesondere auf die Anwendung auf Multi-Physik-Systeme mit proprietären Simulationswerkzeugen. Sie funktionieren jedoch auch am bekannten Beispiel des Zwei-Massen-Schwingers aus 4.3 und 4.4. Da die Verfahren speziell für das Zielsystem Gesamtfahrzeug mit der Forderung $S_{CS} \overset{!}{\geq} 1$ zu applizieren sind, wird das Vorgehen hier anhand numerischer Experimente mit solchen Modellen gezeigt. Alle Experimente in diesem Abschnitt werden am gleichen Gesamtmodell mit gleichbleibender Parametrierung und Partitionierung sowie Konfiguration der Co-Simulation durchgeführt. Sie unterscheiden sich allein in den Verfahren zur Makroschrittweitensteuerung und deren Parametrierung. Der Startwert wird jeweils nicht via $\boldsymbol{p}$, sondern automatisch zu $H_0 = H_{min}$ gesetzt.

Es kommt wieder ein Pkw-Modell entsprechend Abbildung 2.9 zum Einsatz, welches im Detail in Kapitel 5 beschrieben ist. Für die Co-Simulation wird eine Partitionierung in zwei Modelica-Slaves gewählt - Rumpffahrzeug (RFZ) und thermisches Subsystem (THS), welches Detailmodelle des Öl-und Wasserkreislaufs enthält. Deren Kopplung ist über eine konstante Inzidenzmatrix $\boldsymbol{K}$ der Dimension $m_u \times m_y = 14 \times 14$ umgesetzt, die in Anhang A zu finden ist. Die Teilmodelle sind mit den Kausalschnittstellen aus Abschnitt 4.3.2 ausgestattet. So sind keine zusätzlichen Koppelgesetze $\boldsymbol{f}_M$ und Approximationen $\boldsymbol{a}_M$ nötig. Es gilt $\bar{\boldsymbol{u}} = \boldsymbol{K}\bar{\boldsymbol{y}}$

bzw. $\tilde{\boldsymbol{u}} = \boldsymbol{a}(\boldsymbol{K}\bar{\boldsymbol{y}}, \ldots)$, wobei $\boldsymbol{a}$ aus den Approximationen $\boldsymbol{a}_{RFZ}$ und $\boldsymbol{a}_{THS}$ gemäß Abbildung 4.14 besteht. Entsprechend der Ergebnisse aus Abschnitt 4.4.2 wird für alle Komponenten in $\bar{\boldsymbol{u}}$ *Taylor1* mit $T_G^* = 0$ verwendet. Der durch *Taylor2* entstehende Vorteil rechtfertigt den Mehraufwand hier nicht. Im Folgenden wird jeweils der ECE-Teil des NEFZ (Co-)simuliert, $T_M = 206s$. Zunächst wird das gradientenbasierte Verfahren, danach das Verfahren konstanter Änderungsraten behandelt. Anschließend werden beide mit der Zyklen(vor)steuerung kombiniert und schließlich verglichen.

Numerische Experimente: Gradientenbasiertes Verfahren

Um eine sinnvolle Funktion der Verfahren zu gewährleisten, müssen diese konfiguriert und numerisch appliziert werden. Im vorliegenden Fall betrifft dies zum Einen die Konfiguration der Methode anhand der Wahl der Mittelung $\dot{\boldsymbol{y}}_{l,(M)}$ und der Vorschrift für IDC in (4.39). Zum Anderen sind die Parameter der Strategie ν zu bedaten, d.h. α_+ und α_-. H_{min} und H_{max} gehören in diesem Fall zu den Eingangsparametern $\boldsymbol{p}$, die jedoch ebenfalls der Anwendung entsprechend appliziert werden können. Als Werkzeug für das Vorgehen dient das in 4.2.3 beschrieben Batchverfahren. Die Konfiguration und Strategieparameter werden mittels (Parameter-)Variationen in Voruntersuchungen bestimmt, welche im Weitern als Grundlage für die Analyse des sinnvollen Bereichs der Komponenten von $\boldsymbol{p}$ dienen. Ein als deren Ergebnis gefundenes Optimum $\boldsymbol{p}_{opt}$ nach den Kriterien aus Kapitel 3.3, insbesondere der Effizienz, wird wiederum verwendet, um die Konfiguration der Methode zu überprüfen bzw. zu korrigieren und die Alternativen zu vergleichen.

Die Voruntersuchungen zum gradientenbasierten Verfahren führen auf die Konfiguration mit $\dot{\boldsymbol{y}}_{l,(M)}$ nach (4.41) und IDC nach (4.46). Die Mittelung mit Speichern des Höchstwertes ist in Summe effizienter und die IDC-Wahl fällt auf den etablierten Ansatz, [61]. Die Strategieparameter sind hier sehr offen zu wählen, siehe Abschnitt 4.5.1: $\alpha_+ = 10$ erlaubt ein schnelleres Anwachsen, das via H_{max} in (4.60) enger begrenzt wird. $\alpha_- = 0.01$ muss aufgrund der Modelldynamik und des expliziten Verfahrens klein gewählt werden, wobei H_{min} in (4.60) den Aufwand eingrenzbar macht. Mit diesen Randbedingungen kann die Analyse des $\boldsymbol{p}$-Bereiches für (4.47) durchgeführt werden, welche zunächst H_{min} & H_{max} betrachtet und im Anschluss p_0 & p_1 variiert. Zunächst sind p_0 & p_1 festzuhalten: das Ergebnis der Voruntersuchung liefert $p0 = 2$ und $p1 = 2$, was einer Normierung mit $\dot{\boldsymbol{y}}_{max} = 2\dot{\boldsymbol{y}}_{l,(M)}$ entspräche. Für den zu untersuchenden Bereich für H_{min} & H_{max} gelten im Weitesten die Grenzen, die via H_{min} & H_{max} in Abschnitt 4.5.1 beschrieben sind. In einer Parameterstudie für H_{fix}, die auch in Abbildung 4.28 dargestellt ist, findet sich der Zielbereich für $H_{(m)}$ für die Maßgabe $S_{CS} \overset{!}{\geq} 1$ und $\tau = \left\| \hat{\tilde{\boldsymbol{\tau}}}^*_{(m)}(T_M) \right\|_h < 5\%$. Damit ist anhand der Voruntersuchungen eine weitere Eingrenzung möglich: $H_{min} = 10^{-2}s \ldots 10^{0}s$ und $H_{max} = 2 \cdot 10^{-1}s \ldots 10^{1}s$. Da sich die Intervalle überlappen, fehlt jeweils die entsprechende Ecke in den Ergebnissen in Abbildung 4.20.

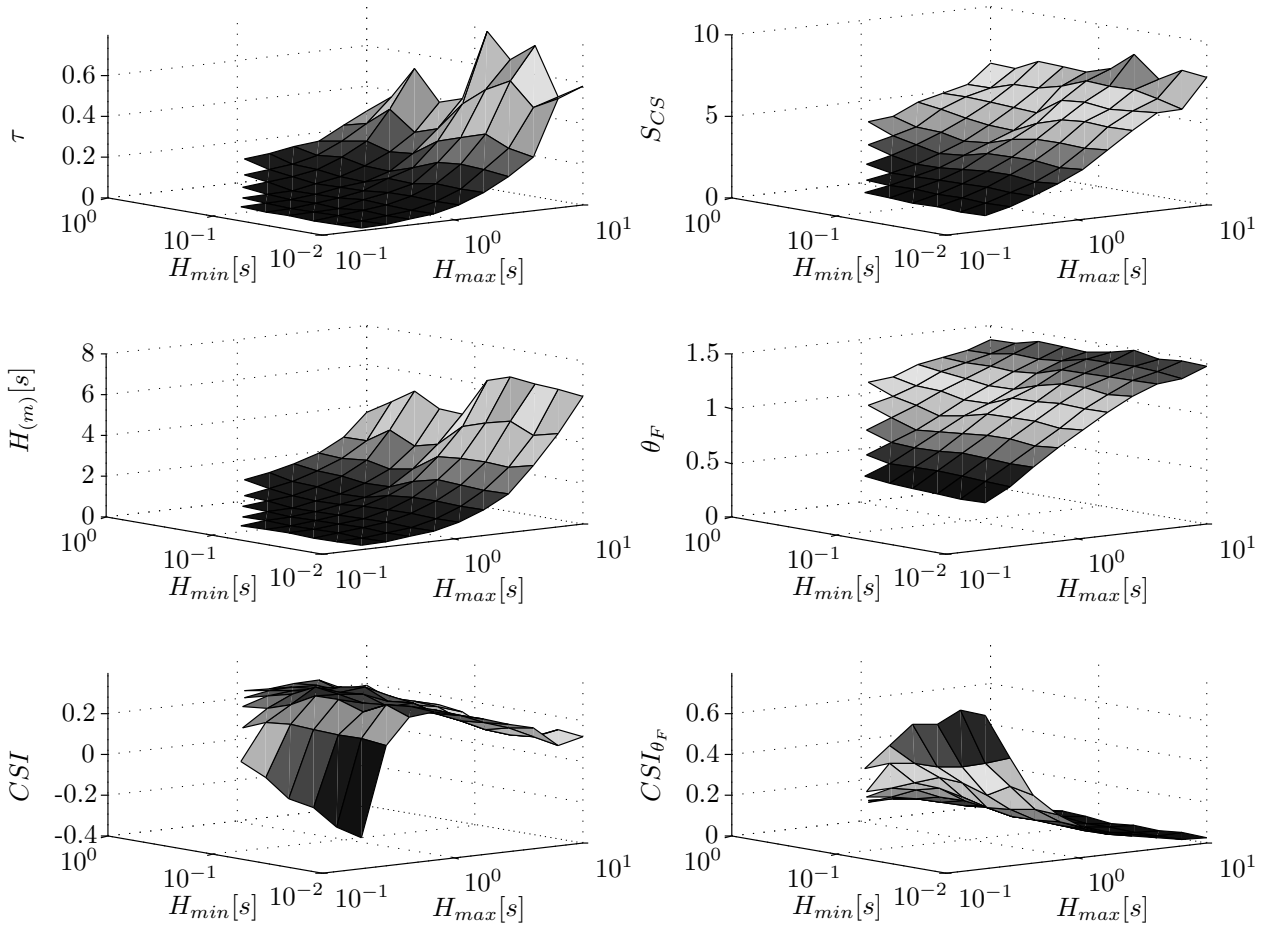


Abbildung 4.20.: Analyse der Variation von H_{min} und H_{max} beim gradientenbasierten Verfahren

Offensichtlich hat H_{max} den deutlich größeren Einfluss auf den Aufwand und damit die mittlere Schrittweite $H_{(m)}$. Somit steigen die Werte für τ, S_{CS} und θ_F ebenfalls v. A. mit H_{max}. Das Verhalten ist jedoch nicht überall monoton, was durch die Zyklensimulation und die unterschiedlich gute Lage zu Ereignissen einzelner T_l zu erklären ist. Für den Bereich $H_{min} < 0.3s$ und $H_{max} < 1,2s$ ergeben sich akzeptable Fehlerwerte von $\tau < 5\%$. In diesem Bereich erreicht auch der Speedup Werte $S_{CS} > 2$, so dass sich ein größeres, flaches Maximum beim CSI ausbildet. Die hohen Werte von θ_F deuten auf eine gute Entkopplung hin; so werden aufgrund der insgesamt großen H_l^* schon bei kleineren $H_{(m)}$ hohe Werte erreicht. Dies führt zu dem Anstieg von CSI_{θ_F} zu kleinem H_{max}, wo jedoch S_{CS} zu klein ist. Somit ist die Wahl effizienter Werte für die H-Grenzen innerhalb des Bereichs hoher CSI-Werte zielführend. Für das weitere Vorgehen wird als Kompromiss zwischen $CSI(\rightarrow 0.32)$ und $S_{CS}(\rightarrow 2.97)$ das Wertepaar: $H_{min} = 0.08s$ & $H_{max} = 1.2s$ verwendet ($CSI_{max} = 0.35$ hätte $S_{CS} = 1.70$).

Um die restlichen Werte in $\boldsymbol{p}$, p_0 & p_1, zu variieren, werden deren Intervalle festgelegt. Sieht man p_0 als Entsprechung des Exponenten eines I-Reglers in Ansätzen nach (2.26), so wäre $p_0 = 1/(q+1)$ mit q als Ordnung der Approximation zu wählen. Mit $q \leq 4$ ergäbe sich

$p_0 \geq 0.2$, was als Untergrenze gewählt wird. Für $p_{0,max}$ soll diese Analogie jedoch nicht gelten, so dass gemäß den Voruntersuchungen für das Intervall gilt: $p_0 = 0.2 \ldots 5$,wobei noch größere Werte zu $S_{CS} < 1$ führen. p_1 ist eher als Applikationsfaktor der Anwendung zu betrachten, für den gilt: $p_1 \downarrow$: $H_{(m)} \rightarrow H_{min}$ bzw. $S_{CS} \downarrow$ und $p_1 \uparrow$: $H_{(m)} \rightarrow H_{max}$ bzw. $\tau \uparrow$ Nach Voruntersuchung wird das Intervall auf $p_1 = 0.5 \ldots 10$ festgelegt.

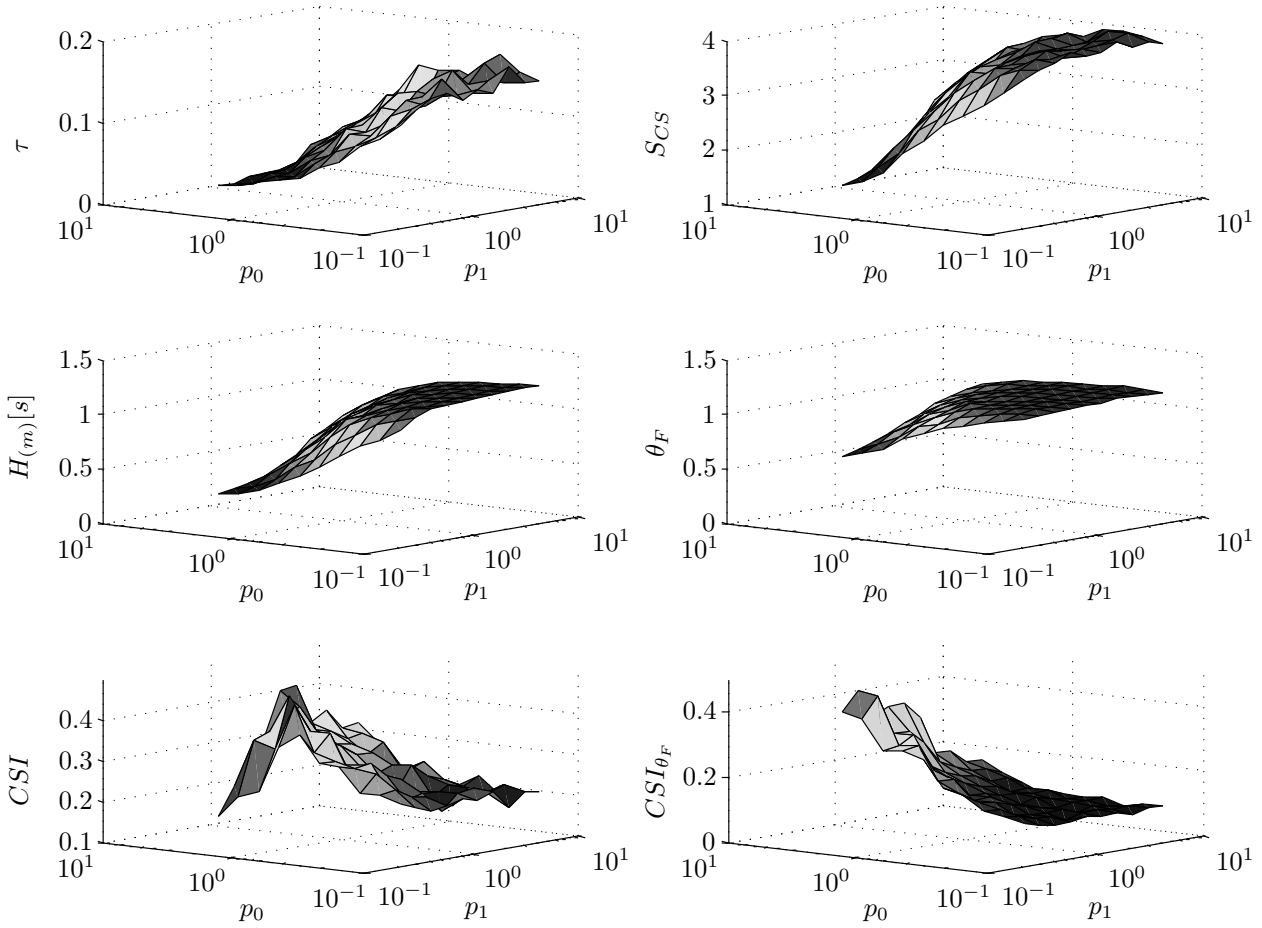


Abbildung 4.21.: Analyse der Variation von p_0 und p_1 beim gradientenbasierten Verfahren

Insgesamt zeigen die Ergebnisse in Abbildung 4.21 eine geringere Varianz als oben bei H_{min} & H_{max}. Die Ergebnisse für τ, S_{CS}, $H_{(m)}$ und θ_F steigen in etwa gleichem Maße mit sinkendem p_0 bzw. steigendem p_1. Jedoch gilt in einem großen Bereich bereits $H_{(m)} \rightarrow H_{max}$, und das Verhalten flacht ab. Die Effizienz ist -wie oben- bei kleinen $H_{(m)}$ größer, wobei im Gegensatz zu CSI_{θ_F} bei CSI ein Maximum durch den abnehmenden S_{CS} entsteht. Dieses ausgeprägte Maximum verläuft diagonal durch den analysierten Bereich. So fällt die Wahl für das Wertepaar $p_0 = 2,5$ & $p_1 = 1$ auf die Mitte dieser Diagonalen.

Somit wird eine optimierte Wahl von $\boldsymbol{p}_{opt} = [0,08 \quad 1,2 \quad 2,5 \quad 1]^T$ für die weiteren Experimente getroffen. Dieser Parametersatz $\boldsymbol{p}_{opt}$ soll auch für eine vergleichende Simulation mit *offline* im Preprocessing nach (4.38) bestimmten Schrittweiten H_l verwendet werden. Hierzu werden mit $\boldsymbol{p}_{opt}$ und *ALGORITHMUS 4.1* mit der Startschrittweite $H_0 = H_{min}$ die Vektoren

$[\boldsymbol{T}_Z\ \boldsymbol{M}_Z]$ bestimmt, was auf $H_{(m)} = 0,518s$ führt. Das Ergebnis findet sich in Abbildung 4.22 bzw. Tabelle 4.6 und wird somit mit den Varianten der *online*-Steuerung vergleichbar.

Für diese Varianten wird zuerst IDC anhand einzelner Koppelgrößen, einzelner Subsysteme und für alle Größen anhand der verschiedenen Normierungen konfiguriert. Diese IDC-Variation wird -wie oben- mit $\dot{\boldsymbol{y}}_{l,(M)}$ nach (4.41) durchgeführt. Anschließend wird eine andere Mittelung $\dot{\boldsymbol{y}}_{l,(M)}$ betrachtet. Die wichtigsten Ergebnisse sind jeweils in Tabelle 4.6 und Abbildung 4.22 dargestellt, deren Lesart mit Abbildung 3.6 erklärt wird. Hier kommt der Wert für θ_F der Konfigurationen hinzu. Weiterhin sind im Bild die beiden Verläufe der Variation fester Makroschritte H_{fix} eingezeichnet.

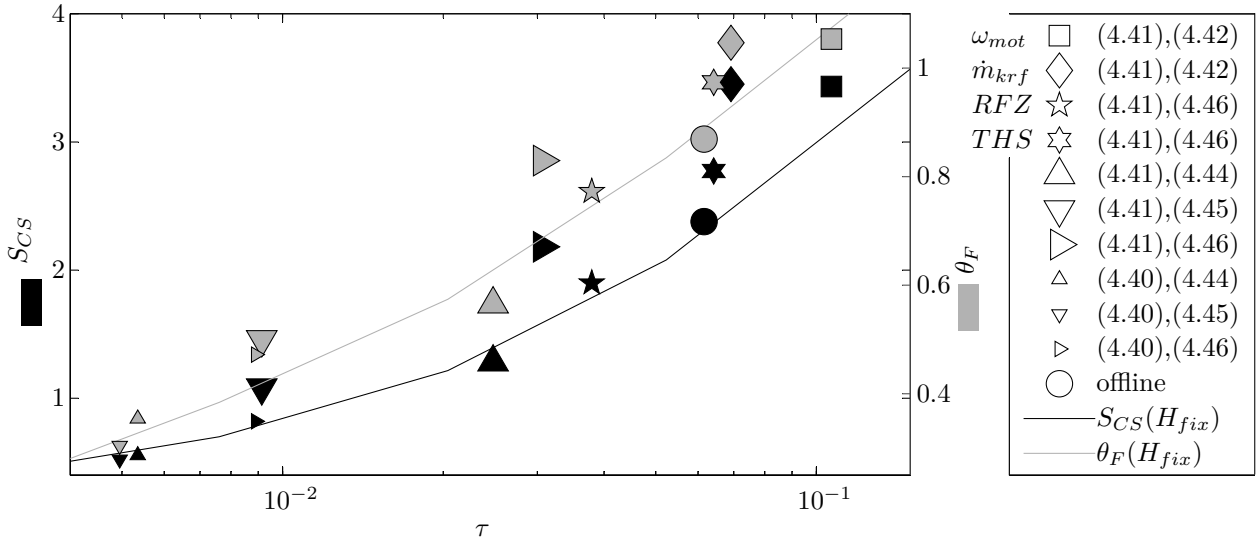


Abbildung 4.22.: CSI-Bewertung für die Varianten des gradientenbasierten Verfahrens

Beispielhaft soll IDC nach (4.42) und mit der Vorgabe i, j gemäß (4.43) für die Einzelgrößen Motordrehzahl ω_{mot} und Kraftstoffmassenstrom $\dot{m}_{krf}$ bestimmt werden. Im Ergebnis mit $\dot{m}_{krf}$ liegen beide Werte, mit ω_{mot} der S_{CS} deutlich oberhalb der H_{fix}-Linien. Die Steuerung würde also schon mit manchen Einzelwerten besser liegen, als H_{fix}.

Nun soll IDC anstatt anhand einer Komponente anhand eines der Subsysteme j bestimmt werden. Die Normierung dessen Komponenten erfolgt -wie oben- mit (4.46). Hier zeigen beide Subsysteme in der Effizienz ähnliche Ergebnisse, wobei THS im Verglich mit H_{fix} etwas besser liegt, jedoch auf zu hohem Fehlerniveau.

Für die folgende Konfiguration mit IDC anhand aller Komponenten und Subsysteme werden schließlich die Normierungen nach (4.44) bis (4.46) verglichen. Im Ergebnis ist die Maximumsnorm nach (4.44) nicht ausreichend für eine Steuerung, die alle Komponenten berücksichtigt, und schneidet so schlechter ab als H_{fix}. Die anderen sind jeweils geeignet, wobei die Euklidische Norm (4.45) mit den gewählten $\boldsymbol{p}_{opt}$ erwartungsgemäß zu kleine H_l liefert, so dass die etablierte, gewichtetet Norm (4.46) auch bei Co-Simulation am effizientesten arbeitet.

Die gleichen Variationen -(4.44) bis (4.46)- werden nun mit $\dot{\boldsymbol{y}}_{l,(M)}$ nach (4.40) durchgeführt. Im Ergebnis sind die Lagen der Normierungen relativ zueinander, wie im Vorigen. Durch die, der Modelldynamik folgend, kleiner werdenden Mittelwerte, ergibt sich jedoch insgesamt ein zu hoher Aufwand.

Tabelle 4.6.: CSI-Ergebnisse der Varianten

$\dot{\boldsymbol{y}}_{l,(M)}$	(4.41)							(4.40)			offline
IDC	(4.42): ω_{mot}	(4.42): $\dot{m}_{krf}$	(4.46), nur RFZ	(4.46), nur THS	(4.44)	(4.45)	(4.46)	(4.44)	(4.45)	(4.46)	
	□	◊	⋆	∗	△	▽	▷	▵	▿	▹	◯
CSI	0,23	0,35	0,24	0,28	0,11	0,09	**0,39**	-0,83	-0,96	-0,20	0,22
CSI_M	0,09	0,13	0,10	0,11	0,08	0,19	0,15	0,19	0,17	0,19	0,08
CSI_{θ_F}	0,10	0,15	0,20	0,15	0,23	0,55	0,27	0,66	0,61	0,53	0,14

Aufgrund des unterschiedlichen Fehlerniveaus in obiger Abbildung sind die Varianten besser anhand Tabelle 4.6 vergleichbar. Sie zeigt die drei Indikatoren nach (3.32), (3.34) und (3.35), wobei CSI nach (3.32) die größte Bedeutung zukommt. Nicht in der Abbildung ersichtlich sind etwa die sehr guten Werte für CSI_{θ_F} für $\dot{\boldsymbol{y}}_{l,(M)}$ nach (4.40). Dies entspricht auch den Erkenntnissen aus den Abbildungen 4.20 und 4.21. Gemäß den Anforderungen wird jedoch $\dot{\boldsymbol{y}}_{l,(M)}$ nach (4.41) nötig, und es bestätigt sich die Beobachtung aus den Voruntersuchungen, ebenso mit IDC nach (4.46).

Für diese Konfiguration ist der Verlauf von H_l in Abbildung 4.23 in der Mitte dargestellt. Die Abbildung verdeutlicht, welche Koppelgröße die jeweilige H_l-Berechnung dominiert. Es sind nur die Größen dargestellt, welche dieses Kriterium im Verlauf von t_{sim} erfüllen; oben die Ausgänge des Fahrzeugteils RFZ, unten die des thermischen Systems THS. Den Ergebnissen der IDC-Variation entsprechend erscheinen hier *mehrere* Größen von *beiden* Subsystemen. Obwohl sich $\dot{x}_{Fzg}$ und ω_{mot} in ihrer Dynamik ähneln und $\dot{x}_{Fzg}$ häufig die kleinen Schrittweiten erzwingt, ist oben bei ω_{mot} (□) ein weit schlechteres Ergebnis als bei der letztlich gewählten Konfiguration zu sehen. Das Nicht-Beachten der THS-Größen, die bei Lastwechseln große Gradienten aufweisen, bewirkt dort zu große H_l mit erheblichem Fehler.

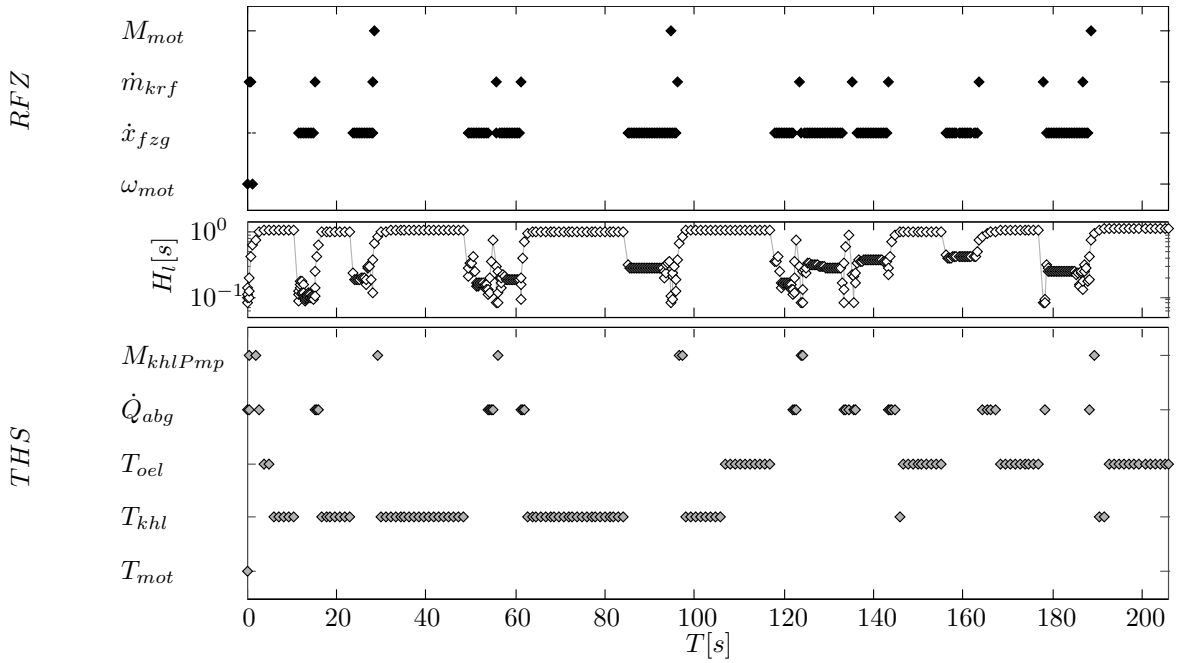


Abbildung 4.23.: Schrittweiten limitierende Komponenten und H_l-Verlauf der gewählten Konfiguration von (4.39) mit $\boldsymbol{p}_{opt} = [0,08 \quad 1,2 \quad 2,5 \quad 1]^T$

Numerische Experimente: Verfahren konstanter Änderungsraten

Das Vorgehen zur Konfiguration der Methode und Applikation des Verfahrens konstanter Änderungsraten (4.48) entspricht grundsätzlich dem des gradientenbasierten Verfahrens (4.39). So werden auch in Voruntersuchungen Werte festgelegt, auf deren Grundlage ein $\boldsymbol{p}_{opt}$ für (4.55) gefunden wird, mit dem wiederum die Konfiguration überprüft wird. Hinzu kommt hier die Bestimmung der Methodenparameter α und q, sowie die der Werte für H_{min} & H_{max}, die hier zu den Strategieparametern gehören. So entsteht der Vorteil, dass der Anwender in $\boldsymbol{p}$ nur die von monolithischen Verfahren gewohnten $\boldsymbol{ATOL}$ und $\boldsymbol{RTOL}$ wählen muss. In Konsequenz dieser Vereinfachung sollen hier skalare Werte verwendet werden, so dass: $ATOL_{ky} = ATOL \; \forall \; k_y$ und $RTOL_{ky} = RTOL \; \forall \; k_y$ gilt.

Zur Analyse der Vorgenannten werden die Konfiguration und Parametrierung des Verfahrens festgelegt: die Mittelung $\boldsymbol{y}_{l,(R)}$ geschieht nach Vorschrift (4.56) und die Normierung IDC obiger Wahl entsprechend nach dem etablierten Ansatz, (4.53). Der sinnvolle Wertebereich für α aus der Literatur bestätigt sich in den Experimenten und wird mit $\alpha = 0,9$ gewählt. Der o.g. Bedatung des I-Reglers entsprechend müsste $q = 0,5$ gewählt werden, was aufgrund der Ergebnisse auf $q = 0,4$ korrigiert wird. Der Strategieparameter $\alpha_+ = 2,5$ fällt kleiner aus als oben, da hier die H-Grenzen offener gewählt werden müssen. Die Werte sind entsprechend: $\alpha_- = 0,1$ bzw. $H_{min} = 10^{-6}s$ und $H_{max} = 10^{1}s$. Damit wird der Wertebereich für $ATOL$ & $RTOL$ analysiert, wobei die Intervalle zu $ATOL = 10^{-3} \ldots 10^{1}$ bzw. $RTOL = 10^{-3} \ldots 10^{0}$ festgelegt werden.

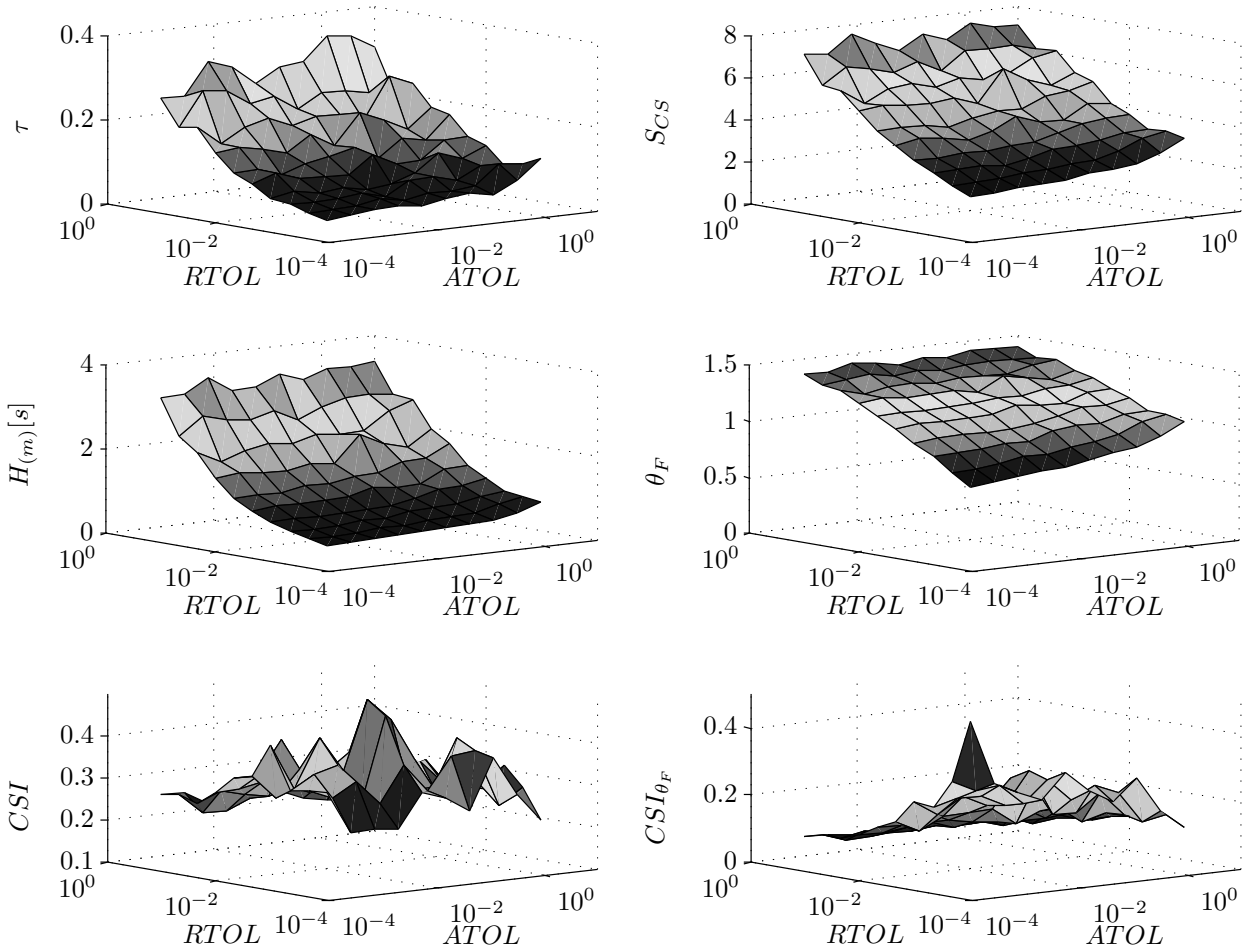


Abbildung 4.24.: Analyse der Variation von $ATOL$ und $RTOL$ beim Verfahren konstanter Änderungsraten

Das Ergebnis ist in Abbildung 4.24 dargestellt. Man sieht, dass der Fehler τ hauptsächlich von $RTOL$ abhängt, was die Funktionalität von $ATOL$ nur als notwendiger Begrenzer bei Nulldurchgängen bestätigt. So werden auch S_{CS}, $H_{(m)}$ und θ_F entsprechend von $RTOL$ dominiert. Aus den o.g. Gründen verläuft das Kennfeld auch hier nicht ganz monoton. Ebenfalls aufgrund der gleichen CS-Konfiguration zeigt sich anhand der hohen Werte von θ_F die gut funktionierende Entkopplung. Dies führt wieder zu hohem θ_F-Niveau bei kleinen $ATOL$ & $RTOL$, so dass CSI_{θ_F} zu kleinen Toleranzen hin ansteigt. Aufgrund des S_{CS}-Verhaltens ergibt sich ein insgesamt flaches, hohes CSI-Niveau, d.h. das Verfahren ist gut für Anwendereingaben $\boldsymbol{p}$ im analysierten Intervall geeignet. Es befindet sich ein kleines Maximum im Bereich $ATOL = 5 \cdot 10^{-2} \ldots 2 \cdot 10^{-1}$ & $RTOL = 5 \cdot 10^{-3} \ldots 2 \cdot 10^{-2}$, so dass für die weitere Untersuchung $\boldsymbol{p}_{opt} = [ATOL = 10^{-1} \quad RTOL = 10^{-2}]^T$ gewählt wird.

Die Variation der Konfiguration erfolgt -wie oben-, indem zunächst IDC bei konstanter Mittelung $\boldsymbol{y}_{l,(R)}$ (nach (4.56)) verglichen und anschließend $\boldsymbol{y}_{l,(R)}$ selbst variiert wird. Die Ergebnisse der wichtigsten Kombinationen sind in der CSI-Bewertung in Abbildung 4.25 sowie in Tabelle 4.7 zusammengefasst.

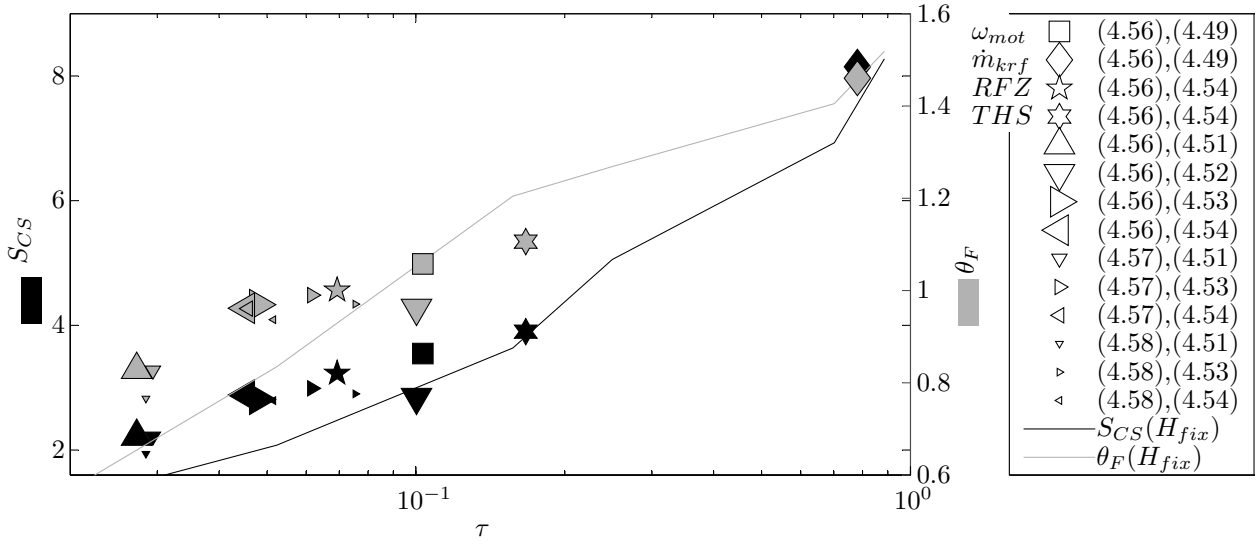


Abbildung 4.25.: CSI-Bewertung für die Varianten des Verfahrens konstanter Änderungsraten

Die Indikation anhand von Einzelgrößen, (4.49), führt bei beiden Beispielen zu inakzeptablem Fehler. Die extremen Werte für $\dot{m}_{krf}$ stellen hierbei einen Ausreißer dar, der in diesem Einzelfall auf ein für die Größenordnung der SI-Einheit $[kg/s]$ viel zu hohes $ATOL$ zurückzuführen ist. Für den Vergleich der Verwendung einzelner Subsysteme wird IDC nach (4.54) bestimmt, vgl. unten. Auch hier ist das Fehlerniveau noch zu hoch, wobei die Effizienz ebenfalls nicht ausreichend wäre.

Der Vergleich der IDC-Varianten (4.51) bis (4.54) ergibt, dass (4.52) für diese Steuerung nicht funktioniert (was bei allen Mittelungen gilt). (4.51) führt zu den höchsten CSI-Werten, ist jedoch zu konservativ beim Aufwand (S_{CS}). Die Modifikation (4.54) des etablierten Ansatzes (4.53) schneidet leicht besser ab, auch im Kompromiss CSI und S_{CS}. Die Variation der Mittelung von (4.56) bis (4.58) zeigt jeweils das gleiche Verhältnis zwischen den IDC-Varianten. Das beste Niveau erreicht, der Tabelle 4.7 entsprechend, $\boldsymbol{y}_{l,(R)}$ nach (4.56), was der direkten Verwendung der Ausgangswerte für $RTOL$ entspricht. Insgesamt wird $\boldsymbol{p}_{opt}$ zusammen mit der Konfiguration (4.56)&(4.54) unter Verwendung sämtlicher Komponenten in $\bar{\boldsymbol{y}}_{ges}$ für die Analysen im Folgenden eingestellt.

Tabelle 4.7.: CSI-Ergebnisse der Varianten

$\boldsymbol{y}_{l,(R)}$	(4.56)								(4.57)			(4.58)		
IDC	(4.49): ω_{mot}	(4.49): $\dot{m}_{krf}$	(4.54), nur RFZ	(4.54), nur THS	(4.51)	(4.52)	(4.53)	(4.54)	(4.51)	(4.53)	(4.54)	(4.51)	(4.53)	(4.54)
	$\square$	$\Diamond$	$\star$	$*$	$\triangle$	$\triangledown$	$\triangleright$	$\triangleleft$	$\triangledown$	$\triangleright$	$\triangleleft$	$\triangledown$	$\triangleright$	$\triangleleft$
CSI	0,25	0,09	0,32	0,17	**0,45**	0,18	0,38	**0,42**	0,41	0,32	0,40	0,33	0,25	0,35
CSI_M	0,09	0,07	0,10	0,06	0,15	0,06	0,14	0,14	0,14	0,11	0,13	0,12	0,09	0,11
CSI_{θ_F}	0,10	0,02	0,14	0,07	0,30	0,10	0,20	0,21	0,28	0,16	0,21	0,27	0,13	0,18

Numerische Experimente: Vorsteuerung Fahrzyklen

Gemäß seiner Umsetzung ist *ALGORITHMUS 4.3* aus 4.5.1 für alle Steuerungen anwendbar, vgl. *ALGORITHMUS 4.2*. So soll hier die Wirksamkeit dieser Vorsteuerung an beiden Vertretern nachgewiesen werden: dem gradientenbasierten Verfahren mit (4.41)&(4.46) sowie dem Verfahren konstanter Änderungsraten mit (4.56)&(4.54). Die Verfahren erkennen nach *ALGORITHMUS 4.2* selbstständig, ob die Zyklenvorsteuerung aktiv ist. Dadurch wird auch das automatische Setzen von $H_0 := H_{min}$ geändert in $H_0 := t_{Z,1} - T_0$. Zur Vergleichbarkeit soll jedoch hier $H_0 := H_{min}$, einstellbar über $\boldsymbol{p}$, weiterhin gelten.

Die Bedatung der Wertetabelle $[\boldsymbol{t}_Z\ \boldsymbol{\alpha}_Z]$ erfolgt in Anlehnung an Abbildung 4.18. Hierzu wird der Zyklus in Phasen eingeteilt, deren Änderung häufig mit dem Auftreten von Ereignissen zusammenfällt. Die hier verwendete Einteilung geht aus Tabelle 4.8 hervor und orientiert sich an Kriterien aus den Zyklusvorgaben: $v_Z = \dot{x}_{soll}$ bzw. $a_Z = \ddot{x}_{soll}$. Die Schaltphasen, mit resultierendem hohen $|\dot{\omega}_{mot}|$, *können* anhand einer Gangvorgabe, a priori aus dem Preprocessing, bekannt sein. Dies ist beim NEFZ und WLTC möglich. Es wird eine konservative

Tabelle 4.8.: Zyklenphasen für die Bedatung von $[\boldsymbol{t}_Z\ \boldsymbol{\alpha}_Z]$

α_Z - Phasen	Stand $v_Z = 0$	Beschl. $\lvert a_Z\rvert > a_{lim}$	konstant $\lvert a_Z\rvert < a_{lim}$	Schalten $\lvert\dot{\omega}_{mot}\rvert \uparrow$
konservativ	0	0,5	1	0,25
schnell	0	1	2	0,5

mit einer schnellen Variante verglichen. Für dynamischere Zyklen wäre ferner eine Bedatung nach *ALGORITHMUS 4.1* denkbar.

Das so bestimmte $[\boldsymbol{t}_Z\ \boldsymbol{\alpha}_Z]$ ist wie der Algorithmus selbst unabhängig vom Verfahren. Die Anpassung ans jeweilige Verfahren erfolgt über die Parameter α_Z^0 und $\alpha_Z^{H_{lim}}$, wobei α_Z^0 in den Standphasen und $\alpha_Z^{H_{lim}}$ in den restlichen Phasen wirkt. Zusätzlich zur allgemeingültigen

Anpassung der Methoden-/Verfahrensparameter in Abbildung 4.19 werden die folgenden verfahrensabhängigen Anpassungen durchgeführt: bei der

- gradientenbasierten Methode (4.39) wird der *ALGORITHMUS 4.3* um

$$H_{min} := \begin{cases} H_{min,fix} \cdot \alpha_Z^0 & \text{für } \alpha_{Z,k} = 0 \\ H_{min,fix} \cdot \alpha_{Z,k} \cdot \alpha_Z^{H_{lim}} & \text{für } \alpha_{Z,k} \neq 0 \end{cases} \tag{4.62}$$

$$\alpha_+ := \begin{cases} max(1,01, (1 + \alpha_{+,0}/5\alpha_Z^0)) & \text{für } \alpha_{Z,k} = 0 \\ max(1,01, (1 + \alpha_{+,0}/5\alpha_{Z,k})) & \text{für } \alpha_{Z,k} \neq 0, \end{cases} \tag{4.63}$$

- bei der Methode konstanter Änderungsraten (4.48) wird der *ALGORITHMUS 4.3* um

$$\alpha := \begin{cases} \alpha_0(1 + \alpha_Z^0/10) & \text{für } \alpha_{Z,k} = 0 \\ \alpha_0(\alpha_{Z,k})^{0,1} & \text{für } \alpha_{Z,k} \neq 0 \end{cases} \tag{4.64}$$

$$\alpha_+ := \begin{cases} \alpha_{+,0} \cdot \alpha_Z^0 & \text{für } \alpha_{Z,k} = 0 \\ max(1,01,\ \alpha_{+,0} \cdot \alpha_{Z,k}) & \text{für } \alpha_{Z,k} \neq 0 \end{cases} \tag{4.65}$$

erweitert.

Anhand dieser Parameter kann die Vorsteuerung eingestellt und Vergleiche durchgeführt werden. Wie in den Abschnitten oben sind die Ergebnisse dargestellt in Abbildung 4.26 und Tabelle 4.9. Die Basis bildet jeweils das Ergebnis des Verfahrens ohne Vorsteuerung. Im ersten Schritt ist jeweils der Einfluss der einfachen „Eventvorsteuerung“ mit gleichem $\boldsymbol{t}_Z$, aber $\alpha_{Z,k} = 1 \ \forall \ k$ aufgezeigt.

Beim gradientenbasierten Verfahren führt dies bereits zu einer Steigerung der Effizienzwerte um 50%, wobei hauptsächlich der Fehler reduziert wird. Der Vergleich zweier Bedatungen für $\alpha_Z^{H_{lim}}$ für beide Varianten der Wertetabelle zeigt, dass H_l durch $\alpha_Z^{H_{lim}} = 0,5$ zu stark begrenzt wird. Vergleichen der Ergebnisse der konservativen mit der schnellen Umsetzung und der „Eventvorsteuerung“ zeigt, dass jeweils die Effizienzwerte für CSI bzw. CSI_{θ_F}, wenn auch leicht, gesteigert werden.

Aufgrund der offeneren H-Grenzen ist beim Verfahren konstanter Änderungsraten analog zu den Werten für α_+ und α_- auch mehr Begrenzung durch α_Z^0 und $\alpha_Z^{H_{lim}}$ notwendig. Somit wird $H_{(m)}$ durch die Vorsteuerung infolge des Setzens zusätzlicher Schritte bei den Events auch jeweils verringert. Der Einfluss der einfachen „Eventvorsteuerung“ führt hier dagegen schon zu mehr als verdoppelten Effizienzwerten. Durch die Wahl der schnellen Variante von $[\boldsymbol{t}_Z\ \boldsymbol{\alpha}_Z]$ ist wieder eine leichte Steigerung möglich.

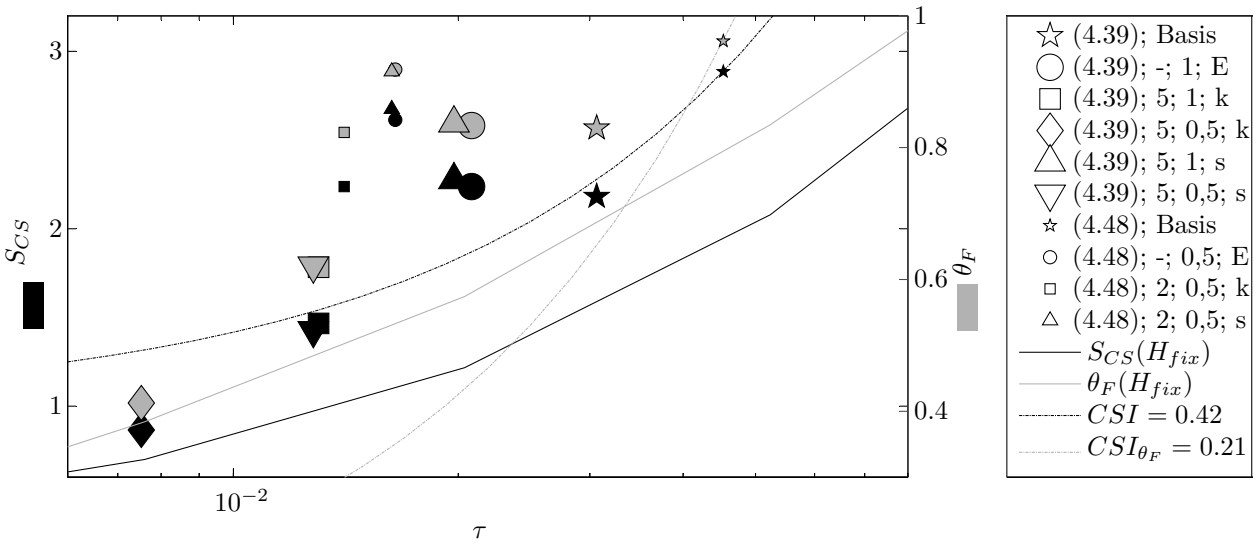


Abbildung 4.26.: CSI-Bewertung beider Verfahren mit Variation der Vorsteuerung

Tabelle 4.9.: CSI-Ergebnisse der Varianten

Verfahren	**gradientenbasiert** (4.39)						**konst. Änderung** (4.48)			
α_Z^0	Basis ohne Vorsteuerung	-	5	5	5	5	Basis ohne Vorsteuerung	-	2	2
$\alpha_Z^{H_{lim}}$		1	1	0,5	1	0,5		0,5	0,5	0,5
$[\boldsymbol{t}_Z\ \boldsymbol{\alpha}_Z]$		nur Events	konservativ	konservativ	schnell	schnell		nur Events	konservativ	schnell
	$\star$	○	□	◊	△	▽	$\star$	○	□	△
CSI	0,39	0,59	0,36	-0,18	0,65	0,33	0,42	0,98	0,88	**1,03**
CSI_M	0,15	0,22	0,18	0,16	0,22	0,19	0,14	0,33	0,29	0,34
CSI_{θ_F}	0,27	0,40	0,48	0,55	0,42	0,49	0,21	0,56	0,58	0,56

Insgesamt kommt der Hauptvorteil der Vorsteuerung aus der Information zu den Events $\boldsymbol{t}_Z$, wie beim simulierten Zyklus auch zu erwarten. Durch die Bedatung $\boldsymbol{\alpha}_Z$ besteht dann noch die Möglichkeit, den Schwerpunkt zu mehr Genauigkeit oder weniger Aufwand zu verschieben. Ein direkter Vergleich der beiden Verfahren mit Vorsteuerung findet sich im folgenden Abschnitt.

Numerische Experimente: Vergleiche

Um die zeitlichen Verläufe der Makroschrittweitensteuerung zu vergleichen, werden zur Wahrung der Übersichtlichkeit in Abbildung 4.27 je ein Vertreter der beiden unterschiedlichen Methoden (4.39) und (4.48) mit aktiver Vorsteuerung (Bedatung: schnell) verglichen. Die mittlere Schrittweite $H_{(m)}$ liegt bei beiden in der gleichen Größenordnung, so dass der direkte Vergleich mit $H_{fix} = H_{(m)} = 0,46s$ hinzugefügt ist. In den beiden rechten Diagrammen

ist ein voller ECE-Abschnitt dargestellt. Links zeigt die Vergrößerung den Übergang in eine Stillstandsphase, ab $t_{soll} = 96s$. In den oberen Diagrammen ist der Verlauf der Makroschritte H_l dargestellt, unten der *relative lokale Fehler* $\tilde{\tau}^*(T_l)$. Dies ist der *lokale* Fehler aller Komponenten $\tilde{\tau}_{k_y}$ aus (3.16), bezogen auf $y_{(m),k_y}$ und normiert gemäß (3.31), worin sich τ_{k_y} entsprechend nach

$$\tau_{k_y} := \tilde{\tau}^*_{k_y} = \frac{\tilde{\tau}_{k_y}(T_l)}{y_{(m),k_y}(T_M)} \tag{4.66}$$

berechnet. $\tilde{\tau}^*(T_l)$ kann in einzelnen Schritten sehr große Werte annehmen, da es sich nicht um einen mittleren Fehler, sondern um jenen am Ende des Makroschrittes handelt. Es können jedoch bereits wenige große lokale Fehler genügen, um den globalen Fehler τ aus $\hat{\tilde{\boldsymbol{\tau}}}^*_{(m)}(T_M)$, der zur CSI-Berechnung herangezogen wird, zu verändern. So kann besonders bei bei einer auf Effizienz ausgelegten Wahl eines H_{fix} ein Ergebnis evtl. unbrauchbar werden.

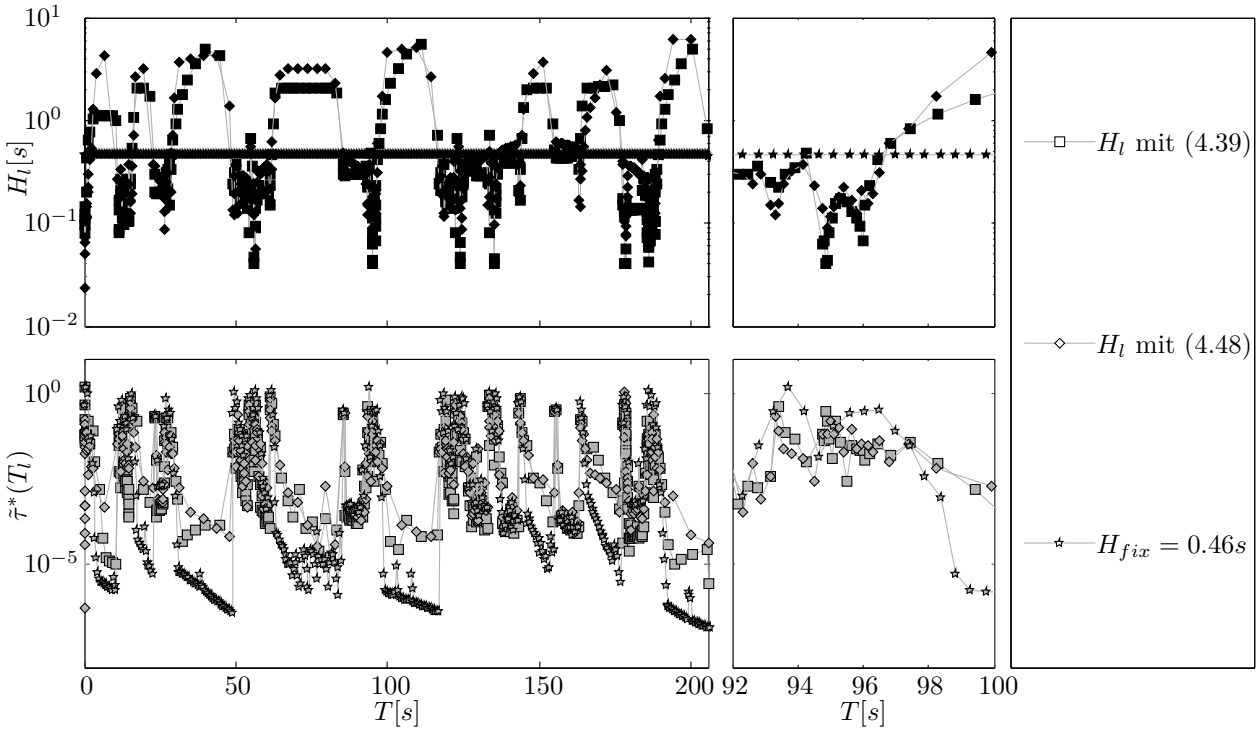


Abbildung 4.27.: Verlauf der Schrittweiten H_l und des Fehlers $\tilde{\tau}^*(T_l)$ beider Verfahren mit Vergleich bei fester Schrittweite H_{fix}

Weiterhin ist anhand der Verläufe zu sehen, dass etwa das Erhöhen der Schrittweite in Standphasen keinen zu großen Fehler zur Folge hat, oder die Wahl von kleinen α_- -Werten in den Strategien nötig und zielführend ist.

Der Gesamtvergleich der entwickelten Verfahren, auch ohne Vorsteuerung, gegen die CS mit festen Schrittweiten ($H_{(m)} = H_{fix}$) hinsichtlich Effizienz wird in Abbildung 4.28 gezogen. Es sind jeweils die Ergebnisse für CSI, CSI_{θ_F} und CSI_M über der entsprechenden mittleren Schrittweite $H_{(m)}$ abgebildet. Der linke Teil umfasst ein größeres $H_{(m)}$ -Intervall,

für das die CSI-Verläufe der Variation von H_{fix} sichtbar sind. Aufgrund des Verlaufs von CSI ist der Bereich erkennbar, der den Anforderungen an die CS genügt, $CSI > 0$ bzw. $\tau < 5\%$(nicht dargestellt). Entsprechend sind nun die Verfahren appliziert, so dass ihre Ergebnisse in diesem Bereich liegen. Er ist im rechten Teil vergrößert dargestellt und bietet eine Zusammenfassung über die zur Verfügung stehende Auswahl der entwickelten Verfahren.

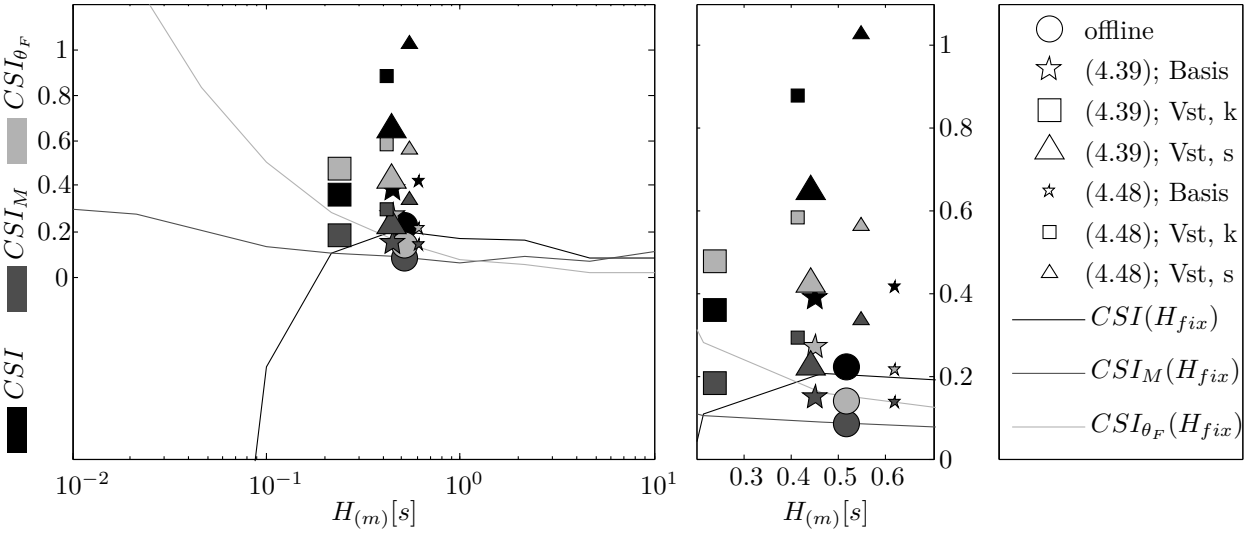


Abbildung 4.28.: Vergleiche der CSI-Werte der Verfahren zur Makroschrittweitensteuerung und Vergleich mit der Variation fester Schrittweiten H_{fix}

Insbesondere die Verfahren mit aktiver Vorsteuerung ('Vstrg.') funktionieren sehr effizient, doch auch ohne Vorsteuerung ('Basis') liegen die Resultate besser als bei H_{fix} und können somit angewendet werden. Allein das *offline-Verfahren* rechtfertigt den vorab nötigen Aufwand der $[\boldsymbol{T}_Z\ \boldsymbol{M}_Z]$ -Bestimmung nicht. Beim *gradientenbasierten Verfahren* (4.39) besteht der Aufwand nur noch im Bedaten von $\boldsymbol{p}$, (4.47), wobei er sich bei ähnlichem Anwendungsfall auf die Wahl von H_{max} reduzieren lässt. Das setzt aber weiterhin eine genauere Kenntnis des Modells und der CS voraus. Dagegen ist das *Verfahren konstanter Änderungsraten* (4.48) mit der üblichen Bedatung $ATOL$ & $RTOL$ eher geeignet, mit oder ohne der von $\boldsymbol{p}$ unabhängigen Vorsteuerung.

4.5.3. Umsetzung der Makroschrittweitensteuerung

Die Implementierung der Makroschrittweitensteuerung innerhalb des *MDPCosim* -Masters wird bereits in Abbildung 4.3 und Abschnitt 4.2.1 erläutert. Im Programmablauf wird an den Stellen $INIT\ H\ Algorithm$ bzw. $H := f(H\ Algorithm)$ je der *ALGORITHMUS 4.2* ausgeführt. Die Konfiguration durch den Benutzer erfolgt über die Datei *masterpar.dat*, die in Anhang C beschrieben ist. Hier wird festgelegt, ob, mit welchem Verfahren und welchen Verfahrensparametern $\boldsymbol{p}$ eine variable Makroschrittweite verwendet wird. Deren Umsetzung

ist unabhängig vom Masteralgorithmus modular aufgebaut, ähnlich dem Einsatz unterschiedlicher Konnektorklassen. Das Einlesen der Wertetabellen $[\boldsymbol{T}_Z\ \boldsymbol{M}_Z]$ für ein offline-Verfahren erfolgt ebenso, wie das der Wertetabellen $[\boldsymbol{t}_Z\ \boldsymbol{\alpha}_Z]$ für die Zyklen-Vorsteuerung mithilfe weiterer Konfigurationsdateien, *masterpar.varstep.dat* bzw. *cycle.dat*.

Bei der Bedatung von $\boldsymbol{p}$ beim *gradientenbesierten Verfahren* können Aufwand und Genauigkeit der CS über die Wahl von H_{max} innerhalb der Grenzen $H^*_{max} = 5 \cdot 10^{-4} \ldots 5 \cdot 10^{-3}$ variiert werden.

[116] schlägt eine Wahl $macTOL = 100 micTOL$ vor, wobei $micTOL$ der Toleranz der Solver der Slaves *tol* entspricht. $macTOL$ (bei Entsprechung $ATOL = RTOL$: „$macTOL$“) bezieht sich auf den lokalen Fehler, wohingegen sich TOL beim *Verfahren konstanter Änderungsraten* auf den Diskretisierungsfehler bezieht: $TOL = \Delta y_{zul} \approx \bar{\tau}_{zul} > \tilde{\tau}_{zul} \approx macTOL$ und somit größer gewählt werden kann. Es kann hier jedoch als Empfehlung einer Untergrenze $RTOL_{min} = 100 tol$ gelten. Eine Ausnahme sind Komponenten in $\boldsymbol{y}$, welche sich monoton steigend verhalten (etwa eine Fahrstrecke) und für die $RTOL$ entsprechend klein bzw. $RTOL = 0 \wedge ARTOL > 0$ zu setzen ist. Aufgrund der bei Multi-Physik-Simulation auftretenden unterschiedlichen Größenordnungen sollte $ATOL$ im Hinblick auf jene gewählt werden.

5. Anwendung in der Gesamtfahrzeugsimulation

In Kapitel 5 wird schließlich die Anwendung des oben behandelten Verfahrens zur Co-Simulation in der entwickelten Fahrzeugmodellbibliothek beschrieben. Die Fahrzeugmodelle, -Bibliothek und -Bedatung werden zunächst in Abschnitt 5.1 beschrieben, worauf in 5.2 die Co-Simulation in diesem Kontext bewertet und die Ergebnisse gezeigt werden.

Ziel der Simulation

Die Motivation und Definition der Gesamtfahrzeugsimulation ist in Kapitel 2.4 bereits beschrieben. Hier sollen alle energieflussrelevanten physikalischen Domänen entsprechend Abbildung 2.9 in einer Zyklusvorgabe simuliert werden. Derzeit in Europa noch aktuell ist der NEFZ [34], der in Bild 5.1 dargestellt ist. Er gibt das Fahrprofil $v(t) = \dot{x}(t)$ sowie

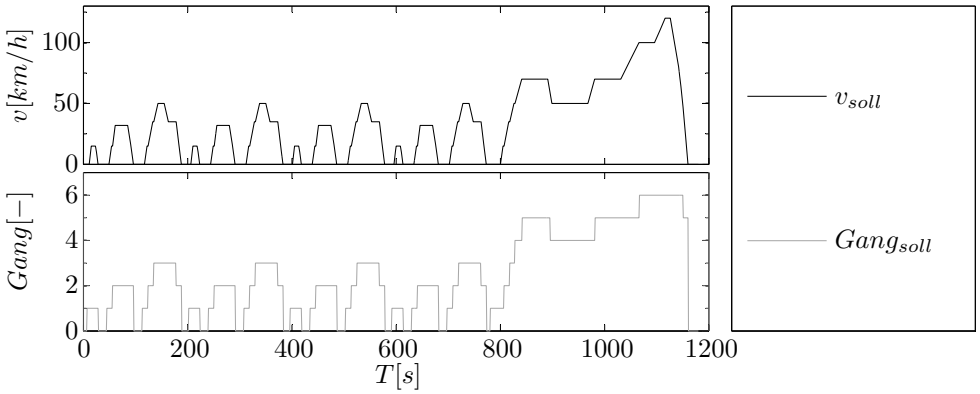


Abbildung 5.1.: *N*euer *E*uropäischer *F*ahr*Z*yklus mit Vorgabe für 6-Gang-Handschalter

für Fahrzeuge mit manuellem Getriebe auch die Schaltpunkte $Gang(t)$ vor. Auf Basis solcher Verbrauchs-/und Emissionierungszyklen wird die Erreichung gesetzlich vorgeschriebener Grenzwerte untersucht. Vor dem Hintergrund der künftigen CO_2-Flottenwerte ist das globale Ziel der Simulation hier die Bewertung des CO_2-Ausstoßes.

Hierzu dient der zeit- und kostengünstige simulative Vergleich unterschiedlicher (alternativer) Antriebstechnologien sowie einzelner Komponenten. Mittels Energieflussbetrachtung, etwa der rekuperierbaren Energie, können so CO_2-Einsparpotentiale identifiziert werden. Mit

dem gleichen Werkzeug sollen so zum Einen Fahrzeugtopologien: konventionelles Fahrzeug, (P)HEV[1], (B)EV[2] und weitere betrachtet werden, zum Anderen bei gleicher Topologie verschiedene Subsysteme bzw. Komponenten. Dies sind Betriebsstrategie, Elektrifizierung von Nebenaggregaten oder thermische Maßnahmen, wie die Nutzung der Abgasenthalpie oder der Abwärme im Kühlsystem, beispielsweise durch thermoelektrische Generatoren, Rankine-Prozesse oder Wärmespeicher. Ferner können neben der Potentialbewertung auch Lastspuren für den Motorprüfstand, insbesondere für Nutzfahrzeuge, für die keine Fahrzeugrollenprüfstände verfügbar sind, erzeugt werden.

5.1. Aufbau der Fahrzeugmodelle

Dieser Abschnitt liefert zunächst die Beschreibung des (physikalischen) Fahrzeugmodells und dessen Umsetzung in Modelica, bevor anschließend in 5.1.1 und 5.1.2 die Entwicklung des modularen Aufbaus der Fahrzeugmodelle und die Fahrzeugbibliothek beschrieben werden. Es handelt sich um Modellierung für Vorwärtssimulation gemäß Abbildung 2.8b) und 2.9. Als Beispiel für ein konventionelles oder HEV-Pkw-Modell zeigt Abbildung 5.2 die graphische Oberfläche in Modelica.

Der mittlere Teil stellt das mechanische Modell des Antriebsstranges als Kern des Fahrzeugmodells dar. Die einzelnen Subsystemblöcke sind mit mechanischen Konnektoren - vergleiche Abschnitt 2.2.1 - verbunden. Im oberen Teil finden sich die kausalen, signalbasierten Modellteile, die wie jedes Subsystem über einen Signalbus gekoppelt sind. Dies sind ein Block mit experimentunabhängigen, globalen Parametern, z.B. zur Konfiguration, weiterhin die Steuerung, die ihrerseits in die Subsysteme Fahrermodell, Motorsteuerung, Getriebesteuerung, etc. aufgeteilt ist, und ein Monitoring-Block, in dem z.B. Energiebilanzen wie der Verbrauch berechnet werden. Unterhalb des Antriebsstrangs sind die weiteren physikalischen Domänen angeschlossen. Das thermische System Ths, der Abgasstrang Exh, der zu Beginn von Kapitel 4 als AS beschrieben ist, sowie das elektrische Bordnetzmodell Pnt, das etwa die Batterie enthält. Zusätzlich sind elektrische Verbinder zwischen allen physikalischen Subsystemblöcken vorgesehen. Ganz unten finden sich schließlich die Systemparameter, wie Gravitationswert und die Experimentparameter, die z.B. Anfangszustände oder den Zyklus definieren.

Der mechanische Modellteil beginnt links in Bild 5.2 mit den Nebenaggregaten Acs: sie beinhalten den Riementrieb, sowie Starter und Ölpumpe. Abbildung 5.6 zeigt den Aufbau. Daran schließt das Modell des Verbrennungsmotors Ced an, welches hier als Ein-Massen-Modell kennfeldbasiert umgesetzt ist. Neben Verbrauchs- und Reibkennfeld beinhaltet es auch Kennfelder, die die Dynamik des Luftsystems darstellen etc. Daneben befindet sich

[1](Plug-In) Hybrid Electric Vehicle
[2](Battery) Electric Vehicle

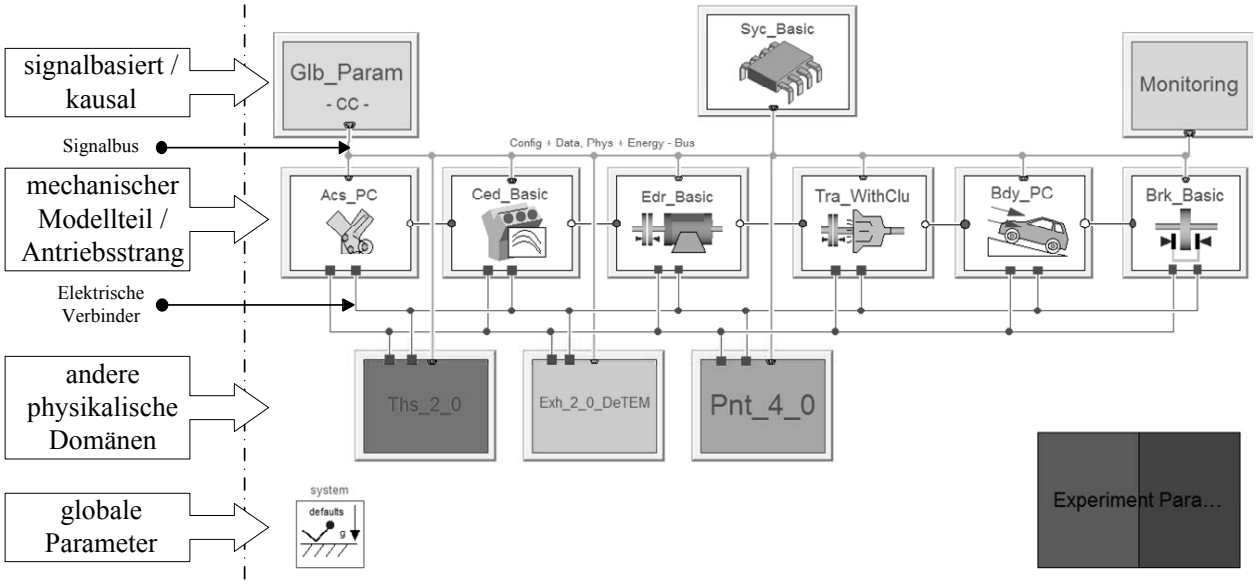


Abbildung 5.2.: Oberste Modellebene des Pkw-Modells in Modelica

der (optionale) E-Antrieb *Edr*, der eine Trennkupplung für eine Umsetzung als P2-HEV[3] haben kann. Neben paralleler ist auch eine serielle Topologie mit zwei E-Maschinen machbar oder die Modellierung als konventionelles Fahrzeug, welches bei *Edr* dann aus einem mechanischen Durchtrieb besteht. In *Tra* sind Anfahrelement und Getriebe modelliert, hier mit kennfeldbasierten Verlusten und mit zwei Massen. Innerhalb des Modells *Tra* besitzt etwa das Modell einer Anfahrkupplung, siehe Abbildung 5.3, weitere Massen. Das Fahrzeugchassis *Bdy* beinhaltet den Abtrieb mit Differential- und evtl. Verteilergetriebe. Weiterhin sind darin die Radträgheiten, die Fahrzeugmasse und die Fahrwiderstände gemäß (2.32) abgebildet. Schließlich befindet sich unter *Brk* das Modell der Reibbremsen. Hier wird -analog zu den anderen Blöcken- die Reibleistung berechnet und auf den Signalbus geschrieben. Dies erfolgt auch innerhalb des *Tra*-Submodells, dessen Kupplungsmodell in Abbildung 5.3 gezeigt wird. Es könnte hier entsprechend durch ein Modell eines Drehmomentwandlers ersetzt werden. Links im Bild ist die Ansicht des Kupplung innerhalb des *Tra*-Modells, rechts der Inhalt zu sehen. Die Reibung ist entsprechend Abbildung 2.3 a) umgesetzt. Details des thermischen Submodells sind unter Abschnitt 5.1.3 beschrieben.

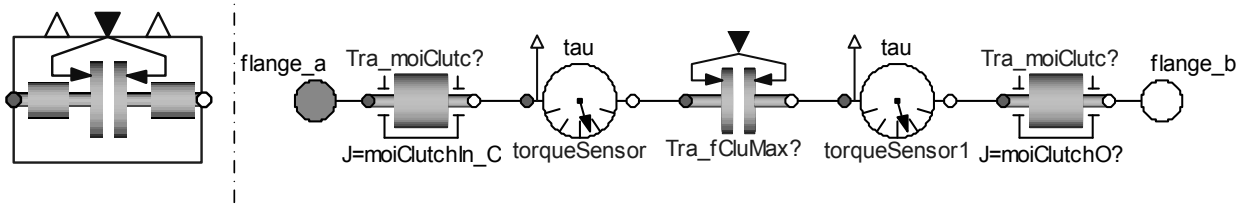


Abbildung 5.3.: Detailansicht des Modells der Anfahrkupplung

[3]Parallelhybrid mit der Option rein elektrischen Fahrens

5.1.1. Modulare Bibliothek

Im Rahmen dieser Arbeit wurde eine modular aufgebaute Fahrzeugmodellbibliothek unter dem Namen *AdAM*[4] maßgeblich mitentwickelt. Sie stützt sich auf Modellierungsrichtlinien, die die Modellarchitektur, einheitliche Schnittstellen sowie Namen betreffen. Damit genügt sie vergleichbaren Anforderungen an große Systembibliotheken, wie etwa in [73], und ermöglicht so eine organisationsübergreifende Zusammenarbeit. Die Bibliothek umfasst Komponentenmodelle unterschiedlichen Detaillierungsgrades sowie Gesamtfahrzeugmodelle.

Die Modularität bietet hier drei Vorteile: zunächst wird so Entwicklern unterschiedlicher technischer Domänen ermöglicht, gleichzeitig an der Bibliothek oder gar an einem Modell zu arbeiten; ein Weiteres ist die Vererbung bei objektorientierter Modellierung: So können einheitliche Basismodelle erstellt werden, deren Änderungen dann konsistent in allen weiteren Instanzen des Modells wirksam werden. Weiterhin ist so keine Doppelarbeit, z.B bei Behebung eines Fehlers, nötig. Drittens können Modelle eines Subsystems in verschiedenen Detaillierungsleveln, vgl. Abbildung 2.2, bzw. externe Modelle, oder eben Schnittstellen für Co-Simulation eingesetzt werden.

Umsetzung

Die Umsetzung erfolgt in *AdAM* durch konsequente Anwendung der Eigenschaft von Modelica als objektorientierte Sprache. So zeigt Abbildung 5.4 rechts die Grundarchitektur des Pkw-Modells. Hier wird definiert, welche *Base*−Klassen als Subsysteme zum Gesamtmodell gehören. Diese sind als *partialmodel* deklariert und enthalten lediglich Schnittstellendefinitionen. Über *replaceable* und *redeclare* werden sie mit den entsprechenden Modellobjekten belegt, wie z.B. dem Kennfeldmodell des Verbrennungsmotors. Aufgrund der hierarchischen

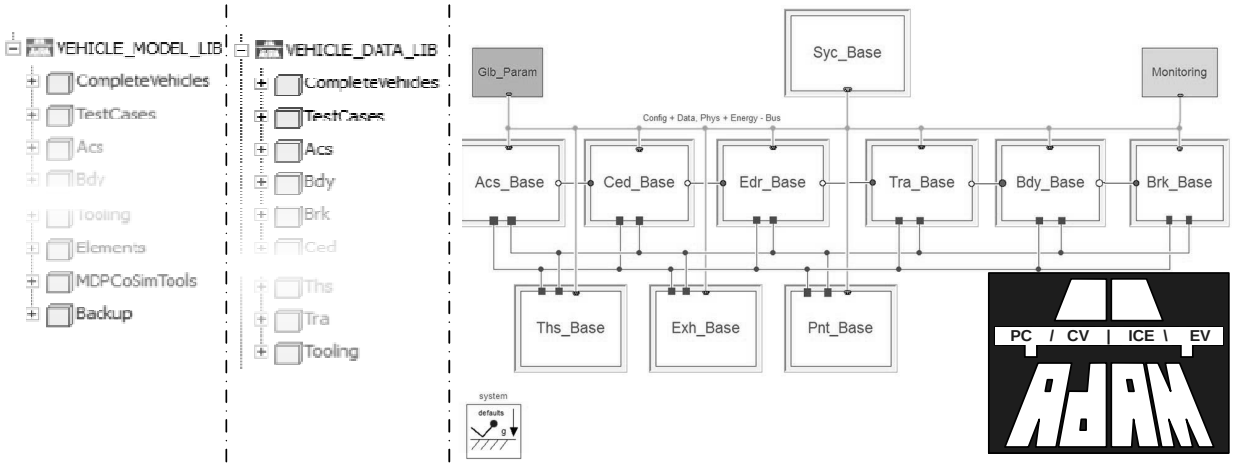


Abbildung 5.4.: Modell-und Datenbibliothek, sowie modulare Pkw-Architektur

[4] *Ad*vanced *A*utomotive *M*odelling

Modellstruktur setzt sich dies innerhalb der Subsysteme (*Acs*, *Tra*, *Ths*, ...) fort, innerhalb derer wieder Architekturen mit *replaceable*-Komponenten -etwa für Kupplung oder Drehmomentwandler- hinterlegt sind. Entsprechend ist auch die Modellbibliothek aufgeteilt in Gesamtfahrzeug- und Subsystemmodelle. Außer den elektrischen und den mechanischen Verbindern im Antriebsstrang sind die physikalischen Kopplungen über die Domänengrenzen hinweg mittels der Kausalschnittstellen aus Abbildung 4.11 umgesetzt, welche über den bidirektionalen Signalbus verbunden werden. Dies gilt für monolithische wie auch für Co-Simulation, vgl. 5.1.2, so dass für die CS keine zusätzlichen Modelle nötig werden.

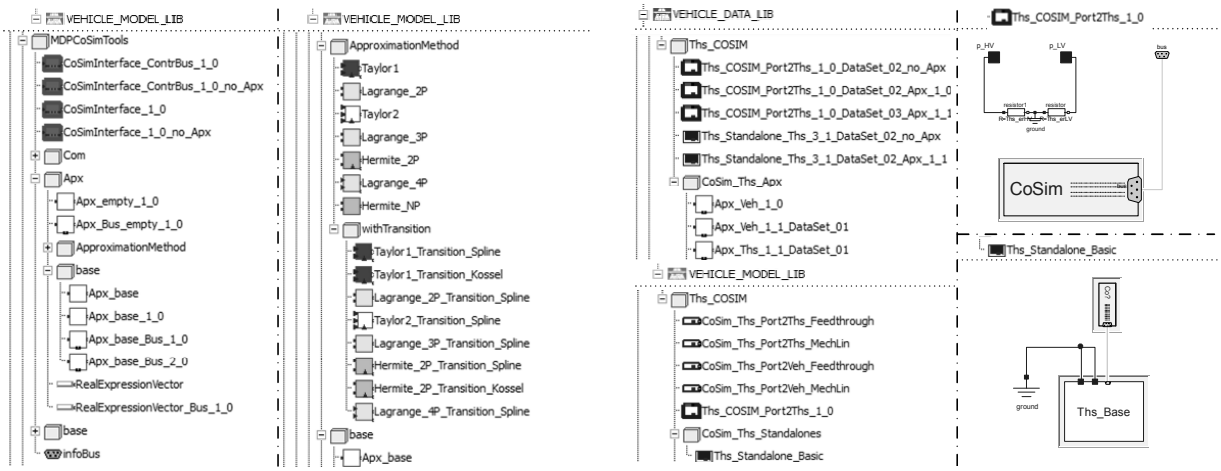
Ein weiterer Grundsatz in *AdAM* ist die Trennung von Daten und Modell. So ist im Bild links die *MODEL_LIB* sowie die *DATA_LIB* dargestellt. Letztere spiegelt weitestgehend die Struktur der Modellbibliothek wieder. Die unbedateten Modelle aus dem Modellteil werden in den Datenteil vererbt und mit Parameterwerten vervollständigt, siehe 5.1.4. Simuliert werden schließlich die Modelle aus der Datenbibliothek. Auf diese Weise ist es möglich, mehrere Datensätze für einen konsistent bleibenden Modellstand zu verwenden. Die Datenhaltung erfolgt mittels Versionierungswerkzeug innerhalb des Firmennetzwerks und ermöglicht so die kontinuierliche, parallele Nutzung und Weiterentwicklung der Modelle.

5.1.2. Modulare Schnittstellen

Das Konzept der modularen Schnittstellen besteht hier aus zwei Bausteinen: die Modellstruktur mit Kausalschnittstellen und Bussystem sowie die Modellbibliothek mit CS-Schnittstellen.

Wie oben erwähnt, finden über die Subsystemgrenzen hinweg die Kausalschnittstellen Verwendung, die die physikalischen Verbindungen über Signale ermöglichen, sowohl in monolithischer wie auch Co-Simulation. Bei monolithischer Simulation bleibt der Vorteil der Formelvorverarbeitung, vgl.2.2.1 erhalten, wobei die CS die benötigten kausalen Koppelgrößen erhält. Unterstützt wird dies durch den generellen Modellaufbau aus Abbildung 5.4 mit dem Signalbus, der alle Blöcke verbindet und einen *CausalSubBus* enthält, wie am Beispiel in Abbildung 5.6 dargestellt. Für die CS werden die Vektoren $\boldsymbol{y}_j$ und $\boldsymbol{u}_j$ zusammen mit den Kontrollbussignalen, vgl. 4.2.2, in einem *CoSimSubBus* zusammengefasst.

Die Modellbibliothek für die Co-Simulation besteht wiederum aus zwei Bestandteilen. So zeigt Abbildung 5.5(a) die Modelica-seitige Umsetzung für *MDPCosim* und 5.5(b) die Schnittstellenbibliothek in *AdAM* für die jeweiligen Subsystemmodelle. Die *MDPCoSimTools* sind innerhalb *AdAM* nur in der *MODEL_LIB* implementiert, da ihre jeweilige Bedatung innerhalb der Instanziierung in den *Ports* der Schnittstellenbibliothek erfolgt. Der Aufbau der *MDPCoSimTools* folgt entsprechend den oben genannten Richtlinien und teilt sich in die Subsysteme *Com* für die Verbindung mit dem Master sowie *Apx* für die Approximationsmethoden auf, vgl. Abbildung 4.14. Die Schnittstellenbibliothek spiegelt sich in *MODEL_LIB* und *DATA_LIB* wieder: So zeigt Bild 5.5(b) das entsprechende Beispiel



(a) Modelica-MDPCS-Schnittstellenbibliothek

(b) AdAM-Schnittstellenbibliothek: Beispiel thermisches Subsystem

Abbildung 5.5.: Bibliothek für den Modelica-Slave, sowie modulare Schnittstellen in *AdAM*

für das thermische System Ths_COSIM. Unten links ist der Teil, der die *Ports* beinhaltet, über die die Extrahierung der Vektoren $\boldsymbol{y}_j$ und $\boldsymbol{u}_j$ modelliert ist. Auch die Subsystemschnittstelle $Ths\_COSIM\_\ldots$, die in das Ths-Subsystem in Bild 5.4 eingesetzt wird, sowie der darüber gekoppelte Ths-Slave als $Ths\_Standalone\_\ldots$ sind in der Bibliothek. Darüber ist der korrespondierende Teil der $DATA\_LIB$, über den die Bedatung u. A. mit $n_{y,j}, n^c_{y,j}, n_{u,j}, n^c_{u,j}$ und $n^i_{y,j}$ sowie die Auswahl der Approximation vorgenommen wird. Rechts davon sind die entsprechenden Architekturen der Subsysteme für die CS dargestellt.

5.1.3. Thermisches Subsystem

Zwei Drittel der Kraftstoffenergie gehen in Wärme über, so dass die Modellierung des thermischen Systems Ths für die Energieflüsse im Fahrzeug wichtig ist [54, 75, 83, 96, 127, 128]. Eine Nutzung der Wärmeenergie kann auch beim konventionellen Fahrzeug zu einer Reduktion der CO_2-Emissionen führen. Zur Potentialabschätzung dieser Nutzung wird folglich eine detaillierte Modellierung des Ths notwendig.

Zunächst soll die mechanische Ankopplung des Ths an den Antriebsstrang erläutert werden. Abbildung 5.6 zeigt im linken Teil das Subsystem *Acs* mit der Anbindung der Nebenaggregate: Wasserpumpe, Ölpumpe, Klimakompressor und Generator (sowie Ritzelstarter). Möglich ist jeweils die elektrische oder mechanische Ankopplung, wobei im Vorliegenden jeweils mechanisch an die Kurbelwelle gekoppelt wird. So ist im rechten Bildteil die Verbindung mit der Ölpumpe dargestellt, die meist über ein Ritzel im Ölsumpf des Verbrennungsmotors mitläuft. Somit wird aus Modellsicht das Ritzel $\theta_{\text{Öl}P}$ über eine Übersetzung starr mit der Kurbelwelle θ_{Ced} verbunden. Freigeschnitten wird dahinter mittels Kausalschnittstelle, die als Potentialsender die Signale mit dem $CausalSubBus$ austauscht.

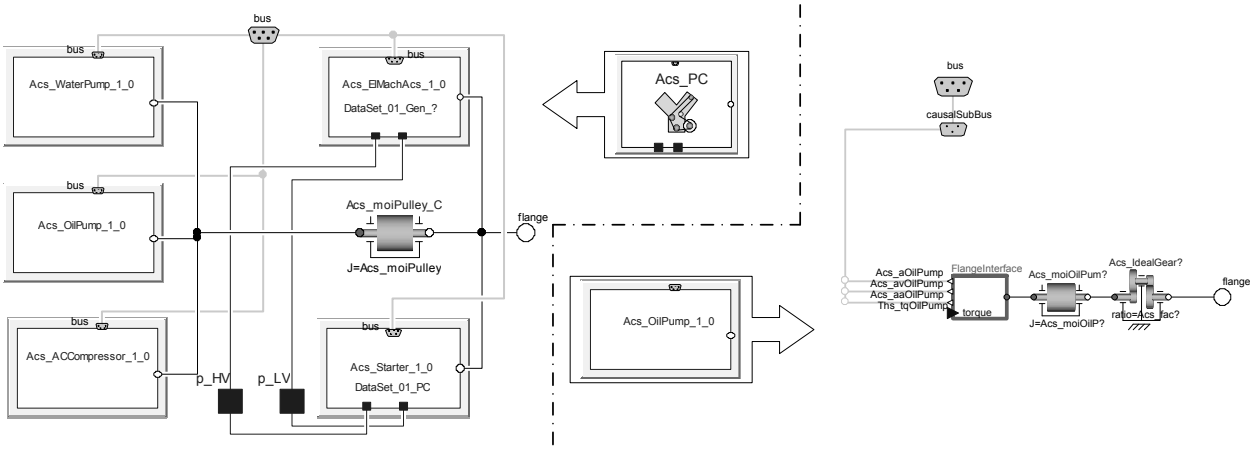


Abbildung 5.6.: Mechanische Kopplung der Nebenaggregate mit Kausalschnittstellen

Das thermische Subsystem selbst kann in verschiedenen Detaillierungsgraden modelliert sein: die Trivialumsetzung gibt lediglich konstante Werte für die abgefragten Temperaturen und Aggregatsmomente auf den Bus. Die nächste Ausbaustufe ist ein Eine-Masse-Modell zur Bestimmung der Temperatur. Die detaillierte Modellierung schließlich beinhaltet ihrerseits wieder Sub(sub)systeme und wird anhand Abbildung 5.7 erläutert. Oben links ist das Mo-

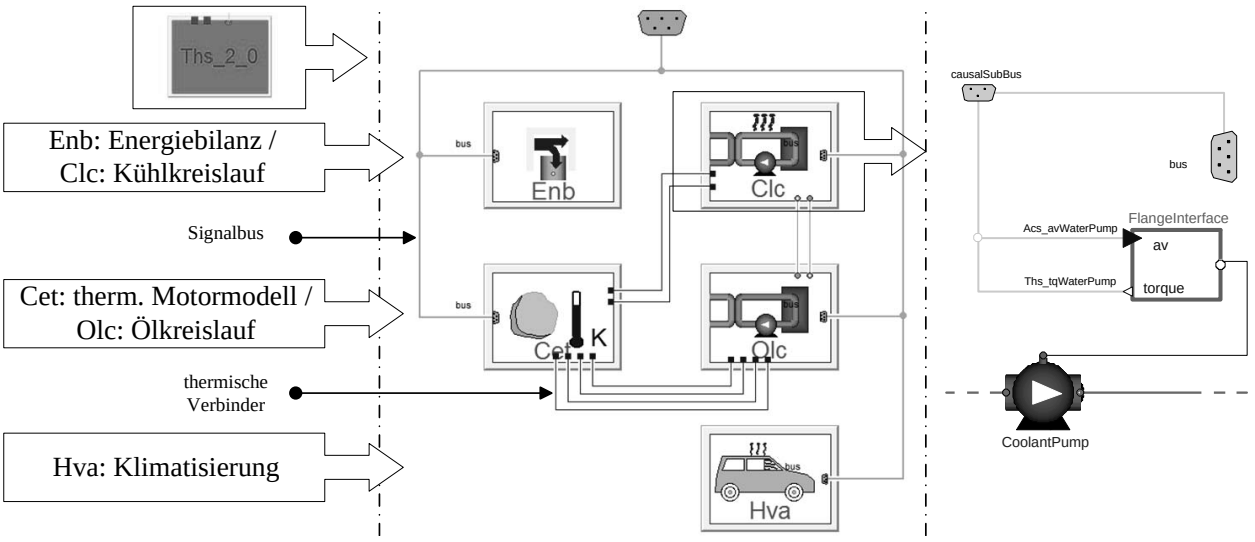


Abbildung 5.7.: Subsystemebene des thermischen Systems mit der weiteren Untergliederung

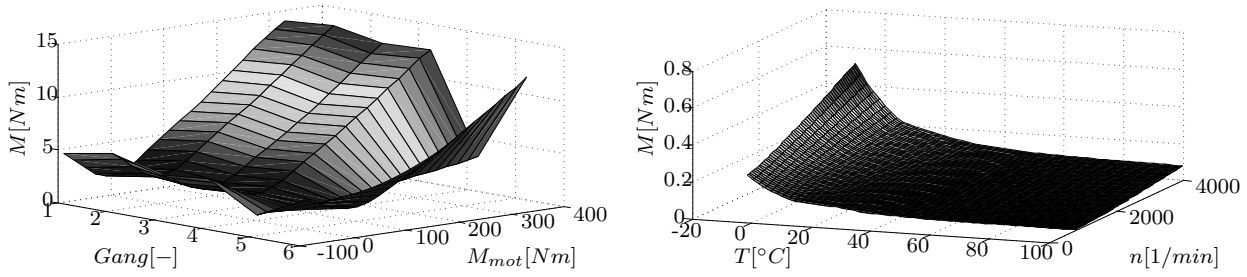
dell für die Energiebilanz *Enb*, über das die Aufteilung der Kraftstoffenergie in Antrieb und Reibung sowie Abgas- und restliche Wärme bestimmt wird. Das Modell des Kühlkreislaufs *Clc* beinhaltet die hydraulische Modellierung mit Wasserpumpe und Kühler. *Cet* fasst das Motormodell in thermischen Massen zusammen und *Olc* beinhaltet den Ölkreislauf mit Pumpe, Sumpf und Wärmetauscher zum Kühlwasser. Unter *Hva* ist der Klimakompressor und -kreislauf dargestellt. Die Blöcke sind über den *CausalSubBus* und weitere Signale an des restliche Fahrzeugmodell gekoppelt. Innerhalb des *Ths* können einzelne Blöcke wiederum modular getauscht werden. So kann ein einfaches Kennfeldmodell oder eine physikalische Model-

lierung z.B. für den Kühl- oder Ölkreislauf -auch mit Komponenten zur Wärmespeicherung- eingesetzt werden. Auf *Ths*-Ebene sind die Teilmodelle mit hydraulischen und thermischen Konnektoren verbunden. Über die Domänengrenze hinweg kommen aber auch dafür wieder Kausalschnittstellen zum Einsatz, etwa für den Wärmeeintrag eines AGR-Kühlers aus dem Subsystem Abgasstrang ins Kühlwasser.

5.1.4. Vermessung und Bedatung

Die Bedatung der Modelle erfolgt in *AdAM* in der *DATA_LIB*, wobei unterschieden werden muss in Konfiguration des Fahrzeugs und in die eigentliche Bedatung. Die Konfiguration ist die Auswahl der bedateten Subsystemmodelle der *DATA_LIB* via *redeclare*-Befehl. Auf diese Wiese wird das Gesamtahrzeug zusammengefügt, wobei die Konfiguration der Hierarchie folgend auf Subsystemebene und schließlich auf Fahrzeugebene erfolgt. Gleichermaßen können so eines oder mehrere Subsysteme für die Ausführung in Co-Simulation konfiguriert werden. Für die eigentliche Bedatung werden den (Komponenten-)Modellen der *MODEL_LIB* ihre Parameterwerte zugewiesen. Dies erfolgt über Vererbung des Modells via *extend*-Befehl, wobei lediglich die Werte bzw. Kennfelder beschrieben werden, was auch aus externen Dateien bedient werden kann. Je nach Detaillierungslevel sind die Parameter in den Subsystemen zusammengefasst, was die Handhabbarkeit erhöht.

Abbildung 2.5 folgend müssen die physikalischen Daten vom realen System, sofern möglich, durch Messung abgeleitet und andernfalls durch Schätzung oder Literaturwerte gefunden werden. So ist im Hause Bosch aufgrund des Betriebs zahlreicher Motorprüfstände der Zugriff auf die Bedatung von Verbrennungsmotoren problemlos möglich. Fahrzeugdaten, die auch für Rollenmessungen im Hause benötigt werden, können vom Fahrzeughersteller oder aus Ausrollmessungen ermittelt werden. Damit können die Fahrwiderstände in (2.32) parametriert werden. Bei anderen Modellkomponenten muss auf Abschätzen, etwa bei den Radträgheiten θ_{Rad} oder Berechnen, wie für Wandlerkennlinien gemäß [57], zurückgegriffen werden.



(a) Verlustmoment beim Handschalter unter Teillast

(b) Schleppmoment der Ölpumpe

Abbildung 5.8.: Gemessene Komponentenkennfelder für Verlustmomente

Neben der Vermessung der Komponente Verbrennungsmotor werden auch andere anhand

von Prüfstandsmessungen bedatet, teilweise extern beauftragt. So zeigt Abbildung 5.8(a) die Verlustkennlinien in Abhängigkeit der Last für jeden Gang eines manuellen Sechsganggetriebes im warmen Zustand, das auf einem Antriebsstrangprüfstand vermessen wurde. In 5.8(b) ist das Schleppmomentenkennfeld einer außerhalb des Motors vermessenen Ölpumpe über Temperatur und Drehzahl dargestellt. Wird mit diesen Daten die Ölpumpe gesondert modelliert, wie bei Verwendung des Ths aus 5.1.3, so ist bei der Bedatung des Verbrennungsmotors zu beachten, dass die Vermessung auf dem Motorprüfstand inklusive der Ölpumpe erfolgt. Die Momentenaufnahme der Ölpumpe muss dann entsprechend

$$M^*_{mot,skf}(n,T) = M_{mot,skf}(n,T) - M_{olp}(n,T) \tag{5.1}$$

aus dem Schleppkennfeld des Motors abgezogen werden. In der $DATA\_LIB$ werden entsprechend zwei Bedatungen des gleichen Motors hinterlegt, jeweils mit und ohne Ölpumpe, je nach Konfiguration des Ths.

5.2. Ergebnisse der Co-Simulation des Gesamtfahrzeugs

Im folgenden Abschnitt wird nun die Anwendung des in Kapitel 4 vorgestellten, hier entwickelten Verfahrens für die Co-Simulation mit *MDPCosim* und mit der oben beschriebenen Fahrzeugmodellbibliothek *AdAM* gezeigt. Zunächst wird die Umsetzung anhand der numerischen Aspekte aus 3.2 erläutert und später anhand der Kriterien aus 3.3 bewertet.

Das Gesamtfahrzeugmodell des Pkw mit detaillierter Abbildung des thermischen Systems Ths führt in monolithischer Simulation zu einem Echtzeitfaktor von $\theta_E \approx 1,5$. Mit dem Ziel, besser als Echtzeit zu werden, d.h. $S_{CS} \overset{!}{\geq} 1,5$ wird eine **Partitionierung** des Modells vorgenommen; gemäß Tabelle 3.1 also mit einer Werkzeugkopplung (IIb) zum Zwecke einer partitionierten Simulation (IIa). Simuliert man das Ths bzw. das restlichen Fahrzeugmodell Rfz getrennt, durch Ersetzen der fehlenden Busgrößen mit Verläufen $\boldsymbol{u}_j(t)$, so erhält man mit $\theta_A = 12,1$ nach (2.34) einen sehr großen Wert, welcher eine Partitionierung in die Slavemodelle RFZ und THS befürwortet. Entsprechend der Abschnitte 5.1.2 bis 5.1.4 wird die Konfiguration, die in Abbildung 5.9 zu sehen ist, unter Verwendung der $CS-Ports$ aus der Bibliothek vorgenommen. Die Konfiguration von *MDPCosim* erfolgt wie unter 4.2.1 beschrieben: Die Definition der Subsysteme wird mit *mastersys.dat*, die der Kommunikatorklassen (vgl. 4.2.2) mit *Slave<Name>.dat* durchgeführt, siehe Anhang C.

Die Umsetzung der **Kausalität** wird teilweise in 5.1.3 gezeigt. Die Kausalschnittstellen in den *Acs*-Blöcken für Anschluss von Wasserpumpe, Ölpumpe und Klimakompressor sind als Potentialsender ausgeführt, vgl. 4.3.2. Es gilt je Aggregat Acs_i für die Bedatung der Speicherträgheit θ_K: $\theta_K << \theta_{Acs_i} + \theta_{mot}$. Im THS finden sich in den Teilmodellen die daran gekoppelten Flusssender, die die Momente von Pumpen und Kompressor, resultierend aus

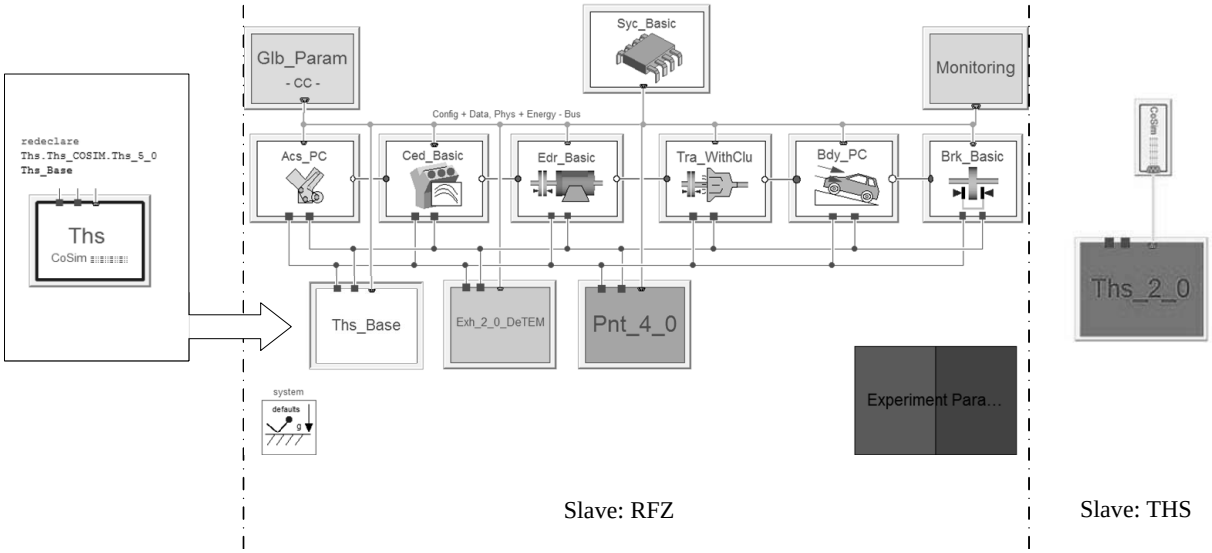


Abbildung 5.9.: Konfiguration des Gesamtfahrzeugmodells für die Co-Simulation und Partitionierung in die Modelica-Slave-Modelle: *RFZ* und *THS*

hydraulischer Leistungsaufnehme sowie Schleppmoment, ausgeben. Sie sind in der Ausführung $r_{Ths,Clc} = r_{Ths,Olc} = r_{Ths,Hva} = 1$ gewählt und jeweils mit $\Omega^*_{krit} = 10^3$ bedatet (da eine Wahl von 10^4 zu viel Aufwand nach sich zöge). Die Konfiguration der $m_u \times m_y = 14 \times 14$-Inzidenzmatrix $\boldsymbol{K}$ in Anhang A für die Kopplung in *MDPCosim* erfolgt mithilfe von *mastercon.dat* aus Anhang C.

Wie bereits in Kapitel 4.5.2 wird hier die **Approximation** für alle Komponenten in $\tilde{\boldsymbol{u}}$ gleich gewählt, so dass a_{k_y}: $Taylor1$ und $T^*_{G,k_y} = 0\ \forall k_y$. Zur Bedatung der Limitierung aus (4.30) und (4.31) wird bei den im Ths ermittelten Temperaturen gewählt: $\tilde{u}_{RFZ,i,min} = 0K$; $\tilde{u}_{RFZ,i,max} = 500K$. In Abhängigkeit der aus dem Motormodell bekannten Maximaldrehzahl $\omega_{Mot,max}$ gilt für die Drehzahlen der Nebenaggregate mit Übersetzung i_{Acs_i}: $\tilde{u}_{THS,i,min} = 0\frac{rad}{s}$; $\tilde{u}_{THS,i,max} = 1,1\omega_{Mot,max} i_{Acs_i}$. Entsprechend wird für die Momente und weitere Größen der Inzidenzmatrix vorgegangen.

Aufgrund ihres Schwerpunkts in der vorliegenden Arbeit sollen verschiedene Varianten bei den **Makroschrittweiten** verglichen werden. So wird ein Verfahren zur Steuerung mit und ohne Zyklusvorsteuerung angewandt und einer vergleichenden Wahl fester Makroschritte gegenübergestellt. Die für die Steuerung Angewandte Parametrierung $\boldsymbol{p}$ des *MDPCosim* -Masters ist, mit einem Ausschnitt aus $[\boldsymbol{t}_Z\ \boldsymbol{\alpha}_Z]$, in Anhang C dargestellt.

Es wird jeweils ein NEFZ, vgl. Abbildung 5.1, mit Kaltstart bei $25°C$ simuliert. Dafür werden die verschiedenen CS-Varianten mit dem Nicht-partitionierten Modell in monolithischer Simulation verglichen. Simuliert wird hierzu auf demselben System mit zwei Kernen unter gleichen Randbedingungen, was auch die Einstellungen der lokalen Solver von Dymola betrifft. Es kommt jeweils der DASSL-Solver ($Atol = Rtol = 10^{-4}$) mit den gleichen Speicherausgabeeinstellungen zum Einsatz. Für die Varianten mit Makroschrittweitensteuerung wird das

Verfahren konstanter Änderungsraten, vgl. Abb. 4.28, mit $ATOL = 10^{-1}$ & $RTOL = 10^{-2}$ verwendet. In den Abbildungen 5.10 und 5.11 werden schließlich die Ergebnisse der CS *ohne* (O) sowie *mit* Vorsteuerung (V) entsprechend Variante „schnell“ aus Tabelle 4.8 mit einer dem Ergebnis der Vorsteuerung entsprechenden festen Schrittweite $H_{fix} = H_{V,(m)}$ verglichen. $H_{fix} = 0.663s$ entspricht hier einem $H^*_{fix} = 5,62 \cdot 10^{-4}$.

Ergebnisse

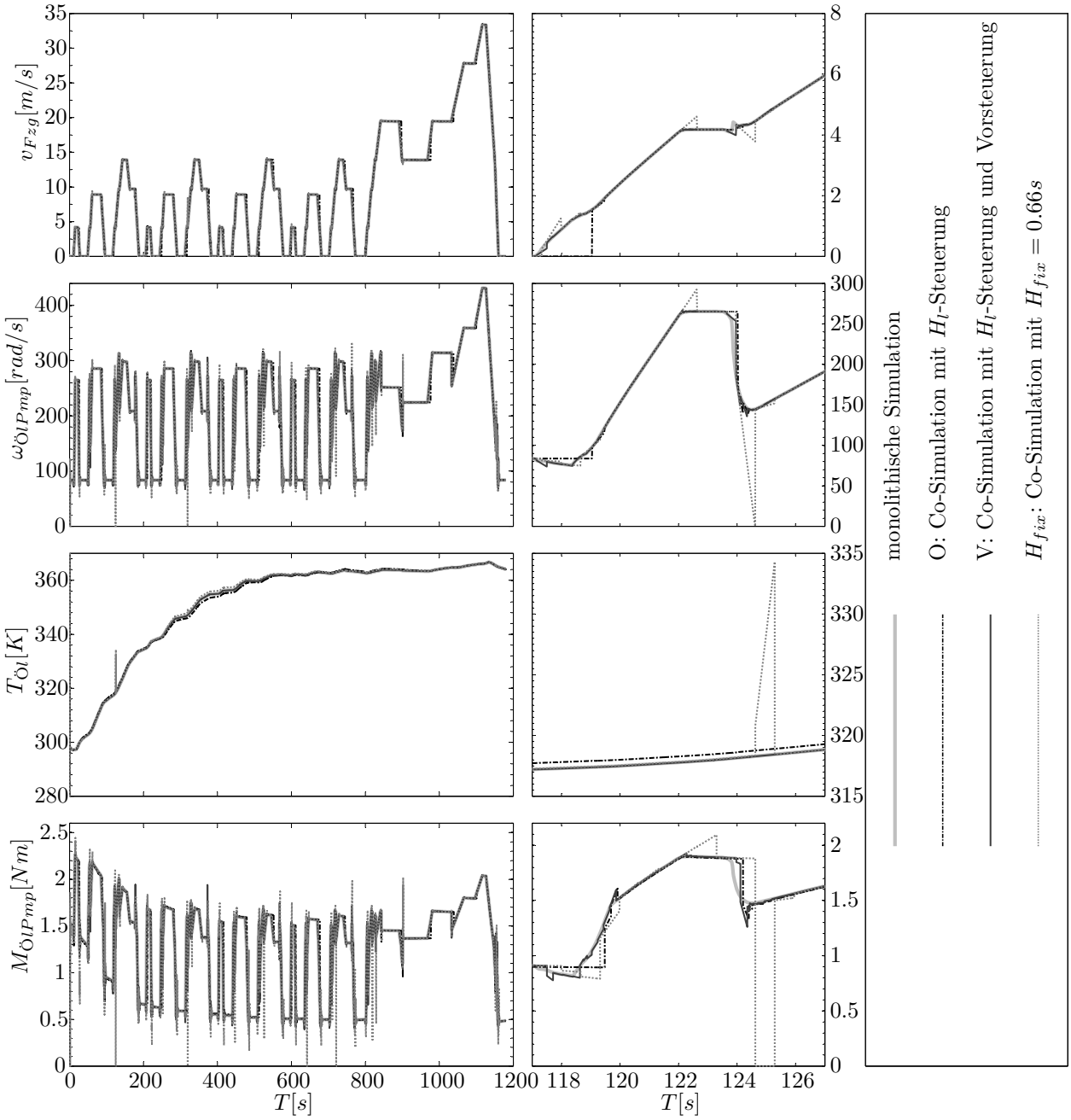


Abbildung 5.10.: Vergleich der physikalischen Ergebnisse der Zyklussimulation des Gesamtfahrzeugs in Co-Simulation mit dem Ergebnis in monolithischer Simulation

Abbildung 5.10 zeigt die Ergebnistrajektorien aus den drei Varianten der CS, aufgetragen über denen der monolithischen Simulation für vier Koppelgrößen.

Links ist jeweils der gesamte Zyklus, rechts eine Beschleunigungsphase vergrößert dargestellt. Es sind bei allen Größen jeweils der approximierte Eingang $\tilde{u}_{k_y}(t_j)$ bzw. die monolithische Entsprechung $\hat{y}(\hat{t})$ aufgetragen. Von oben nach unten sind dies: die Fahrzeuggeschwindigkeit $v_{Fzg} = \dot{x}_{ist}(:= \tilde{u}_{THS,4})$, die Drehzahl des Ölpumpentriebs $\omega_{\ddot{O}lPmp}(:= \tilde{u}_{THS,2})$ als Potentialgröße ans THS, die Öltemperatur $T_{\ddot{O}l}(:= \tilde{u}_{RFZ,3})$, sowie das von der Ölpumpe aufgenommene Moment $M_{\ddot{O}lPmp}(:= \tilde{u}_{RFZ,6})$ als Flussgröße ans RFZ.

Entsprechend der linken Ansicht liegen die Größen der CS insgesamt gut über denen der monolithischen Simulation. Es sind jedoch rechts Unterschiede in den Varianten und leichte Abweichungen erkennbar: So ist erwartungsgemäß des Ergebnis von V besser als O, welches besser als das von H_{fix} ausfällt. Ein Nachteil von O, die fehlende Eventerkennung, wird um $T = 118s$ erkennbar, wo nach einem großen $H_{l,O}$ im Stillstand das Anfahren verpasst wird. Hier liegt der Zeitpunkt $T_{l,fix}$ zufällig günstig, wobei sich beim Schaltvorgang später die zu großen H_{fix} als nachteilig erweisen: so wird der Drehzahleingang im THS fast zu 0 extrapoliert, was danach zu einem Fehler in der Temperatur und dem Moment führt.

Abbildung 5.11 vergleicht die Schrittweite[5] H_{fix} mit denen der gesteuerten Varianten und die drei entsprechenden relativen lokalen Fehler $\tilde{\tau}^*(T_l)$, nach der Definition auf Seite 105, sowie die normierten mittleren relativen globalen Fehler $\hat{\tilde{\tau}}^*_{(m)}(T_l) = \left\|\hat{\tilde{\boldsymbol{\tau}}}^*_{(m)}(T_l)\right\|_h$. Links ist wieder der gesamte Zyklus, rechts vergrößert eine Verzögerungsphase dargestellt. Man kann erkennen, dass durch den Kaltstart und die Aufwärmphase in den ersten 500 s die Schrittweiten etwas geringer und die lokalen Fehler etwas größer ausfallen. Weiterhin zeigt sich, dass der Bereich $H_{l,min} \dots H_{l,max}$ ohne Vorsteuerung (O) enger ausfällt als mit (V), wohingegen es sich bei $\tilde{\tau}^*(T_l)$ umgekehrt verhält. So kommt die Variante O der Makroschrittweitensteuerung im Mittel auf $H_{O,(m)} = 0,843s$ bei einem Gesamtfehler $\tau = \hat{\tilde{\tau}}^*_{(m)}(T_M) = 6,00\%$. Bei V sind es entsprechend $H_{V,(m)} = 0,663s$ und $\tau = 1,20\%$. Bei O ergibt sich dann mit $S_{CS} = 3,60$ die Effizienz $CSI = 0,43$ (im Vgl. hat $H_{O,fix} = 0,843$: $CSI = 0,35$). Bei V entspricht dies mit $S_{CS} = 2,95$ der Effizienz $CSI = 1,62$ (im Vgl. hat $H_{V,fix} = 0,663s$ auch $CSI = 0,35$).

Betrachtet man statt der Koppelgrößen die Energieflüsse, davon insbesondere den Kraftstoffverbrauch oder den dazu äquivalenten CO_2-Ausstoß, so fällt die Abweichung von der monolithischen Simulation noch viel geringer aus: bei allen Varianten liegt $\hat{\tilde{\tau}}^*_{CO_2,(m)}(T_M) < 0,1\%$. Auch die Werte von $S_{CS} > 1,5$ erfüllen mehr, als gefordert war. Somit kann das erweiterte Verfahren zur Co-Simulation in seiner Umsetzung gemäß den Anforderungen aus 4.1 sehr gut für die Gesamtfahrzeugsimulation mit Multi-Physikmodellen eingesetzt werden.

[5] der letzte Wert fällt $H_{fix,M-1} < H_{fix}$ aus, um T_M zu treffen

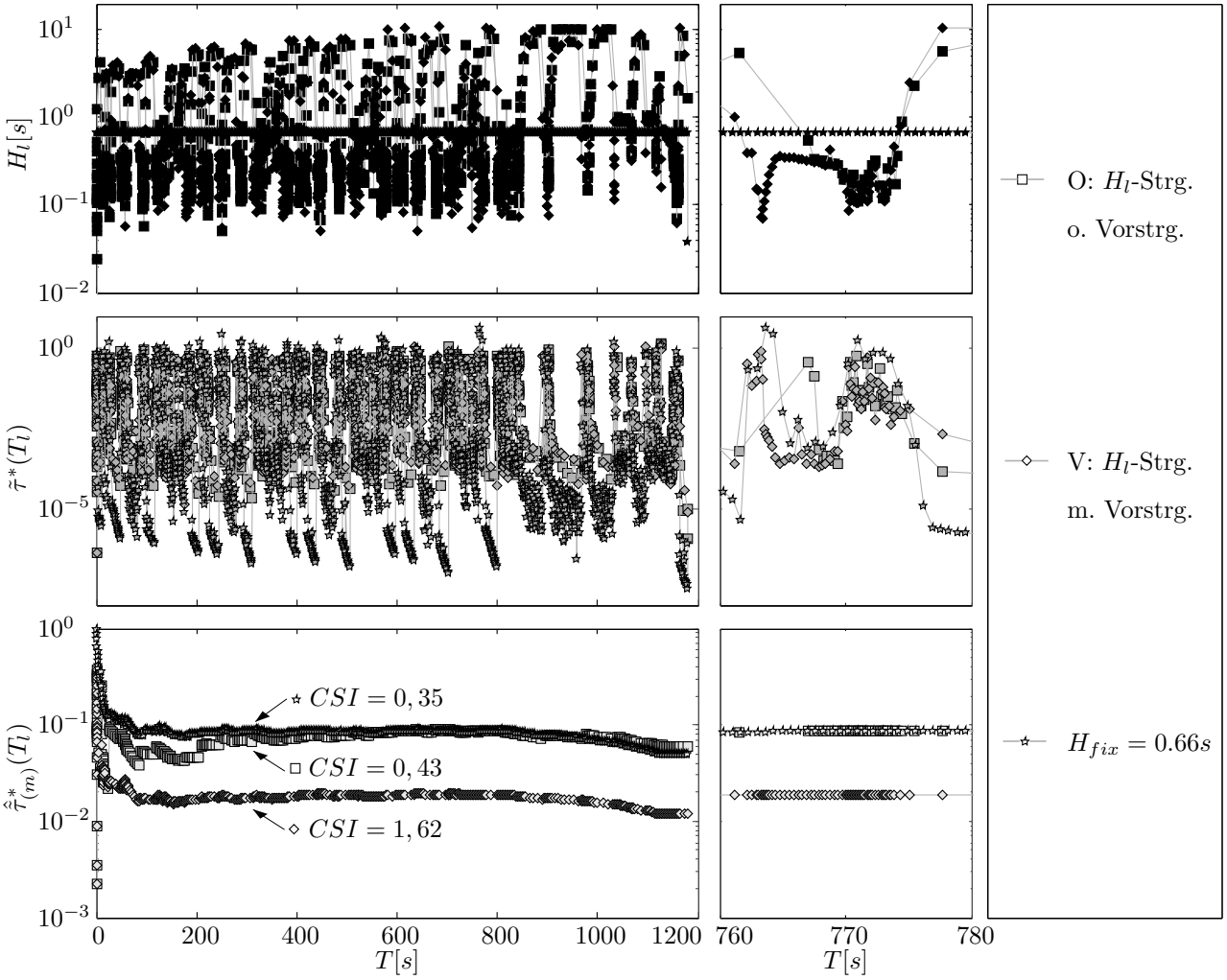


Abbildung 5.11.: Vergleich der numerischen Ergebnisse: Verlauf der Schrittweiten H_l sowie des lokalen Fehlers $\tilde{\tau}^*(T_l)$ und des mittleren globalen Fehlers $\hat{\tilde{\tau}}^*_{(m)}(T_l)$ beider Verfahren mit Vergleich bei fester Schrittweite H_{fix}

6. Zusammenfassung und Ausblick

Zusammenfassung

Um die vielseitigen Anforderungen in der heutigen Fahrzeugentwicklung erfüllen zu können, ist die Simulation des Gesamtfahrzeugs mit allen physikalischen Domänen -Mechanik, Fluiddynamik, Elektrik, und weitere- notwendig. In der vorliegenden Arbeit werden sowohl ein numerisches Verfahren zur Co-Simulation für Fahrzyklen entwickelt, das handhabbare Rechenzeiten für den Systementwickler ermöglicht, als auch eine Modellbibliothek erstellt.

Grundlage ist die Darstellung der mechanischen Teilsysteme und anderer Domänen in gleichungsbasierten Modellen. Diese wurden mit dem Programm Dymola in der objektorientierten Sprache Modelica implementiert und zu Gesamtfahrzeugmodellen in Vorwärtssimulation zusammengeschaltet. Das durch die geschlossene Modellierung entstandene Differentialalgebraische Gleichungssystem mit Diskontinuitäten ergab mehrfach höhere Rechenzeiten als die Summe der einzelnen Teilsysteme. Diese Erkenntnis führte zum Ansatz der verteilten Simulation durch Modellpartitionierung und Co-Simulation des Gesamtsystems.

Hierzu erfolgte zunächst eine Diskussion der Numerik der Co-Simulation, für die eine Kategorisierung in einzelne Teilaspekte entwickelt wurde: Synchronisierung, Kausalität, Approximation, Makroschrittweiten und Ereignisbehandlung. Es wurden quantitative Kriterien eingeführt, die die numerischen Verfahren nach Genauigkeit, Aufwand und in deren Kombination nach Effizienz bemessen und eine Basis für die Verfahrensentwicklung schaffen.

In Voruntersuchungen mit dem kommerziellen Kopplungsprogramm TISC und der Partitionierung eines detaillierten Modells konnte der Ansatz der Co-Simulation als Multi-Rate-Verfahren zur Aufwandsreduktion bestätigt werden. Um die nötige Genauigkeit der Integration zu erhalten, wurde, auf Anforderungen an die Co-Simulation mit kommerziellen Programmen aufbauend, ein Konzept für die Verfahrensentwicklung hergeleitet. Die Umsetzung erfolgte mit dem am Lehrstuhl entwickelten Kopplungsprogramm MDPCoSim, das um eine Modelica-Slave-Schnittstelle sowie numerische Master-Algorithmen erweitert wurde.

Mithilfe dieser Werkzeugkette wurden die vorgenannten numerischen Teilaspekte einzeln untersucht und das Verfahrenskonzept schrittweise umgesetzt. Dies erfolgte anhand der Kopplung eines Zwei-Massen-Schwingers sowie der Verallgemeinerung und Implementierung für die Zielanwendung Gesamtfahrzeugsimulation. Hierzu wurden Steifigkeit und Kopplung des ungedämpften Zwei-Massen-Schwingers und die Kausalität anhand der Potential-Fluss-

Kopplung untersucht. Es wurden die Bewegungsgleichungen für die möglichen Varianten der Partitionierung aufgestellt und eine Definition der Kopplungsrichtung vorgenommen. Im Ergebnis ist eine stabile parallele Kopplung möglich, wofür der Potentialsender mit differentiellem Zustand beim Teilmodell mit dem größeren Speicherglied am besten geeignet ist. Für die Approximation wurden die bei paralleler Co-Simulation notwendigen Extrapolationsverfahren sowie Glättungsverfahren untersucht. Es wurde die experimentelle Analyse einer Kombinationsmatrix beim Zwei-Massen-Schwinger durchgeführt, sowie die gleichen Verfahren am Fahrzeugmodell für unterschiedliche Makroschrittweiten untersucht. Bei Letzterem zeigten sich die Taylorpolynome erster Ordnung als am besten geeignet.

Für die Steuerung der Makroschrittweiten wurden neue Verfahren entwickelt: aus bekannten Verfahren für lokale Schrittweiten wurde ein Ansatz für explizite heuristische Verfahren zur allgemeinen Anwendung mit beliebigen Programmen hergeleitet. Zwei Methoden wurden zu diesem Ansatz entwickelt, das gradientenbasierte Verfahren und das Verfahren konstanter Änderungsraten sowie eine Vorsteuerung für Fahrzyklussimulation. Es erfolgten ausführliche numerische Analysen sowie die Applikation für die Zielanwendnung in Experimenten. Die Effizienzkriterien wiesen dem Verfahren konstanter Änderungsraten die beste Eignung nach.

Neben den im Rahmen dieser Arbeit entwickelten Verfahren zur Co-Simulation wurde die Modellbibliothek AdAM für Multi-Physik-Fahrzeugsimulation maßgeblich mitentwickelt. Objektorientierten Richtlinien folgend, ist sie modular aufgebaut und ermöglicht so die geschlossene sowie Co-Simulation innerhalb *einer* Bibliothek. Dazu enthält sie Modelle für Schnittstellen in den Domänen und zu MDPCoSim. Neben dem Antriebsstrang wurde in dieser Arbeit ein Modell des thermischen Systems mit Öl-und Kühlkreislauf eines Pkw aufgebaut. Die Bedatung erfolgte anhand von Fahrzeug- und Komponentenmessungen und wird in der Bibliothek getrennt vom Modell umgesetzt. Anhand der Partitionierung von Fahrzeug und thermischem System wird die Konfiguration des Co-Simulationsverfahrens in AdAM gezeigt. So kann schließlich die Funktion der Entwicklungen dieser Arbeit gemäß den Anforderungen an Genauigkeit und Rechenzeit in der Anwendung nachgewiesen werden.

Ausblick

Das vorgestellte Verfahren wurde für die explizite Kopplung von Simulationswerkzeugen und im Hinblick auf kürzere Simulationszeiten entwickelt. Mit den allgemein gehaltenen Schnittstellen ist es jedoch möglich, weitere Verfahren zu integrieren. So könnten, bei impliziter Kopplung, die klassischen Methoden zur Schrittweitensteuerung mit Fehlerschätzer implementiert werden. Eine Fortführung wäre die Kombination dieser Methoden mit einer Steuerung der Approximationsordnung über den Kontrollbus zum Master-Algorithmus, als eine Art Co-Simulations-Solver, auch mit Ereignissteuerung. Weiterhin besteht die Möglichkeit, MDPCoSim für den FMI-Standard zu erweitern und zusammen mit AdAM auch für Hardware-in-the-Loop-Anwendungen einzusetzen.

A. Koppel-/Inzidenzmatrizen

Potentialnachweis in Kapitel 4 (Beispiel auf den Seiten 57 ff.)

Der Gesamteingangsvektor $\bar{\boldsymbol{u}} = \bar{\boldsymbol{u}}_{ges} = \begin{pmatrix} \bar{\boldsymbol{u}}_{RFZ} \\ \bar{\boldsymbol{u}}_{ASM} \\ \bar{\boldsymbol{u}}_{ASN} \end{pmatrix} \in \mathbb{R}^{24}$ setzt sich aus den einzelnen Eingängen $\bar{\boldsymbol{u}}_{RFZ} \in \mathbb{R}^2$, $\bar{\boldsymbol{u}}_{ASM} \in \mathbb{R}^6$ und $\bar{\boldsymbol{u}}_{ASN} \in \mathbb{R}^{16}$ zusammen. Enstprechend gilt für $\bar{\boldsymbol{y}} = \bar{\boldsymbol{y}}_{ges} = \begin{pmatrix} \bar{\boldsymbol{y}}_{RFZ} \\ \bar{\boldsymbol{y}}_{ASM} \\ \bar{\boldsymbol{y}}_{ASN} \end{pmatrix} \in \mathbb{R}^{20}$ mit $\bar{\boldsymbol{y}}_{RFZ} \in \mathbb{R}^5$, $\bar{\boldsymbol{y}}_{ASM} \in \mathbb{R}^{14}$ und $\bar{\boldsymbol{y}}_{ASM} \in \mathbb{R}^1$ Darin enthalten ist auch die Abgaszusammensetzung aus neun Komponenten $\bar{\boldsymbol{x}}_{Gas} \in \mathbb{R}^9$. Es ergibt sich die Inzidenzmatrix $\boldsymbol{K}$ für $\bar{\boldsymbol{u}} = \boldsymbol{K}\bar{\boldsymbol{y}}$:

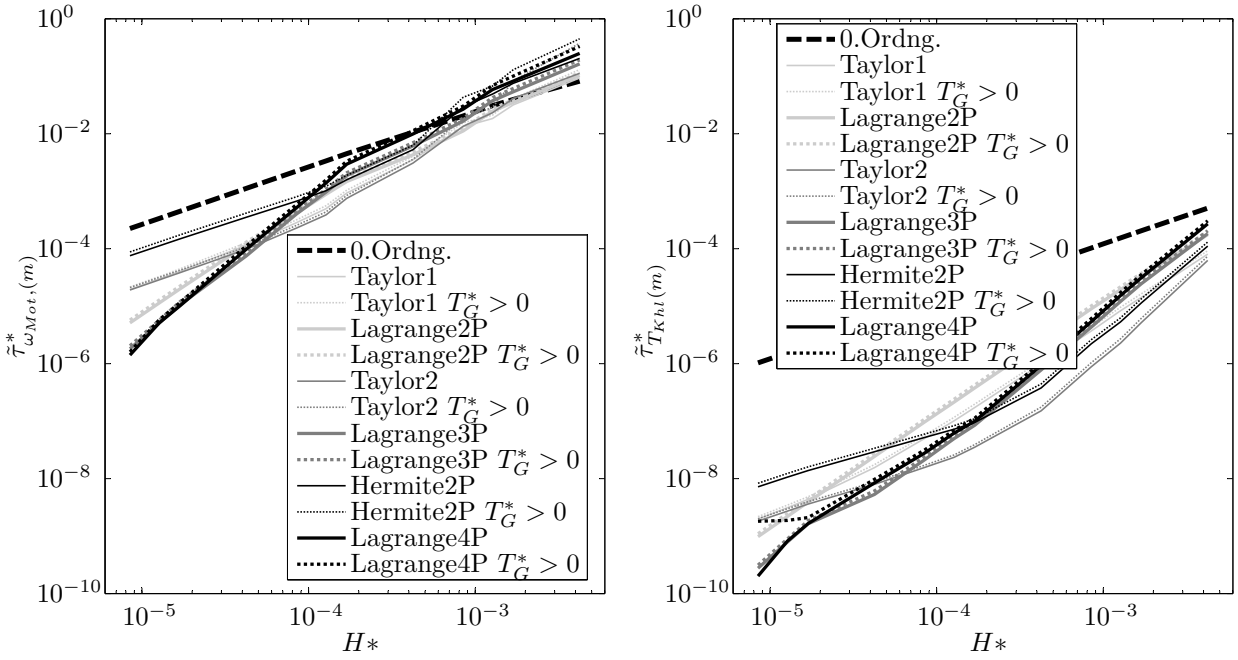
$$
\begin{pmatrix}
 & RFZ\begin{cases} \dot{Q}_{Abgas} \\ T_{AGR} \end{cases} \\
 & ASM\begin{cases} \dot{m}_{Kraftstoff} \\ \omega_{mot} \\ M_{mot} \\ \dot{x}_{fzg} \\ T_{K\ddot{u}hl} \\ \dot{Q}_{KatFlansch} \end{cases} \\
 & ASN\begin{cases} \dot{m}_{Kraftstoff} \\ \omega_{mot} \\ M_{mot} \\ \dot{x}_{fzg} \\ \dot{m}_{KatFlansch} \\ h_{KatFlansch} \\ T_{KatFlansch} \\ \bar{\boldsymbol{x}}_{Gas} \end{cases}
\end{pmatrix}
=
\underbrace{\begin{pmatrix}
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0
\end{pmatrix}}_{\boldsymbol{K}}
\begin{pmatrix}
 & RFZ\begin{cases} \dot{m}_{Kraftstoff} \\ \omega_{mot} \\ M_{mot} \\ \dot{x}_{fzg} \\ T_{K\ddot{u}hl} \end{cases} \\
 & ASM\begin{cases} \dot{Q}_{Abgas} \\ T_{AGR} \\ \dot{m}_{KatFlansch} \\ h_{KatFlansch} \\ T_{KatFlansch} \\ \bar{\boldsymbol{x}}_{Gas} \end{cases} \\
 & ASN\begin{cases} \dot{Q}_{KatFlansch} \end{cases}
\end{pmatrix}
$$

Beispiel aus Kapitel 4.5.2 und 5 (Seiten 93 ff. bzw. 117 ff.)

Der Gesamteingangsvektor $\bar{\boldsymbol{u}} = \bar{\boldsymbol{u}}_{ges} = \begin{pmatrix} \bar{\boldsymbol{u}}_{RFZ} \\ \bar{\boldsymbol{u}}_{THS} \end{pmatrix} \in \mathbb{R}^{14}$ setzt sich aus den einzelnen Eingängen $\bar{\boldsymbol{u}}_{RFZ} \in \mathbb{R}^7$ und $\bar{\boldsymbol{u}}_{THS} \in \mathbb{R}^7$ zusammen. Enstprechend gilt für $\bar{\boldsymbol{y}} = \bar{\boldsymbol{y}}_{ges} = \begin{pmatrix} \bar{\boldsymbol{y}}_{RFZ} \\ \bar{\boldsymbol{y}}_{THS} \end{pmatrix} \in \mathbb{R}^{14}$ mit $\bar{\boldsymbol{y}}_{RFZ} \in \mathbb{R}^7$ und $\bar{\boldsymbol{y}}_{THS} \in \mathbb{R}^7$. Es ergibt sich die Inzidenzmatrix $\boldsymbol{K}$ für $\bar{\boldsymbol{u}} = \boldsymbol{K}\bar{\boldsymbol{y}}$:

$$
\begin{pmatrix}
\overset{RFZ}{\left\{\begin{matrix} T_{mot} \\ T_{K\ddot{u}hl} \\ T_{\ddot{O}l} \\ \dot{Q}_{Abgas} \\ M_{K\ddot{u}hlPmp} \\ M_{\ddot{O}lPmp} \\ M_{ACKmpr} \end{matrix}\right.} \\
\overset{THS}{\left\{\begin{matrix} \omega_{K\ddot{u}hlPmp} \\ \omega_{\ddot{O}lPmp} \\ \omega_{ACKmpr} \\ \dot{x}_{fzg} \\ \dot{m}_{Kraftstoff} \\ \omega_{mot} \\ M_{mot} \end{matrix}\right.}
\end{pmatrix}
=
\underbrace{\begin{pmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{pmatrix}}_{\boldsymbol{K}}
\begin{pmatrix}
\overset{RFZ}{\left\{\begin{matrix} \omega_{K\ddot{u}hlPmp} \\ \omega_{\ddot{O}lPmp} \\ \omega_{ACKmpr} \\ \dot{x}_{fzg} \\ \dot{m}_{Kraftstoff} \\ \omega_{mot} \\ M_{mot} \end{matrix}\right.} \\
\overset{THS}{\left\{\begin{matrix} T_{mot} \\ T_{K\ddot{u}hl} \\ T_{\ddot{O}l} \\ \dot{Q}_{Abgas} \\ M_{K\ddot{u}hlPmp} \\ M_{\ddot{O}lPmp} \\ M_{ACKmpr} \end{matrix}\right.}
\end{pmatrix}
$$

B. Co-Simulationsergebnisse



(a) Fehler der Drehzahl $\tilde{\tau}^*_{\omega_{Mot},(m)}$

(b) Fehler der Kühlmitteltemperatur $\tilde{\tau}^*_{T_{Khl}(m)}$

Abbildung B.1.: lokaler Fehler $\tilde{\tau}^*_{j,(m)}$ der Approximationen mit ($T^*_G = 0,1$) und ohne ($T^*_G = 0$) Glättung in Abhängigkeit fester Makroschritte H^*

C. Konfiguration der Co-Simulation

`0 1180`	$\rightarrow T_0 \rightarrow T_M$
`0`	$\rightarrow$ `>0`: H_{fix} / `0`: H_l -Steuerung / `<0`: H_l aus *masterpar.varstep.dat*
`Caa`	$\rightarrow$ Auswahl H_l-Verfahren, hier: (a)lle Slaves, (a)lle Ausgänge
`1e-6`	$\rightarrow H_0$
`0.1 0.01`	$\rightarrow ATOL \rightarrow RTOL$

Abbildung C.1.: *masterpar.dat*-Datei zur Konfiguration des Masteralgorithmus und der Makroschrittweitensteuerung

`slave_rfz.bat slave_rfz.dat`	$\rightarrow$ Pfad Slave 0 $\rightarrow$ Pfad Schnittstellendefinition 0
`slave_ths.bat slave_ths.dat`	$\rightarrow$ Pfad Slave 1 $\rightarrow$ Pfad Schnittstellendefinition 1

Abbildung C.2.: *mastersys.dat*-Datei zur Definition der Slaveprogramme

`ConnectionFeedThroughLin`	$\rightarrow$ Auswahl Konnektorklasse für $\boldsymbol{f}_M$, hier: Durchgriff ohne $\boldsymbol{a}_M$
`0 1`	$\rightarrow$ Richtung: Slave 0 $\longmapsto$ Slave 1
`7`	$\rightarrow$ Vektorgröße des Durchgriffs, hier: $n_{u,THS}$
`0 0`	$\rightarrow y_{RFZ,1} \rightarrow u_{THS,1}$
`1 1`	$\rightarrow y_{RFZ,2} \rightarrow u_{THS,2}$
`...`	

Abbildung C.3.: Auszug aus einer *mastercon.dat*-Datei zur Auswahl der Konnektorklassen und Eingabe von $\boldsymbol{K}$

`8.000000e-2 1`	$\rightarrow T_{Z_1} \rightarrow M_{Z_1}$
`1.208000e+1 10`	$\rightarrow T_{Z_2} \rightarrow M_{Z_2}$
`1.230501e+1 1`	$\rightarrow T_{Z_3} \rightarrow M_{Z_3}$
`...`	

Abbildung C.4.: Auszug aus einer *masterpar.varstep.dat*-Datei zur Vorgabe variabler Makroschritte über $[\boldsymbol{T}_Z \boldsymbol{M}_Z]$

```
7 7                    → n_{u,j} → n_{y,j}
2 3                    → n^c_{u,j} → n^c_{y,j}
```

Abbildung C.5.: *slaveTHS.dat*-Datei zur Definition der Koppelgrößen- und Kontrollbusbandbreite

```
10                     → Anzahl der Co-Simulations-Durchläufe
parameterAutoVariLog   → Art der Variation, hier: ein Parameter
RTOL                   → zu variierender Parameter
1e-3 1e0               → Variationsintervall, hier: → RTOL_1 ... → RTOL_10
```

Abbildung C.6.: *batchpar.dat*-Datei zum Starten von automatisierten Batch-Co-Simulationsdurchläufen

```
1.100000e+1 1          → t_{Z_1} → α_{Z_1}
1.500000e+1 2          → t_{Z_2} → α_{Z_1}
2.300000e+1 1          → t_{Z_3} → α_{Z_3}
2.800000e+1 0          → t_{Z_4} → α_{Z_4}
...
```

Abbildung C.7.: Auszug aus einer *cycle.dat*-Datei zur Vorsteuerung bei Makroschrittweitensteuerungen mittels $[\boldsymbol{t}_Z\ \boldsymbol{\alpha}_Z]$

Literaturverzeichnis

[1] A. e. Abel. Hardware-in-the-Loop-Simulation mit detaillierten physikalischen Getriebemodellen. In: Virtual Powertrain Creation 2011 13. MTZ-Fachtagung, Unterschleißheim (2011) .

[2] D. Abel, A. Bollig. Rapid Control Prototyping. Springer Berlin Heidelberg New York (2006).

[3] ANSYS. ANSYS (2016). Http://www.ansys.com/.

[4] ANSYS. Ansys Fluent (2016). Http://www.ansys.com/Products/Fluids/ANSYS-Fluent.

[5] M. Arnold. Multi-Rate Time Integration for Large Scale Multibody System Models. In: IUTAM Bookseries, 1, Volume 1, IUTAM Symposium on Multiscale Problems in Multibody System Contacts, Stuttgart (2006) 1–10.

[6] M. Arnold. Stability of Sequential Modular Time Integration Methods for Coupled Multibody System Models. *J. Comput. Nonlinear Dynam* **5 Issue 3** (2010) 031003.

[7] M. Arnold, M. Günther. Preconditioned dynamic iteration for coupled differential-algebraic systems. *BIT Numerical Mathematics* **41** (2001) 1–25.

[8] M. Arnold, T. Schierz. Effizienz und Robustheit numerischer Kopplungsalgorithmen im MODELISAR Co-Simulation-Interface. In: Tagungsband ASIM-Treffen STS/GMMS 2009 und DASS 2009, Dresden (2009) .

[9] M. Arnold, et al. Simulation techniques for multidisciplinary problems in vehicle system dynamics. *Vehicle System Dynamics* **40** (2004) 17–36.

[10] M. Arnold, et al. FMI for Co-Simulation. In: 1st International Conference on Multiphysics Simulation - Advanced Methods for Industrial Engineering, Bonn (2010) .

[11] ATZonline. ICOS 2.0: Neuartige Kopplungsalgorithmen ermöglichen Fehlerkompensation (2012).

[12] G. Baumann, A. Piegsa, C. Liedecke. Der neue Fahrsimulator der Universität Stuttgart. In: Tagungsband ASIM-Treffen STS/GMMS 2010, Ulm (2010) .

[13] C. Beidl. Antriebsentwicklung 2020 Evolution oder Revolution? In: Virtual Powertrain Creation 2011 13. MTZ-Fachtagung, Unterschleißheim (2011) .

[14] M. Benedikt, H. Stippel, D. Watzing. An adaptive coupling methodology for fast time-domain distributed heterogeneous co-simulation. *Reliability and Robust Design in Automotive Engineering* **2010-01-0649** (2010) 203–212.

[15] M. Benedikt, D. Watzenig, J. Bernasch. Kopplungsmethodik für nicht-iterative Co-Simulation (2012).

[16] T. e. Blochwitz. An External Model Interface for Modelica. In: Proceedings of the 6th International Modelica Conference, Bielefeld, March 3-4 (2008) .

[17] M. Boumans, et al. Einsatz von Simulation in der Kalibrierung von Motorsteuergeräten. In: Virtual Powertrain Creation 2011 13. MTZ-Fachtagung, Unterschleißheim (2011) .

[18] H. Brückmann, et al. Model-based Development of a Dual-Clutch Transmission using Rapid Prototyping and SiL. In: Getriebe in Fahrzeugen 2009, Friedrichshafen (2009) .

[19] P. Breedveld. Simulation Techniques for Applied Dynamics, Kapitel Port-based modelling of multidomain physikal systems in terms of bond graphs, 141–190. Arnold, M. and Schielen, W. (2008) .

[20] J. Brembeck, M. Otter, D. Zimmer. Nonlinear Observers based on the Functional Mockup Interface with Applications to Electric Vehicles. In: Proceedings of the 8th International Modelica Conference, Dresden (2011) .

[21] K. Brenan, S. Campbell, L. Petzold. Numerical Solution of Initial-Value Problems in Differential-Algebraic Equations. SIAM, Philadelphia, PA (1996).

[22] M. Busch. Entwicklung einer SIMPACK-Modelica/Dymola-Schnittstelle. Diplomarbeit, Martin-Luther-Universitat Halle-Wittenberg (2007).

[23] M. Busch. Zur effizienten Kopplung von Simulationsprogrammen. Dissertation, Universität Halle (2012).

[24] M. Busch, B. Schweizer. Explizite und implizite Solver-Kopplung: Stabilitätsanalyse auf Basis eines 8-parametrigen Testmodells. In: Proceedings GAMM-Tagung 2010, Karlsruhe. (2010) .

[25] M. Busch, B. Schweizer. Numerical Stability and Accuracy of Different Co-Simulation Techniques: Analytical Investigations Based on a 2-DOF Test-Model. In: The 1st Joint International Conference on Multibody System Dynamics, Lappeenranta, Finland, Paper-ID 20, pp. 1-13, May 2527, 2010 (2010) .

[26] Cadence. Verilog-XL (2012). Http://www.cadence.com.

[27] F. Casella. Modelling, Simulation, and Control of a Geothermal Power Plant. Dissertation, Politecnico di Milano (1999).

[28] F. Casella. Exploiting Weak Dynamic Interactions in Modelica. In: Proceedings of the 4th International Modelica Conference, March 7-8, Hamburg (2005) pp. 97–102.

[29] F. E. Cellier, E. Kofman. Continuous System Simulation. Springer Science+Business Media, Inc. (2006).

[30] ChiasTek. CosiMate - Heterogeneous Co-simulation in Distributed Environments (2011). Http://www.chiastek.com/products/cosimate.html.

[31] C. Clauss, et al. Simulation mikrosystemtechnischer Aufgaben mit gekoppelten Simulatoren. In: 2. Chemnitzer Fachtagung Mikrosystemtechnik, Mikromechanik, und Mikroelektronik '95. Proceedings. (1995) .

[32] C. Clauss, et al. Simulatorkopplung mit FEMLAB. In: Proceedings of the COMSOL Multiphysics User's Conference 2005, Frankfurt (2005) .

[33] C. Clauss, et al. Master zur Simulatorkopplung via FMI. In: ASIM-Konferenz STS/GMMS 2012 Simulation technischer Systeme Grundlagen und Methoden in Modellbildung und Simulation, Wolfenbüttel (2012) .

[34] E. Commission. NEDC. Consolidated Directive 70/220/EEC (2006).

[35] DLR. The Powertrain Library (Version 2.1.0) (2011). Http://www.dlr.de/rm.

[36] S. Dronka. Die Simulation gekoppelter Mehrkörper- und Hydraulikmodelle mit Erweiterung für Echtzeitsimulation. Dissertation, TU Dresden (2004).

[37] Dymola. Dymola - multi-engineering modeling and simulation. Dassault Systèmes (2016). Http://www.3ds.com/de/products/catia/portfolio/dymola.

[38] H. Elmqvist, M. Otter, F. E. Cellier. Inline Integration: A new mixed symbolic/numeric approach for solving differential-algebraic equation systems. In: Proc. ESM'95, SCS European Simulation MultiConference, Prague, Czech Republic (1995) .

[39] H. Elmqvist, et al. Modelica for Embedded Systems. In: 7th international Modelica Conference, Como (2009) .

[40] C. Engstler, C. Lubich. Multirate extrapolation methods for differential equations with different time scales. *Computing* **58** (1997) 173–185.

[41] C. Engstler, C. Lubich. MUR8: a multirate extension of the eighth-order Dormand-Prince method. *Appl. Numer. Math. , special issue Time Integrators* **25** (1997) 185–192.

[42] A. Ennemoser, H. Schreier, H. Petutschnig. Optimierte Betriebsstrategie für Nebenaggregate im Lkw. *MTZ - Motortechnische Zeitschrift* **2012-03** (2012) 220–224.

[43] M. Enright, S. Kamel. Automatic Partitioning of Stiff Systems and Exploiting the Resulting Structure. *Transactions on Mathematical Software (TOMS)* **Volume 5 Issue 4** (1979) 374–385.

[44] W. EXTESSY AG. EXITE Handbuch, Version 1.4.4. EXTESSY AG, Wolfsburg (2005).

[45] C. Faure, et al. Methods for real-time simulation of Cyber-Physical Systems: application to automotive domain. In: Proc. of First IEEE Workshop on Design, Modeling and Evaluation of Cyber Physical Systems (CyPhy'11), Istanbul, Turkey, July 2011. (2011) .

[46] M. Filippi, H. Ubrich. Simulation der Dynamik von Steuertrieben und Maßnahmen zur Schwingungsreduktion. In: 7. VDI-Tagung Schwingungen in Antrieben 2011; Leonberg (2011) .

[47] Flowmaster. Flowmaster - Fluid thinking for systems engineers (2016). Https://www.mentor.com/products/mechanical/flowmaster/.

[48] FMI-Standard. FMI Specification 2.0 (2014).

[49] M. Friedrich. Parallel Co-Simulation for Mechatronic Systems. Dissertation, TU München (2011).

[50] M. Friedrich, M. Schneider, H. Ulbrich. A Parallel Co-Simulation for Mechatronic Systems. In: The 1st Joint International Conference on Multibody System Dynamics, Lappeenranta, Finland (2010) .

[51] M. Friedrich, H. Ulbrich. A Parallel Co-Simulation for Multibody Systems. In: SEECCM 2009 2nd South-East European Conference on Computational Mechanics, Rhodos, Greece (2009) .

[52] C. W. Gear, B. Leimkuhler, G. K. Gupta. Automatic integration of Euler-Lagrange equations with constraints. *Journal of Computational and Applied Mathematics, Netherlands* **12&13** (1985) 77–90.

[53] M. Geimer, T. Krüger, P. Linsel. Co-Simulation, gekoppelte Simulation oder Simulatorkopplung? Ein Versuch der Begriffsvereinheitlichung. *O+P Zeitschrift für Fliudtechnik - Aktorik Steuerelektronik und Sensorik* **11-12/2006** (2006) 572–576.

[54] M. Genger, M. Weinrich. Optimiertes Thermomanagement: Dokumentation des thermischen Fahrzeugmodells TheFaMoS. *Technischer Bericht*, FVV-Vorhaben Nr. 068540 (2006).

[55] M. Gipser. Systemdynamik und Simulation. ISBN-10: 3519027437. Teubner, Stuttgar (1999).

[56] G. R. Gómez. Absolute Stability Analysis of Semi-Implicit Multirate Linear Multistep Methods. Dissertation, Instituto Nacional de Astrofísica, Óptica y Electrónica, Mexiko (2002).

[57] F. Günther. Modelle des Drehmomentwandlers für die Systemsimulation. Diplomarbeit, unveröffentlicht, Universität Stuttgart (2009).

[58] F. Günther, G. Mallebrein, H. Ulbrich. A Modular Technique for Automotive System Simulation. In: Proceedings of the 9th International Modelica Conference, München, Sept.03-05 (2012) .

[59] K.-H. Grote, J. Feldhusen. Dubbel - Taschenbuch fi¿$\frac{1}{2}$r den Maschinenbau. Springer Verlag, 22 Auflage (2007).

[60] B. Gu, H. Asada. Co-Simulation of Algebraically Coupled Dynamic Subsystems Without Disclosure of Proprietary Subsystem Models. *Journal of Dynamic Systems, Measurement, and Contro* **Vol126** (2004) 1–13.

[61] E. Hairer, S. P. Norsett, G. Wanner. Solving Ordinary Differential Equations I - Nonstiff Problems. Springer-Verlag, 2nd revised edition Auflage (2000).

[62] E. Hairer, G. Wanner. Solving Ordinary Differential Equations II - Stiff and Differential-Algebraic Problems. Springer-Verlag, 2nd revised edition Auflage (1996).

[63] E. Hessel, J. Haase. Aspekte der Transformationen zwischen VHDL-AMS- und Modelica-Modellen. In: ASIM-Treffen STS/GMMS 2009, Dresden (2009) .

[64] C. von Heyden, et al. Bewertung von Antriebssträngen mit abgasturboaufgeladenen Ottomotoren Eine Simulation des Gesamtsystems. In: Virtual Powertrain Creation 2011 13. MTZ-Fachtagung, Unterschleißheim (2011) .

[65] G. Hippmann, M. Arnold, M. Schittenhelm. Efficient Simulation of bush and roller chain drives. In: Multibody Dynamics 2005, Eccomas Thematic Conference, Madrid (2005) .

[66] D. Hülsebusch, et al. Multidisziplinäre Simulation von Fahrzeugen. *ATZ* **10 2009** (2009) 772 – 779.

[67] M. Hommel. Parallelisierte Simulationsprozesse für virtuelles Prototyping in der Automobilindustrie. Dissertation, TU Braunschweig (2006).

[68] R. Howe. Real-Time Multirate asynchronous Simulation with single and multiple Processors. *Proceedings of the SPIE - The international society for optical engineering* **Vol 3369** (1998) 331–342.

[69] R. Howe. Accuracy and Stability Tradeoffs in Multirate-Simulation. *Proceedings of the SPIE - The international society for optical engineering* **Vol. 4367** (2001) 113–126.

[70] ITI. SimulationX (2016). Https://www.simulationx.de/.

[71] Z. Jackiewicz, M. Kwapisz. Convergence of waveform relaxation methods for differential-algebraic systems. *SIAM Journal on Numerical Analysis* **33 / 6** (1996) 2303–2317.

[72] K. Jackson. A survey of parallel numerical methods for IVPs for ODEs. *IEEE Transactions on Magnetics* **27** (1991) 3792–3797.

[73] C. Janz, B. Hageroth. Antriebsstrangmodelle im Fahrzeugentwicklungsprozess. In: 7. VDI-Tagung Schwingungen in Antrieben 2011, Leonberg (2011) .

[74] B. Johansson. Modelica in a Distributed Environment Using Transmission Line Modelling. In: Proceedings Modelica Workshop 2000 (2000) .

[75] C. Kaiser, et al. Untersuchungen von Regelstrategien für die Omnibusklimatisierung mit Hilfe einer Gesamtfahrzeugsimulation. In: ASIM-Konferenz STS/GMMS 2012 Simulation technischer Systeme Grundlagen und Methoden in Modellbildung und Simulation, Wolfenbüttel (2012) .

[76] D. Kanth. Zur steifigkeits- und kopplungsbasierten Partitionierung mechatronischer Systeme. Dissertation, Universität Stuttgart (2010).

[77] D. Kanth, M. Lehner. Coupling of modular subsystems in a mechatronic simulatoin environment. In: Proceedings of the EUROMECH Nonlinear Dynamics Conference, 2005 (2005) .

[78] R. Kübler. Numerische Stabilität und Genauigkeit bei modularer Integration mit Anwendung auf die Hardware-in-the-Loop-Simulatuion. *Z. Angew. Math. Mech* **78 S2** (1998) 569–570.

[79] R. Kübler. Modulare Modellierung und Simulation mechatronischer Systeme. Dissertation, Universität Stuttgart (2000).

[80] R. Kübler, W. Schiehlen. Two Methods of Simulator Coupling. *Mathematical and Computer Modelling of Dynamical Systems: Methods, Tools and Applications in Engineering and Related Sciences* **Volume 6, Issue 2** (2000) 93–113.

[81] R. Kübler, W. Schielen. Modular Simulation in Multibody System Dynamics. *Multibody System Dynamics* **4** (2000) 107–127.

[82] S. Knorr. Multirate-Verfahren in der Co-Simulation gekoppelter dynamischer Systeme mit Anwendung in der Fahrzeugdynamik. Diplomarbeit, Universität Ulm (2002).

[83] R. Kossel. Hybride Simulation thermischer Systeme am Beispiel eines Reisebusses. Dissertation, TU Braunschweig (2011).

[84] R. Kossel, S. Försteling, W. Tegethoff. Einsatz hybrider Simulationstechnik für die Bewertung mobiler Heiz- und Kühlkonzepte. *Haus der Technik Fachbuch* **93** (2008) 150–162.

[85] R. Kossel, et al. Simulation of Complex Systems using Modelica and Tool Coupling. In: Proceedings of the 5th International Modelica Conference 2006, Vienna, Austria (2006) 485–490.

[86] G. Kron. Diakoptics: The Piecewise Solution of LargeScale Systems. *Macdonald Publishing, London, United Kingdom* **2** (1963) –.

[87] P. D. Lax, R. D. Richtmyer. Survey of the stability of linear finite difference equations. *Communications on Pure and Applied Mathematics* **Vol. 9, Issue 2** (1956) 267293.

[88] E. Maten, M. Günther. Multirate Time Integration for Multiscaled Systems. In: Minisymposium of ECMI Special Interest Group on Scientific Computing in Electronics Industrie (2008) .

[89] S. E. Mattsson, G. Soderling. Index Reduction in Differential-Algebraic Equations Using Dummy Derivatives. *SIAM Journal of Computation* **Vol. 14, No 3** (1993) pp. 667692.

[90] H. Mertens. Aussagegüte und Zeitaufwand Kriterien zur Auswahl von Berechnungsmethoden im Konstruktionsprozess. *Technischer Bericht Nr. 1442*, VDI-Berichte Nr. 1442, Düsseldorf (1998).

[91] R. Mirkheshti. Zur Entwicklung von Berechnungsmethoden und deren Integration in den Produktentwicklungsprozess. Dissertation, TU Berlin (2006).

[92] S. Misera. Simulation von Fehlern in digitalen Schaltungen mit SystemC. Dissertation, TU Cottbus (2007).

[93] Modelica. Modelica - A Unified Object-Oriented Language for Physical Systems Modeling Language Specification, Version 3.3 (2016). Http://www.modelica.org.

[94] Modelisar. Functional Mock-up Interface for Co-Simulation 1.0 (2010).

[95] Modelisar. Functional Mock-up Interface for Model Exchange 1.0 (2010).

[96] L. Morawietz, et al. Modellierung des Antriebsstrangs und des elektrischen Energiebordnetzes eines Fahrzeugs für die Echtzeitsimulation unter Verwendung von Modelica. In: ASIM: Simulations- und Testmethoden für Software in Fahrzeugsystemen, TU Berlin (2005) .

[97] P. J. Mosterman. An Overview of Hybrid Simulation Phenomena and Their Support by Simulation Packages. *Lecture notes in Computer Science,* **Vol1, Nr2** (1999) 165–177.

[98] R. Mustafa, F. Küçükay. Effizientes Thermomanagement Einsatz der Gesamtfahrzeugsimulation zur Quantifizierung von Kraftstoffverbrauchseinsparungen im PKW. In: ASIM-Konferenz STS/GMMS 2012 Simulation technischer Systeme Grundlagen und Methoden in Modellbildung und Simulation, Wolfenbüttel (2012) .

[99] M. Nägeli, D. Lichtenthäler. Eine virtuelle Simulationsumgebung als Prüfplatz für Hochvoltbatterie-Steuergeräte. In: ASIM-Konferenz STS/GMMS 2012 Simulation technischer Systeme Grundlagen und Methoden in Modellbildung und Simulation, Wolfenbüttel, ISBN 978-3-901608-39-1 (2012) .

[100] R. Nikoukhah. Hybrid Dynamics in Modelica: Should all Events be Considered Synchronous. In: P. Fritzson, F. Cellier, C. Nytsch-Geusen (Herausgeber), Proceedings of the 1st International Workshop on Equation-Based Object-Oriented Languages and Tools (2007) .

[101] K. e. a. Nyström. Parallelization in Modelica. In: Proceedings of the 4th International Modelica Conference, Hamburg (2005) 169–172.

[102] M. Otter. Elektrische Antriebe - Regelung von Antriebssystemen, Kapitel Objektorientierte Modellierung von Antriebssystemen, 1049–1165. Springer-Verlag Berlin Heidelberg, 3. bearbeitete auflage Auflage (2009) .

[103] M. Otter, et al. Das Functional Mockup Interface zum Austausch Dynamischer Modelle. In: ASIM-Workshop, Ulm (2010) .

[104] C. Pantelides. The consistent initialization of differential-algebraic systems. *SIAM Journal of Scientific and Statistical Computing,* **Vol. 9, No 2** (1988) pp. 213–231.

[105] K. Petridis, A. Klein, M. Beitelschmidt. Asynchronous method for the coupled simulation of mechatronic systems. In: Proceedings in Applied Mathematics and Mechanics, Bremen (2008) .

[106] L. Petzold. DASSL: A Differential/Algebraic System Solver. *Technischer Bericht*, Sandia National Laboratories Livermore, CA (1982).

[107] A. Pfeiffer. Numerische Sensitivitätsanalyse unstetiger multidisziplinärer Modelle mit Anwendungen in der gradientenbasierten Optimierung. Dissertation, Universität Halle (2008).

[108] G. Puntigam, et al. Transient Co-Simulation of Comprehensive Vehicle Models by Time Dependent Coupling. In: SAE 2006 Worldcongress, Thermal systems & Climate control (2006) .

[109] QTronic. Silver 2.4 (2016). Http://www.qtronic.de/de/silver.html.

[110] A. Rückgauer. Modulare Simulation mechatronischer Systeme mit Anwendung in der Fahrzeugdynamik. Dissertation, Universität Stuttgart (1999).

[111] P. Rentrop. Partitioned Runge-Kutta methods with stiffness detection and stepsize control. *Numerische Mathematik* **Volume 47, Number 4** (1985) 545–564.

[112] P. Rumbolz, G. Baumann, H.-C. Reuss. Messung der fahrzeuginternen Leistungsflüsse im Realverkehr. *ATZ* **05 2011** (2011) 416 – 421.

[113] A. Schiela, H. Olsson. Mixed-Mode Integration for Real-Time Simulation. In: Modelica Workshop 2000 Proceedings (2000) 69–75.

[114] T. Schierz, M. Arnold. Advanced numerical methods for co-simulation algorithms in vehicle system dynamics. In: Proc. of 1st Fraunhofer Conference on Multiphysics Simulation, Bonn (Germany) (2010) .

[115] T. Schierz, M. Arnold. Stabilized overlapping modular time integration of coupled differential-algebraic equations. *Applied Numerical Mathematics* **62 No. 10** (2012) 1491 – 1502.

[116] T. Schierz, et al. Co-simulation with communication step size control in an FMI compatible master algorithm. In: Proceedings of the 9th International Modelica Conference, München, Sept.03-05 (2012) .

[117] M. Schneider. Interdisziplinäre Simulation am Beispiel hydraulischer Nockenwellenversteller. Dissertation, TU München (2013).

[118] M. e. a. Schneider. Wechselwirkungen hydraulischer Nockenwelenversteller mit Ventil- und Steuertrieb. In: 6. FachtagungSchwingungen in Antrieben 2009, Leonberg (2009) 133–142.

[119] C. Scholz. Zur Simulatorkopplung für mechatronische Systeme. Dissertation, Universität Stuttgart (2004).

[120] C. Scholz, W. Schielen. Step size control of simulator coupling for multibody systems. In: Co-Simulation für Mechatronic Systems, Stuttgart; Materials of a Workshop (2001) .

[121] H. R. Schwarz, N. Köckler. Numerische Mathemathik. ISBN 978-3-8348-0683-3. Vieweg + Teubner, Wiesbaden (2009).

[122] P. Schwarz. Simulation of systems with dynamically varying model structure. *Mathematics and Computers in Simulation 79* **79** (2008) 850863.

[123] L. F. Shampine, A. Witt. A simple step size selection algorithm for ODE codes. *Journal of . Comput. Appl. Math* **58(3)** (1995) 345–354.

[124] Y. Shi. Reevaluating Amdahl's Law and Gustafson's Law. *Technischer Bericht*, Computer and Information Sciences Department, Temple University, Philadelphia (1996).

[125] D. Simic, M. Noll, F. Pirker. Innovative Entwicklungstools zur effizienten Auslegung von Hybrid- und Elektrofahrzeugen. *Elektrotechnik & Informationstechnik* **11 2008** (2008) 377381.

[126] Simpack. SIMPACK. Dassault Systèmes (2016). Http://www.simpack.com.

[127] B. Stegmann, et al. Prognose Thermomanagement Phase I und Phase II. *Technischer Bericht*, FVV, Vorhaben Nr. 939 und 1004 (2012).

[128] I. Stotz, et al. Prognose Thermomanagement. In: Virtual Powertrain Creation 2011 13. MTZ-Fachtagung, Unterschleißheim (2011) .

[129] M. Striebel, M. Günther. A charge oriented mixed multirate method for a special class of index-1 network equations in chip design. *Appl. Numer. Maths* **Vol. 53** (2005) 489– 507.

[130] Y. Sun, S. Vogel, H. Steuer. Combining Advantages of Specialized Simulation Tools and Modelica Models using Functional Mock-up Interface (FMI). In: Proceedings of the 8th International Modelica Conference, Dresden, Germany (2011) .

[131] Synopsys. Saber (2012). Http://www.synopsys.com/Systems/Saber/Pages/default.aspx.

[132] R. Tarjan. Depth First Search and Linear Graph Algorithms. *SIAM Journal of Computation* **1** (1972) pp. 146–160.

[133] G. Technologies. GT-POWER - Engine Simulation Software (2016). Http://www.gtisoft.com.

[134] W. Tegethoff, et al. Co-Simulation und Sprach-Standardisierung am Beispiel des Wärmemanagements. *Wärmemanagement des Kraftfahrzeugs V, Haus der Technik, expert verlag, Renningen* **68** (2006) 231–242.

[135] S. Terwen. Vorausschauende Längsregelung schwerer Lastkraftwagen. Dissertation, Universität Karlsruhe (TH) (2009).

[136] I. The MathWorks. Simulink - Simulation and Model-Based Design (2012). Http://www.mathworks.com.

[137] TISC. TISC Suite Software zur Kopplung mehrerer Simulationswerkzeuge. (2016). Http://www.tlk-thermo.com/index.php/de/softwareprodukte/tisc-suite.

[138] M. Trcka, M. Wetter, J. Jan Hensen. Comparison of co-simulation approaches for building and HVAC-R system simulation. In: Proceedings of the10th IBPSA Building Simulation Conference, Beijing, China (2007) pp. 1418 1425.

[139] H. Ulbrich. Some Selected Research Activities in Mechatronic Applications - Case Studies. In: Proc. 2nd International Conference on Engineering Mechanics, Structures, Engineering Geology ,Rodos Island, Greece (2009) .

[140] UNECE. WLTC v4 (2012). Http://www.unece.org/.

[141] J. Ungethüm, et al. Simulation von alternativen Fahrzeugantrieben in Modelica. In: Tagungsband ASIM-Treffen STS/GMMS 2010, Ulm (2010) .

[142] VDI. VDI3633 - 1: Simulation von Logistik-, Materialfluß- und Produktionssystemen (1993).

[143] VDI. VDI2206: Entwicklungsmethodik für mechatronische Systeme (2004).

[144] A. Verhoeven. Automatic control for adaptive time stepping in electrical circuit simulation. Diplomarbeit, TU Eindhoven, Dept. of Math. and Comp. Sience (2003).

[145] A. Verhoeven, et al. BDF compound-fast multirate transient analysis with adaptive stepsize control. *Journal of Numerical Analysis, Industrial and Applied Mathematics* **vol. 3, no. 3-4,** (2008) 275–297.

[146] L. Völker. Untersuchung des Kommunikationsintervalls bei der gekoppelten Simulation. Dissertation, Karlsruher Institut für Technik, KIT (2010).

[147] D. Wells. Multirate linear multistep methods for the solution of systems of ordinary differential equations. *Technischer Bericht*, Tech. Report Rept. No.UIUCDCS-R-82-1093, Dept. Computer Sci., Illinois, (1982).

[148] W. Wickord. Zur Anwendung probabilistischer Methoden in den frühen Phasen des Systementwurfs. Dissertation, Universität Paderborn (2003).

[149] S. Wünsche, et al. Electro-Thermal Circuit Simulation Using Simulator Coupling. *IEEE Transactions on very large scale integration (vlsi) systems* **VOL. 5 NO. 3** (1997) 277–282.

[150] L. Wunderlich. Analysis and Numerical Solution of Structured and Switched Differential-Algebraic Systems. Dissertation, Technischen Universität Berlin (2008).

Lebenslauf

Dipl.-Ing. Felix Günther

Geburtsdatum/-ort	16. April 1983, Stuttgart
Staatsangehörigkeit	deutsch
Familienstand	verheiratet

Schulbildung und Zivildienst

08/1989 – 07/1993	Osterholzschule (Grundschule), Ludwigsburg
09/1993 – 06/2002	Mörike-Gymnasium Ludwigsburg
09/2002 – 06/2003	Zivildienst beim Deutschen Roten Kreuz, Ludwigsburg

Akademische Bildung

10/2003 – 03/2009	Studium des Maschinenbaus (Studiengang Maschinenwesen) Universität Stuttgart Hauptfächer: Angewandte Thermodynamik, Konstruktionstechnik
02/2008 – 08/2008	Auslandspraktikum bei Robert Bosch France SAS Drancy bei Paris, Frankreich

Promotion und Berufstätigkeit

09/2009 – 08/2012	Doktorand am Lehrstuhl für Angewandte Mechanik der Technischen Universität München als Mitarbeiter im Doktorandenprogramm der Robert Bosch GmbH, Schwieberdingen Forschung und Vorausentwicklung, Fahrzeugantriebssysteme
seit 10/2012	Entwicklungsingenieur bei der Robert Bosch GmbH, Stuttgart Systementwicklung Diesel Antriebsstrang

Stuttgart, 16. März 2017