INAUGURAL-DISSERTATION

zur

Erlangung der Doktorwürde

der

Naturwissenschaftlich-Mathematischen Gesamtfakultät

der

RUPRECHT-KARLS-UNIVERSITÄT
HEIDELBERG

vorgelegt von

Dipl.-Math. Thomas Loderer

aus Augsburg

Tag der mündlichen Prüfung:

Echtzeitfähige Simulation steifer Modelle für Anwendungen in Fahrzeug-Steuergeräten

Betreuer: Professor Dr. Vincent Heuveline

Zweitbetreuer: Professor Dr.-Ing. Carsten Proppe

Zusammenfassung

In vorliegender Arbeit wird die Echtzeitsimulation steifer Modelle für den Einsatz auf Fahrzeug-Steuergeräten untersucht. Am Beispiel eines steifen Modells einer Rohrströmung wird untersucht, wie durch den Einsatz geeigneter numerischer Verfahren eine echtzeitfähige Simulation des Modells auf einem Fahrzeug-Steuergerät erreicht werden kann. Im Rahmen der Lösung der zu Grunde liegenden Differentialgleichung mittels eines linear-impliziten Euler-Verfahrens werden für die Berechnung einer Approximation der Jacobi-Matrix anstelle eines Differenziationsverfahrens verschiedene auf das Rohrmodell zugeschnittene Arten der Approximation der Einträge eingesetzt, was eine Verallgemeinerung der von Schiela und Bornemann beschriebenen *Sparsing* Methode darstellt. Dies führt zu einer signifikanten Reduktion der für die Approximation der Jacobi-Matrix benötigten Rechenzeit sowie zu einer Vereinfachung des linearen Gleichungssystems des linear-impliziten Euler-Verfahrens. Insgesamt wird hierdurch ein echtzeitfähiger Einsatz des Rohrmodells erreicht. Darüber hinaus wird untersucht, wie der Rechenaufwand zur Lösung des dünn besetzten linearen Gleichungssystems des linear-impliziten Euler-Verfahrens reduziert werden kann, falls der Ansatz der maßgeschneiderten Approximationsweise für die Jacobi-Matrix des Rohrmodells nicht eingesetzt wird. Hierzu wird eine Block-Gauß Elimination eingesetzt, bei der die mit dem Schur-Komplement assoziierten Zustände beim Rohrmodell dahingehend bestimmt werden, dass sowohl das reduzierte System als auch das Restsystem eine günstige Struktur besitzen. Für den potentiellen Einsatz auf zukünftigen Fahrzeug-Steuergeräten wird das Parareal Verfahren untersucht, das eine Parallelisierung der Zeitintegration ermöglicht. Am Beispiel der Leitanwendung wird eine Wahl der Parameter des Verfahrens diskutiert. Desweiteren wird der Einsatz eines vereinfachten Modells für die grobe Stufe des Parareal Verfahrens beim Rohrmodell angewandt. Insgesamt wird durch die Arbeit aufgezeigt, dass die Eignung von Simulationen steifer Modelle für den Einsatz bei Echtzeitanwendungen auf Fahrzeug-Steuergeräten durch die untersuchten numerischen Verfahren signifikant erhöht werden kann.

Abstract

In this thesis, the real-time simulation of stiff models is investigated regarding applications in vehicle control units. Using the example of a stiff pipe model, it is studied, how a real-time-capable simulation of this model on an electronic control unit can be obtained by employing suitable numerical methods. Within the solution of the underlying differential equation of the pipe model with a linearly implicit Euler method, several tailored variants for the approximation of the Jacobian matrix are used in a manner that generalizes the sparsing method of Schiela and Bornemann. This leads to a significant reduction of the required computation time for the approximation of the Jacobian matrix and a simplification of the arising linear equation system within the linearly implicit Euler method. By this means, a real-time-capable simulation of the pipe model on an engine control unit is achieved. Furthermore, it is investigated, how the computational effort for the solution of the sparse linear equation system within the linearly implicit Euler method can be reduced in case that the approach of a tailored Jacobian approximation is not applied. For this purpose, a central tool is a block Gauß elimination, that is applied in a way that the Schur complement is chosen such that both the reduced system and the remaining system obtain an advantageous structure. With regard to a potential application in future vehicle control units, the parareal algorithm, which enables a parallelisation of the time integration of a differential equation, is investigated. Using the example of the pipe model, the determination of suitable parameters of the algorithm is studied. Furthermore, the application of a simplified model in the coarse stage of the parareal algorithm is regarded. Altogether, this work shows, that the suitability of simulations of stiff models within real-time applications on electronic control units can be increased significantly by the investigated numerical methods.

Danksagung

An dieser Stelle möchte ich meinen herzlichen Dank an alle richten, die die Entstehung dieser Arbeit unterstützt haben. Mein herzlicher Dank gilt Prof. Dr. Vincent Heuveline, bei dem die Betreuung dieser Arbeit in jeder Hinsicht in besten Händen war, der ein überaus fruchtbares Umfeld an der Fakultät für Mathematik und Informatik an der Universität Heidelberg geschaffen hat und der sich regelmäßig und ausgiebig Zeit genommen hat, um eine Vielfalt an Ideen und Punkten zu diskutieren. Ich bedanke mich bei Prof. Dr.-Ing. Carsten Proppe für die Zweitbetreuung der Arbeit. Desweiteren möchte ich mich bei Prof. Dr. Rudolf Lohner, Prof. Dr. Götz Alefeld und allen Mitarbeitern des EMCL für die guten Diskussionen und die angenehme Atmosphäre bedanken.

Ich möchte mich bei meinem Umfeld bei Bosch bedanken, wo ich exzellente Rahmenbedingungen bei der zurückliegenden Anstellung als Doktorand erhalten hatte. Ich möchte mich herzlich bei Christian Bertsch bedanken sowohl für den häufigen und tiefgreifenden fachlichen Austausch als auch für die darüber hinausgehende Begleitung, die nicht hätte wertvoller sein können. Ebenso möchte ich mich herzlich bei Elmar Ahle bedanken, der mich als Gruppenleiter in einer Weise geführt hat, die ich bewundere und der an den hervorragenden Rahmenbedingungen der Promotion einen erheblichen Anteil und zahlreiche wertvolle methodische Ratschläge hatte. Desweiteren möchte ich mich bei Reinhard Gantenbein für die hervorragenden Rahmenbedingungen bedanken. Allen Kollegen aus meiner Gruppe danke für die schöne Zeit und den guten Austausch.

Ich möchte mich außerdem bei Ana Gallet, Michael Volpp und Wolfgang Lengerer für die Bereitstellung des Leitanwendungsmodell und für die gute Zusammenarbeit bedanken.

Meinen herzlichen und besonderen Dank sage ich meiner Familie.

Inhaltsverzeichnis

1 Einführung

Mathematische Modelle, die durch eine physikalische Beschreibung eines Systems motiviert und in Form einer Differentialgleichung beschrieben sind, ermöglichen eine kompakte Beschreibung der Wirkzusammenhänge eines mechatronischen Systems und finden auf Fahrzeug-Steuergeräten Anwendung im Rahmen einer modellbasierten Regelung und Diagnose sowie von virtuellen Sensoren. Da bei mechatronischen Systemen Effekte unterschiedlicher physikalischen Domänen involviert sind, sind die zu Grunde liegenden Differentialgleichungen der Modelle in vielen Fällen steif. Insbesondere bei steifen Modellen stellt eine echtzeitfähige stabile Simulation eine erhebliche numerische Herausforderung dar. Aus diesem Grund werden in vorliegender Arbeit Ansätze untersucht, die bei Modellen einer gegebenen Problemklasse angewandt werden können, um eine stabile Echtzeitsimulation auf einem Fahrzeug-Steuergerät zu ermöglichen.

In Kapitel 2 wird zunächst eine Beschreibung der Einsatzmöglichkeiten derartiger Modelle, ihrer grundsätzlichen Vorzüge und Herausforderungen und der relevanten Problemklasse gegeben. Die Problemklasse umfasst Differentialgleichungen, wie sie bei komplexen steifen Modellen mit Anwendung auf Fahrzeugsteuergeräten auftreten. Insbesondere wird eine innere Struktur angenommen, so dass das System zusammengesetzt aus Subsystemen mit relativ gering ausgeprägten Querabhängigkeiten angesehen wird. Die Problemklasse umfasst Modelle bis zu 75 Zuständen. An die Glattheit der Modelle werden geringe Anforderungen gestellt, so dass lediglich angenommen wird, dass sie fast überall stetig differenzierbar sind. Die Untersuchung wird am Beispiel einer Leitanwendung geführt. Als solche wird ein Modell verwendet, das den Massen- und Enthalpiestrom durch ein Rohr beschreibt und die Eigenschaften der Modelle der Problemklasse gut repräsentiert.

Ein Einsatzzweck des Modells auf einem Fahrzeug-Steuergerät besteht beispielsweise darin, dazu beizutragen, die Aufwärmphase eines Katalysators nach einem Kaltstart zu modellieren. Im Rahmen einer Diagnosefunktion der Sauerstoff-Speicherfähigkeit kann hierdurch der Aktivitätsbeginn des Katalysators und somit ein geeigneter Startzeitpunkt für die Diagnose ermittelt werden. Dadurch dass das Modell auch den Strom verschiedener Gasfraktionen erfasst, ist es zudem geeignet, im Rahmen einer Bauteilschutzfunktion eines Katalysators, die ein Überhitzen verhindert, dazu beizutragen, den Strom der für die exotherme Umsetzung im Katalysator wesentlichen Gasfraktionen zu beschreiben.

Die zu Grunde liegende gewöhnliche Differentialgleichung ist eine Semidiskretisierung einer nichtlinearen partiellen Differentialgleichungen, bei der ein Großteil der physikalischen Koeffizienten basierend auf physikalischen Abhängigkeiten aus den Zuständen in jedem Zeitschritt berechnet wird. Die Berechnung der Koeffizienten verursacht einen nennenswerten Anteil des Rechenaufwands im Rahmen einer Auswertung der rechten Seite der Differentialgleichung und trägt zur Nichtlinearität der Differentialgleichung bei. Zudem führt sie dazu, dass die rechte Seite nicht mehrfach differenzierbar ist. Die Anzahl an Diskretisierungszellen liegt zwischen fünf und 15 Zellen und dient als Applikationsparameter, um die physikalische Dispersion durch die numerische Diffusion zu imitieren.

In Kapitel 3 wird auf die Rahmenbedingungen einer Echtzeitsimulation auf Fahrzeug-Steuergeräten eingegangen. Als Testhardware wird das Motorsteuergerät MDG1C Device 4 von Bosch gewählt. Gleitkommaberechnungen werden hierbei in 32 Bit Genauigkeit durchgeführt. Für das Rohrmodell liegt durch einen geeigneten Software-Stand ein praxisnaher Programmablauf vor.

Dadurch dass auf einem Steuergerät über 1000 Tasks betrieben werden, führt die im Vergleich zu Desktop-Rechnern geringe Rechenleistung zur Notwendigkeit einer effizienten Software. Für

die Simulation des Rohrmodells hat dies zur Konsequenz, dass eine geringe Rechenzeit je Zeitschritt notwendig ist, damit sie im Rahmen einer Software-Funktion eines Fahrzeug-Steuergeräts zum Einsatz kommen kann. Aus diesem Grund ist die Reduktion des Rechenaufwands für einen Zeitschritt des Rohrmodells wesentlich, um einen echtzeitfähigen Betrieb auf einem Fahrzeug-Steuergerät zu ermöglichen.

In Kapitel 4 wird auf Implementierung und Rechenzeitoptimierung für die Echtzeitanwendung des Rohrmodells eingegangen. Zur Lösung der gewöhnlichen Differentialgleichung des Rohrmodells wird in vorliegender Arbeit ein linear-implizites Euler-Verfahren gewählt. Zur Überprüfung der Lösungsverläufe dient ein gemessener Verlauf der Gastemperaturen am Rohrausgang.

Bei einer Lösung des im Rahmen des linear-impliziten Euler-Verfahrens auftretenden linearen Gleichungssystems mit einer Gauß Elimination mit Spaltenpivotisierung und einer Bildung der Jacobi-Matrix durch numerisches Differenzieren beansprucht die Differenziation je nach Anzahl an Diskretisierungszellen des Rohrmodells zwischen etwa 60 % und 81 % der Rechenzeit eines Zeitschritts des linear-impliziten Euler-Verfahrens. Somit ist die Reduktion der Rechenzeit zur Ermittlung einer Approximation der Jacobi-Matrix essentiell zur Reduktion der Rechenzeit einer Simulation des Rohrmodells.

Das linear-implizite Euler-Verfahren ist eine W Methode. Daher führt eine Abweichung bei der Approximation der Jacobi-Matrix von der exakten Jacobi-Matrix nicht unmittelbar dazu, dass die berechnete Lösung durch das linear-implizite Euler-Verfahren in einer ähnlichen Größenordnung von der exakten Lösung abweicht. Beim sogenannten Sparsing wird diese Eigenschaft des linear-impliziten Euler-Verfahrens dazu genutzt, ausgewählte Einträge der Jacobi-Matrix durch eine Null zu ersetzen, um hierdurch zu erreichen, dass das resultierende lineare Gleichungssysteme dünner besetzt und somit bei einem geeigneten Lösungsverfahren mit einem geringeren Rechenaufwand gelöst wird. Falls durch das Sparsing ganze Spaltenvektoren der Jacobi-Matrix zu einem Nullvektor werden oder im Fall der Kombination von Sparsing mit einer Färbung der Jacobi-Matrix führt dies zudem zu einer möglichen Reduktion des Rechenaufwands bei der Bildung der Jacobi-Matrix.

Bei der Bestimmung einer Menge von Einträgen der Jacobi-Matrix, die geeignet ist, dass ihre Einträge jeweils durch eine Null ersetzt werden, wird beim Sparsing gefordert, dass durch das Ersetzen der Einträge die Eigenwerte der Matrix kaum geändert werden. Diese Bestimmung wird nicht zur Laufzeit, sondern bereits während der Einstellung des Lösungsverfahrens durchgeführt. Um sie mit praktikablem Rechenaufwand zu erzielen, wird beim Sparsing ein Zusammenhang genutzt, der es erlaubt, für hinreichend kleine Störungen die Summe der Abweichungen der Eigenwerte der gestörten Matrix gegenüber der originalen Matrix zu approximieren, ohne die Eigenwerte zu berechnen. Ein Nachteil dieser Vorgehensweise besteht darin, dass eine Störung einer Matrix zu einer wesentlichen Veränderung der Eigenwerte führen kann und gleichzeitig die Summe der Abweichungen gering ausfällt, falls sich die Abweichungen bei verschiedenen Eigenwerten auslöschen. In diesem Fall führt das Ersetzen der Einträge durch Nullen zu einer unerwünschten Veränderung der Matrix, ohne dass dies festgestellt wird.

Um diesen Fall zu verhindern, wird in vorliegender Arbeit eine Modifikation der Vorgehensweise aufgezeigt, durch die für hinreichend kleine Störungen die Summe der Betragsquadrate der Abweichungen entsprechender Eigenwerte der gestörten und originalen Matrix approximiert wird, ohne hierzu die Eigenwerte zu berechnen. Hierdurch tritt keine Auslöschung von Abweichungen bei verschiedenen Eigenwerten auf und eine Veränderung der Eigenwerte ist im Wert der Summe erkennbar.

Anstatt eine Jacobi-Matrix dahingehend zu stören, dass Einträge durch Nullen ersetzt werden, kann eine Störung auch dahingehend gewählt werden, dass die Bildung der gestörten Matrix während der Laufzeit mit einem geringen Rechenaufwand verbunden ist. Dieses Vorgehen wird in vorliegender Arbeit verfolgt und als Vergröberung der Jacobi-Matrix bezeichnet. Um die Wirksamkeit dieses Vorgehen zu beurteilen, wird neben einer numerischen Differenziation eine weitere Vergleichsvariante betrachtet, bei der eine numerische Differentiation mit einer Färbung der Jacobi-Matrix sowie einer Mixed Mode Integration eingesetzt wird.

Bei der Störung werden drei Varianten betrachtet: Zum einen wird im Rahmen der Bildung der Jacobi-Matrix die Berechnung der Koeffizienten der rechten Seite der Differenzialgleichung ersetzt durch einen polynomiellen Approximationsausdruck, um so den Rechenaufwand einer Auswertung der rechten Seite zu reduzieren und die Differenziation zu beschleunigen. Dies führt dazu, dass die Rechenzeit der Bildung der Jacobi-Matrix zwischen 66 % bis 70 % gegenüber der regulären numerischen Differentiation reduziert wird.

Desweiteren wird der Ansatz betrachtet, einige Einträge der Jacobi-Matrix anhand fester Beziehungen aus anderen Einträgen zu errechnen. Bei der zugehörigen Variante wird dieser Ansatz mit einem Sparsing sowie einer Färbung der Jacobi-Matrix kombiniert. Diese Variante führt zu erheblich geringeren Rechenzeiten: Gegenüber der originalen Variante beträgt die Reduktion der Rechenzeit 96 % bis 99 %, gegenüber der Variante mit Mixed Mode Integration und einer Färbung der Jacobi-Matrix beträgt die Reduktion zwischen 87 % und 88 %.

Außerdem wird der Ansatz untersucht, für die Berechnung ausgewählter Einträge der Jacobi-Matrix die Auswertung von Approximationsausdrücke einzusetzen, die erhalten werden, indem die Zustandsabhängigkeit gewisser physikalischer Koeffizienten vernachlässigt wird. Bei der zugehörigen Implementierungsvariante wird dieser Ansatz mit den vorangehenden Ansätzen kombiniert. Bei dieser Variante wird die Rechenzeit zur Bildung der gestörten Matrix gegenüber der vorangehenden Varianten nochmals um etwa 51 % bis 52 % reduziert. Insgesamt ergibt sich hierbei also eine Reduktion der Rechenzeit, die je nach Anzahl an Diskretisierungszellen zwischen etwa 98 % und über 99 % gegenüber einer numerischen Differenziation der rechten Seite liegt und die je nach Anzahl an Diskretisierungszellen um 93 % bis zu 94 % gegenüber derjenigen Variante beiträgt, bei der eine numerische Differentiation mit einer Färbung der Jacobi-Matrix sowie einer Mixed Mode Integration eingesetzt wird.

Da die mathematische Beschreibung der Störung der Jacobi-Matrix dieselbe ist wie beim Sparsing, kann das dort verwendete Vorgehen zur Einschätzung der Eignung einer Störung der Jacobi-Matrix übernommen werden. Im Fall des Rohrmodells wird zur Prüfung der Eignung die Abweichung des sich ergebenden Simulationsverlaufs gegenüber der Referenztrajektorie verwendet. Die sich ergebende Änderung des Verlaufs des relativen Fehlers ist gering und beträgt bei jeder Varianten weniger als einen halben Prozentpunkt.

Insgesamt erweist sich der Ansatz der Verwendung einer geeignet gewählten Störung der Jacobi-Matrix zur Rechenzeitreduktion beim untersuchten Rohrmodell als sehr zweckmäßig. Die erzielte Rechenzeitreduktion ist immens. Zu beachten ist, dass die Referenztrajektorie entscheidend ist dafür, welche Störungen als geeignet anzusehen sind. Daher ist die Verfügbarkeit einer für den Einsatzzweck repräsentativen Referenztrajektorie bei diesem Vorgehen wesentlich.

Eine Konsequenz der Vergröberung der Jacobi-Matrix besteht darin, dass durch das beinhaltete Sparsing im Fall des Rohrmodells die Struktur des im Rahmen des linear-impliziten Euler-Verfahrens auftretenden linearen Gleichungssystems signifikant vereinfacht wird. Aus diesem Grund wird die Rechenzeit zur Lösung des linearen Gleichungssystems im Fall der Vergröberung der Jacobi-Matrix dementsprechend verringert, sofern ein geeignetes Lösungsverfahren eingesetzt wird. Hierdurch ist die resultierende Rechenzeit für einen Zeitschritt des linear-impliziten Euler-Verfahrens gering genug, um für einen Einsatz im Rahmen einer Software-Funktion auf einem Fahrzeug-Steuergerät geeignet zu sein.

In Kapitel 5 wird zum Zwecke der Übertragbarkeit auf andere Modelle über die Betrachtungen aus Kapitel 4 zur Reduktion der Rechenzeit der Bildung einer Approximation der Jacobi-Matrix hinaus eine Rechenzeitreduktion für die Lösung des linearen Gleichungssystems im Rahmen des linear-impliziten Euler-Verfahrens untersucht. Im Hinblick auf eine betrachtenswerte Gleichungsstruktur werden hierbei diejenigen Gleichungssysteme herangezogen, die entstehen, wenn bei der Simulation des Rohrmodells lediglich eine Mixed Mode Integration und kein Sparsing eingesetzt wird.

Zunächst wird die Kondition der Systeme untersucht. Diese sind bei der Leitanwendung gut konditioniert, so dass die Lösung mittels einer Gauß Elimination auch mit 32 Bit Gleitkommava-

riablen unproblematisch ist. Darüber hinaus liegt sogar der Fall vor, dass eine Gauß Elimination ohne Pivotisierung eingesetzt werden kann. Dies wird dadurch gerechtfertigt, dass sich die Wachstumsfaktoren im Rahmen einer Gauß Elimination für die Referenztrajektorie abschätzen lassen und klein sind.

Das Leitmotiv zur Rechenzeitreduktion der linearen Gleichungssysteme besteht darin, die Struktur der Systeme im Rahmen von Voruntersuchungen ausgiebig zu analysieren und eine auf das Modell zugeschnittene Einstellung des Lösungsverfahren zu erstellen, die die gewonnenen Erkenntnisse über die Struktur in eine Reduktion der Rechenzeit ummünzt. Hierzu wird der Adjazenzgraph der Matrix des linearen Gleichungssystems untersucht.

Ein zentraler Ansatz basiert hierbei auf einer sogenannten *nested dissection*: Für den Adjazenzgraph wird ein starker Separator bestimmt. Zur Lösung des Gleichungssystems wird eine Block-Gauß Elimination eingesetzt. Die zum Separator korrespondierenden Zeilen und Spalten der Matrix werden durch eine Permutation zusammen gruppiert. Bei einer Block-Gauß Elimination, bei der die zum Separator korrespondierenden Variablen die Variablen des reduzierten Systems bilden, besitzt das Restsystem dann eine Blockdreiecksform. Sofern der Separator ein schwacher Separator ist, besitzt das Restsystem sogar Blockdiagonalform. Für die entstehenden Restsysteme wird dieses Vorgehen wiederholt.

Im Fall der untersuchten Struktur ergibt sich hierdurch eine Block-Gauß Elimination, bei der das reduzierte System die Dimension 2 hat und das Restsystem eine Blockdreiecksform, bei der die maximale Dimension eines Diagonalblocks 4 ist. Letztlich wird die Lösung des Gleichungssystems also auf die Lösung von Gleichungssysteme der Dimension $d \leq 4$ zurückgeführt.

Neben einer Lösung mit einer Block-Gauß Elimination wird zudem die Lösung mittels einer Gauß Elimination mit Verwendung eines geeigneten Orderings betrachtet. Dieses wird basierend auf der Untersuchung des Adjazenzgraphen der Matrix des Gleichungssystems manuell bestimmt. Im Vergleich mit Orderings, die sich durch ein Approximate Minimum Degree Verfahren oder ein Reverse Cuthill McKee Verfahren aus der initialen Form ergeben, führt dieses zu einem geringeren Fill-in. Im Vergleich mit einer physikalisch natürlichen Anordnung der Zustände führt es zu einem gleichwertigen Fill-in. Die numerische Stabilität ist bei Verwendung des Orderings gewährleistet.

Damit die dünn besetzte Struktur und der geringe Fill-in des ermittelten Orderings im Rahmen einer Gauß Elimination zu einer geringen Rechenzeit führt, ist das Verwenden eines Sparse-Formats nötig. Hierfür wird das CRS Format gewählt. Darüber hinaus wird eine Variation des CRS Formats beschrieben, um die Rechenzeit weiter zu reduzieren. Die Idee hierbei beruht darauf, dass die Struktur des Gleichungssystems und somit der gesamte potentiell auftretende Fill-in festgelegt ist. Indem anstatt der Einträge, die ungleich Null sind, genau die Einträge durch das Sparse-Format berücksichtigt werden, die *potentiell* nicht verschwinden, tritt nicht der Fall ein, dass im Rahmen der Elimination zusätzliche zu speichernde Einträge auftreten. Folglich lassen sich zu jeder Spalte der Matrix die Indizes der Einträge ablegen, die potentiell ungleich Null sind. Hierdurch entfällt die Notwendigkeit einer Indexsuche bei der Ermittlung der Einträge einer gegebenen Spalte.

Für eine Lösung des linearen Gleichungssystems werden neben einer Elimination, die die Struktur des Systems nicht nutzt, vier Varianten implementiert: eine Variante nutzt eine Elimination mit einem CRS Format als Sparse-Format, eine weitere verwendet die vorangehend beschriebene Modifikation des CRS Formats, eine dritte Variante setzt darüber hinaus das manuell bestimmte Ordering ein. Die vierte Variante nutzt die maßgeschneiderte Lösungsweise mit einer Block-Gauß Elimination. Aus den ermittelten Rechenzeiten ergibt sich, dass durch die Verwendung eines Sparse-Formats eine beträchtliche Reduktion der Rechenzeit zwischen 64 % und 89 % erzielt wird. Durch die modifizierte Variante des Sparse-Formats kann die Rechenzeit um weitere 45 % bis 49 % reduziert werden und in Kombination mit dem manuell bestimmten Ordering verringert sich die Rechenzeit zusätzlich um 8 % bis 59 %. Die Rechenzeit bei einer Lösung durch die Block-Gauß Elimination ist im Fall von fünf Diskretisierungszellen nochmals zwischen 10 % und 42 % geringer als bei einer Elimination mit dem manuell bestimmten Ordering und dem modifizierten Sparse-Format.

Insgesamt wird deutlich, dass im Fall des Rohrmodells der Einsatz maßgeschneiderter Lösungsverfahren eine signifikante Reduktion der Rechenzeit der Lösung der betrachteten linearen Gleichungssysteme ermöglicht, so dass gegenüber einer Lösung mit einer Gauß Elimination mit Spaltenpivotisierung und ohne Strukturberücksichtigung bei jeder betrachteten Anzahl an Diskretisierungszellen eine Rechenzeitreduktion von über 90 % erzielt wird.

In Kapitel 6 wird im Hinblick darauf, dass entsprechend der Entwicklung bei eingebetteten Systemen auch bei Fahrzeug-Steuergeräten die Anzahl an Rechenkernen zunimmt und somit eine Möglichkeit zur Parallelisierung der Lösung eines Anfangswertproblems relevant ist, die Eignung eines Einsatzes des Parareal Verfahrens untersucht. Dieses Verfahren gehört zu einer Klasse von Verfahren zur Lösung einer Differentialgleichung, die eine Parallelisierung der Zeitintegration ermöglichen.

Das Parareal Verfahren besitzt verschiedene Parameter wie die Anzahl an verwendeten Mikrozeitintervalle, die Anzahl an Zeitschritten der feinen Stufe je Mikrozeitintervall sowie die Anzahl an durchgeführten Parareal Iterationen. Am Beispiel eines Zeitschrittes des Rohrmodells wird eine Untersuchung durchgeführt, wie diese Parameter zu wählen sind, um ein günstiges Verhältnis von Fehler und Rechenaufwand zu erzielen. Der Rechenaufwand wird hierbei nicht durch Messungen ermittelt, sondern basiert auf einer heuristischen Berechnung. Durch diese Untersuchung wird deutlich, dass das Parareal Verfahren auch bei einer geringen Anzahl an Rechenkernen sinnvoll einsetzbar ist. Um beim Parareal Verfahren eine hohe parallele Effizienz zu erzielen, ist es notwendig, dass die Anzahl an Iterationen gering ist und idealerweise nur eine Iteration durchzuführen ist. Im Rahmen der Untersuchung wird festgestellt, dass das Rohrmodell ein Beispiel darstellt, bei dem das Parareal Verfahren bereits mit einer Iteration zweckmäßig einsetzbar ist. Somit besitzt das Verfahren in diesem Fall eine hohe parallele Effizienz und die Parallelisierung durch das Parareal Verfahren führt nicht zu einer signifikanten Vergrößerung des Gesamtrechenaufwands gegenüber einer nicht parallelisierten Lösung.

Ein wesentlicher Faktor für diese Vergrößerung des Gesamtrechenaufwands beim Parareal Verfahren gegenüber einer nicht parallelisierten Lösung ist der Rechenaufwand der groben Stufe. Um die parallele Effizienz des Verfahrens weiter zu erhöhen, ist es daher sinnvoll, den Rechenaufwand der groben Stufe zu reduzieren. Hierzu werden zwei Möglichkeiten betrachtet:

Zum einen wird die Methode der Vergröberung der Jacobi-Matrix im Rahmen der groben Stufe des Parareal Verfahrens eingesetzt. Diese Kombination führt zu Simulationsverläufen, deren Abweichung von der Referenztrajektorie sich kaum von einem Parareal Verfahren ohne eine Vergröberung der Jacobi-Matrix unterscheidet und ist somit zweckmäßig.

Eine weitere Möglichkeit, den Rechenaufwand der groben Stufe bei einer Integration mit dem linear-impliziten Euler-Verfahren besteht darin, lediglich ein reduziertes System zu lösen, das die schnellen Anteile der Dynamik umfasst, da dieses über den kurzen Zeitraum eines Zeitschrittes hinweg zu einer adäquaten Beschreibung der Dynamik des Systems ausreichend sein kann. Es wird nachgewiesen, dass bei einem Parareal Verfahren, das in der groben und feinen Stufe jeweils ein linear-implizites Euler-Verfahren verwendet, dieser Ansatz zu keiner Reduktion der Konsistenzordnung des Verfahrens führt. Im Fall des Rohrmodells geht die Bestimmung eines derartigen Teilsystems aus den Untersuchungen im Rahmen der Vergröberung der Jacobi-Matrix hervor. Hierbei ist im untersuchten Fall keine Transformation der Jacobi-Matrix nötig. Im Fall einer geringen Anzahl an Zeitschritten der feinen Stufe je Mikrozeitintervall, wie sie bei einem Einsatz auf einem Fahrzeug-Steuergerät zu erwarten ist, führt dieser Ansatz beim Rohrmodell zu keiner signifikanten Änderung des Simulationsverlaufs und ist somit ebenfalls als geeignet anzusehen.

In Kapitel 7 ist der Schluss vorliegender Arbeit beschrieben. Hierin sind die zentralen Erkenntnisse und Ergebnisse der Arbeit zusammengestellt.

2 Problemstellung

Als mathematisches Modell wird in vorliegender Arbeit ein Modell bezeichnet, das durch physikalische Zusammenhänge motiviert und durch eine Differentialgleichung beschrieben wird (vgl. 2.1.2). Ein steifes mathematisches Modell wird kurz auch als steifes Modell bezeichnet.

Die Relevanz steifer Modelle und ihrer Simulation auf Fahrzeug-Steuergeräten wird im Folgenden vorgestellt. Im Anschluss wird das Leitanwendungsmodell dieser Arbeit beschrieben. Am Beispiel der Leitanwendung werden in vorliegender Arbeit die numerischen Untersuchungen betrieben, die einen Beitrag leisten, eine echtzeitfähige Simulation steifer Modelle auf Fahrzeug-Steuergeräten zu ermöglichen.

2.1 Einsatz mathematischer Modelle in Fahrzeug-Steuergeräten

Im Zuge der voranschreitenden Entwicklung mechatronischer Systeme ist die zunehmende Bedeutung der Simulation von Modellen nicht nur während der Entwicklung und Herstellung, sondern auch innerhalb von Echtzeitanwendungen auf den Produkten selbst naheliegend. Für den Fall von Fahrzeug-Steuergeräten wird in dieser Arbeit untersucht, wie die Simulation eines steifen Modells auf einem Steuergerät echtzeitfähig betrieben werden kann.

2.1.1 Anwendungsfelder mathematischer Modelle in Fahrzeug-Steuergeräten

Die zentralen Einsatzgebiete mathematischer Modelle in Fahrzeug-Steuergeräten sind als virtueller Sensor, bei modellbasierter Diagnose und modellbasierter Regelung. Zukünftig ist daneben ein vermehrtes Auftreten mathematischer Modelle im Rahmen einer modellprädiktiven Regelung denkbar.

Virtueller Sensor

Ein wichtiger Anwendungsfall mathematischer Modelle in Echtzeitanwendungen auf Fahrzeug-Steuergeräten ist als virtueller Sensor. Unter einem virtuellen Sensor versteht man eine Software-Routine, die dazu dient, einen realen Sensor zu ersetzen durch das Zurückgreifen auf ein Modell sowie gegebenenfalls auf weitere aus dem Kontext zur Verfügung stehenden Daten wie etwa Systemparameter und die Werte anderer realer Sensoren im betrachteten System (vgl. [70]).

Ein Mehrwert von virtuellen Sensoren liegt darin, dass die mit einem realen Sensor verbundenen Kosten eingespart werden. Ein anschauliches Beispiel hierfür ist im Patent [17] beschrieben, durch das es möglich wurde, unter anderem den Drucksensor in der Niederdruck-Abgasrückführung im Luftsystem eines aufgeladenen Dieselmotors durch einen virtuellen Sensor einzusparen. Darüber hinaus kommen virtuelle Sensoren auch dann zum Einsatz, wenn reale Sensoren zur Bestimmung einer Größe nicht verfügbar sind, nicht die technischen Anforderungen wie etwa Latenzen einhalten können oder unverhältnismäßig aufwändig sind.

Modellbasierte Diagnose

Eine weitere wichtige Einsatzmöglichkeit von mathematischen Modellen auf Steuergeräten ist zu Zustandsüberwachungs- und Diagnosezwecken (vgl. [38]): Typischerweise wird dabei eine Größe auf mindestens zwei unabhängige Wege ermittelt, beispielsweise einerseits durch einen Sensor

und andererseits durch einen modellbasierten Vergleichswert. Die entstehende Informationsredundanz wird genutzt, um bei einer Abweichung der Werte voneinander auf einen Fehler zu schließen und diesen möglichst genau zu lokalisieren.

In [45] wurde die modellbasierte Diagnose von Zylinderdrucksensoren bei Dieselmotoren untersucht. Die Zylinderdrucksensoren sind sehr anspruchsvollen Bedingungen ausgesetzt, was eine Überwachung des Verschleißes notwendig macht. Durch die Überwachung ist es möglich, die Qualität des Zylinderdrucksignals einzuordnen und bei Bedarf Steuergerätefunktionen dem Verschleißzustand entsprechend abzuschalten.

Modellbasierte Regelung

Darüber hinaus bietet der Einsatz mathematischer Modelle auch im Rahmen von Regelungen erhebliche Chancen: In [12] wird für die Einspritzung bei einem Dieselmotor mit Abgasrückführung und Turbolader auf ein mathematisches Modell für die Zylinderfüllung zurückgegriffen.

Der Einsatz eines mathematischen Modells ist hierfür besser geeignet als der Einsatz von Kennfeldern für die stationären Zustände, die durch empirische Funktionen für die Übergangsphasen angepasst werden. Nachteil dieser kennfeldbasierten Vorgehensweise ist, dass es in den Übergangsphasen oder auch durch veränderliche Umgebungsparameter dazu kommt, dass die tatsächliche Zylinderfüllung vom entsprechenden Wert des Kennfeldes abweicht. Durch den Einsatz eines mathematischen Modells für die Zylinderfüllung kann diese insbesondere für derartige Umgebungsparameter und Übergangsphasen akkurater beschrieben werden. Die genauere Erfassung des Zylinderdrucks wird in dieser Quelle bei der Ermittlung der Einspritzparameter ausgenutzt. Hierdurch wird eine Reduktion der CO_2-Emissionen um 4 % und eine Reduktion des Rußanteils um etwa 50 % gegenüber der üblichen Einspritzfunktion erreicht.

Modellprädiktive Regelung

Ein weiterer Rahmen, in dem mathematische Modelle in Echtzeit-Anwendungen in Fahrzeug-Steuergeräten zum Einsatz kommen können, ist die modellprädiktive Regelung (vgl. [29]). Hierbei wird durch ein mathematisches Modell berechnet, wie sich die Wahl einer Steuergröße auf eine Zielgröße in einem bevorstehenden Zeitintervall auswirkt und die Wahl der Steuergröße anhand dieser Prognosen derart bestimmt, dass die Zielgröße optimiert wird. In [64] wird dies beispielsweise eingesetzt, um in einem Nutzfahrzeug die Gangwahl derart zu treffen, dass der Spritverbrauch minimiert wird.

Bei einer modellprädiktiven Regelung ist eine Prädiktion, die eine Rechnung über mehrere Zeitschritte hinweg erfordert, und eine Optimierung durchzuführen. Durch den relativ hohen Rechenbedarf im Vergleich zu den vorangehenden Anwendungsfällen mathematischer Modelle wird die modellprädiktive Regelung voraussichtlich mit zunehmender Rechenleistung der Steuergeräte an Bedeutung gewinnen.

2.1.2 Beschreibungsweise mathematischer Modelle

Als mathematisches Modell wird in dieser Arbeit ein Modell bezeichnet, das ein System vorrangig basierend auf mathematischen Gesetzmäßigkeiten beschreibt und das in Form einer gewöhnlichen Differentialgleichung der Gestalt

$$\dot{x}\left(t\right) = f\left(t, x\right)$$

dargestellt ist, wobei x die Zustände des Systems bezeichnet und t die Zeit. Außerdem werde die Funktion $f : I \times D \to \mathbb{R}^n$ als rechte Seite bezeichnet, wobei I ein reelles Intervall und $D \subseteq \mathbb{R}^n$ offen sei. Die rechte Seite wird als stückweise stetig differenzierbar angenommen.

Der Begriff des mathematischen Modells kann auch Systembeschreibungen mittels eines differentiell-algebraischen Gleichungssystems oder eines partiellen Differentialgleichungssystems umfassen.

Diese werden in dieser Arbeit allerdings nicht in gezielter Weise betrachtet, da das Ziel verfolgt wird, zunächst die echtzeitfähige Lösung von steifen gewöhnlichen Differentialgleichungen auf einem Fahrzeug-Steuergerät zu ermöglichen.

An dieser Stelle wird darauf hingewiesen, dass bei der Simulation eines mathematischen Modells letztlich ein Anfangswertproblem zu lösen ist. Der Anfangswert ist allerdings nicht unbedingt bekannt und eine geeignete Initialisierung unter Umständen nicht trivial. In solchen Fällen ist die Stabilität des Systems ein essentieller Faktor, um trotz einer ungenauen Initialisierung einen guten Simulationsverlauf zu erhalten.

Ein weiterer Aspekt von mathematischen Modellen mechatronischer Systeme sind Eingangsdaten des Modells. Sie werden als rein zeitabhängig angenommen, mit $u(t)$ bezeichnet und die Abhängigkeit der rechten Seite von den Eingangsdaten wird nicht explizit dargestellt, sondern ihre Darstellung in Abhängigkeit der Zeit wird in der rechten Seite subsumiert. Sie stellen eine wesentliche Ursache für die Zeitabhängigkeit der rechten Seite des Systems dar.

2.1.3 Datenbasierte Modelle und ihre Eigenschaften

Dem mathematischen Modell steht der Begriff des datenbasierten Modells gegenüber, das ein System basierend auf einem diskreten Datensatz beschreibt, der beispielsweise empirisch gewonnen wird. Typischerweise wird eine Interpolations- oder Approximationsmethode eingesetzt, um den Einsatz des Modells auch auf Zustände auszudehnen, für die kein empirischer Wert im Datensatz vorliegt. Musterbeispiel eines datenbasierten Modells ist das Kennfeld. Daneben sind neuronale Netzen oder Modelle basierend auf einer Gauß-Prozess-Regression nennenswerte Vertreter (vgl. [101, 61, 112]).

Datenbasierte Modelle, allem voran in der Form von kennfeldbasierten Modelle, sind auf Fahrzeug-Steuergeräten weit verbreitet (vgl. [103]). Sie stellen derzeit den Stand der Technik dar. Für ihren Einsatz gibt es eine Reihe von Gründen: Das Prinzip eines kennfeldbasierten Modells ist einfach, mit einem geringen Rechenbedarf verbunden und für eine Vielzahl an Systemen einsetzbar.

Allerdings haben kennfeldbasierte und auch allgemeine datenbasierte Modelle neben diesen Vorteilen auch nachteilige Eigenschaften und Grenzen.

Begrenzte Parameteranzahl Der Speicherbedarf eines Kennfeldes und auch verschiedener weiterer datenbasierter Modelle wächst exponentiell mit der Anzahl an Parametern. Dies wird als Fluch der Dimensionalität bezeichnet und führt zu einer natürlichen Grenze für die Anzahl an Parametern bei derartigen datenbasierten Modell.

Bei Kennfelder sind bereits drei Parameter eine typische Grenze. Bei Modellen, die auf einer Gauß-Prozess-Regression basieren, liegt die Grenze etwa eine Größenordnung höher. Für Modelle, die mehr Parameter benötigen, sind andere Modellierungsweisen nötig.

Aufwändige Parametrierung Für die Parametrierung datenbasierter Modelle sind teils aufwändige Messungen nötig. Hierbei ist das Risiko von fehlerhaften Messungen gegeben, beispielsweise durch Fehler in den Sensoren oder auch im Messaufbau. Im Falle einer leichten Variation des Systems, beispielsweise der Veränderung einer Rohrlänge, sind unter Umständen sämtliche Messungen erneut durchzuführen.

Geringe Extrapolationsfähigkeit Nicht für alle Betriebspunkte, die durch ein datenbasiertes Modell abgedeckt werden müssen, sind Messungen zugänglich. Am Beispiel eines Turboladers wird dies in [77] beschrieben. Gewisse Betriebspunkte können zudem in einem Messaufbau nicht hergestellt werden, ohne das Bauteil zu schädigen oder zu zerstören.

In derartigen Fällen kommt zum Tragen, dass datenbasierte Modelle typischerweise schwache Extrapolationseigenschaften haben. Das naive Extrapolieren der Werte von verfügbaren Betriebs-

punkten auf die benötigten Betriebspunkte ist ungeeignet und zusätzlicher Aufwand ist nötig, um für die Betriebspunkte, für die keine Messungen vorliegen, geeignete Werte zu bestimmen.

Nachteile beim dynamischen Verhalten Bei kennfeldbasierten Modellen und auch Modellen basierend auf neuronalen Netzen oder Gauß-Prozess-Regression ist das dynamische Verhalten von mechatronischen Systemen in ihren Übergangsphasen nicht trivial abzubilden. Um ein dynamisches Verhalten in Übergangsphasen bei einem datenbasierten Modell zu berücksichtigen, sind gezielt Maßnahmen zu ergreifen, wie beispielsweise in [89] beschrieben.

Wie in Abschnitt 2.1.1 bereits erwähnt, ist es ein gängiger Ansatz, bei einem Kennfeld mittels empirischer Funktionen eine Anpassung an die Übergangsphasen vorzunehmen. Auch im Falle einer solchen Anpassung ist das dynamische Verhalten in transienten Phasen der Systeme allerdings oft nicht akkurat wiedergegeben. Daneben kann eine solche Anpassung anfällig sein für den Fall, dass sich Umgebungsparameter ändern (vgl. [12]).

2.1.4 Vor- und Nachteile mathematischer Modelle

Ein grundlegendes Motiv dafür, Modelle auf mathematische Weise zu beschreiben, ist, dass diese im Stande sind, Prozesse in realen Systeme entsprechend ihrer Wirkzusammenhänge zu beschreiben und durch ihre mathematische Formulierung in Gestalt von Differentialgleichungen hierfür zudem eine kompakte Beschreibung bereitzustellen.

Vorteile mathematischer Modelle

Mathematische Modelle bieten bei einigen Eigenschaften Vorteile gegenüber datenbasierten Modellen. In [103] werden sie insbesondere als eine vielversprechende Möglichkeit beschrieben, die Anzahl an Kennfeldern auf Fahrzeug-Steuergeräten zu reduzieren. In mehreren Aspekten sind die Stärken mathematischer Modelle komplementär zu den Nachteilen von datenbasierten Modellen.

Modellübersicht Dadurch dass mathematische Modelle Systeme entsprechend ihrer Wirkzusammenhänge beschreiben und die Beschreibung in Form von Gleichungen vorliegt, sind sie im Vergleich zu datenbasierten Modellen prinzipiell gut nachvollziehbar und überschaubar. Die Modellübersicht und das Modellverständnis sind bei der Fehlersuche und -vermeidung von Nutzen.

Dynamisches Verhalten Ein mathematisches Modell beinhaltet von Natur aus eine akkurate Beschreibung auch für die Übergangsphasen des Systems. Die Anwendungsfälle, bei denen die die Übergangsphasen eines Systems von Bedeutung sind, sind regelrecht prädestiniert dafür, durch ein mathematisches Modell beschrieben zu werden.

Parameteranzahl Der Speicher- und Rechenbedarf in Abhängigkeit der Parameteranzahl skaliert bei mathematischen Modellen nicht exponentiell, sondern im Allgemeinen linear. Hierdurch ergeben sich keine restriktiven Grenzen der Parameteranzahl wie es bei datenbasierten Modellierungsarten typisch ist.

Parametrierung Die Parametrierung eines mathematischen Modells ist mit geringem Aufwand möglich, sofern die Parameter des Modells eine mathematische Bedeutung und mit geringem Aufwand messbar sind (vgl. auch [84]). Beispielsweise ist die Länge eines Rohres leicht messbar und die spezifische Wärmekapazität eines Gasgemisches mit Hilfe von Tabellenwerken leicht zugänglich.

In solchen Fällen ist bei einer Variante des Modells ein geringer Aufwand für die Modellanpassung ausreichend. Vor dem Hintergrund einer zunehmenden Variantenvielfalt, die beispielsweise unter

anderem durch die Hybridisierung des Antriebstrangs bedingt ist, ist dies eine potentialträchtige Qualität mathematischer Modelle.

Extrapolationsfähigkeit Durch die Beschreibung eines Systems entsprechend seiner Wirkzusammenhänge sind bei mathematischen Modellen auch bei einer Extrapolation physikalisch plausible Werte zu erwarten. Wird ein mathematisches Modell für ein gewissen Nutzungsbereich konzipiert, so beschreibt es häufig auch in einem etwas größeren Bereich vernünftige physikalische Abhängigkeiten. Beispielsweise ist das in Abschnitt 2.2 ausführlich beschriebene Rohrmodell, das unter anderem den Enthalpiestrom im Rohr abbildet, aus praktischen Gründen für Temperaturen über $-50°C$ konzipiert, weil es für den Einsatz bei niedrigeren Temperaturen nicht vorgesehen ist. Sofern kein unmodellierter Effekt, wie etwa ein unberücksichtigter Phasenübergang, hinzukommt, ist aber zu erwarten, dass das Modell auch bei Temperaturen von $-60°C$ plausible Werte liefert.

Nachteile mathematischer Modelle

Natürlich weisen mathematische Modelle neben Vorteilen auch Nachteile auf und führen teilweise zu spezifischen zusätzlichen Herausforderungen. Die wichtigsten Aspekte sind folgende:

Modellierungsaufwand Der anfängliche Modellierungsaufwand für mathematische Modelle ist relativ hoch und erfordert sehr spezifische Kompetenzen im Bereich Numerik und Modellierung. Da sich zahlreiche Bestandteile mathematischer Modelle für Anwendungen auf Fahrzeug-Steuergeräten über die verschiedenen Modelle hinweg wiederholen, lässt sich der Modellierungsaufwand aber durch den Aufbau einer Bibliothek für mathematische Modelle dieser Bestandteile auf Grund entstehender Synergien reduzieren.

Komplexe Vorgänge Für komplexe Vorgänge, wie beispielsweise Verbrennungsprozesse im Motor, liegt eine akkurate Beschreibung mittels einfacher Differentialgleichungen, die im Rahmen einer Software-Funktion eines Fahrzeug-Steuergeräts gelöst werden können, nicht vor. In solchen Fällen ist der Einsatz eines mathematischen Modells nicht möglich. Datenbasierte Modelle liefern in diesen Fällen basierend auf den verwendeten Messungen hingegen häufig gute Resultate.

Numerische Herausforderung Insbesondere bei steifen mathematischen Modellen kommt es beim Einsatz in Echtzeit-Anwendungen auf Fahrzeug-Steuergeräten zu erheblichen numerischen Herausforderungen. Dieser Herausforderung widmet sich vorliegende Arbeit. Die Schwierigkeiten werden in Kapitel 3 ausführlich vorgestellt.

Symbiose der Modellierungsansätze

Für die Frage, ob datenbasierte oder mathematische Modelle eingesetzt werden, erscheint nicht eine pauschale Empfehlung angemessen. Eine Kombination der Stärken beider Modellierungsweisen ist wünschenswert.

Hinsichtlich Eigenschaften wie der möglichen Parameteranzahl, der Extrapolationsfähigkeit, dem Parametrierungsaufwand und der Modellierung komplexer Vorgänge ergänzen sich datenbasierte und mathematische Modelle in ihren Qualitäten. Aus diesem Grund ist es naheliegend, auf beide Modellierungsarten zurückzugreifen, um die Defizite der einen Art durch den Einsatz der anderen Art vermeiden zu können.

Ein gängiges Beispiel für die vorteilhafte Symbiose der Modellierungsweisen findet sich bei der Modellierung von Verbrennungsabläufen: Die grundlegende Dynamik der beteiligten Gasmassen, dem Druck und der Temperatur lässt sich zweckmäßig durch ein mathematisches Modell

beschreiben. Für die Wärmefreisetzung hingegen ist der Einsatz betriebspunktabhängiger Vibe-Funktionen und damit eines datenbasierten Modells sinnvoll (vgl. [82]).

Gemischte Modelle lassen sich oft in Form einer Differentialgleichung beschreiben, bei der die Koeffizienten der Gleichung durch ein datenbasiertes Modell berechnet werden. Sofern der Einfluss der physikalischen Beziehungen bei einem solchem Modell dominiert, ist im Bezug auf die Simulation des Modells kein allzu großer Unterschied zu einem rein mathematischen Modell gegeben. Aus diesem Grund werden derartige Modelle in vorliegender Arbeit ebenfalls als mathematisch aufgefasst.

2.1.5 Steifheit bei mechatronischen Systemen

Der Begriff der Steifheit einer gewöhnlichen Differentialgleichung ist im Kontext mechatronischer Systeme von großer Bedeutung. Zum einen ist diese Eigenschaft bei mechatronischen Systemen relativ häufig und in einem gewissen Sinne sogar in der Natur der Systeme. Dies wird am Ende des Abschnitts heuristisch erläutert. Zum anderen ist es eine Eigenschaft, die für die Wahl des Lösungsverfahrens essentiell ist. Hier ist auf die historische Entdeckung des Begriffs selbst hinzuweisen:

Der Begriff der Steifheit wurde erst 1952 durch Curtiss und Hirschfelder [26] eingeführt. Sie haben festgestellt:

“ [...] a type of differential equation arises which is exceedingly difficult to solve by ordinary numerical procedures.” [26]

Das Lösen eines Typs von Differentialgleichungen mit gewöhnlichen numerischen Verfahren gestalte sich also als herausragend schwierig. Die Schwierigkeit äußert darin, dass bei der Lösung der Differentialgleichung mit expliziten Integrationsverfahren Instabilität anstatt Konvergenz eintritt. In [26] wird der Begriff der Steifheit eingeführt und charakterisiert und außerdem die BDF-Verfahren als geeignete numerische Integrationsverfahren vorgestellt. Die Arbeit war Ausgangspunkt für eine weitergehende Untersuchung und Entwicklung von impliziten Integrationsverfahren. Die Schwierigkeiten, die in diesem historischen Beispiel beobachtet werden, können als mahnendes Exempel dienen und dem Entwickler von Echtzeit-Anwendungen, die die Simulation mechatronischer Systeme umfassen, vor Augen führen, dass es im Sinne einer zuverlässigen Simulation wesentlich ist, sich des Begriffs der Steifheit bewusst zu sein und echtzeitfähige stabile Integrationsverfahren zur Verfügung zu haben.

Die Steifheit einer gewöhnlichen Differentialgleichung ist eine weich definierte Eigenschaft und als kontextabhängig anzusehen. Für sie gibt es verschiedene Charakterisierungen. Im Folgenden wird zunächst eine Charakterisierung dargestellt, die sich weitgehend an [32] orientiert:

Hierbei werden die Begriffe der intervallweisen und der diskreten Kondition eines Anfangswertproblems

$$\dot{x}\left(t\right) = f\left(t, x\right), \quad x\left(t_0\right) = x_0$$

verwendet. Die intervallweise Kondition ermöglicht eine Abschätzung des Einflusses einer Störung des Anfangsvektors auf den Verlauf des Zustandsvektors. Sie wird formal definiert als

$$\kappa\left[t_0; t\right] = \inf\left\{\tilde{\kappa} \in \mathbb{R} \,\middle|\, \lim_{\bar{x}_0 \to x_0,\, \bar{x}_0 \neq x_0} \frac{\left\|x\left(s\right) - \bar{x}\left(s\right)\right\|}{\left\|x_0 - \bar{x}_0\right\|} \leq \tilde{\kappa} \text{ für alle } s \in \left[t_0; t\right]\right\}.$$

Die diskrete Kondition schätzt den Einfluss einer Störung des Anfangsvektors auf die Werte ab, die bei Integration mit dem expliziten Euler-Verfahren erhalten werden. Für eine formale Definition werden die numerischen Werte der Integration des betrachteten Anfangswertproblems mit dem expliziten Euler-Verfahren auf folgende Weise beschrieben: Sei $\Gamma \subseteq I$ ein äquidistantes Gitter mit $t_0 \in \Gamma$ und Schrittweite Δt. Ferner sei $x_\Gamma : \Gamma \to \mathbb{R}^n$ eine Gitterfunktion mit

$$x_\Gamma\left(t + \Delta t\right) = x_\Gamma\left(t\right) + \Delta t f\left(t, x_\Gamma\left(t\right)\right) \quad \text{für } t \in \Gamma.$$

Dann wird die diskrete Kondition des Anfangswertproblems definiert als

$$\kappa_\Gamma [t_0; t] = \inf \left\{ \tilde{\kappa} \in \mathbb{R} \left| \lim_{\bar{x}_0 \to x_0, \bar{x}_0 \neq x_0} \frac{\|x_\Gamma (s) - \bar{x}_\Gamma (s)\|}{\|x_0 - \bar{x}_0\|} \leq \tilde{\kappa} \text{ für alle } s \in \Gamma \cap [t_0; t] \right. \right\}.$$

Mit diesen Begriffen lässt sich die Steifheit charakterisieren. Grundlage für die Definition ist die Tatsache, dass für stetige rechte Seiten $\kappa_\Gamma [t_0; t] \to \kappa [t_0; t]$ für $\Delta t \to 0$ gilt.

Ein Anfangswertproblem wird als steif bezeichnet, wenn erst für sehr kleine Schrittweiten Δt die Beziehung $\kappa_\Gamma [t_0; t] \approx \kappa [t_0; t]$ gilt. Eine gewöhnliche Differentialgleichung $\dot{x}(t) = f(t, x)$ werde als steif bezeichnet, wenn es einen Anfangswert (t_0, x_0) gibt, so dass das zugehörige Anfangswertproblem steif ist.

Ein Anfangswertproblem heißt dissipativ wenn es ein $\nu \leq 0$ gibt, so dass gilt:

$$\langle f(t, x_2) - f(t, x_1), x_2 - x_1 \rangle \leq \nu \|x_2 - x_1\|^2 \quad \text{für alle } (t, x_1), (t, x_2) \in I \times D.$$

Für ein dissipatives Anfangswertproblem bedeutet obige Definition von Steifheit für $t \to \infty$ im Wesentlichen, dass es steif ist, wenn das explizite Euler-Verfahren erst für sehr kleine Schrittweiten Δt stabil ist, also die Menge $\{\|x_\Gamma (t)\| \, | t \in \Gamma\}$ beschränkt ist.

Weitere gängige Charakterisierungen von Steifheit sind:

- Sei die rechte Seite f für Punkte auf der Lösungstrajektorie zum Zeitintervall $[t_0; t_f]$ Lipschitz-stetig im zweiten Argument zur Lipschitz-Konstante L. Dann wird das zugehörige Anfangswertproblem gemäß [110] als steif bezeichnet, wenn gilt

$$(t_f - t_0) L \gg 1.$$

- Für steife Anfangswertprobleme wird in [58] mit Hinweis auf [26] folgende Charakterisierung angegeben:
  "stiff equations are equations where certain implicit methods [...] perform better [...] than explicit ones."
  Steife Anfangswertprobleme seien also dadurch gekennzeichnet, dass implizite Integrationsverfahren besser geeignet sind als explizite Integrationsverfahren.
- Für lineare stabile autonome Systeme $\dot{x}(t) = Ax(t)$ ist die Steifheit durch einen betragsmäßig großen Quotient von Eigenwertbeträgen gekennzeichnet, wenn A also Eigenwerte $\lambda_{\min}$ und $\lambda_{\max}$ besitzt mit

$$\frac{|\lambda_{\max}|}{|\lambda_{\min}|} \gg 1.$$

Im Folgenden wird heuristisch erläutert, weshalb mechatronische System zur Steifheit neigen:

Ein mathematisches Modell tendiert zur Steifheit, wenn Effekte inhärent sind, die in unterschiedlichen Zeitskalen ablaufen. Bei mechatronischen Systemen ist die Beteiligung unterschiedlicher physikalischer Domänen kennzeichnend. Beispielsweise können Vorgänge aus Mechanik, Pneumatik, Elektrik oder Hydraulik involviert sein. Die Vorgänge der unterschiedlichen Domänen laufen teilweise in dem Sinne in stark unterschiedlichen Zeitskalen ab, dass beispielsweise Zeitkonstanten eines hydraulischen Subsystems häufig um Größenordnungen größer sind als Zeitkonstanten eines mechanischen Subsystems. Natürlich ist dies nicht generell gültig: Beispielsweise hat auch ein Masse-Federdämpfer-Systeme mit einer hinreichenden großen Federhärte eine entsprechend große Zeitkonstante, die auch über den Zeitkonstanten von vielen hydraulischen Systemen liegt. Eine typische Konstellation für ein steifes Systems ist beispielsweise, dass die Zustände mechanische Größen sind und im Modell hydraulische oder andere hochdynamische Effekte auftreten.

Die Beteiligung von schnellen und langsamen Effekten äußert sich bei einer Linearisierung des Systems in betragsmäßig entsprechend unterschiedlich großen Eigenwerten, so dass die linearisierten Modelle nach obiger Charakterisierung steif sind. Dies gilt nahe des Linearisierungspunktes dann auch für das System selbst.

Im Kontext von Echtzeitsimulation mathematischer Modelle auf Fahrzeug-Steuergeräten sind Rahmenbedingungen gegeben, so dass sich auch eine weitere Charakterisierung der Steifheit anbietet:

Zum einen wird hierbei davon ausgegangen, dass die Aufrufintervalle des Software-Tasks, der die Simulation des mathematischen Modells umfasst, weitgehend vorgegeben ist. Beispielsweise ist bei einer Software-Funktion zur Zündwinkelsteuerung das Aufrufintervall in einer Größenordnung von wenigen Millisekunden, während bei einer Software-Funktion zur Diagnose der Sauerstoff-Speicherfähigkeit eines Katalysators auch ein Aufrufintervall im Bereich von Zehntelsekunden ausreichend ist. Da kleinere Aufrufintervalle zu mehr Verbrauch an Rechenkapazität führen würden und sich dies als Kostenfaktor darstellt, ist eine Verringerung der Aufrufintervalle aus numerischen Gründen nicht wünschenswert.

Zum anderen wird davon ausgegangen, dass durch die Modellierung sichergestellt ist, dass keine hochdynamischen Systeme in Tasks mit im Verhältnis dazu zu großen Aufrufintervallen verwendet werden.

Da die Systeme auch als stabil angenommen werden, führt dies insgesamt zu folgender Charakterisierung: Ein mathematisches Modell, das in einem Software-Task auf einem Fahrzeug-Steuergerät simuliert wird, wird in vorliegender Arbeit als steif bezeichnet, wenn es bei Integration mit dem expliziten Euler-Verfahren und einer Zeitschrittweite entsprechend dem Aufrufintervall instabil ist. Die Fälle, bei denen bereits eine geringe Reduzierung der Zeitschrittweite zur Stabilität der Integration führt, stellen hierbei eine Grauzone dar.

2.1.6 Durchgängigkeit bei modellbasierter Entwicklung

Als mechatronisches System wird wie in der VDI-Richtlinie 2206 [2] beschrieben, ein System angesehen, bei dem "das enge Zusammenwirken von Maschinenbau, Elektrotechnik und Informationstechnik" kennzeichnend ist (vgl. [2]). Diese VDI-Richtlinie befasst sich mit der Entwicklungsmethodik für mechatronische System und sieht bei mechatronischen Systemen eine im Vergleich zu mechanischen Systemen hohe Komplexität (vgl. [2]).

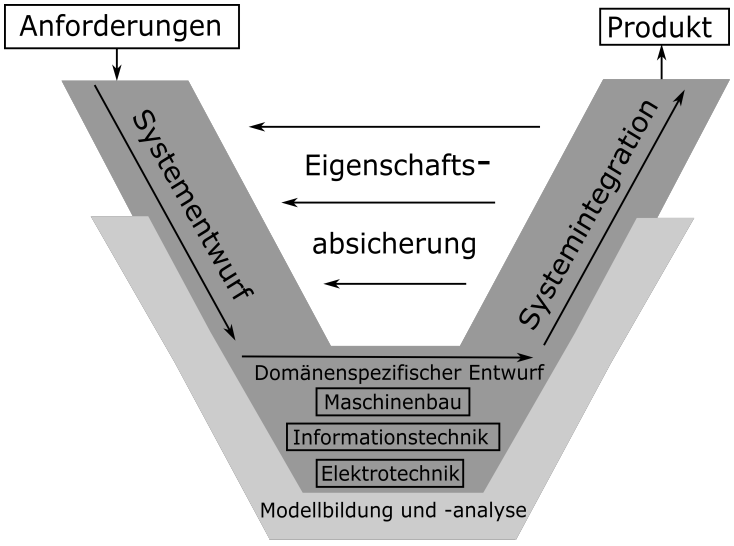


Abbildung 2.1: V-Modell als Makrozyklus gemäß VDI-Richtlinie 2206 aus [2].

Um in diesem Kontext effizient zu entwickeln, wird durch die VDI-Richtlinie 2206 eine Entwicklung nach dem V-Modell wie in Abbildung 2.1 gezeigt empfohlen. Hierdurch wird die unter Umständen kaum überschaubare Komplexität der gesamten Entwicklungsaufgabe auf Teilschritte heruntergebrochen, die jeweils von einem Team bewältigbar sind. Hierbei wird ein iteratives Anwenden des V-Modells vorgeschlagen (vgl. [2]), um beispielsweise Fehlverhalten beim Zusammenspiel der verschiedenen Systembestandteile bereits frühzeitig erkennen zu können. Für

eine zeit- und kostensparende Entwicklung ist die modellbasierte Entwicklung, also das Entwickeln und Testen mit Hilfe von Modellen, wesentlich. Diese ermöglicht Entwicklungszeiten zu verkürzen, indem bereits vor der Verfügbarkeit realer Prototypen Produkteigenschaften überprüft werden können. Zudem kann die Zahl realer Prototypen reduziert werden, wodurch die entsprechenden Kosten für Material und Fertigung eingespart werden.

Um den Aufwand für die Eigenschaftsabsicherung im V-Modell über die verschiedenen Iterationen hinweg gering zu halten, ist beim Einsatz von Modellen die Durchgängigkeit der Modelle nützlich. Dies wird am Beispiels eines Modells für ein Rohrstück veranschaulicht, in dem Massen- und Enthalpiestrom abgebildet sind:

Ein solches Rohrmodell kommt zum einen während der modellbasierten Entwicklung einer Diagnose-Funktion auf einem Motor-Steuergerät zum Einsatz, die die Sauerstoff-Speicherfähigkeit eines Katalysators im Abgassubsystem eines Diesel-Luftsystems überwacht. Zum anderen kann ein solches Modell auch während der Tests eines Steuergeräts mit dieser Diagnose-Funktion durch ein Hardware-in-the-Loop System eingesetzt werden. Schließlich kann zudem die Software-Funktion zur Diagnose der Sauerstoff-Speicherfähigkeit eines Katalysators selbst ein solches Modell einsetzen. Um Synergien bei der Verifikation und Validierung der Rohrmodelle zu schaffen, ist der Einsatz durchgängiger Rohrmodelle notwendig. Idealerweise kann ein und dasselbe Rohrmodell für alle drei Zwecke eingesetzt werden, so dass der Aufwand für die Verifikation und Validierung bis auf den Einfluss der unterschiedlichen Ziel-Hardware nur einmal aufgebracht werden muss. Zudem werden Fehler vermieden, die auf der Unterschiedlichkeit der eingesetzten Rohrmodelle beruhen.

Im Sinne der Durchgängigkeit ist es sinnvoll, den Einsatz von Modellen, wie sie bei der Entwicklung von mechatronischen Systemen Standard geworden sind und beim Testen von Steuergeräten durch Hardware-in-the-Loop Systemen ebenfalls eingesetzt werden, auch auf Fahrzeug-Steuergeräten zur Verfügung zu haben. Dies ist derzeit für eine wichtige Klasse von Modellen noch Forschungsgegenstand: Für steife Modelle stellt eine echtzeitfähige und stabile Simulation auf einem Fahrzeug-Steuergerät ein numerisch anspruchsvolles Problem dar. Am Beispiel eines Rohrmodells wird in dieser Arbeit untersucht, wie durch geeignete numerische Methoden die Echtzeitfähigkeit der Simulation eines steifen Modells auf einem Steuergerät erzielt werden kann.

2.1.7 Beschreibung der Problemklasse

Die in dieser Arbeit untersuchten Vorgehensweisen zur echtzeitfähigen Simulation auf Fahrzeug-Steuergeräten zielen auf nicht-lineare steife Modelle, wie sie für mechatronische Systeme im Rahmen von Echtzeit-Anwendungen auf Steuergeräten zum Tragen kommen. Die Attribute, die sich bei der mathematischen Beschreibung dieser Modelle regelmäßig feststellen lassen, werden im Folgenden dargestellt.

Systemstruktur

Ab mehreren Zuständen sind die gewöhnlichen Differentialgleichungen der Modelle strukturell dünn besetzt in dem Sinne, dass die Jacobi-Matrizen $J_f(t, x)$ dünn besetzt sind und eine Substruktur erkennen lassen.

Die gewöhnlichen Differentialgleichungen bestehen aus Subsystemen mit einer relativ geringen Zahl an Querabhängigkeiten. Für diese innere Struktur gibt es zwei Ursachen:

1. Das reale System selbst kann aus Subsystemen und Komponenten aufgebaut sein. Dies ist bei mechatronischen Systemen sehr typisch. In Abbildung 2.2 ist der Plan eines Pkw-Luftsystems abgebildet. Eine natürliche Untergliederung ist in Frischluft- und Abgas-Subsystem. Diese Subsysteme interagieren ausschließlich über den Motor, die Abgasrückführungen und die Turbolader. Das Abgas-Subsystem ist wiederum in natürlicher Weise untergliederbar in Hochdruck- und Niederdruck-Bereich. Strukturell betrachtet sind diese

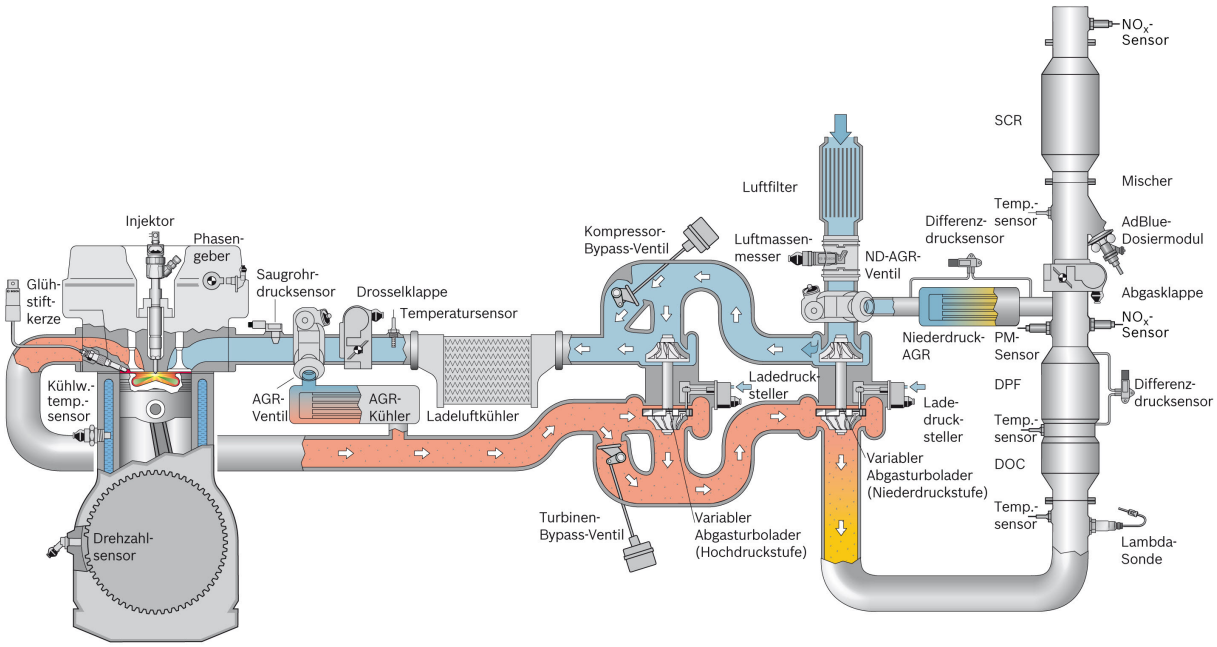


Abbildung 2.2: Luftsystemplan PKW aus [85].

sequentiell angeordnet, so dass eine direkte Interaktion lediglich an den Schnittstellen gegeben ist. Indirekte Abhängigkeiten ergeben sich über das Frischluft-Subsystem. Eine weitere Unterteilung bis auf Komponentenebene ist durch den sequentiellen Aufbau in sinnvoller Weise möglich.

2. Die unterschiedlichen beteiligten physikalischen Domänen bilden weitgehend eigenständige und nur wenig gekoppelte Subsysteme. Ein Beispiel hierfür ist das Gesamtsystem eines Pkw-Modells, das unter anderem ein Luftsystemmodell, ein Motormodell und ein Antriebstrangmodell enthält. Diese entsprechen einer Unterteilung nach den physikalischen Domänen Pneumatik, Verbrennung und Mechanik. Verbindungen zwischen diesen Bestandteilen sind nur gegeben durch die Ein- und Auslassventile des Motors, die das Luftsystem mit dem Motor verbinden, sowie über die Kurbelwelle, die den Motor mit dem Antriebstrang verbindet. Insgesamt ergibt sich also auch in diesem Fall eine Struktur, die sich aus drei Bestandteilen mit relativ wenigen Querabhängigkeiten zusammensetzen.
3. Sofern eine gewöhnliche Differentialgleichung durch eine örtliche Semidiskretisierung einer partiellen Differentialgleichung zu Stande kommt, weist diese ebenso eine innere Struktur auf und die Jacobi-Matrizen sind typischerweise dünn besetzt.

Systemgröße

Neben der Struktur ist die Größe der Systeme ein Charakteristikum. Als Stand der Technik für steife Modelle, die auf Fahrzeug-Steuergeräten eingesetzt werden können, wird eine Systemgröße bis zu etwa vier Zuständen angesehen werden (vgl. [17, 116]). In vorliegender Arbeit sind Systeme mit mehr als vier und hin zu 75 Zuständen in der Betrachtung. Motivation hierfür ist zum einen, dass steife Systeme dieser Größe jenseits des Standes der Technik liegen. Zum anderen führt dies auch zu wesentlichen zusätzlichen Herausforderungen: In dieser Größenordnung ist der Einsatz numerischer Verfahren für Teilaufgaben wie Differenziation oder Gleichungslösung nötig. Denn ein Zurückgreifen auf explizite Terme oder Umformungsschritte, die während einer Analyse in der Entwicklungsphase bestimmt werden, ist bei dieser Systemgröße in vielen Fällen nur noch wenig praktikabel.

Systeme mit mehr als 75 Zuständen führen zu einer Verschiebung der Gewichtung der Herausforderungen und bringen zudem noch weitere Herausforderungen mit sich. Beispielsweise wird

bei Systemen dieser Größe neben der Rechenzeit auch der Speicherbedarf zunehmend zu einem Flaschenhals. Untersuchungen, die hierauf eingehen, erscheinen dann sinnvoll, wenn Systeme bis 75 Zuständen beherrscht werden.

Somit stellt der betrachtete Bereich die Größe von steifen Modellen dar, die den nächsten Schritt für den Einsatz von steifen Modellen auf Fahrzeug-Steuergeräten darstellen.

Aufrufintervall

Für Echtzeit-Anwendungen auf Fahrzeug-Steuergeräten wird in dieser Arbeit angenommen, dass das Aufrufintervall der Software-Funktion vorgegeben ist. Beispielsweise ist bei einer Software-Funktion, die einen aktuellen Sensorwert ausliest und ablegt, das Aufrufintervall in einer Größenordnung von wenigen Millisekunden sinnvoll, während bei einer Software-Funktion zur Diagnose der Sauerstoff-Speicherfähigkeit eines Katalysators auch ein Aufrufintervall im Bereich von Zehntelsekunden ausreichend ist. Je nach Anwendung liegt das Aufrufintervall typischerweise zwischen 1 ms und 1000 ms.

Darüber hinaus wird angenommen, dass das Aufrufintervall weitgehend konstant bleibt. Der Aspekt schwankender Aufrufintervalle ist nicht Schwerpunkt dieser Arbeit. Dies betrifft beispielsweise die Klasse der winkelsynchronen Tasks, bei denen der Aufruf nicht in einem konstanten zeitlichen Abstand erfolgt, sondern dann erfolgt, wenn die Kurbelwelle gewisse Winkel überschreitet. In diesem Fall ist das Aufrufintervall von der Drehzahl der Kurbelwelle abhängig. Ein Beispiel hierfür ist eine Software-Funktion zur Zündwinkelsteuerung. Eine echtzeitfähige Simulation mathematischer Modelle in winkelsynchronen Anwendungen lässt sich dadurch erhalten, dass die Worst Case-Bedingungen, also die Aufrufintervalle samt der darin zur Verfügung stehenden Rechenzeit, wie sie bei einer sehr hohen Drehzahl auftreten, als Rahmen angenommen werden und hierfür eine echtzeitfähige Lösung erarbeitet wird.

Weitere Eigenschaften

Eine weitere verbreitete Eigenschaft von mechatronischen Systemen, die für Anwendungen auf Fahrzeug-Steuergeräten von Interesse sind, ist ihre Stabilität im Sinne von der Beschränktheit der Zustände $x(t)$ für $t \to \infty$. Dies ist aus dem Grund naheliegend, dass ein Zustand mit physikalischer Bedeutung in einem mechatronischen System, der über alle Grenzen wächst, ein klarer Indikator für einen unerwünschten Systemverlauf ist und praktisch nicht auftreten sollte, insbesondere wenn berücksichtigt wird, welche Konsequenzen sich hierbei allein durch die Speicherung als Maschinenzahl beispielsweise im Gleitkommaformat gemäß [4] ergibt. Idealerweise ist dies durch die Beschaffenheit des Systems sichergestellt und spiegelt sich im mathematischen Modell in Form von Stabilität wider.

Für die vorliegende Arbeit ist darüber hinaus die Annahme sinnvoll, dass die Systeme nicht ungedämpft sind. Der Grund hierfür liegt darin, dass in dieser Arbeit implizite Verfahren betrachtet werden, die eine dämpfende Wirkung haben und somit für den Einsatz bei ungedämpften Vorgängen wenig geeignet sind. Hierfür sind andere Integrationsverfahren wie geometrische Integratoren [56] geeignet, die hier nicht betrachtet werden.

Desweiteren wird davon ausgegangen, dass durch die Modellierung sichergestellt ist, dass die mathematischen Modelle in Relation zum Aufrufintervall der Software-Funktion, in der sie eingesetzt werden, nicht hoch oszillativ sind. Dies äußert sich darin, dass die Imaginärteile der Eigenwerte $\lambda_i(x(t_n))$ von Linearisierungen des mathematischen Modells nicht groß gegenüber der Aufruffrequenz Δt^{-1} der Softwarefunktion sind im Sinne von $\mathfrak{Im}(\lambda_i(x(t_n)))\,\Delta t \lessapprox 1$. Hoch oszillative Systeme führen zu speziellen zusätzlichen Herausforderungen wie beispielsweise, dass implizite Runge-Kutta Verfahren recht kleine Schrittweiten benötigen, um eine akzeptable Genauigkeit zu liefern. Daher sind in diesem Fall andere Integrationsverfahren wie exponentielle Integratoren von Vorteil [60]. Diese sind in vorliegender Arbeit nicht im Fokus. Stattdessen sind Herausforderungen von steifen, nicht hoch oszillativen Systemen im Fokus.

Darüber hinaus wird angenommen, dass die mathematischen Modelle in Form von gewöhnlichen Differentialgleichungen beschrieben werden und keine Umschaltung für die Differentialgleichungen vorliegt, die basierend auf den Werten des Zustandsvektors oder äußeren Ereignisse zwischen verschiedenen Beschreibungen für das Modell wechselt. Herausforderungen wie die Bestimmung eines Nulldurchgangs, auf Englisch mit *zero crossing detection* bezeichnet, erfordert bei einem Einsatz unter Echtzeit-Bedingungen gesonderte Maßnahmen (vgl. [75]).

Eine Eigenschaft, die nicht einheitlich für die Modelle der Problemklasse sind, ist ihre Glattheit: Durch die Beteiligung nicht-glatter Funktionen wie Beträge, Minimumsfunktion oder Treppenfunktionen können die rechten Seiten der Differentialgleichungen nicht-glatt sein. Da sie zudem von Eingangsdaten abhängen, die im Rahmen der Lösung der Differentialgleichung nur zu diskreten Zeitpunkten verändert werden, ist zu beachten, dass die rechten Seiten der Differentialgleichungen über diese Abtastzeitpunkte hinweg in der Regel nicht stetig sind. Daneben gibt es aber auch Modelle, deren Differentialgleichungen eine glatte rechte Seite aufweisen. In vorliegender Arbeit werden Systeme betrachtet mit einer rechten Seite, die höchstens in den Abtastzeitpunkten nicht stetig und zudem fast überall glatt im Sinne von beliebig oft differenzierbar ist.

2.2 Leitanwendungsmodell

Als Leitanwendung dieser Arbeit dient ein Modell für ein Rohrstück. Dieses ist schematisch in Abbildung 2.3 dargestellt. Es beschreibt den Massenstrom, aufgegliedert in mehrere Gasfraktionen, und den Enthalpiestrom durch ein Rohr. Die Modellierung wird aus [46] übernommen und als gegeben betrachtet. Die Modellierung ist nicht Gegenstand der Diskussion vorliegender Arbeit.

2.2.1 Einsatzzweck des Leitanwendungsmodells

Für ein mathematisches Rohrmodell, das den Massen- und Enthalpiestrom im Rohr beschreibt, gibt es mehrere Software-Funktionen auf einem Fahrzeug-Steuergerät, in dem es zweckmäßig eingesetzt werden kann. Im Folgenden wird der Einsatz im Rahmen einer Überwachung der Temperatur eines Katalysators skizziert.

Eine Temperaturüberwachung kommt wiederum bei verschiedenen Anwendungen zum Tragen. Hier sind beispielsweise eine Diagnosefunktion der Sauerstoff-Speicherfähigkeit eines Katalysators und eine Funktion zum Bauteilschutz des Katalysators zu nennen.

Der Katalysator hat seine volle Konvertierfähigkeit lediglich innerhalb eines beschränkten Temperaturfensters. Insbesondere ist er nach einem Kaltstart des Motors zunächst nicht aktiv. Die Diagnosefunktion für die Sauerstoff-Speicherfähigkeit eines Katalysators soll daher erst im aktiven Bereich des Katalysators wirksam werden, da vorher eine geringe Sauerstoff-Speicherfähigkeit keinen Fehler darstellt. Für die Temperatur des Katalysator sind drei Faktoren dominant:

1. Wärme-Eintrag durch das eingehende Rohrstück
2. Wärme-Austrag durch das ausgehende Rohrstück
3. Exotherme Umsetzung im Katalysator

Im Rahmen der Aufwärmphase bei einem Kaltstart ist die exotherme Umsetzung im Katalysator durch seine Inaktivität vernachlässigbar. Für die anderen beiden Punkte ist der Einsatz eines mathematischen Modells für ein Rohrstück zweckmäßig, um die Temperatur des Katalysators und somit den Aktivitätsbeginn des Katalysators zu ermitteln. Mit dem Aktivitätsbeginn des Katalysators hat dann die Diagnose der Sauerstoff-Speicherfähigkeit zu starten.

Auch bei hohen Temperaturen des Katalysators ist die Ermittlung seiner Temperatur relevant, um im Rahmen einer Funktion zum Bauteilschutz des Katalysators ein Überhitzen zu verhindern. In diesem Temperaturbereich ist die exotherme Umsetzung im Katalysator zentral für den Temperaturverlauf. Hierzu ist der Anteil an Kohlenwasserstoffen, Kohlenmonoxid und Stickoxiden im

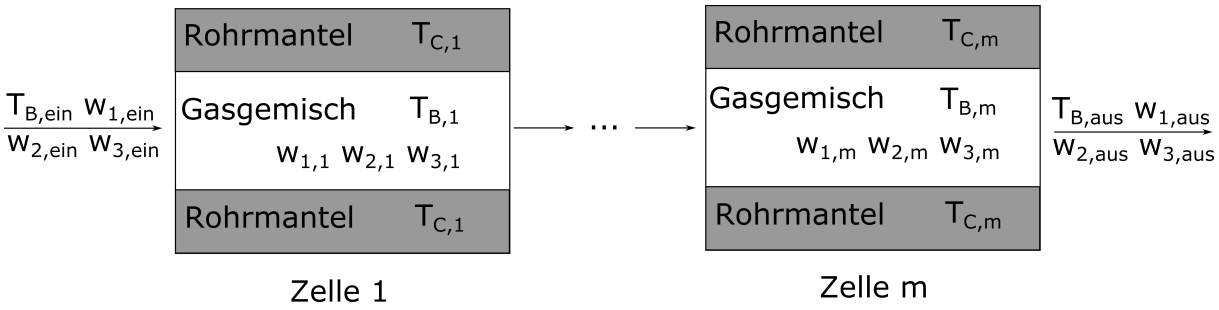


Abbildung 2.3: Schema des betrachteten Rohrmodells mit m Diskretisierungszellen.

einströmenden Gas signifikant. Aus diesem Grund ist es essentiell, dass im oben beschriebenen Rohrmodell die Massenanteile dieser Stoffe separat erfasst sind.

2.2.2 Modellbeschreibung

Das Leitanwendungsmodell beschreibt den Massen- und Enthalpiestrom durch ein Rohr. Es berücksichtigt die Gaszusammensetzung dahingehend, dass der Massenanteil w_1 der Kohlenwasserstoffe, der Massenanteil w_2 von Kohlenmonoxid sowie der Massenanteil w_3 der Stickoxide separat erfasst werden. Neben diesen Größen bilden die Temperatur im Rohrinneren T_B und die Temperatur des Rohrmantels T_C die Zustände, die sich je Diskretisierungszelle ergeben.

Für die Enthalpiebilanz werden berücksichtigt:

- Konvektion im Gasgemisch
- Konvektion zwischen Gasgemisch und Rohrmantel
- Wärmeleitung im Rohrmantel
- Wärmeabstrahlung vom Rohrmantel in die Umgebung

Darüber hinaus ist ein Druckabfall modelliert. Insgesamt lassen sich die Modellgleichungen in Form nachstehender partiellen Differentialgleichung charakterisieren. Eine Liste der verwendeten Kürzel ist im Anhang angegeben.

$$\frac{pm_{\mathrm{mol}}}{RT_B}\frac{\partial w_j}{\partial t} = -\frac{\dot{M}}{A_B}\frac{\partial w_j}{\partial z}, \quad j = 1,\, 2,\, 3, \tag{2.1}$$

$$\frac{p}{T_B}\left(\frac{M_{\mathrm{mol}}c_{p,B}}{R} - 1\right)\frac{\partial T_B}{\partial t} = -\frac{1}{A_B}\left(\dot{M}c_{p,B}\frac{\partial T_B}{\partial z} + \alpha_{B,C}l_B\left(T_B - T_C\right)\right), \tag{2.2}$$

$$\begin{aligned}\rho_C c_C \frac{\partial T_C}{\partial t} = {} & \lambda_C\frac{\partial^2 T_C}{\partial z^2} + \frac{1}{A_C}\Big(\alpha_{B,C}l_B\left(T_B - T_C\right) - \\ & \alpha_{A,C}l_C\left(T_C - T_A\right) - \epsilon_C l_C\sigma\left(\left(T_C\right)^4 - \left(T_A\right)^4\right)\Big).\end{aligned} \tag{2.3}$$

Diese partielle Differentialgleichung wird nach der Linienmethode (vgl. [35]), auf Englisch mit *method of lines* bezeichnet, und mit Verwendung von Rückwärtsdifferenzen semidiskretisiert. Diese Semidiskretisierungen der partiellen Differentialgleichung sind Grundlage der Untersuchungen vorliegender Arbeit. Sie werden als gegeben betrachtet und stellen geeignete Repräsentanten für die untersuchte Modellklasse dar (vgl. Abschnitt 2.1.7). Dass eine partielle Differentialgleichung zu Grunde liegt, ist ein Umstand, der im Rahmen der Untersuchungen nicht herangezogen wird.

Der Druckabfall ist auf folgende modelliert: Der Druck p_i in der Diskretisierungszelle i wird bei

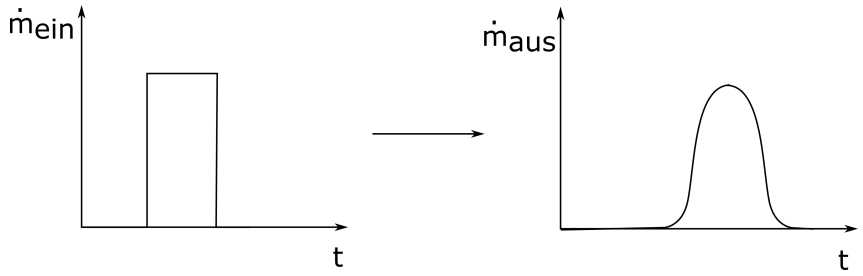


Abbildung 2.4: Dispersion im Rohr schematisch dargestellt am Beispiel eines Verlaufs des Masseneinstroms in Treppenform.

einer Zellanzahl m auf folgende Weise berechnet:

$$p_i = \sqrt{p_{\text{ein}}^2 - \frac{\dot{m}_{\text{ein}} f R T_B}{A_B^2 m_{\text{mol}} d_{h,B}} \frac{i}{m} l}. \tag{2.4}$$

Für den Massenstrom wird Quasistationarität angenommen, also dass der Masseneinstrom und -ausstrom jeder Zelle gleich ist und dass gilt:

$$\dot{m}_i = \dot{m}_{\text{ein}}, \quad i = 1, \ldots, m. \tag{2.5}$$

Die wesentliche Komplexität der Modellgleichung liegt in der Berechnung der physikalischen Koeffizienten. Diese wird zu einem großen Teil anhand physikalischer Abhängigkeiten und nur zum einem geringen Teil anhand von datenbasierten Methoden vorgenommen. Sie wird im Unterabschnitt 2.2.2 beschrieben.

Wie in [106] wird im Folgenden im Kontext eines Massenstroms durch ein Rohr die Veränderung des Verlaufs des Massenausstroms aus dem Rohr gegenüber dem Verlauf des Masseneinstroms in das Rohr als Dispersion oder physikalische Dispersion bezeichnet, wobei zeitliche Verschiebungen der Verläufe vernachlässigt werden. Die Dispersion ist ein typischer Effekt bei einem Gasmassenstrom durch ein Rohr, da der Verlauf des Gasmassenausstroms aus einem Rohr eine Glättung des Verlaufs des Gasmasseneinstroms darstellt. Beispielhaft ist dies in Abbildung 2.4 illustriert. Das Rohrmodell ist in der Lage die Dispersion widerzuspiegeln. Hierfür wird folgender Umstand genutzt: Bei der Semidiskretisierung obiger partieller Differentialgleichung tritt numerische Diffusion auf. Diese wird genutzt, um die physikalische Dispersion zu imitieren (vgl. [106]). Die Intensität der numerischen Diffusion hängt von der Anzahl an Diskretisierungszellen ab. Diese Anzahl wird als Applikationsparameter genutzt, so dass die numerische Diffusion auf das gewünschte Maß eingestellt und so die physikalische Dispersion abgebildet werden kann. Hierdurch entsteht die Notwendigkeit, das Rohrmodell für verschiedene Anzahlen an Diskretisierungszellen zu beherrschen. Im weiteren Verlauf der Arbeit wird das Rohrmodell mit fünf bis 15 Diskretisierungszellen betrachtet.

Die Parameter des Modells sind entweder leicht ermittelbar oder werden aus den Zustandsgrößen berechnet. Geometrische Parameter wie Rohrlänge, Querschnittsfläche oder Rohrdicke sind gut bestimmbar und Materialeigenschaften wie die Dichte des Rohrmantels oder seine Wärmeleitfähigkeit sind ebenfalls gut zugänglich. Für die verbleibenden Parameter Rohrreibungszahl und Wärmeübergangskoeffizient von Gasgemisch zu Rohrmantel werden die großteils physikalischen Beziehungen des nachstehenden Unterabschnitts genutzt, die diese in Abhängigkeit der Zustände beschreiben. Insgesamt hat das Modell hierdurch einen geringen Parametrieraufwand.

Physikalische Koeffizienten

Für die Berechnung verschiedener physikalischer Koeffizienten werden physikalische Beziehungen verwendet, bei denen die Koeffizienten in Abhängigkeit der Zustände des Modells dargestellt

werden. Diese Abhängigkeiten müssen nicht für ein einzelnes Modell separat zusammengestellt werden, sondern können in einer Bibliothek gesammelt werden, so dass diese mit geringem Aufwand verfügbar sind.

Die Berechnung der Koeffizienten basiert großteils auf physikalisch recht allgemeinen Beziehungen und ist dadurch für ein breites Spektrum an Umständen einsetzbar. Beispielsweise ist sowohl laminare als auch turbulente Strömung umfasst und der Einsatz des Modells somit nicht auf Strömungen mit geringen oder großen Reynolds Zahlen eingeschränkt.

Nicht physikalisch, sondern datenbasiert sind folgende Größen berechnet: die Wärmeleitfähigkeit λ sowie die dynamische Viskosität η des durchströmenden Gases und die spezifischen Wärmekapazitäten $c_{p,j}$ verschiedener Gasfraktionen.

Eine Konsequenz aus der Zustandsabhängigkeit der Koeffizienten ist, dass beim Einsatz impliziter Integratoren die Koeffizienten zu zusätzlichen Querabhängigkeiten zwischen den Zuständen führen, die allerdings auch physikalisch bestehen und bei einer kennfeldbasierten Berechnung der Koeffizienten vernachlässigt sind. Darüber hinaus führt die Zustandsabhängigkeit der Koeffizienten zusammen mit der Komplexität ihrer Berechnung zu einer entsprechenden Komplexität der Jacobi-Matrix $J_f(t,x)$.

Zur Berechnung der Rohrreibungszahl werden bei einer Diskretisierung mit m Zellen folgende Beziehungen verwendet:

$$\begin{cases} f = \frac{64}{Re} & , Re \leq 2320, \\ f = \frac{0.3164}{\sqrt[4]{Re}} & , 2320 < Re \leq 10^5, \\ \frac{1}{\sqrt{f}} = 2\log\left(Re\sqrt{f}\right) - 0.9 & , 10^5 < Re \leq 10^7, \end{cases} \tag{2.6}$$

$$Re = \frac{\dot{M} d_{h,B}}{A_B \eta}, \tag{2.7}$$

$$T = \frac{T_{B,1} + T_{B,m}}{2}. \tag{2.8}$$

Um die Übergänge zu glätten, wird in den Übergangsbereichen $[a;b]$ zwischen den verschiedenen Berechnungsweisen der Rohrreibungszahl jeweils eine Glättung eingesetzt folgender Art:

$$G(x,a,b) = \begin{cases} a + (b-a)\sin\left(\left(2\frac{x-a}{b-a} - 1\right)\frac{\pi}{2}\right) & , a < x < b, \\ a & , x \leq a, \\ b & , x \geq b. \end{cases} \tag{2.9}$$

Für die Berechnung der Wärmeübergangskoeffizienten von Gasgemisch zu Rohr werden zusätzlich zu den Beziehungen (2.6) - (2.8) zur Ermittlung der Rohrreibungszahl sowie Gleichung (2.4) für den Druck folgende Beziehungen verwendet:

$$\mathrm{Pr} = \frac{\eta c_p}{\lambda},$$

$$\mathrm{Nu} = \mathrm{Nu}_{\mathrm{lam}} + \mathrm{Nu}_{\mathrm{tur}},$$

$$\mathrm{Nu}_{\mathrm{lam}} = \begin{cases} \left(\mathrm{Nu}_{\mathrm{lam},1}^3 + 0.7^3 + \left(\mathrm{Nu}_{\mathrm{lam},2} - 0.7\right)^3 + \mathrm{Nu}_{\mathrm{lam},3}^3\right)^{\frac{1}{3}} & , \mathrm{Re} \leq 2300, \\ 0 & , \mathrm{Re} > 2300, \end{cases}$$

$$\mathrm{Nu}_{\mathrm{lam},1} = 3.66,$$

$$\mathrm{Nu}_{\mathrm{lam},2} = 1.077 \left(\frac{\mathrm{RePr}D}{z} \right)^{\frac{1}{3}},$$

$$\mathrm{Nu}_{\mathrm{lam},3} = 0.5 \left(\frac{2}{1 + 22\mathrm{Pr}} \right)^{\frac{1}{6}} \left(\frac{\mathrm{RePr}D}{z} \right)^{\frac{1}{2}},$$

$$\mathrm{Nu}_{\mathrm{tur}} = \begin{cases} 0.0214 \left(\mathrm{Re}^{0.8} - 100 \right) \mathrm{Pr}^{0.4} \left(1 + \left(\frac{D}{L} \right)^{\frac{2}{3}} \right) & , 0.5 < \mathrm{Pr} < 1.5 \text{ und } \mathrm{Re} > 2300, \\ 0.012 \left(\mathrm{Re}^{0.87} - 281 \right) \mathrm{Pr}^{0.4} \left(1 + \left(\frac{D}{L} \right)^{\frac{2}{3}} \right) & , 1.5 < \mathrm{Pr} < 500 \text{ und } \mathrm{Re} > 2300, \\ 0 & , \mathrm{Re} \leq 2300, \end{cases}$$

$$\alpha_{B,C} = \frac{\mathrm{Nu}\lambda}{2r}.$$

Wiederum wird eine Glättung gemäß 2.9 eingesetzt.

Die Wärmekapazität des Gasgemischs ergibt sich in natürlicher Weise aus den spezifischen Wärmekapazitäten und den Massenanteilen der beteiligten Stoffe:

$$c_p = \sum_{j=1}^{6} w_j c_{p,j}.$$

Neben den Zuständen $w_{1,i}$, $w_{2,i}$ und $w_{3,i}$, die die Massenanteile von Kohlenwasserstoffen, Kohlenmonoxid und Stickoxiden im Gasgemisch beschreiben, werden hierbei der Stickstoff-, Sauerstoff- und Kohlendioxid-Anteil berücksichtigt.

2.2.3 Repräsentativität für die Problemklasse

Das vorangehend beschriebene Rohrmodell wird als Leitanwendung dieser Arbeit verwendet. Es ist geeignet, die in Abschnitt 2.1.7 beschriebene Problemklasse gut zu repräsentieren. Hierzu tragen folgende Eigenschaften bei:

- Das System ist nicht-linear und steif.
- Die Struktur des Rohrmodells ist dünn besetzt im dem Sinne, dass die Jacobi-Matrizen $J_f\left(x\left(t\right)\right)$ dünn besetzt sind.
- Das Modell untergliedert sich durch die Zellen der Semidiskretisierung der partiellen Differentialgleichung in Submodelle, die eine relativ geringe Anzahl an Querabhängigkeiten aufweisen.
- Die Systemgröße liegt für die untersuchten Anzahlen an Diskretisierungszellen zwischen 25 und 75 Zuständen.
- Das Aufrufintervall einer Software-Funktion, die dieses Modell simuliert, kann mit 100 ms angenommen werden.

Weitere Eigenschaften, die das Modell charakterisieren und die in der Problemklasse zwar bei vielen, aber nicht bei allen Modell aus der Problemklasse auftreten, sind folgende:

- Die rechte Seite weist eine geringe Glattheit auf: Bei glatten Verläufen der Eingangsdaten $u\left(t\right)$ ist die rechte Seite stetig differenzierbar, aber nicht in C^2. Bei einem nicht stetigen, lokal konstanten Verlauf der Eingangsdaten, was die Rahmenbedingungen bei Steuergeräte-Anwendungen widerspiegelt, ist die rechte Seite nicht stetig, sondern nur stückweise stetig.
- Der Auswertung der rechten Seite ist rechenaufwändig.

Das Rohrmodell besitzt auch wenige Eigenschaften, die im Hinblick auf die übrige Problemklasse als individuelle Eigenschaft des Rohrmodells einzustufen sind und nicht als Repräsentation der

Problemklasse:

- Die Eigenwerte der Jacobi-Matrizen $J_f\left(x\left(t\right)\right)$ sind rein reell.
- Die Struktur der Jacobi-Matrizen weist Substrukturen auf, die untereinander nahezu identisch sind. Diese ergeben sich aus der Semidiskretisierung der partiellen Differentialgleichung.

Das Rohrmodell ist insgesamt als Repräsentant der Problemklasse gut geeignet. Dieser repräsentative Charakter ist Grund für die Untersuchung des Rohrmodells. Dieses dient als Beispiel, anhand dessen gezeigt wird, welche Möglichkeiten für die Echtzeitsimulation steifer Modelle im Rahmen von Anwendungen auf Fahrzeug-Steuergeräten durch den Einsatz geeigneter numerischer Verfahren bestehen.

2.3 Fokus der Arbeit

In vorliegender Arbeit wird am Beispiel eines Rohrmodells als Leitanwendung untersucht, wie für ein steifes Modell eine echtzeitfähige und stabile Simulation auf einem Fahrzeug-Steuergerät durch numerische Mittel erreicht werden kann. Das Leitanwendungsmodell ist in der Ausgangsimplementierung nicht echtzeitfähig rechenbar. Durch eine umfassende Laufzeitreduktion soll Echtzeitfähigkeit erzielt werden. Hierzu werden die wesentlichen Anteile des Rechenaufwands bei Betrieb auf einem Fahrzeug-Steuergerät identifiziert. Die Laufzeitreduktion betrifft die numerischen Teilaufgaben der Lösung der Anfangswertprobleme, die die wesentlichen Rechenzeitanteile beanspruchen. Die Modellierung wird hierbei als gegeben betrachtet und ist nicht Gegenstand der Untersuchungen in vorliegender Arbeit. Die Echtzeitfähigkeit wird durch Untersuchungen der Rechenzeiten auf einem Fahrzeug-Steuergerät ermittelt.

3 Echtzeitsimulation auf Fahrzeug-Steuergeräten

In ersten Abschnitt des Kapitels wird auf die Hardware und die Software-Rahmenbedingungen bei Fahrzeug-Steuergeräten eingegangen. Die numerischen Auswirkungen von Hardware und Software-Rahmenbedingungen werden im zweiten Abschnitt dargestellt.

3.1 Technische Rahmenbedingungen

In modernen Premiumklasse-Automobilen können über hundert Steuergeräte zum Einsatz kommen. Viele essentielle technologische Fortschritte im Automobilbereich sind eng verzahnt mit den Entwicklungen der Steuergeräte und ihrer Software-Funktionen. Beispiele hierfür sind die Reduktion der Schadstoffemission bei Verbrennungsmotoren basierend auf elaborierten Regelungen (vgl. [17, 116]) oder die zunehmenden Automatisierungsmöglichkeiten im Fahrerassistenzbereich. Sowohl hinsichtlich der Hardware wie auch auch der Rahmenbedingungen für die Anwendungssoftware unterscheiden sich Fahrzeug-Steuergeräte weitreichend von üblichen PCs. Hierauf wird im Folgenden eingegangen.

3.1.1 Hardware

Als Testhardware wird in vorliegender Arbeit das Motorsteuergerät MDG1C Device 4 von Bosch verwendet. Bosch ist führend auf dem Gebiet der Fahrzeug-Steuergeräte (vgl. [94]). Die MDG1-Plattform hat ein sehr breites Einsatzgebiet: Sie wird für verschiedenste Fahrzeugkategorien wie Zweiräder, Pkw, Lkw und weitere Nutzfahrzeuge eingesetzt. Als gemeinsamer Nachfolger der Steuergeräte-Plattform ME17 bzw. MED17 für Benzin-Fahrzeuge und der Plattform EDC 17 für Diesel-Fahrzeuge kommt sie sowohl für Diesel- wie auch Benzin- oder Hybrid-Fahrzeuge zum Einsatz. Somit stellt diese Steuergeräte-Plattform ein Musterbeispiel für eine Fahrzeug-Steuergeräte-Plattform dar. Die Mikrocontroller werden von Infineon Technologies und Freescale Semiconductors in gegenseitig austauschbarer Form geliefert (vgl. [94]).

Das Steuergerät MDG1C Device 4 kommt vorrangig in der Premiumklasse im Pkw-Sektor zum Einsatz. Es ist in Abbildung 3.1 gezeigt. Eine schematische Beschreibung ist in Abbildung 3.2 aufgeführt.

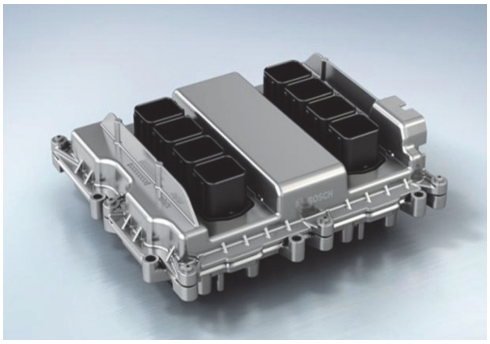


Abbildung 3.1: Bosch Motorsteuergerät MDG1.

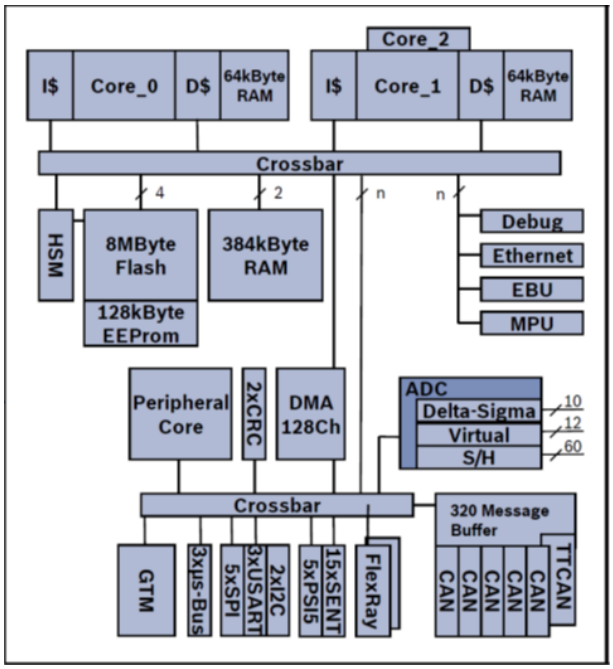


Abbildung 3.2: Schematische Beschreibung der Architektur des Mikrocontrollers Bosch MDG1C Device 4 aus [94].

Der Mikrocontroller verfügt über zwei Hauptkerne und einen Peripheriecontroller. Bei den Infineon-Produkten, die auf dem Mikrocontroller TC29x aus der AURIX^{TM}-Familie basieren, kommen TriCore 1.6P als Hauptkerne und ein TriCore 1.6E als Peripheriecontroller zum Einsatz. Bei den Freescale-Versionen, die auf dem Mikrocontroller MPC5777M basieren, sind die Hauptkerne jeweils ein e200z7 aus der PowerPC-Familie und der Peripheriecontroller ist ein e200z4.

In jeder der Varianten sind die Hauptkerne mit 300 MHz getaktet und der Peripheriecontroller mit 200 MHz. Einer der beiden Hauptkerne hat einen Sicherheitskern im Lockstep Modus, der also dieselben Operationen wie der Hauptkern um wenige Zyklen versetzt ausführt, um die entstehende Redundanz zur Fehlerdetektion zu nutzen. Die Hauptkerne verfügen jeweils über 4 kByte Daten-Cache. Jedem der beiden Hauptkerne stehen zudem 64 kByte an Daten-RAM zur Verfügung. Über eine Crossbar sind sie mit einem gemeinsamen RAM-Speicher von 384 kByte verbunden. Der Mikrocontroller besitzt 8 Mbyte Programmspeicher.

Auf dem Mikrocontroller sind durch eine Gleitkomma-Recheneinheit binäre Gleitkomma-Rechnungen in 32 Bit verfügbar. Für elementare Funktionen wie Exponential-, Logarithmus und die trigonometrischen Funktionen kommen im Rahmen von Software-Funktionen auf dem Steuergerät Software-Realisierungen zum Einsatz.

Um bei Programmabläufen, die von Interrupts geprägt sind, in effizienter Weise eine Vielzahl von Tasks auf einem Kern zu rechnen, sind die Mikrocontroller darauf ausgelegt, zu schnellen Kontextwechseln im Stande zu sein. Die Mikrocontroller können in einem Temperaturbereich von -40°C bis 125°C eingesetzt werden.

Sicherheitsrelevante Funktionen Für sicherheitsrelevante Funktionen eine Ausführung mit höchster Zuverlässigkeit gewährleisten zu können, ist eine wesentliche Anforderung an Fahrzeug-Steuergeräte (vgl. [117]). Die Norm ISO 26262 [109] gibt Anforderungsstufen an elektrische, elektronische und Software-Komponenten vor, die an einer sicherheitsrelevanten Funktion in einem Straßenfahrzeug beteiligt sind. Die Funktionen in einem Fahrzeug werden hierbei entsprechend ihrer Sicherheitsrelevanz verschiedenen Anforderungsniveaus zugeordnet, die von der höchsten Anforderungsstufe ASIL D bis zur Stufe ASIL A reichen. ASIL steht hierbei für *Automotive Safety Integrity Level*. Im Falle einer noch geringeren Sicherheitsrelevanz wird eine Funktion in

keine der vier Stufen eingeteilt. Beim Steuergerät MDG1C Device 4 ist der Hauptkern mit Sicherheitskern für den Betrieb sicherheitskritischer Software-Funktionen vorgesehen. Er genügt der höchsten Anforderungsstufe ASIL D der Norm ISO 26262. Dies ist Voraussetzung dafür, dass ein Einsatz auch für sicherheitskritische Funktionen möglich ist.

3.1.2 Software-Rahmenbedingungen

Auf einem modernen Motorsteuergerät kommen über Tausend Software-Funktionen zum Einsatz. Diese teils sicherheitsrelevanten Funktionen sind in zuverlässiger Weise unter Einhaltung von Echtzeitbedingungen zu betreiben. Im Folgenden werden wesentliche Aspekte der Software auf einem Fahrzeug-Steuergerät beschrieben.

Betriebssystem

Auf Steuergeräten mit Echtzeit-Anwendungen kommen Echtzeitbetriebssysteme zum Einsatz. Diese sind stark angepasst auf die Steuergeräte-Hardware sowie das eingesetzte Programm. Für ihre Spezifikation sind der OSEK/VDX-OS-Standard sowie der AUTOSAR-Standard zentral.

OSEK/VDX Für Echtzeitbetriebssysteme für eingebettete Systeme im Automobilbereich ist der Standard von OSEK/VDX prägend. Die Organisation OSEK/VDX ist entstanden als Vereinigung von VDX (Vehicle distributed execution) und OSEK (Offene Systeme und deren Schnittstellen für die Elektronik im Kraftfahrzeug) und ihr Standard für Echtzeitbetriebssysteme seit 2005 in der Norm ISO 17356 [3] beinhaltet.

Die Hauptaufgabe der Betriebssysteme liegt im Scheduling. Die spezifizierten Betriebssysteme werden statisch konfiguriert: die Tasks und ihre Ressourcenzuteilung werden nicht während der Laufzeit ermittelt, sondern werden im Rahmen der Generierung des maßgeschneiderten Betriebssystems statisch festgelegt (vgl. [87]). Die Implementierung eines OSEK-OS-konformen Betriebssystems wird also basierend auf den Spezifikationen des OSEK-OS-Standards anwendungsspezifisch angepasst. Hierdurch soll einerseits ein möglichst geringer Bedarf an Rechen- und Speicherressourcen durch das Betriebssystem erzielt werden. Andererseits ist dieses Vorgehen förderlich für einen bekannten Programmablauf. Ein bekannter Ablauf hat Vorteile für Echtzeitprogramme: Er ermöglicht es, ein hohes Maß an Zuverlässigkeit zu gewährleisten. Zudem erleichtert ein festgelegter Programmablauf die Fehlersuche.

AUTOSAR Das Konsortium AUTOSAR (AUTomotive Open System ARchitecture) hat ein breites Mitgliederfeld im Automobil-Bereich und umfasst neben führenden Automobilherstellern und -zulieferern auch Halbleiterhersteller und Hersteller von Entwicklungswerkzeugen. In dem durch das Konsortium geschaffenen Standard wurden die OSEK-Spezifikationen übernommen. Dieser ist damit abwärtskompatibel zum OSEK-OS-Standard. Das Ziel von AUTOSAR ist, die Wiederverwendbarkeit und Portierbarkeit von Steuergeräte-Softwarefunktionen durch verschiedene Vereinheitlichungen zu erhöhen. Dies erleichtert beispielsweise die verteilte Entwicklung von Steuergeräte-Softwarefunktionen zwischen Automobilherstellern und -zulieferern.

Hierzu wird eine Aufteilung in verschiedene Software-Schichten vorgenommen: Die Basis-Software ist hardwarenah und -spezifisch. Sie umfasst beispielsweise Implementierungen der elementaren Funktionen wie der Exponentialfunktion oder den trigonometrischen Funktionen. Die Anwendungssoftware ist weitgehend hardwarefern und -unabhängig. Software-Funktionen zur Regelung oder Diagnose gehören in diese Schicht. Dazwischen sorgt eine weitere Schicht, die AUTOSAR-Laufzeitumgebung, auch als AUTOSAR-Runtime environment oder AUTOSAR-RTE bezeichnet, für eine geeignete Verbindung der Basis-Software mit der Anwendungssoftware. Auf diese Weise kann die Anwendungssoftware unabhängig von der genauen Hardware des Steuergeräts entwickelt und eingesetzt und die gewünschte Portierbarkeit zwischen Steuergeräten unterstützt werden.

Eine Sonderrolle nehmen die sogenannten *complex drivers* ein. Diese können aus der Anwendungssoftware-Schicht heraus direkt auf die Hardware zugreifen, um so für spezielle Aufgaben eine hardwarenahe Implementierung zu ermöglichen.

RTA-OS Im Rahmen der Untersuchungen vorliegender Arbeit wird das Werkzeug RTA-OS der ETAS GmbH eingesetzt, um das Echtzeitbetriebssystem zu generieren. Diese Software stellt einen optimierten Betriebssystemkern mit geringem Speicherbedarf und niedrigen Laufzeitoverheads zur Verfügung. Der erstellte Code ist OSEK/VDX-OS sowie AUTOSAR-konform. Zusammen mit weiteren Tools der ETAS GmbH stehen auch für das Debuggen und für Laufzeitmessungen geeignete kompatible Werkzeuge zur Verfügung.

Programmablauf

Die Software eines modernen Motorsteuergeräts umfasst über 1000 Software-Funktionen (vgl. [67]). Diese werden wie in [67] als Prozesse bezeichnet. Diese Prozesse werden entsprechend ihrem Aufrufintervall verschiedenen Tasks zugeordnet. Die Anzahl der periodischen Tasks ist festgelegt und entspricht gerade der Anzahl an verfügbaren Aufruf-Zeitrastern für die Software-Funktionen. Ihre Perioden reichen vom Millisekunden- bis Sekundenbereich. Die Prioritäten der Tasks sind umso höher je kürzer ihre Periode ist. Neben diesen zeitgesteuerten Prozessen gibt es auch ereignisgesteuerte Tasks. Diese werden beispielsweise ausgelöst, wenn der Kurbelwellenwinkel gewisse Werte überschreitet. Die zugehörigen Tasks werden auch als winkelsynchron bezeichnet. Daneben können unter anderem auch Lenk- oder Bremssignale Ereignisse auslösen. Die Priorität der ereignisgesteuerten Tasks ist höher als die der zeitgesteuerten Tasks. Als Zeitaufteilungsverfahren, auch Scheduling genannt, wird ein verdrängendes Prioritätsscheduling eingesetzt, auf Englisch mit *preemptive priority scheduling* bezeichnet. Bei diesem wird ein laufender Task unterbrochen, sobald ein Task höherer Priorität ausgelöst wird. Erst nach Terminieren des höherprioren Tasks wird zum niederprioren Task zurückgekehrt. Die Prozesse innerhalb eines Tasks werden nacheinander berechnet.

Für die periodischen Tasks stimmen die relative Deadline, innerhalb derer die Erledigung des Tasks fällig ist, und die Aufrufperiode jeweils überein. Daher ist auch nicht zu unterscheiden zwischen einem Scheduling entsprechend der relativen Deadline, was auf Englisch mit *rate monotonic scheduling* bezeichnet wird, und einem Scheduling entsprechend der Periodenlänge, was auf Englisch mit *deadline monotonic scheduling* bezeichnet wird. Eine schematische Darstellung aus einem Programmablauf ist in Abbildung 3.3 gezeigt. In dieser Abbildung ist zu erkennen, dass die Rechenzeit eines Prozesses häufig geringer ist als die Zeitspanne zwischen Beginn und Ende der Berechnungen des Prozesses.

Gelingt es nicht, einen Task innerhalb seiner Periode abzuarbeiten, wird das System durch Sicherheitsfunktionen typischerweise neu gestartet. Ein großer Aufwand wird betrieben, um derartige Echtzeitverletzungen zu vermeiden. Sollte dieser Fall dennoch eintreten, ist der Neustart schnell durchführbar und führt nicht grundsätzlich zu einer Gefährdung.

Mit den unterbrechenden Wechseln zwischen den Tasks ist jeweils ein Kontextwechsel verbunden. Hierbei sind Teile des Datenspeichers und der Register zu sichern beziehungsweise zu laden. Die Kontextwechsel sind durch die vielen Prozesse innerhalb der verschiedenen Tasks zahlreich. Daher soll ein Kontextwechsel wenig Zeit in Anspruch nehmen. Zur Vermeidung von Dateninkonsistenzen im Rahmen dieser Kontextwechsel wird folgendermaßen vorgegangen (vgl. [83]): Für globale Variablen des Programmcodes werden feste Speicheradressen reserviert. Zu Beginn eines Prozesses wird von den verwendeten Variablen eine Kopie erstellt. Auf diesen Kopien arbeitet der Prozess. Hierdurch führen Veränderungen an den globalen Variablen, die durch höherpriore Tasks verursacht werden, nicht zu Dateninkonsistenzen. Nach Abschluss des Prozesses werden die Werte der Variablen auf die für sie vorgesehenen festen Speicheradressen kopiert. Dies kommt beispielsweise bei mathematischen Modellen, bei denen Sensorwerte verwendet werden, zum Tragen: Das Abtasten von Sensorwerten erfolgt in kurzperiodigen Tasks mit einer Periode

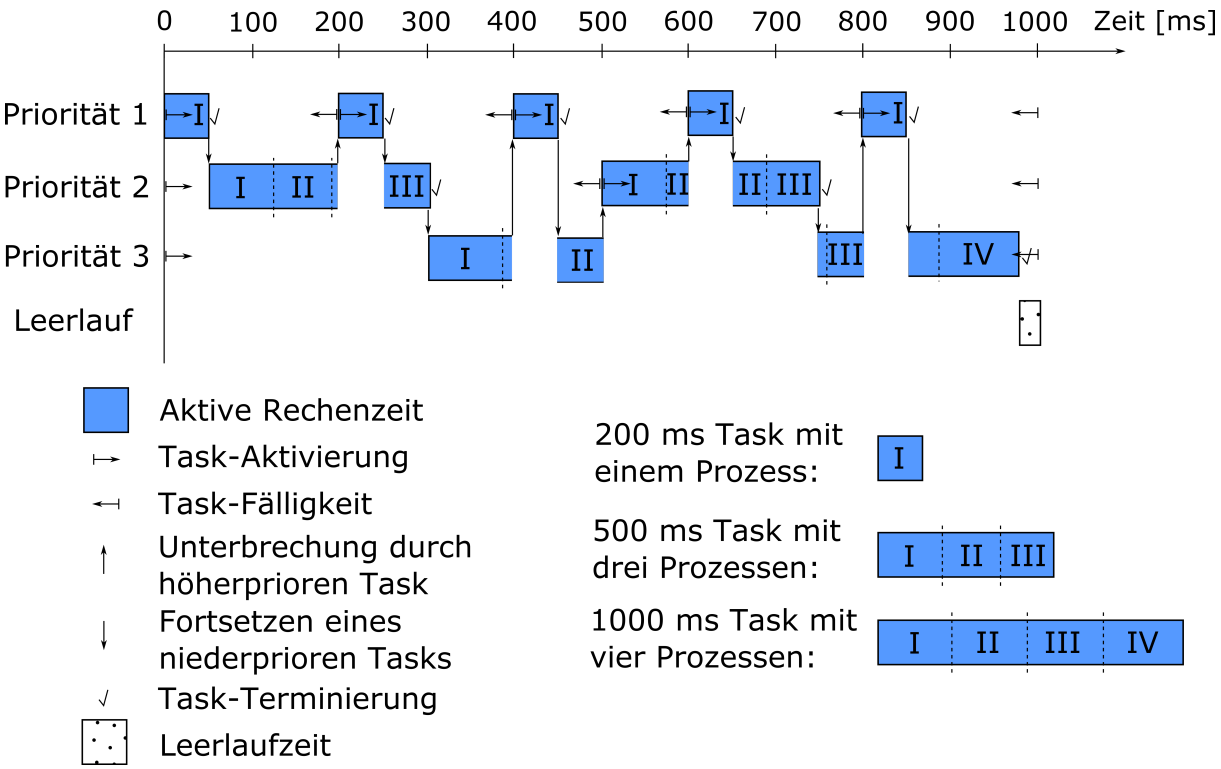


Abbildung 3.3: Schema zur Rechenzeitaufteilung auf verschiedene Prozesse innerhalb einer Hyperperiode im Falle einer zu Veranschaulichungszwecken vereinfachten Task- und Prozessstruktur. Hierbei ist Priorität 1 höher als Priorität 2 und Priorität 3. Außerdem umfasst der Übersichtlichkeit halber der 200 ms Task 1 Prozess, der 500 ms Task 3 Prozesse und 1000 ms Task 4 Prozesse.

im Millisekundenbereich. Folglich kann sich ein verwendeter Sensorwert während der Berechnung eines Zeitschrittes bei der Simulation eines mathematischen Modells mehrfach ändern. Durch die beschriebene Vorgehensweise wird eine hieraus resultierende Dateninkonsistenz verhindert. Dadurch, dass die Software eines modernen Motorsteuergerätes über 1000 Software-Funktionen umfasst, ist es derzeit auch im Falle einer Multicore-Architektur in der Regel unzweckmäßig, eine einzelne dieser Software-Funktionen zu parallelisieren. Stattdessen werden die Prozesse auf die Tasks der verschiedenen Kerne aufgeteilt.

Programmiersprache

Als Programmiersprache für Echtzeit-Software hat auf Steuergeräten im Automobilbereich die Sprache C derzeit eine dominierende Stellung. In [1] werden als Gründe hierfür unter anderem die Flexibilität der Sprache, die grundsätzlich weitgehenden Portierungsmöglichkeiten und die im Hinblick auf Echtzeitanwendungen wichtigen Möglichkeiten zur hardwarenahen Implementierung schneller Ein- und Ausgabeoperationen aufgeführt. Als Ausnahme sind Teile der hochoptimierten Basis-Software anzusehen, bei denen Assemblersprachen eingesetzt werden.

Um dem Zuverlässigkeitsanspruch gerecht zu werden, der bei sicherheitsrelevanten Software-Funktionen im Automobilbereich nötig ist, hat die Organisation MISRA (Motor Industry Software Reliability Association) 1998 erstmals den Programmierstandard MISRA-C geschaffen. Dieser umfasst Einschränkungen gegenüber anderen C-Standards, die für den Einsatz auf eingebetteten Systemen im Automobilbereich angepasst sind. Sie dienen insbesondere einem fehlerfreien Betrieb und einem verständlichen und gut wartbaren Code. Dies wird durch Regeln erreicht, die verbreiteten Programmierfehlern vorbeugen, die Lesbarkeit erhöhen und solche Konstrukte der C-Syntax ausschließen, die je nach Compiler eine unterschiedliche Interpretation haben (vgl. [1]).

Zwei beispielhafte Richtlinien sind folgende:

- Gleitkommazahlen sollen nicht mittels der Vergleichsoperatoren auf Gleichheit oder Ungleichheit getestet werden:

  ```
  if (x == 10.0f)
  ```

  Stattdessen wird gefordert, eine Vergleichsroutine zu verwenden, die die Maschinengenauigkeit und die Größe der beteiligten Werte berücksichtigt.

- Die Zeichenfolge /* darf nicht innerhalb eines Kommentars verwendet werden:

  ```
  /* Kommentar 1
          function_alt();
          /* Kommentar 2 */
  function_neu();
  ```

  Grund hierfür ist folgender: In C ist eine Verschachtelung von Kommentaren nicht vorgesehen, so dass ein mit /* eröffneter Kommentar durch */ geschlossen wird. Manche Compiler hingegen unterstützen eine Verschachtelung von Kommentaren. Hierdurch ist es von der Wahl des Compilers abhängig ob, die Funktion `function_neu`() im Beispiel aufgerufen oder als Kommentar interpretiert wird.

3.2 Simulation mathematischer Modelle in Echtzeitanwendungen

Die Anforderung der Echtzeitfähigkeit führt nicht nur beim Betriebssystem (vgl. Abschnitt 3.1.2), sondern auch bei der Simulation eines mathematischen Modells zu einer gravierenden Verschiebung der Herausforderungen und der Beschaffenheit geeigneter Lösungsverfahren. Die Notwendigkeit, dass ein Algorithmus nicht bloß ein passendes Resultat liefert, sondern binnen einer vorgegebenen Frist überhaupt zu einem Resultat kommt, stellt eine zusätzliche Dimension im Anforderungsraum an einen Algorithmus dar. Diese führt dazu, dass einige numerische Verfahren bei der Lösung einer gewöhnlichen Differentialgleichung für den Echtzeiteinsatz weitgehend ungeeignet sind. Dies betrifft beispielsweise eine Schrittweitensteuerung und wird in Abschnitt 3.2.3 beleuchtet. Für den Echtzeiteinsatz auf einem Fahrzeug-Steuergerät sind durch eine Software-Funktion sowohl die Anforderungen an die Qualität des Ergebnisses als auch die Anforderungen an die eingesetzte Rechenzeit einzuhalten. Es ist zu beachten, dass hierbei weniger die Durchschnittswerte für die Qualität der Lösung oder für die Rechenzeit geeignete Kriterien sind, als vielmehr die Qualität der Lösung, die im ungünstigsten praktisch auftretenden Fall erreicht wird, oder die Rechenzeit, die im ungünstigsten praktisch auftretenden Fall benötigt wird. Auf diese Weise betrachtet ändert sich die Eignung numerischer Verfahren teilweise erheblich.

3.2.1 Definition von Echtzeitfähigkeit

In der Norm DIN 44300 [5] , die mittlerweile zurückgezogen ist, ist folgende Definition für Echtzeit angegeben:

> "Unter Echtzeit versteht man den Betrieb eines Rechensystems, bei dem Programme zur Verarbeitung anfallender Daten ständig betriebsbereit sind, derart, dass die Verarbeitungsergebnisse innerhalb einer vorgegebenen Zeitspanne verfügbar sind. Die Daten können je nach Anwendungsfall nach einer zeitlich zufälligen Verteilung oder zu vorherbestimmten Zeitpunkten anfallen."

Im Hinblick auf Echtzeitanwendungen auf Fahrzeug-Steuergeräten ist der Begriff weiter zu differenzieren. Eine gängige Unterteilung von Echtzeitanforderungen ist in weiche, feste und harte Echtzeit [87]:

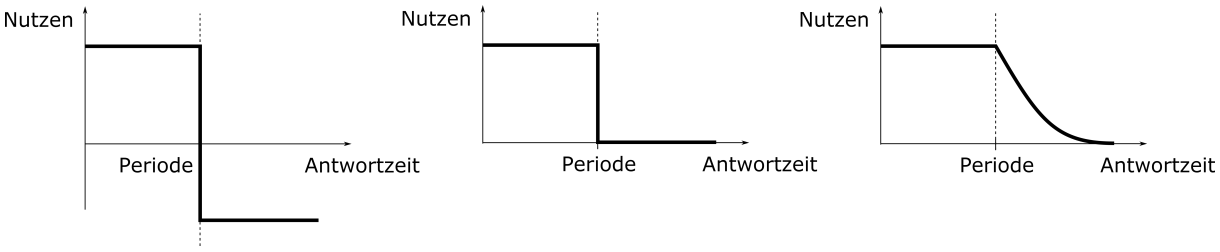


Abbildung 3.4: Qualitative Beschreibung des Nutzens des Resultates einer Software-Funktion in Abhängigkeit seiner Antwortzeit für verschiedene Arten von Echtzeitanforderungen nach [87]: harte Echtzeit (links), feste Echtzeit (Mitte) und weiche Echtzeit (rechts).

- Harte Echtzeit: Die Einhaltung der Echtzeitanforderung ist unabdingbar, andernfalls droht eine Gefahr oder ein Schaden. Diese Art der Echtzeitanforderung betrifft sicherheitsrelevante Funktionen bei Anwendungen auf Steuergeräten. Als Konsequenz der ISO 26262 (vgl. Abschnitt 3.1.1) ergeben sich feste Reaktionszeiten für verschiedene sicherheitsrelevanten Funktionen, deren Einhaltung Voraussetzung für die Erfüllung der Norm ist. Einige Beispiele, bei denen harte Echtzeitanforderungen vorliegen, sind bei Bremssystemen, bei elektronischen Stabilitätsprogrammen und bei der Motorsteuerung.
- Feste Echtzeit: Bei festen Echtzeitanforderungen ist das Resultat einer Software-Funktion nach Verstreichen der zeitlichen Frist wertlos. Allerdings droht durch die Verletzung einer festen Echtzeitanforderung nicht unmittelbar ein Schaden oder eine Gefahr. Diese Art der Echtzeitanforderung tritt im Fahrzeug-Bereich bei Regelungen in Komfortfunktionen auf. In [87] wird eine Geschwindigkeitsregelung als Beispiel angeführt: eine Verletzung der Echtzeitanforderung wird als nicht sicherheitskritisch eingestuft. Ein verspätetes Resultat ist aber dadurch wertlos, dass eine rückwirkende Ansteuerung eines Aktors nicht möglich ist.
- Weiche Echtzeit: Bei weicher Echtzeit sind geringe Verletzungen der Echtzeitanforderung nur von geringem Nachteil. Musterbeispiel sind Multimediasysteme: Bei der Menüführung eines Multimediasysteme gibt es Anforderungen an die Reaktionszeit, die gewährleisten, dass die Reaktion des Systems durch den Nutzer als instantan wahrgenommen wird. Verletzungen dieser Reaktionszeit werden durch den Nutzer als verzögertes Systemverhalten wahrgenommen. Auch zu einem verspäteten Zeitpunkt hat die Systemreaktion hierbei aber noch einen Wert.

Die in vorliegender Arbeit untersuchten numerischen Verfahren sollen für einen Einsatz in sicherheitskritischen Software-Funktionen geeignet sein. Somit ist die harte Echtzeitanforderung die angemessene Anforderung. Im Folgenden wird unter einer Echtzeitanforderung, sofern sie nicht weiter spezifiziert wird, stets eine harte Echtzeitanforderung verstanden.

In Abbildung 3.4 ist der Nutzen des Resultats einer Software-Funktion in Abhängigkeit seiner Antwortzeit für die verschiedenen Arten von Echtzeitanforderungen qualitativ illustriert. Eine Konsequenz aus Abbildung 3.4 ist, dass der Nutzen des Resultates einer Software-Funktion für Antwortzeiten, die die Echtzeitanforderung einhalten, konstant ist. Der Nutzen ist also insbesondere nicht umso größer, je schneller das Resultat verfügbar ist. Stattdessen ist die Einhaltung der Frist das einzige zeitliche Kriterium für den Nutzen des Resultates einer Software-Funktion.

3.2.2 Zentrale Anforderungen für Einsatz auf Steuergeräten

Damit die Simulation eines mathematischen Modells für den Einsatz in einer Software-Funktion eines Steuergeräts geeignet ist, ist es zentral, dass diese verschiedene Anforderungen erfüllt. Die folgenden vier Eigenschaften bilden hierbei die Leitkriterien.

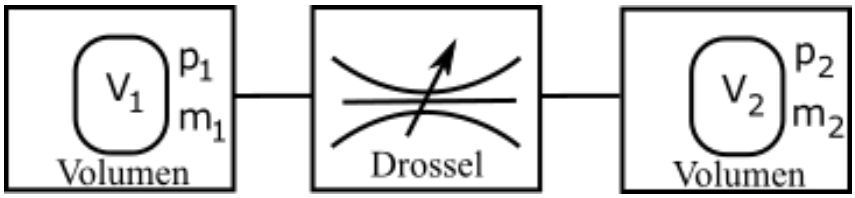


Abbildung 3.5: Schema eines Modells mit zwei Volumen, die durch eine Drossel verbunden sind.

Echtzeitfähigkeit

Die Einhaltung der Echtzeitanforderungen ist essentiell. Ist diese nicht gegeben, ist nicht nur eine zuverlässige Ausführung der Simulation nicht gegeben, sondern es kann auch Auswirkungen auf weitere Funktionen nach sich ziehen. Im Hinblick auf den Betrieb sicherheitsrelevanter Funktion ist dies inakzeptabel (vgl. 3.1.1). Daher ist das Gewährleisten der Echtzeitfähigkeit unabdingbar.

Notwendigkeit zur Rechenzeitreduktion auf Steuergeräten

Eine Echtzeitanforderung führt zunächst nicht zur Anforderung einer besonders kurzen Rechenzeit. Beispielsweise wird in [36] eine Regelung mit Echtzeitanforderungen für eine chemische Anwendung untersucht, bei der ein Zeitschritt von 10 Sekunden vorliegt. Wie in Abbildung 3.4 dargestellt, bedeutet eine Echtzeitanforderung auch nicht, dass der Nutzen des Resultates umso höher ist, je schneller es vorliegt. Stattdessen ist der Nutzen innerhalb der vorgesehenen Zeitintervall als konstant anzusehen. Allerdings führt der Einsatz auf einem Fahrzeug-Steuergerät zur Anforderung eines möglichst geringen Verbrauchs an Rechenkapazität:

Wie in Abschnitt 3.1 beschrieben, ist die Rechenkapazität eines Steuergeräts limitiert und wird auf eine große Zahl an Software-Funktionen verteilt, so dass die Rechenzeit für eine einzelne Software-Funktion beschränkt und wertvoll ist. Der Markt der Fahrzeug-Steuergeräte ist kostensensitiv [87] und die Rechenzeit einer Software-Funktion entspricht letzten Endes einem monetären Gegenwert. Im Hinblick auf die Simulation steifer Modelle auf einem Fahrzeug-Steuergerät ist davon auszugehen, dass die Rechenkapazität, die für einen Zeitschritt eines mathematischen Modells zur Verfügung steht, aus wirtschaftlichen Gründen so gering ist, dass sie derzeit eine zentrale Hürde für den Einsatz solcher mathematischer Modelle in Echtzeitanwendungen auf einem Steuergerät darstellen, wie sie in Abschnitt 2.1.7 beschrieben sind. Somit ist eine Kernherausforderung für den Einsatz einer Simulationen eines steifen mathematischen Modells in einer Echtzeitanwendung auf einem Fahrzeug-Steuergeräten, dass die benötigte Rechenzeit zur Lösung der gewöhnlichen Differentialgleichung gering genug ist.

Stabilität des Simulationsverlauf

Mit Stabilität des Simulationsverlaufs wird im Folgenden bezeichnet, dass der Verlauf der Zustände über den Simulationsverlauf hinweg beschränkt ist und dass der Simulationsverlauf zudem keine starken Oszillationen aufweist, die durch die Steifheit des Systems numerisch bedingt sind. Dies wird im Folgenden an einem einfachen, aber durchaus relevanten Beispielmodell veranschaulicht.

Das Beispielmodell ist ein Modell von zwei nicht örtlich diskretisierten Volumen, die über eine Drossel verbunden sind. Es ist in Abbildung 3.5 schematisch dargestellt. Relevant ist das Beispiel dadurch, dass es in einem Luftsystemmodell mehrfach als Bestandteil auftreten kann und wesentlich zur Steifheit des Luftsystems beiträgt (vgl. [102]).

Das Modell hat vier Zustände: jeweils den Druck und die Gasmasse für beide Volumen. Betrachtet wird folgende Konfiguration: Beide Volumen seien nicht variabel mit einem Volumen von $V_1 = V_2 = 5\,\mathrm{l}$. Gefüllt seien sie mit Frischluft. Als effektive Querschnittsfläche der Drossel wird $1.75\,\mathrm{cm}^2$

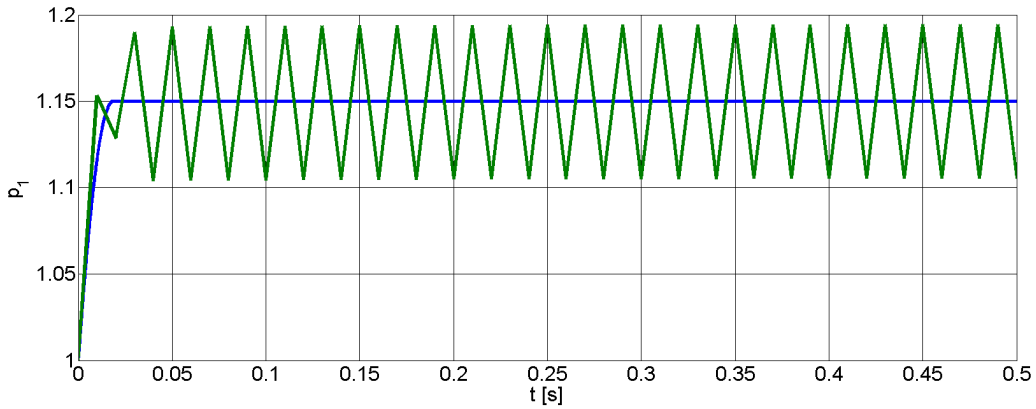


Abbildung 3.6: Simulationsverlauf der Beispielkonfiguration des Zwei-Volumen-Modells mit dem expliziten Euler-Verfahren mit einer Schrittweite von 10 ms (grün) und mit einer Schrittweite von 0.1 ms (blau).

angenommen. Dies entspricht einer nur geringen Öffnung der Drosselklappe. Die Steifheit des Modells nahe dem Druckgleichgewicht ist umso größer, je weiter die Drosselklappe geöffnet ist.

Bei einer Modellierung gemäß [102] ergeben sich im Fall $p_1 > p_2$ folgende Modellgleichungen:

$$\begin{pmatrix} \dot{p_1} \\ \dot{m_1} \\ \dot{p_2} \\ \dot{m_2} \end{pmatrix} = \begin{pmatrix} -3.10\,\frac{{p_1}^2}{m_1}\,\min\left(0.484, \sqrt{1.87\left(\frac{p_2}{p_1}\right)^{1.43} - 1.87\left(\frac{p_2}{p_1}\right)^{1.71}}\right)\sqrt{\frac{m_1}{p_1}} \\ -2.21\,p_1\,\min\left(0.484, \sqrt{1.87\left(\frac{p_2}{p_1}\right)^{1.43} - 1.87\left(\frac{p_2}{p_1}\right)^{1.71}}\right)\sqrt{\frac{m_1}{p_1}} \\ p_1\,\min\left(0.484, \sqrt{1.87\left(\left(\frac{p_2}{p_1}\right)^{1.43} - \left(\frac{p_2}{p_1}\right)^{1.71}\right)}\right)\left(\frac{1.89\,p_1}{m_1} - \frac{1.61\,p_2}{m_2}\right)\sqrt{\frac{m_1}{p_1}} \\ 2.21\,p_1\,\min\left(0.484, \sqrt{1.87\left(\frac{p_2}{p_1}\right)^{1.43} - 1.87\left(\frac{p_2}{p_1}\right)^{1.72}}\right)\sqrt{\frac{m_1}{p_1}} \end{pmatrix}.$$

Im Falle $p_1 < p_2$ werden die Rollen von p_1 und m_1 mit p_2 und m_2 in obiger Gleichung vertauscht. Die Überprüfung, welches der Volumen den größeren Druck besitzt, erfolgt hierbei nur zu jedem Zeitschritt.

Als Beispielverlauf wird ein Druckausgleich zwischen den beiden Volumen betrachtet. Hierzu werden als Initialisierung folgende Werte gewählt: Der Druck in Volumen V_1 sei $p_{1,0} = 1.3\,\text{bar}$ und der Druck in Volumen V_2 sei $p_{2,0} = 1.0\,\text{bar}$. In beiden Volumen wird eine initiale Lufttemperatur von $T_1 = T_2 = 300\,\text{K}$ angenommen, so dass sich über das allgemeine Gasgleichung als Startwerte für die Massen $m_{1,0} = m_{2,0} = 0.581\,\text{g}$ ergeben.

In Abbildung 3.6 ist zu sehen, dass sich bei der Lösung mit dem expliziten Euler-Verfahren je nach gewählter Zeitschrittweite qualitativ signifikant unterschiedliche Simulationsverläufe ergeben. Die zu beobachtenden Schwingungen im Falle einer Zeitschrittweite von $\Delta t = 10\,\text{ms}$ sind typisch bei einer Integration steifer Systeme mittels expliziter Lösungsverfahren und einer nicht hinreichend kleinen Schrittweite. Treten bei der Simulation eines steifen mathematischen Modells Schwingungen oder eine Instabilität auf, die im physikalischen System selbst nicht zu beobachten sind, so ist eine der denkbaren Ursachen, dass ein explizites oder ein anderes nicht A-stabiles Integrationsverfahren verwendet wird (vgl. [58]).

Um qualitativ falsche Resultate auszuschließen, ist es eine selbstverständliche und wesentliche Anforderung an die Simulation eines steifen mathematischen Modells, dass der Simulationsverlauf weder unbeschränkte Zustände besitzt noch allein durch die Steifheit des Systems aus numerischen Gründen stark oszilliert.

Da die Ausgänge eines mathematischen Modells durch andere Software-Funktionen weiterverarbeitet werden, sind Oszillationen kritisch, die allein numerisch bedingt sind. Wird beispielsweise das Ausgangssignal eines mathematischen Modells durch eine weitere Software-Funktion zeitlich differenziert, so wirken sich allein numerisch bedingte Oszillationen besonders ungünstig aus. Daher sind numerisch bedingt Oszillationen zu vermeiden.

Wird im obigen Beispiel die Zeitschrittweite $\Delta t = 100\,\text{ms}$ gewählt, so tritt der Fall ein, dass die Zustände nicht beschränkt bleiben. Dies führt zu einem Überlauf der jeweiligen Gleitkommavariable. Hierdurch wird die Simulation letztlich vollständig unbrauchbar. Der Fall von Zuständen, die über den Simulationsverlauf hinweg unbeschränkt sind, ist daher unbedingt zu verhindern.

Genauigkeit des Simulationsverlaufs Neben der Stabilität des Simulationsverlaufs stellt auch die Genauigkeit eine weitere Anforderung dar. Der letztliche Fehler, also die Abweichung vom Resultat aus der Simulation und dem real vorliegenden Wert, setzt sich aus mehreren Faktoren zusammen:

- Diskretisierungsfehler durch Integrationsverfahren
- Numerische Fehler im Rahmen des Integrationsverfahrens
- Modellierungsfehler
- Fehler durch Eingangsdaten wie beispielsweise Sensorwerte

Die Untersuchungen vorliegender Arbeit spielen lediglich bei dem Fehler durch das Integrationsverfahren und dessen numerische Teilaufgaben wie der Differenziation oder Gleichungslösung eine Rolle. Die Modellierung oder die Sensoreigenschaften sind nicht Gegenstand der Untersuchung.

Aus numerischer Sicht sind die Anforderungen an die Genauigkeit weit weniger kritisch und herausfordernd als es die Anforderung eines stabilen Simulationsverlaufs ist. Dies liegt daran, dass der Diskretisierungsfehler und die numerischen Fehler im Falle einer stabilen Integration bei den üblichen Zeitschrittweiten oft bereits klein sind gegenüber dem Modellierungsfehler.

Sollte der Diskretisierungsfehler bei der Simulation eines mathematischen Modells bei der standardmäßigen Wahl von Zeitschrittweite und Integrationsverfahren größer als zulässig ausfallen, so ist Folgendes zu berücksichtigen: Aussagen über den Diskretisierungsfehler eines Integrators in Form von Konsistenz- oder Konvergenzordnung sind nur von eingeschränkter Aussagekraft darüber, zu welchem Fehler der Integrator bei Einsatz im Rahmen einer Echtzeitsimulation eines mathematischen Modells führt. Dies liegt daran, dass diese Ordnungen jeweils über eine Grenzwertbildung $\lim_{\Delta t \to 0}$ zu Stande kommt. Die Zeitschrittweiten im Falle der Simulation eines mathematischen Modells auf einem Steuergerät sind aber entweder fest vorgegeben oder nur eingeschränkt veränderbar. Das Verhalten eines Integrators bei $\lim_{\Delta t \to 0}$ hat daher höchstens keinen unmittelbaren Bezug zur Eignung des Integrators im Rahmen einer Echtzeitsimulation in einer Software-Funktion eines Steuergeräts. Falls theoretische Aussagen über die Genauigkeit eines Integrators herangezogen werden, so sollten dies Fehlerschätzungen sind, die ohne Grenzwertbildung auskommen und die konkreten Werte für Zeitschrittweite sowie die Modellgleichungen verwenden. Bei einer Taylor-Entwicklung eines Fehlerterms sind also nicht allein die Exponenten der nicht verschwindenden Summanden, sondern auch deren Koeffizienten zu bestimmen.

Ein weiterer Aspekt ist, dass die Ordnung eines Integrationsverfahren dadurch nur bedingt relevant sind, dass die Modelle der Problemklasse aus Abschnitt 2.1.7 nicht glatt sind oder nur eine Glattheit geringer Ordnung aufweisen. Daher führen Verfahren hoher Ordnung nicht notwendigerweise zu einer großen Genauigkeit. Grundsätzlich ist deren Einsatz im Falle der unzureichenden Glattheit der Modellgleichungen unnötig. Falls der Fall eintritt, dass der Diskretisierungsfehler bei einem mathematischen Modell größer als zulässig ausfällt, ist daher das Verringern der Zeitschrittweite ein typisches Mittel.

3.2.3 Der Spagat einer echtzeitfähigen und stabilen Integration

In diesem Abschnitt wird erläutert, weshalb verschiedene gängige Ansätze bei der Lösung von steifen Anfangswertproblemen für den Einsatz unter Echtzeitbedingungen nicht geeignet sind oder einer speziellen Sorgfalt bedürfen.

Verringerung der Schrittweite Der simpelste aller Ansätze zur stabilen Integration steifer Anfangswertprobleme, ist die Reduktion des Zeitschrittes beim Einsatz eines expliziten Integrators. Für sehr steife Systeme ist dieser Ansatz ungeeignet. Gerade für Systeme, die sich nahe dem Graubereich zwischen steif und nicht-steif bewegen, ist dies aber eine relevante Möglichkeit. Aus diesem Grund wird im Folgenden dargelegt, wie und ob die Zeitschrittweite im Rahmen der Simulation des Zwei-Volumen-Modell aus Abschnitt 3.2.2 im Rahmen einer Software-Funktion auf einem Steuergerät reduziert wird:

Wie bereits in Abbildung 3.6 gezeigt, liefert für die in Abschnitt 3.2.2 beschriebene Konfiguration des Zwei-Volumenmodells eine Zeitschrittweite des Integrators von $\Delta t = 0.1\,\text{ms}$ einen stabilen Simulationsverlauf. Auf dem Steuergerät steht kein Task mit einer Periode dieser Größenordnung zur Verfügung. Um eine solche Zeitschrittweite auf dem Steuergerät zu realisieren, ist folgendermaßen vorzugehen: Die Software-Funktion, die die Simulation des Modells umfasst, wird einem Zeitraster ΔT ausgeführt, wie es für die Software-Funktion grundsätzlich geeignet ist, beispielsweise $\Delta T = 10\,\text{ms}$. Bei jedem Aufruf der Software-Funktion werden $\frac{\Delta T}{\Delta t} = 100$ Zeitschritte der Schrittweite Δt ausgeführt.

Hierdurch werden die starken Oszillationen des Simulationsverlaufs verhindert. Allerdings erfüllt dieses Vorgehen den Anspruch einer effizienten Nutzung der zur Verfügung stehenden Rechenzeit nicht, da 100 Auswertungen der rechten Seite je Zeitschritt deutlich mehr als nötig sind (vgl. [17]). Außerdem ist die Stabilität des Simulationsverlaufs des Zwei-Volumen-Modells mit einer Zeitschrittweite von $\Delta t = 0.1\,\text{ms}$ keineswegs allgemein gültig, sondern zunächst nur die obige Konfiguration erfüllt, so dass eine noch wesentlich kleinere Schrittweite für einen allgemeinen Einsatz des Zwei-Volumen-Modells nötig sein kann.

Schrittweitensteuerung Eine Schrittweitensteuerung ist bei Echtzeit-Anwendungen prinzipiell zwar möglich, aber hierfür im Allgemeinen suboptimal. Dies wird im Folgenden erläutert:

Damit diese unter Echtzeitbedingungen einsetzbar ist, muss auch im Falle, dass durch die Schrittweitensteuerung viele kleine Zeitschritte in einem Aufruf zu absolvieren sind, ausreichend Rechenkapazität bestehen. Diese wäre aber im Fall großer Schrittweiten ungenutzt. Zweckmäßiger ist es daher im Allgemeinen, die ohnehin reservierte Rechenkapazität dadurch zu nutzen, dass der kleinste mit dieser Rechenkapazität echtzeitfähige Zeitschritt bei allen Aufrufen verwendet wird anstatt nur bei einzelnen Aufrufen, wie es bei der Schrittweitensteuerung der Fall ist.

Neben einer Schrittweitensteuerung gibt es etliche weitere Verfahren, die außerhalb von Echtzeitanwendungen etabliert sind, für den Echtzeiteinsatz aber nicht gut geeignet sind. Eine wichtige Ursache hierfür kann folgende sein:

Wie in Abbildung 3.4 dargestellt, führt eine vorzeitige Verfügbarkeit der Resultate nicht zu einem höheren Nutzen. Innerhalb der relativen Deadline ist der Nutzen des Resultates konstant. Insbesondere kann bei der Simulation eines mathematischen Modells die Rechenkapazität, die in einem Zeitschritt gespart wird, nicht in einem anderen Zeitschritt verwertet werden. Aus diesem Grund sind numerische Verfahren, die ihren Rechenbedarf zur Simulation eines mathematischen Modells nicht gleichmäßig über den Integrationszeitraum verteilen, sondern für einige Zeiträume wesentlich mehr Aufwand investieren als für andere, ungünstig für einen Einsatz unter Echtzeitbedingungen. Denn dort ist die Echtzeitanforderung bei jedem Aufruf zu erfüllen, somit ist die Rechenzeit im ungünstigen Fall wesentlich und eine Kompensation der Rechenzeiten zwischen verschiedenen Aufrufen nicht möglich.

Steifheitserkennung Bei der Lösung einer gewöhnlichen Differentialgleichung kann eine Steifheitserkennung eingesetzt werden (vgl. [88, 58]). Deren Idee besteht darin, nur dann eine implizite Integration einer Differentialgleichung durchzuführen, wenn dies im aktuellen Zeitschritt mit dem vorliegenden Zustandsvektor nötig ist. Eine Steifheitserkennung ist unter Echtzeitbedingungen allerdings ohne Mehrwert: Dadurch, dass die Rechenzeit im ungünstigsten Fall ausschlaggebend ist, ist für jeden Zeitschritt die Rechenkapazität für eine implizite Integration vorzusehen. Hierdurch erzeugt es also keinen zusätzlichen Bedarf an Rechenkapazität, wenn in jedem Zeitschritt eine implizite Integration eingesetzt anstatt nur in solchen Zeitschritten, wo dies zur Stabilität nötig ist. Der Rechenaufwand, der dafür investiert wird, die Steifheit zu erkennen, ist somit unnötig und kann eingespart werden.

So wie bei einer Schrittweitensteuerung eine ungleichmäßige Rechenzeitverteilung über die Simulationszeit hinweg die Ursache ist für die Untauglichkeit für den Einsatz unter Echtzeitbedingungen ist, ist bei einer Steifheitserkennnung der unterschiedliche Rechenzeitbedarf über unterschiedliche Zustände des Modells hinweg die Ursache dafür, dass für den Echtzeiteinsatz eine Steifheitserkennung ungünstig ist.

Implizite Integratoren

Wie die vorangehende Diskussion zeigt, ist eine Integration mittels eines impliziten Lösungsverfahren für vorliegendes Beispiel naheliegend. Da die rechte Seite eine geringe Glattheit durch das Auftreten der Minimumsfunktion hat, ist die Verwendung eines Verfahrens erster Ordnung naheliegend. Beim impliziten Euler-Verfahren ergibt sich beim Anfangswertproblem

$$\dot{x}(t) = f(x)$$

im n-ten Zeitschritt die Gleichung

$$x_n - \Delta t f(x_n) - x_{n-1} = 0.$$

Für nichtlineare rechte Seiten f ist somit eine nichtlineare Gleichung zu lösen. Diese Aufgabe ist nicht trivial. Die Art und Weise, wie diese nichtlineare Gleichung gelöst wird, hat wie nachfolgend dargestellt wesentlichen Einfluss auf die Eigenschaften des zu Grunde liegenden Integrationsverfahrens.

Lösung nichtlinearer Gleichungen mittels Picard-Verfahren Die Verwendung eines Picard-Verfahrens ist bei steifen Systemen ungeeignet, da es oft nicht zu einem stabilen Verlauf der Simulation führt (vgl. [32]). Selbst im einfachen Fall von skalaren, linearen rechten Seiten

$$\dot{x} = \mu x, \quad \mu < 0$$

ist ein implizites Euler-Verfahren, das ein Picard-Verfahren einsetzt, oft nicht A-stabil. Als Fixpunktgleichung wird als Beispiel folgende betrachtet:

$$x_n = x_{n-1} + \Delta t \mu x_n. \tag{3.1}$$

Falls im i-ten Zeitschritt k_i Fixpunktiterationen durchgeführt werden, sowie $x_i^{(0)} = x_{i-1}$ gewählt wird, ergibt sich, wie sich induktiv leicht zeigen lässt, im n-ten Zeitschritt für die k_n-te Fixpunktiteration:

$$x_n^{(k_n)} = x_0 \prod_{i=1}^{n} \left(\sum_{j=0}^{k_i} (\Delta t \mu)^j \right).$$

Die Stabilitätsfunktion des Verfahrens ist insbesondere polynomiell und das Verfahren damit

nicht A-stabil, obwohl das implizite Euler-Verfahren bei korrekter Lösung der Gleichung 3.1 A-stabil ist.

Lösung nichtlinearer Gleichungen mittels Newton-Verfahren Das Newton-Verfahren ist ein lokal konvergentes Verfahren. Bei ungeeigneten Startwerten liegt daher unter Umständen keine Konvergenz vor. Sofern das Newton-Verfahren für gegebene Startwerte konvergiert, verhält sich die Qualität der Zwischenlösung in Abhängigkeit von der aufgewandten Rechenzeit nicht notwendigerweise monoton. Selbst im Falle der Konvergenz ist daher das Lösen einer nichtlinearen Gleichung bis auf eine vorgegebene Genauigkeit mit dem Newton-Verfahren im Allgemeinen unter Echtzeitbedingungen heikel.

Für einen echtzeitfähigen Einsatz ist daher die Anzahl der verwendeten Iterationen unter Echtzeitbedingungen fest zu begrenzen oder festlegen. Für die Lösung der während des impliziten Euler-Verfahrens auftretenden nichtlinearen Gleichungen kann dies bedeuten, dass die Resultate, die durch ein Newton-Verfahren mit fixer Iterationszahl erhalten werden, sehr ungenau sind.

Im Rahmen eines impliziten Euler-Verfahrens ist der Einsatz eines Newtons-Verfahrens mit fixer Iterationszahl aber dennoch geeignet: diese Kombination wird als eine Newton-artige Methode bezeichnet (vgl. [34]). Ihre Eignung lässt sich anschaulich dadurch erklären, dass bei ihnen ein Ausiterieren auch über die Zeitschritte hinweg anstatt allein innerhalb eines Zeitschrittes stattfindet.

Führt man je Zeitschritt genau eine Iteration des Newton-Verfahrens aus, so erhält man das linear-implizite Euler-Verfahren (vgl. [110, 111]). Es wird in Kapitel 4 beschrieben und angewandt.

Differenziation Wird ein linear-impliziter Integrator oder ein anderes implizites Integrationsverfahren eingesetzt, bei dem zur Lösung des auftretenden Gleichungssystems ein Newton-Verfahren verwendet wird, so ist die Differenziation eine der wesentlichen Herausforderungen im Rahmen der Integration.

Die numerische Differenziation ist beispielsweise anfällig bei starken Nichtlinearitäten der rechten Seite und bedarf für einen zuverlässigen Einsatz eines Absicherungsmechanismus, durch den unzulängliche Resultate bei der numerischen Differenziation detektiert werden. Das Ermitteln eines symbolischen Ausdrucks für die exakte Jacobi-Matrix im Rahmen einer zur Entwicklungszeit stattfindenden Untersuchung, der zur Laufzeit lediglich auszuwerten ist, ist dadurch nur begrenzt einsetzbar, dass die Terme unter Umständen zu kompliziert sind, als dass eine Auswertung eines solchen symbolischen Ausdrucks zur Laufzeit eine geeignete Möglichkeit auf einem Steuergerät darstellt. Eine interessante Möglichkeit stellt das automatische Differenzieren dar. Dieses ermöglicht eine zuverlässige und akkurate Möglichkeit zur Differenziation, ohne dass zu einem im Vergleich zur numerischen Differenziation erheblich größeren Rechenaufwand führt.

Festhalten der Jacobi-Matrix Das Festhalten der Jacobi-Matrix, auf Englisch mit *time-lagged Jacobian* bezeichnet, ist eine Vorgehensweise im Rahmen der Lösung einer gewöhnlichen Differentialgleichung mittels eines impliziten Lösungsverfahrens (vgl. [114]). Die Idee des Ansatzes ist, dass die Rechenzeit bei der Lösung der Differentialgleichung dadurch verringert wird, dass die Jacobi-Matrix nicht in jedem Zeitschritt neu ermittelt wird, sondern über mehrere Zeitschritte hinweg festgehalten wird. Hierdurch wird in vielen Schritten der Aufwand zur Jacobi-Matrix-Berechnung eingespart.

Unter Echtzeitbedingungen würde dieser Ansatz allerdings zu keinerlei Verringerung der Rechenzeit führen, weil diejenigen Aufrufe, in denen die Jacobi-Matrix neu berechnet wird, die benötigte Rechenkapazität festlegen. Dass bei den anderen Aufrufen Rechenzeit gespart wird, hat keine Wirkung unter Echtzeitbedingungen. Das Festhalten der Jacobi-Matrix ist ein weiteres Beispiel dafür, dass eine Laufzeitreduktion, die nur einen Großteil der Zeitschritte betrifft, unter Echtzeitbedingungen zunächst keinen Mehrwert bietet.

Implementierung des symbolischen Ausdrucks der Lösung

Beachtenswert ist, dass auch an sich sehr gute geeignete Lösungsverfahren abhängig sein können von geeigneten Implementierungen elementarer Funktionen. Dies wird am Beispiel der Exponentialfunktion veranschaulicht.

Für die Dahlquistsche Testgleichung

$$\dot{x}(t) = \mu x, \quad \mu < 0$$

ist die Lösung zum Zeitpunkt t_{n+1} gegeben durch

$$x(t_{n+1}) = e^{\mu \Delta t} x(t_n).$$

In diesem Fall ist das Implementieren der explizit vorliegenden Lösungsfunktion also eine Alternative zum Einsatz eines Integrationsverfahrens. Äquivalent ist der Einsatz eines exponentiellen Integrators, da ein solcher für lineare Systeme exakt ist (vgl. [60]). In beiden Fällen hängt der erhaltene Wert x_{n+1} davon ab, wie $e^{\mu \Delta t}$ berechnet wird. Wird dieser Term durch eine Taylor-Reihenentwicklung bis zum k-ten Glied ermittelt, so führt dies bei einer vorgegebenen Schrittweite für betragsmäßig große Werte μ zu einem instabilen Simulationsverlauf:

Dann gilt nämlich

$$x_{n+1} = \left(\sum_{i=0}^{k} \frac{(\Delta t \mu)^i}{i!} \right) x_n.$$

Die Stabilitätsfunktion des Verfahrens ist also $R(z) = \sum_{i=0}^{k} \frac{z^i}{i!}$ und somit ein Polynom (vgl. [57]). Daher ist das Stabilitätsgebiet beschränkt und der Simulationsverlauf für $\mu \Delta t$ außerhalb des Stabilitätsgebiets instabil. Obwohl also das Verfahren in der Auswertung des exakten Lösungsausdrucks besteht, ist das Verfahren nicht A-stabil. Im Falle einer besonders ungeeigneten Berechnung der Exponentialfunktion durch eine Taylor-Reihe bis zur Ordnung $k = 1$ ist das Verfahren sogar äquivalent mit dem expliziten Euler-Verfahren.

Dieses Beispiel zeigt, dass bei jedem Lösungsverfahren die Art, wie die elementaren Funktionen implementiert sind, beachtet werden muss. Dies betrifft nicht nur die Qualität der Lösung, sondern auch ihre benötigte Rechenzeit: Die Quadratwurzelfunktion, die Exponentialfunktion, die Logarithmusfunktion oder die trigonometrischen Funktionen sind beispielsweise nicht bei jedem Fahrzeug-Steuergerät in Hardware verfügbar, so dass bei einer Software-Funktionen, die grundsätzlich auf derartige Steuergeräte portierbar sein soll, unter Umständen auf Realisierungen dieser elementaren Funktionen in Software zurückgegriffen wird. Dies zieht Konsequenzen für die Wahl der numerischen Verfahren nach sich: Falls beispielsweise eine Givens-Rotation eingesetzt werden soll, ist für die Entscheidung zwischen der Standardvariante der Givens-Rotation oder einer rationalen Givens-Rotation (vgl. [63]) durchaus relevant, wie aufwändig die Berechnung der Wurzel ist. Je nach Implementierung der Wurzelfunktion kann somit die eine oder die andere Variante besser geeignet sein.

3.2.4 Stand der Technik

Für die Simulation steifer Modelle auf Steuergeräten sind wenige Beispiele dokumentiert. Die zugänglichen Beispiele werden im ersten Teil dieses Abschnitts vorgestellt.

Eine weitere wichtige Quelle für grundsätzlich geeignete Methoden stellen die numerischen Verfahren dar, die auf Hardware-in-the-Loop Systemen zum Einsatz kommen. Bei diesen bestehen ebenfalls Echtzeitanforderungen und die dort eingesetzten Modelle sind die gleichen oder ähnlich denen, die für den Einsatz auf Steuergeräten grundsätzlich in Frage kommen. Hierdurch sind die dort eingesetzten Methoden insbesondere anwendbar für Modelle mit den recht spezifischen Eigenschaften, wie sie in Abschnitt 2.1.7 beschrieben sind. Allerdings sind dort die Anforderungen

andere: Auf Grund der leistungsfähigeren Zielhardware und dadurch, dass die Hardware weniger kostensensitiv ist, ist der Aspekt der Rechenzeitreduktion nicht so dominierend wie es bei Steuergeräten der Fall ist (vgl. Abschnitt 3.2.2).

Die Kombination aus den recht spezifischen Modelleigenschaften, wie beispielsweise, dass es sich in Relation zu den in der numerischen Literatur eher untersuchten Systemen um vergleichsweise kleine Systeme handelt, die zudem eine geringe Glattheit aufweisen, sowie den speziellen Anforderungen, nämlich unter harten Echtzeitbedingungen eine Minimierung der Rechenzeit zu erzielen, sehe ich als Ursache dafür, dass durch die numerische Literatur keine umfassenden Untersuchungen oder klare Antworten auf die Frage nach optimalen Lösungsverfahren hierfür zur Verfügung gestellt werden.

Dennoch bildet auch die numerische Literatur eine wesentliche Quelle für die Methoden, die für die Untersuchungen in vorliegender Arbeit in Frage kommen. Eine Beschreibung der für die verschiedenen Bereiche dieser Arbeit relevanten Literatur wird in den jeweiligen Kapiteln gegeben.

Stand der Technik hinsichtlich Steuergeräte

Derzeit ist die Simulation steifer Modelle in Echtzeitanwendungen auf Fahrzeug-Steuergeräten wenig dokumentiert. Anhand zugänglicher Beispiele wird im Folgenden der Stand der Technik beschrieben.

In [17, 116, 6] werden mathematische Modelle eingesetzt, um Luftsystemvariablen zu bestimmen. Die Lösung der steifen Anfangswertprobleme im Rahmen der Simulation steifer Modelle ist verbunden mit ausgiebigen manuellen Anpassungen im Rahmen des Verfahrens. In [17] werden implizite Integratoren vorgeschlagen und besonders das implizite Euler-Verfahren betrachtet. Die Modelle weisen Ähnlichkeiten mit dem in Abschnitt 3.2.2 beschriebenen Modell auf. Die auftretenden nichtlinearen Gleichungen werden nicht durch den Einsatz eines allgemeinen Lösungsverfahrens gelöst. Stattdessen wird - nicht zur Laufzeit, sondern im Rahmen von Untersuchungen - eine Umformung der auftretenden nichtlinearen Gleichung vollzogen. Diese Umformung wird - ebenfalls im Rahmen der Untersuchungen - approximiert, so dass eine symbolisch auflösbare nichtlineare Gleichung erhalten wird. Das auf dem Steuergerät eingesetzte Verfahren besteht darin, den symbolischen Ausdruck für diese nichtlineare Gleichung auszuwerten und ihn entsprechend der Umformungen weiter zu verarbeiten, um auf diese Weise eine gute Approximation an die Lösung der ursprünglichen nichtlinearen Gleichung zu erhalten.

Insbesondere wird hierbei kein allgemeines Verfahren zur Lösung von nichtskalaren linearen oder nichtlinearen Gleichungssystemen eingesetzt. Auch ist kein Differenziationsverfahren auf dem Steuergerät dokumentiert.

In [116] wird ebenfalls ein mathematisches Modell zur Bestimmung einer Luftsystemvariable und das implizite Euler-Verfahren als Integrator betrachtet. Hierbei werden zur Lösung der auftretenden nichtlinearen Gleichung Einschließungsverfahren, unter anderem das Pegasus-, das Illinois- und das Björck-Anderson-Verfahren (vgl. hierzu [42]) erwähnt. Bei diesen nimmt der Fehler monoton mit der Anzahl der durchgeführten Iterationen ab. Zu bemerken ist, dass im Falle, dass sich abschätzen lässt, wie schnell das Einschließungsintervall mindestens abnimmt, zudem eine Iterationszahl angegeben werden kann, bei der das Ergebnis eine bestimmte Genauigkeit erreicht hat, so dass der iterative Charakter des Verfahrens mit den Echtzeitanforderungen vereinbar ist. Eine Übertragung dieser Lösungsweise auf nichtlineare Gleichungssysteme, die nicht skalar sind, ist kaum praktikabel.

Der Einsatz linear-impliziter Integratoren in Software-Funktionen auf Fahrzeug ist noch sehr jung. Dies wurde erstmals im Jahr 2015 in der Publikation [71], der der Verfasser vorliegender Arbeit Mitautor ist, in veröffentlichter Form vorgeschlagen sowie in weiteren Veröffentlichungen betrachtet (vgl. [16, 75]). Der Einsatz linear-impliziter Integratoren führt auf weitere numerische Aufgaben wie die Differentiation oder die Gleichungslösung, die in Steuergeräte-Anwendungen

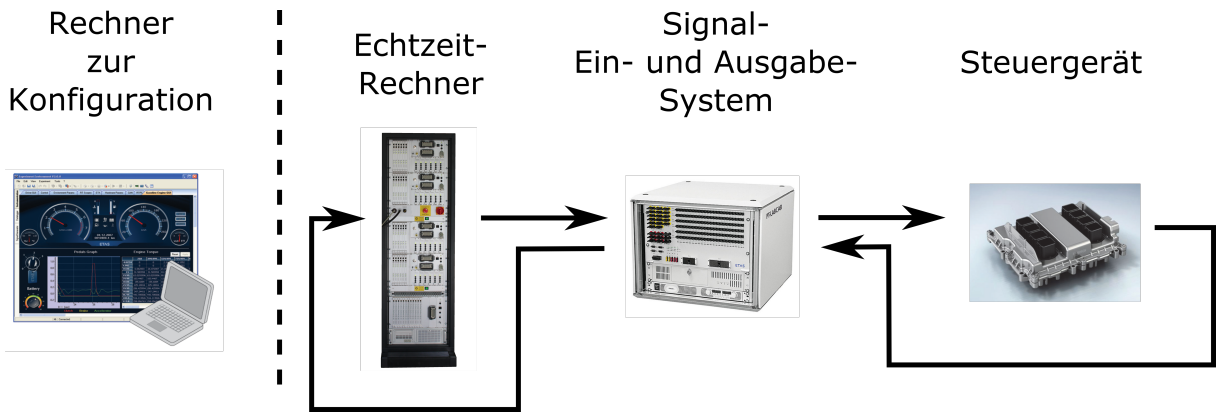


Abbildung 3.7: Schema eines Hardware-in-the-Loop Systems.

Herausforderungen mit sich bringen. Diese Herausforderungen sind in Abschnitt 3.2.3 beschrieben und werden in vorliegender Arbeit angegangen.

Etablierte Ansätze auf HiL Systemen

Ein Hardware-in-the-Loop System, auch als HiL bezeichnet, wird im Fahrzeugbereich eingesetzt, um Steuergeräte oder einzelne Funktionen davon im Echtzeitbetrieb zu testen (vgl. [62, 100, 118]). Ein Schema zu einem HiL ist in Abbildung 3.7 dargestellt.

Das HiL besteht in diesem Fall aus einem Echtzeitrechner, einem Signal-Ein- und Ausgabe-System und einem Desktop-PC zu Konfigurationszwecken. Gegebenenfalls können nach Bedarf weitere Bestandteile hinzukommen. Ein Beispiel ist, dass ein Trennadapter, auf Englisch mit *Break-Out Box* bezeichnet, angeschlossen wird, durch den gezielt Fehler für Testzwecke injiziert werden können. Ein weiteres Beispiel ist, dass eine Drosselklappe angeschlossen ist, da diese zu Herausforderungen bei einer Simulation auf dem Echtzeitrechner führt und es daher teilweise einfacher ist, auf eine reale Drosselklappe zurückzugreifen.

Das HiL erzeugt die benötigten Eingangsdaten für das Steuergerät und erfasst die Ausgabe desselben. Hierbei sollen die erzeugten Eingangsdaten in realistischer Weise durch die Ausgabe des Steuergeräts beeinflusst sein, so dass für das Steuergerät selbst ein Einsatz in einem HiL nicht von einem realen Einsatz unterscheidbar ist. Insbesondere sind durch das HiL die Daten innerhalb eines Abtastintervalls zu erzeugen, so dass also Echtzeitanforderungen bestehen. Eine wesentliche Chance, die durch HiL Systeme geboten wird, ist, dass diese erhebliche Effizienzsteigerungen in der Funktionsentwicklung eröffnen, da aufwändige Tests an realen Fahrzeugen eingespart und eine Schädigung von echten Versuchsträgern vermieden werden. Da auch bei HiL der Einsatz mathematischer Modelle vergleichbare Vorteile wie auf Steuergeräten bietet (vgl. Abschnitt 2.1.4), wurden in den vergangenen Jahren die sich ergebenden mathematischen Herausforderungen bei der Echtzeit-Simulation typischer, häufig steifer Modelle aus dem Automotive-Bereich auf HiL untersucht (vgl. [40, 90, 21, 9]). Eine gute Zusammenfassung gibt [11].

Anforderungen an numerische Verfahren auf HiL Systemen Das Umfeld von HiL bietet eine geeignete Quelle für numerische Untersuchungen an steifen mathematischen Modellen unter Echtzeitbedingungen. Ein wichtiger Unterschied zwischen den Anforderungen an numerische Verfahren auf HiL Systemen und auf Fahrzeug-Steuergeräten besteht darin, dass die Hardware des Echtzeitrechners eines HiL wesentlich leistungsfähiger als ein Steuergerät ist. Darüber hinaus ist ein HiL weniger kostensensitiv als ein Steuergerät. Hierdurch ist bei HiL Systemen die Rechenzeit, die ein numerisches Verfahren in Anspruch nimmt, weniger wertvoll und entscheidend als auf einem Steuergerät. Daher hat die Notwendigkeit zu einer Rechenzeitreduktion nicht den Stellenwert wie auf einem Steuergerät. Dies erklärt, weshalb bei den im Bereich von HiL Systemen

untersuchten Verfahren der Aspekt der Rechenzeitreduktion geringer priorisiert ist als es für die Anwendung auf Fahrzeug-Steuergeräten nötig ist und diese daher noch nicht unmittelbar für den Einsatz auf Steuergeräten günstig sind.

Numerische Verfahren auf HiL Systemen Im Folgenden wird ein Überblick über ausgewählte Methoden gegeben, die auf HiL Systemen eingesetzt werden, um steife mathematische Modelle echtzeitfähig zu simulieren und die für den Einsatz auf Steuergeräten relevant sind. Eine detaillierte Betrachtung der Verfahren wird in den jeweiligen Kapiteln der weiteren Arbeit gegeben.

Der Einsatz linear-impliziter Integrationsverfahren ist auf HiL Systemen etabliert (vgl. [10, 91, 81]). Zudem wird Wissen über das System und seine Struktur im Rahmen der numerischen Verfahren in verschiedenen Formen ausgenützt:

Ein Ansatz ist die sogenannte Mixed-Mode Integration, bei der diejenigen Zustände explizit gerechnet, bei denen dies nicht zu einer Instabilität des Simulation führt (vgl. [99, 104, 23]). Ein weiteres Konzept ist das sogenannte Sparsing wie es in [79] beschrieben ist. In abgewandelter Form wurde Sparsing in [98] auf HiL Systemen eingesetzt. Die Kernidee ist, solche Einträge der Jacobi-Matrizen zu vernachlässigen, bei denen dies nicht zu einem instabilen Simulationsverlauf führt (vgl. Abschnitt 4.2.1). Hierdurch wird die Struktur der zu lösenden linearen Gleichungssysteme dünner besetzt, was durch strukturausnutzende Lösungsverfahren in eine Rechenzeitreduktion umgemünzt wird.

Im Rahmen der Lösung linearer Gleichungssysteme ist eine weitere Form der Strukturausnützung etabliert (vgl. [80, 41, 23]). Die diesbezüglich beschriebenen Ansätze bestehen darin, durch eine geeignete Permutation der Zustände in der Formulierung des Modells zu erreichen, dass die auftretenden Jacobi-Matrizen der rechten Seite eine günstige Besetzungsstruktur aufweisen, insbesondere dass sie Block-Dreiecksform haben. Hierdurch kann die Lösung eines großen Gleichungssystems auf die Lösung mehrerer kleiner Gleichungssysteme zurückgeführt werden.

Ein Ansatz, um im Rahmen der Berechnung der Jacobi-Matrix die Anzahl der Auswertungen der rechten Seite zu reduzieren, stellt die sogenannte Colored Jacobian (vgl. [50]). Hierbei wird im Falle dünn besetzter Matrizen der zur Jacobi-Matrix gehörige Adjazenzgraph in einer Weise gefärbt, aus der hervorgeht, an welchen Stellen die rechte Seite auszuwerten ist, damit sich eine möglichst geringe Anzahl an Auswertungen zur Ermittlung der Jacobi-Matrix ergibt.

Offene Herausforderungen

Um den Einsatz steifer mathematischer Modelle, wie sie im Abschnitt 2.1.7 beschrieben sind, in Echtzeitanwendungen auf Steuergeräten zu ermöglichen, ist es zum einen nötig, für die verschiedenen Teilschritte, die während der Lösung eines steifen Anfangswertproblems durchzuführen sind, also insbesondere für die Differentiation und das Lösen auftretender Gleichungssysteme, geeignete numerische Verfahren für den Einsatz in Steuergeräte-Anwendungen zu finden. Ein hohes Maß an manuellen Lösungsanteilen, wie beispielsweise durch das explizite Ermitteln von Lösungsausdrücken oder das manuelle Festlegen der Umformungs- und Lösungsschritte eines Gleichungssystems, sind für Systeme der in Abschnitt 2.1.7 beschriebenen Größenordnung nicht mehr geeignet.

Zum anderen ist es nötig, solche numerische Verfahren einzusetzen, die zu einer möglichst großen Effizienz bezüglich der Rechenzeit führen. Die auf HiL Systemen etablierten Verfahren dienen hier als Ausgangspunkt. Eine Reduktion der Rechenzeit der Verfahren ist allerdings für den Einsatz auf Fahrzeug-Steuergeräten notwendig.

Insbesondere betrifft dies linear-implizite Integratoren. Damit diese in Steuergeräte-Anwendungen zum Einsatz kommen können, ist es nötig, geeignete Verfahren zur Differentiation und zur Gleichungslösung bereit zu stellen. Insgesamt ist hierzu eine Minimierung der Laufzeit der eingesetzten Verfahren notwendig.

3.2.5 Anforderung der Parallelisierung bei zukünftigen Fahrzeug-Steuergeräten

Ein Anspruch an die untersuchten numerischen Methoden ist ihre Eignung auch für zukünftige Steuergeräte-Generationen. Diesbezüglich ist die Entwicklung der Kernanzahl zukünftiger Steuergeräte ein wesentlicher Aspekt. Derzeit sind Multicore-Architekturen, wie beispielsweise beim Bosch MDG1 Device 4 in Abschnitt 3.1.1 beschrieben, Stand der Technik. Entsprechend der allgemeinen Entwicklung im Bereich Embedded Hardware ist davon auszugehen, dass die Kernanzahl in den kommenden Jahren rapide zunehmen wird. Manycore-Architekturen stellen den natürlichen nächsten Entwicklungsschritt für die Steuergeräte-Hardware dar.

Auf Grund der großen Zahl an Software-Funktionen, die auf einem Fahrzeug-Steuergerät betrieben werden, führt dies noch nicht unmittelbar zur Notwendigkeit auch einzelne Software-Funktionen zu parallelisieren. Eine Verteilung der Funktionen auf verschiedene Kerne, ohne die einzelnen Funktionen zu parallelisieren, ist durchaus denkbar. Allerdings ist die mit Manycore-Architekturen einhergehende, erheblich zunehmende Rechenkapazität ein Schlüssel dazu, auch rechenintensive Software-Funktionen auf Steuergeräten einzusetzen, die heute noch nicht in Frage kommen. Dies betrifft auch mathematische Modelle mit einer Größe und Komplexität jenseits der Problemklasse wie sie in vorliegender Arbeit betrachtet werden.

Gustafsons Gesetz [55] lässt erwarten, dass mit der Zunahme der Kernanzahl der Fahrzeug-Steuergeräte eine entsprechende Zunahme der Größe der mathematischen Modelle bzw. der ihnen zu Grunde liegenden Differentialgleichungssysteme, die echtzeitfähig auf Fahrzeug-Steuergeräten gerechnet werden können, erreichbar ist. Hierzu ist es notwendig, dass die eingesetzten numerischen Verfahren geeignet sind für eine Parallelisierung auf eine entsprechende Kernanzahl.

Insofern ist es eine kollaterale Anforderung an die in vorliegender Arbeit untersuchten numerischen Verfahren, dass sie die grundsätzliche Eignung zur Parallelisierung besitzen, um hierdurch auch für zukünftige Steuergeräte-Generationen geeignet zu sein.

4 Implementierung und Rechenzeitoptimierung für Echtzeitanwendung

In diesem Kapitel wird untersucht, wie sich die Rechenzeit der Simulation eines steifen Modells reduzieren lässt. Zunächst wird die Wahl des linear-impliziten Euler-Verfahrens als Integrator motiviert. Anschließend wird eine Übersicht über die benötigten Rechenzeiten der verschiedenen Anteile dieses Integrationsverfahren im Falle einer nicht weiter optimierten Implementierung gegeben. Im weiteren Kapitel wird die Reduktion der Rechenzeit der Differenziation untersucht, die den Hauptanteil der Rechenzeit in der unoptimierten Implementierung in Anspruch nimmt. Das Kernmotiv bildet hierbei, durch eine umfassende Analyse der rechten Seite eine auf den Einsatzzweck zugeschnittene inexakte Jacobi-Matrix zu wählen, die hinsichtlich Genauigkeit und Stabilität zu keinen relevanten Abstrichen führt, aber zur Laufzeit mit erheblich geringerem Rechenaufwand ermittelt werden kann.

4.1 Übersicht

Im Folgenden wird die Wahl des Integrators erläutert. Desweiteren wird für die Referenztrajektorie das Verhältnis von Modellierungsfehler und Integrationsfehler in Bezug gesetzt. Darüber hinaus werden die Rechenzeitanteile der verschiedenen Teilschritte im Rahmen des linear-impliziten Euler-Verfahrens dargestellt.

4.1.1 Wahl des Integrators

Wie bereits in Abschnitt 3.2.2 beschrieben, führen die Anforderungen einer stabilen echtzeitfähigen und steuergerätetauglichen Lösung von steifen Anfangswertproblem dazu, dass verschiedene gängige Integrationsverfahren ungeeignet sind. In diesem Abschnitt wird die Wahl des Integrators knapp diskutiert.

Ein geeigneter Integrator sollte folgende Eigenschaften besitzen:

- Eignung auch für nicht-glatte rechte Seiten
- Gute Stabilitätseigenschaften wie beispielsweise A-Stabilität
- Geringer Rechenbedarf auch in den ungünstigsten praktisch auftretenden Zeitschritten

Wie im Abschnitt 3.2.2 erläutert ist, führt die Forderung nach einem geringen Rechenbedarf auch in den ungünstigsten Zeitschritten dazu, dass eine Schrittweitensteuerung oder eine Steifheitserkennung nicht geeignet sind.

Mehrschrittverfahren Bei Mehrschrittverfahren werden Integrationswerte verschiedener Zeitschritte genutzt, um den Integrationsfehler im Fall einer hinreichender Glattheit des Systems zu verringern. Da die hier betrachteten Systeme, wie in Abschnitt 2.1.7 beschrieben, aber beispielsweise durch Eingangssignale in Form von Treppenfunktionen insbesondere an den Zeitpunkten der Zeitschritte Unstetigkeiten aufweisen können, ist die Annahme der Glattheit über Zeitschritte hinweg im Allgemeinen nicht erfüllt. Die Konsequenz ist, dass die Berücksichtigung vergangener Zeitschritte für das Resultat sogar zu einer Vergrößerung des Integrationsfehlers führen kann.

Insbesondere sind daher in diesen Fällen Einschrittverfahren einem Mehrschrittverfahren vorzuziehen.

Einschrittverfahren höherer Ordnung Bei Einschrittverfahren höherer Ordnung, wie zum Beispiel Runge Kutta Verfahren höherer Ordnung oder Extrapolationsverfahren, wird typischerweise mindestens eine Glattheit des Systems innerhalb eines Zeitschrittes vorausgesetzt (vgl. [57]).Wie in Abschnitt 2.2.2 beschrieben ist im Fall der Leitanwendung auf Grund der Beteiligung der Minimumsfunktion sowie die Berechnung der Koeffizienten, die unter anderem Betragsfunktionen beinhaltet, auch innerhalb eines Zeitschrittes keine durchgehende Differenzierbarkeit der rechten Seite gegeben. Dies führt ebenso wie bei den Mehrschrittverfahren dazu, dass der Einsatz von Einschrittverfahren höherer Ordnung den Fehler unter Umständen nicht nur nicht verringert, sondern sogar vergrößert. Dies kann eintreten, da die Verfahren auf Glattheitsannahmen beruhen, die bei der Leitanwendung nicht zutreffen und auch bei der betrachteten Problemklasse gemäß Abschnitt 2.1.7 nicht vorausgesetzt werden können. Insbesondere ist der Einsatz von Einschrittverfahren höherer Ordnung in diesen Fällen nicht geeignet.

Einschrittverfahren erster Ordnung Wichtige Vertreter der Klasse der Einschrittverfahren erster Ordnung sind das explizite, das implizite, das linear-implizite sowie das exponentielle Euler-Verfahren. Das explizite Euler-Verfahren benötigt bei steifen Systeme für eine stabile Integration viel Rechenaufwand und ist daher ungeeignet (vgl. Abschnitt 3.2.3). Im Falle eines impliziten Euler-Verfahrens stellt sich die Frage, wie auftretende nichtlineare Gleichungssysteme gelöst werden. Wie in Abschnitt 3.2.3 beschrieben, ist der Einsatz des Newton-Verfahrens zusammen mit einer fix vorgegebenen Anzahl an Newton-Iterationen auf Grund der Echtzeitanforderung zweckmäßig. Im Hinblick auf den nötigen Rechenaufwand ist es günstig, falls eine geringe Anzahl an Newton-Iterationen zu einem zufrieden stellenden Integrationsresultat führt. Insbesondere ist der Einsatz des linear-impliziten Euler-Verfahrens vorzuziehen, sofern es zu guten Integrationsresultaten führt.

Das linear-implizite Euler-Verfahren ist ein A-stabiles Integrationsverfahren, das sich im HiL-Kontext bereits bewährt (vgl. Abschnitt 3.2.4). Somit stellt es einen naheliegenden Kandidaten für das Integrationsverfahren dar.

Linear-implizites Euler-Verfahren Für das autonome Anfangswertproblem

$$\dot{x}(t) = f(x), \quad x(t_0) = x_0$$

ist das linear-implizite Euler-Verfahren definiert durch

$$(I - \Delta t J_f(x_n))\,\Delta x_n = \Delta t f(x_n), \quad x_{n+1} = x_n + \Delta x_n.$$

Der Rechenaufwand eines Zeitschrittes mit dem linear-impliziten Euler-Verfahren ist dadurch, dass keine iterative Elemente enthalten sind, gut vorhersagbar und über die verschiedenen Zeitschritte hinweg tendenziell nicht stark variierend. Zudem ist das Verfahren A-Stabil (vgl. [110, 33]). Insgesamt besitzt es somit für den Einsatz bei steifen Anfangswertproblemen und bei Echtzeitbedingungen günstige Eigenschaften. Das linear-implizite Euler-Verfahren ist eine einstufige W-Methode (vgl. [107]), so dass also eine inexakte Jacobi-Matrix nicht zu einem Ordnungsverlust führt. Es gilt sogar, dass jede Matrix anstelle der Jacobi-Matrix verwendet werden kann, ohne dass das Verfahren seine Konsistenzordnung verliert. Dies hat den Vorteil, dass sich die approximative Ermittlung der Jacobi-Matrix beispielsweise mittels einer numerischen Differenziation nicht wesentlich auf den Fehler auswirkt. Darüber hinaus eröffnet diese Eigenschaft weitergehende Möglichkeiten, um den Rechenaufwand, der mit der Ermittlung der Jacobi-Matrix zusammenhängt, zu reduzieren. Beispiele hierfür sind das Festhalten der Jacobi-Matrix und Sparsing (vgl.

Abschnitt 4.2.1). In vorliegender Arbeit ist diese Eigenschaft grundlegend für den Einsatz der Vergröberung der Jacobi-Matrix (vgl. Abschnitt 4.2.4).

Exponentielles Euler-Verfahren Das exponentielle Euler-Verfahren ist das einfachste Integrationsverfahren in der Klasse der exponentiellen Integratoren (vgl. [60]). Für ein autonomes Anfangswertproblem

$$\dot{x} = f(x), \quad x(t_0) = x_0$$

ist es gemäß [60] definiert durch

$$x_{n+1} = e^{\Delta t J_{f,x_n}} x_n + \Delta t \varphi_1(\Delta t J_{f,x_n}) (f(x_n) - J_{f,x_n} x_n)$$

mit

$$\varphi_1(z) = \frac{e^z - 1}{z}.$$

Es besitzt gute numerische Eigenschaften:

Das Verfahren ist für lineare Systeme der Form

$$\dot{x}(t) = Ax(t)$$

exakt. Darüber hinaus ist das exponentielle Euler-Verfahren wie in [60] beschrieben A-stabil und auch B-stabil. Die B-Stabilität ist ein nichtlinearer Stabilitätsbegriff, der bedeutet, dass die Dissipativität eines mathematischen Modells durch ein Integrationsverfahren erhalten wird. Eine grundsätzliche Stärke von exponentiellen Integratoren besteht bei hoch oszillativen Systemen, bei denen die Linearisierungen des Systems Eigenwerte mit einem großen Imaginärteil besitzen. Bei exponentiellen Integratoren kann, anders als beispielsweise bei impliziten Runge Kutta Verfahren, hierbei nicht das Phänomen der Ordnungsreduktion auftreten. Als Ordnungsreduktion wird bezeichnet, wenn ein Einschrittverfahren der Konsistenzordnung p trotz stabilem Integrationsverlauf eine Konvergenzordnung $q < p$ aufweist. Im Kontext von mathematischen Modellen, wie sie in Abschnitt 2.1.7 beschrieben sind, liegen keine hoch oszillativen Systeme vor, sodass dieser Vorteil bei diesen nicht wesentlich ist.

Ein zentraler Punkt, weshalb in vorliegender Arbeit das linear-implizite Euler-Verfahren gegenüber dem exponentiellen Euler-Verfahren für die gegebene Problemstellung als überlegen eingeschätzt wird, ist, dass der Berechnungsaufwand eines Zeitschrittes beim linear-impliziten Euler-Verfahren in den meisten Fällen geringer ausfallen dürfte als beim exponentiellen Euler-Verfahren und die Vorteile eines exponentiellen Integrators in vorliegender Arbeit nicht benötigt werden. Zudem ist eine Rechenzeitreduktion im Rahmen von Echtzeitsimulationen im Fall des linear-impliziten Euler-Verfahrens dadurch leichter zugänglich, dass bereits Ansätze samt theoretischer Untersuchungen bestehen, auf die aufgebaut werden kann (vgl. Abschnitt 4.2.1).

4.1.2 Referenztrajektorie

Zur Überprüfung der Daten aus der numerischen Simulation des Rohrmodells steht eine auf Messungen basierende Referenztrajektorie für die Gastemperaturen am Rohrausgang sowie der zugehörigen Eingangsverläufe zur Verfügung. Der entsprechende Datensatz erstreckt sich über einen Verlauf von 1000 s und ist mit einer Zeitschrittweite von 0.1 s gemessen. Für die Gasfraktionen stehen keine Messungen zur Verfügung. Aus diesem Grund wird die Überprüfung der Simulationsverläufe anhand der Messungen der Gastemperaturen vorgenommen.

Im weiteren Verlauf der Arbeit wird der Begriff der Referenztrajektorie in der folgenden Weise genutzt: Neben der gemessenen Trajektorie wird als Referenztrajektorie auch jeweils der Verlauf einer Simulation bezeichnet, bei dem die Eingangswerte, Parameter und ungefähren Anfangswerte der Referenztrajektorie zu Grunde gelegt werden.

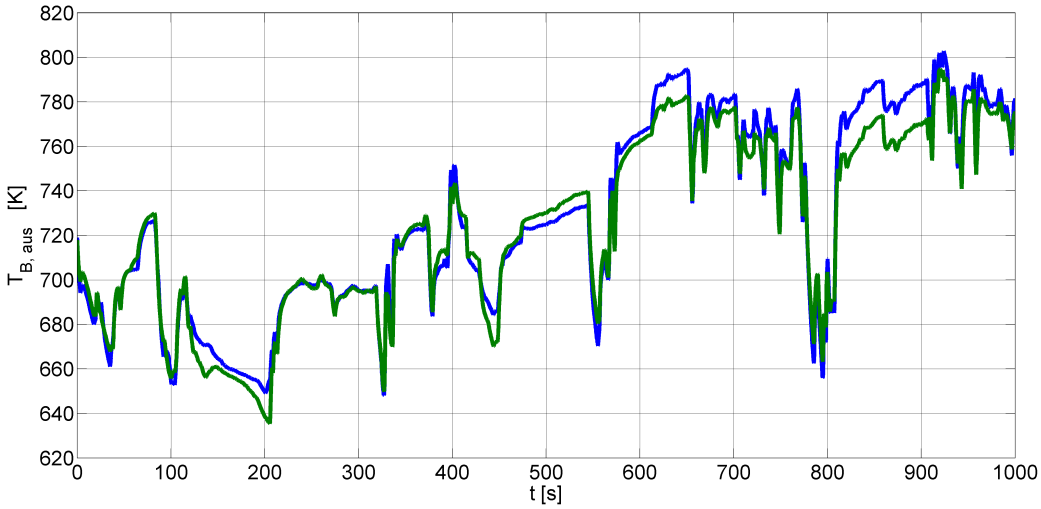


Abbildung 4.1: Verlauf der Gastemperaturen am Rohrausgang. Blau: Messung. Grün: Simulation mit linear-impliziten Euler-Verfahren und einer Zeitschrittweite von 1 ms.

In vorliegender Arbeit sind diverse Fehlerverläufe abgebildet. Die Daten für die Abbildungen werden auf einem PC errechnet. Hierbei werden 32 Bit Gleitkommavariablen genutzt. Dies stellt einen wesentlichen Zwischenschritt bei der Validierung einer Software-Funktion für ein Fahrzeug-Steuergerät dar. Die Messungen der Rechenzeiten in vorliegender Arbeit werden am Steuergerät vorgenommen.

Der Verlauf der Gastemperaturen ist der Abbildung 4.1 zu entnehmen. In dieser Abbildung ist zudem der Simulationsverlauf bei Verwendung eines linear-impliziten Euler-Verfahrens mit einer Zeitschrittweite von 1 ms dargestellt. In Abbildung 4.2 sind die betragsmäßigen Abweichungen für zwei unterschiedliche Zeitschrittweiten des linear-impliziten Euler-Verfahrens gezeigt: zum einen die Zeitschrittweite von 1 ms und zum anderen die Zeitschrittweite von 100 ms. Aus Abbildung 4.2 geht hervor, dass die betragsmäßige Abweichung der Werte aus der Simulation gegenüber den gemessenen Werten bei der Referenztrajektorie unterhalb von 20 K liegen. Der relative Fehler der Simulation liegt dementsprechend unterhalb von 3 %. Dies stellt einen sehr guten Wert für das Modell dar.

In Abbildung 4.3 ist die betragsmäßige Abweichung zwischen den Gastemperaturen am Rohrausgang bei Simulation mit einem linear-impliziten Euler-Verfahren mit einer Zeitschrittweite von 100 ms gegenüber einer Zeitschrittweite von 1 ms dargestellt. Die gegenseitige Abweichung ist in Relation zu den Abweichungen von der gemessenen Referenztrajektorie, wie sie in Abbildung 4.2 dargestellt sind, sehr gering. Dies deutet darauf hin, dass die Abweichungen der Verläufe aus der Simulation gegenüber den gemessenen Verläufen zu einem geringen Teil durch den Integrationsfehler bedingt sind und zu einem überwiegenden Anteil durch den Modellierungsfehler zu Stande kommen sowie durch den Umstand, dass die Messungen der Eingangswerte lediglich im 100 ms-Raster vorliegen.

Die in Abbildung 4.2 gezeigten betragsmäßigen Abweichungen werden durch eine Verringerung der Zeitschrittweite nicht in relevanter Weise reduziert. Aus diesem Grund werden diese Abweichungen als gegeben angenommen. Für die weitere Arbeit wird ein Simulationsfehler als akzeptabel eingestuft, wenn er die in Abbildung 4.2 dargestellten Abweichungen gegenüber den gemessenen Werte nur gering übersteigt. Aus diesem Grund wird eine Abweichung vom Simulationsverlauf des linear-impliziten Euler-Verfahren mit einer Zeitschrittweite von 1 ms von bis zu 2 K als akzeptabel eingestuft.

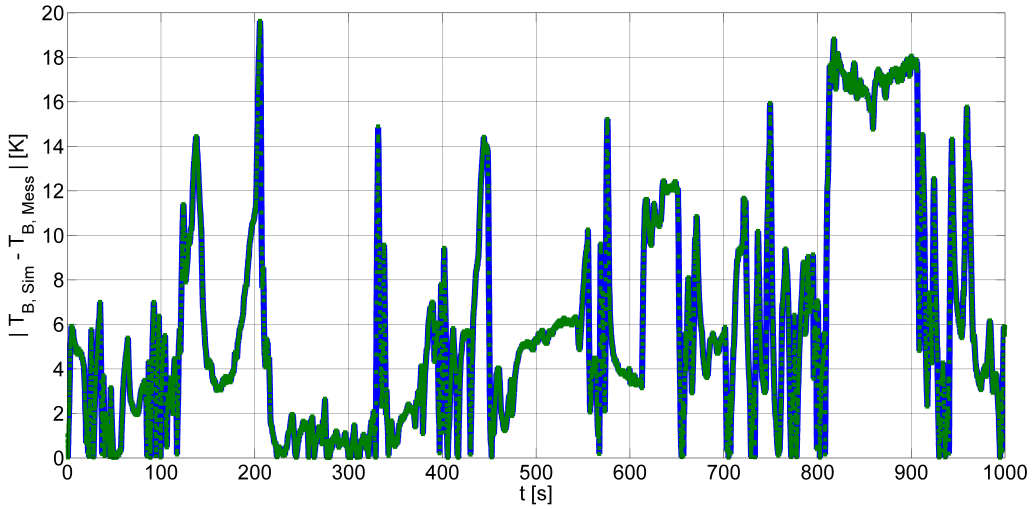


Abbildung 4.2: Verlauf der betragsmäßigen Abweichungen der Gastemperaturen am Rohrausgang aus zwei Simulationen. Blaue Linie: Simulation mit linear-impliziten Euler-Verfahren und einer Zeitschrittweite von 1 ms. Grün gepunktete Linie: Simulation mit linear-impliziten Euler-Verfahren und einer Zeitschrittweite von 100 ms.

Rechenzeit [µs]	5 Zellen	10 Zellen	15 Zellen
Gesamter Zeitschritt	4486	20500	52049
Jacobi-Matrix	3618	14104	31383
Gleichungssystem	799	6024	19917
Auswertung der rechten Seite	73	140	210

Tabelle 4.1: Rechenzeit eines Zeitschrittes des linear-impliziten Euler-Verfahrens, einer Jacobi-Matrix-Ermittlung, einer Lösung des linearen Gleichungssystems sowie einer Auswertung der rechten Seite beim betrachteten Rohrmodell für verschiedene Anzahlen an Diskretisierungszellen.

4.1.3 Initiale Laufzeitverteilung

Im Folgenden werden die Rechenzeitanteile der verschiedenen Teilschritte eines linear-impliziten Euler-Verfahrens dargestellt. Im Hinblick auf den Einsatz in einer Echtzeitanwendung wird hierbei die Rechenzeit, die in den ungünstigsten praktisch auftretenden Zeitschritten benötigt wird, als Maß herangezogen. Zur Ermittlung der Jacobi-Matrix wird hierbei eine numerische Differenziation eingesetzt, bei der zur Absicherung gegenüber Nichtlinearitäten jeweils eine zusätzliche Auswertung der rechten Seite je Richtungsvektor genutzt wird. Das lineare Gleichungssystem wird mit einer Gauß Elimination mit Spaltenpivotisierung gelöst.

In Tabelle 4.1 sind die Rechenzeiten der verschiedenen Bestandteile des Integrators zu finden. Die Rechenzeit eines einzelnen Zeitschrittes beträgt also je nach Anzahl an Diskretisierungszellen zwischen 4 ms und 53 ms. Bei einer Zeitschrittweite von 100 ms stellt dies eine Rechenzeit dar, die relativ zu ihrem Aufrufintervall sehr groß ist und damit einen wesentlichen Teil der Rechenzeit des Steuergeräts in Anspruch nimmt. Damit die Rechenzeit für einen praktischen Einsatz auf einem Steuergerät geeignet ist, ist diese wesentlich zu reduzieren.

An dieser Stelle lässt sich der Mehrwert des linear-impliziten Euler-Verfahrens gegenüber dem expliziten Euler-Verfahren verdeutlichen. Beim expliziten Euler-Verfahren ist für einen stabilen Simulationsverlauf eine Zeitschrittweite von etwa $\Delta t = 0.1$ ms nötig, so dass also etwa 1000 Auswertungen der rechten Seite je Simulationszeitschritt durchzuführen sind. Im Fall von zehn oder 15 Diskretisierungszellen ist somit eine Rechenzeit von etwa 140 ms beziehungsweise 210 ms zu

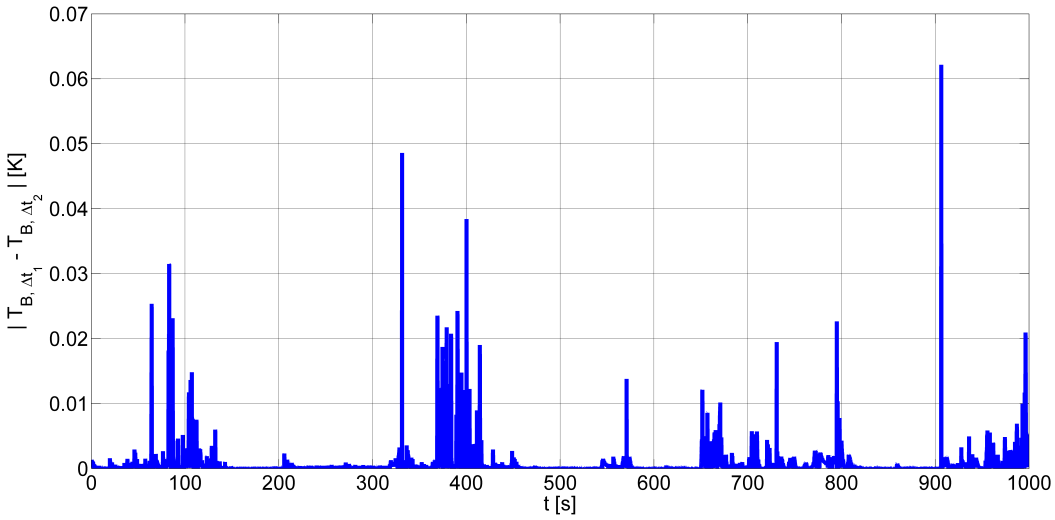


Abbildung 4.3: Verlauf der betragsmäßigen Abweichungen der Gastemperaturen am Rohrausgang einer Simulation mit linear-impliziten Euler-Verfahren und einer Zeitschrittweite von 1 ms gegenüber einer Simulation mit linear-impliziten Euler-Verfahren und einer Zeitschrittweite von 100 ms.

erwarten. Damit ist eine echtzeitfähige Simulation mit dem expliziten Euler-Verfahren nicht möglich. Das linear-implizite Euler-Verfahren ermöglicht demgegenüber eine Rechenzeit, die signifikant geringer ist: Im Fall von fünf Diskretisierungszellen kann für das explizite Euler-Verfahren eine benötigte Rechenzeit von etwa 73 ms je Simulationszeitschritt mit einem Schrittweite von 100 ms angenommen werden, während das linear-implizite Euler-Verfahren lediglich etwa 4.5 ms benötigt.

Für die Rechenzeit der Lösung eines linearen Gleichungssystems ist bei einer zunehmenden Anzahl an Diskretisierungszellen ein kubisches Skalierungsverhalten zu erwarten, während bei der Ermittlung der Jacobi-Matrix ein quadratisches Skalieren mit der Anzahl der Diskretisierungszellen naheliegend ist. Es ist zu erkennen, dass bei den betrachteten Anzahlen an Diskretisierungszellen allerdings der Aufwand der Ermittlung der Jacobi-Matrix dominiert. Somit ist das vorrangige Ziel, die dafür benötigte Rechenzeit zu reduzieren. Dies wird im Teilkapitel 4.2 betrachtet.

Für den Fall, dass die Rechenzeit für die Ermittlung der Jacobi-Matrix erheblich reduziert werden kann, ist bei zehn oder 15 Diskretisierungszellen die Rechenzeit zur Lösung des linearen Gleichungssystems für sich allein ebenfalls größer als es für die Gesamtrechenzeit eines Zeitschrittes wünschenswert ist. Insofern ist in jedem Fall auch eine Reduktion der Rechenzeit der Lösung des linearen Gleichungssystems zu untersuchen. Dieses Ziel wird in Kapitel 5 verfolgt.

4.2 Jacobi-Matrix-Approximation

Im Falle einer nicht optimierten Anwendung des linear-impliziten Euler-Verfahrens nimmt die Differenziation den größten Anteils der Rechenzeit im Rahmen der Simulation des Rohrmodells ein (vgl. 4.1.3). Im Folgenden wird untersucht, wie sich der Rechenaufwand reduzieren lässt, ohne hierbei Einbußen in der Genauigkeit oder Stabilität des resultierenden Simulationsverlaufs in Kauf nehmen zu müssen. Den Schlüssel hierzu bildet das linear-implizite Euler-Verfahren, da es eine W Methode ist und somit auch mit einer inexakten Jacobi-Matrix gute Resultate liefern kann. Diese Eigenschaft wird genutzt, indem durch eine detaillierte Analyse der rechten Seite eine inexakte Jacobi-Matrix gewählt wird, die mit geringem Aufwand berechenbar ist und zu geeigneten Simulationsverläufen führt.

4.2.1 Literaturübersicht

In diesem Abschnitt sind Methoden zusammengestellt und erläutert, die im Rahmen der Berechnung von Jacobi-Matrizen bei der Lösung von Anfangswertproblemen zum Einsatz kommen können. Insbesondere umfassen sie den diesbezüglichen Stand der Technik bei Echtzeitsimulationen steifer Modelle auf HiL Systemen. Einer dieser Ansätze, das sogenannte *Sparsing*, wird in Abschnitt 4.2.2 weiter vertieft, da die damit verbundenen Überlegungen im weiteren Kapitel in modifizierter Weise genutzt werden.

Färbung der Jacobi-Matrix

In [50] wurde die Färbung der Jacobi-Matrix eingeführt, auf Englisch mit *colored Jacobian matrix* bezeichnet. Der Ansatz ist anwendbar bei Differenziationsverfahren, die auf der Berechnung von Richtungsableitungen basieren, beispielsweise bei der numerischen und der automatischen Differenziation. Die Absicht hinter der Färbung besteht darin, die Anzahl der für die Ermittlung einer Jacobi-Matrix nötigen Richtungsableitungen zu reduzieren. Dies wird dadurch erreicht, dass geeignete Richtungsvektoren bei den Richtungsableitungen gewählt werden. Dies wird zunächst an einem Beispiel illustriert:

Für das steife mathematische Modell

$$\dot{x}(t) = f(t, x)$$

mit Jacobi-Matrix $J_f(t,x) = (j_{ik}(t,x))_{i,k}$ sei die Strukturmatrix $S_f = (s_{ik})_{i,k}$ definiert als

$$s_{ik} = \begin{cases} 1 & \text{, falls } j_{ik}(t,x) = 0 \text{ für alle } t,\ x, \\ 0 & \text{, sonst.} \end{cases}$$

Als Beispiel sei ein Modell mit folgender Strukturmatrix gegeben:

$$S_f = \begin{pmatrix} 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 \end{pmatrix}.$$

Dann genügt das Berechnen zweier Richtungsableitungen $\frac{\partial f}{\partial d_1}, \frac{\partial f}{\partial d_2}$ zur Ermittlung von J_f:
Wähle $d_1 = (\delta_{1,1}, \delta_{1,2}, 0, 0, \delta_{1,5})^T$ und $d_2 = (0, 0, \delta_{2,3}, \delta_{2,4}, 0)^T$ mit $\delta_{i,k} \neq 0$. Es ist

$$\frac{\partial f}{\partial d_1} = J_f d_1 = \begin{pmatrix} \delta_{1,1} j_{11} & \delta_{1,2} j_{22} & \delta_{1,2} j_{32} & \delta_{1,1} j_{41} & \delta_{1,5} j_{55} \end{pmatrix}^T.$$

Aus $\frac{\partial f}{\partial d_1}$ ergeben sich somit unmittelbar die Spalten 1, 2 und 5 der Jacobi-Matrix. Aus

$$\frac{\partial f}{\partial d_2} = J_f d_2 = \begin{pmatrix} \delta_{2,4} j_{14} & \delta_{2,3} j_{23} & \delta_{2,3} j_{33} & \delta_{2,4} j_{44} & \delta_{2,4} j_{54} \end{pmatrix}^T$$

ergeben sich unmittelbar die Spalten 3 und 4. Durch die Färbung der Jacobi-Matrix sind somit lediglich zwei anstelle von fünf Richtungsableitungen zur Ermittlung der Jacobi-Matrix ausreichend.

Im allgemeinen Fall lässt sich das Finden geeigneter Richtungsableitungen als Färbungsproblems eines Graphen formulieren (vgl. [50]). Dieses ist im Allgemeinen NP-hart und somit nicht praktikabel lösbar. Allerdings sind effiziente Routinen für Näherungslösungen, wie beispielsweise der MATLAB-Befehl `colgroup`, verfügbar.

Für dünn besetzte Jacobi-Matrizen sind durch diesen Ansatz teilweise nennenswerte Reduktionen des Rechenaufwands möglich. In [20] wird er am Beispiel zweier Fluid-Modelle im Kontext von Kraftwerken untersucht. Hierbei werden durch den Ansatz Laufzeitreduktionen zwischen 50 % und 80 % erzielt.

Der Einsatz einer gefärbten Jacobi-Matrix ist für Echtzeitsimulationen gut geeignet. Dies wird für die Leitanwendung in Abschnitt 4.4 genutzt.

CPR-Heuristik Ein Ansatz, der eine ähnliche Intention wie die Färbung der Jacobi-Matrix hat, ist die CPR-Heuristik (vgl. [24]). Sie stellt eine Verallgemeinerung des Ansatzes der Färbung der Jacobi-Matrix dar. Diese kann dementsprechend eingesetzt werden, um bei einer dünn besetzten Jacobi-Matrix die Anzahl an Richtungsableitungen zu reduzieren, die ermittelt werden, um eine Jacobi-Matrix zu bestimmen. Hierzu wird die Annahme

$$J_f\left(x_n\right) \Delta x^{(i)} \approx f\left(x_n + \Delta x^{(i)}\right) - f\left(x_n\right), \quad i = 1, \ldots, m$$

zu Grunde gelegt. Diese m Gleichungen lassen sich als lineares Gleichungssystem interpretieren, bei dem die nicht verschwindenden Einträge der Jacobi-Matrix die Unbekannten bilden. Somit wird die Jacobi-Matrix durch das Bilden von m Richtungsableitungen und dem Lösen eines linearen Gleichungssystems ermittelt. Das Gleichungssystems hat hierbei eine überaus simple Form, so dass der Aufwand zur Lösung sehr gering ist.

Sei m_0 die maximale Anzahl an strukturell nicht verschwindenden Einträge innerhalb einer Zeile einer Jacobi-Matrix. Dann ist es durch das Vorgehen möglich, dass maximal m_0 Richtungsableitungen für die Ermittlung der Jacobi-Matrix genügen. Dies kann bei einer Färbung der Jacobi-Matrix nicht im allgemeinen Fall erzielt werden. Analog zu der in Abschnitt 3.2.3 beschriebenen Anfälligkeit der numerischen Differenziation gegenüber starken Nichtlinearitäten der rechten Seite weist auch die Ermittlung der Jacobi-Matrix basierend auf der CPR-Heuristik eine entsprechende Anfälligkeit auf. Ein Einsatz bedarf somit einer sorgfältigen Prüfung, ob im jeweiligen Einzelfall die erhaltenen Resultate akzeptabel sind.

Festhalten der Jacobi-Matrix

Das Festhalten der Jacobi-Matrix, auf Englisch mit *time-lagged Jacobian matrix* bezeichnet, wurde in [114] eingeführt. Die zu Grunde liegende Idee ist, dass sich bei dynamischen Systemen die Jacobi-Matrix teilweise über mehrere Zeitschritte hinweg nur wenig verändert. In solchen Fällen ist es möglich, die Neuberechnung der Jacobi-Matrix nicht in jedem Zeitschritt durchzuführen, falls ein Integrator verwendet wird, bei dem eine inexakte Jacobi-Matrix nicht zu einem Ordnungsverlust führt. Bei den im Rahmen dieser Arbeit bevorzugten W Methoden, insbesondere für das linear-implizite Euler-Verfahren, ist die Verwendung einer inexakten Jacobi-Matrix ohne Auswirkungen auf die Konsistenzordnung möglich.

Das Festhalten der Jacobi-Matrix ist zunächst für den Einsatz in Echtzeitsimulationen nicht geeignet, da nur die durchschnittliche Rechenzeit eines Zeitschrittes, nicht aber die größte praktisch auftretende Rechenzeit eines Zeitschrittes reduziert wird (vgl. 3.2.2). In Abschnitt 4.4 wird beschrieben, wie sich der Ansatz für den Einsatz in Echtzeitsimulationen anpassen lässt.

Eine extreme Form des Festhalten der Jacobi-Matrix besteht darin, über alle Zeitschritte hinweg die Jacobi-Matrix nicht neu zu berechnen. In diesem Fall wird also eine konstante Jacobi-Matrix verwendet. Im Kontext von Echtzeitsimulationen ist es möglich, diese Jacobi-Matrix samt einer geeigneten LU- oder QR-Zerlegung vor der Laufzeit zu ermitteln, so dass während der Laufzeit nur ein geringer Rechenaufwand verbleibt.

Mixed-Mode Integration

In [99, 23] wird die sogenannte Mixed-Mode Integration beschrieben. Dieser Ansatz beruht auf der Idee, dass bei einem steifen Modell diejenigen Zustände mit einem expliziten Integrator berechnet werden, bei denen dies in stabiler Weise möglich ist. Nur für die verbleibenden Zustände wird ein stabiler Integrator eingesetzt.

Durch diese Vorgehensweise wird erreicht, dass die im Rahmen einer impliziten Integration zu lösenden Gleichungssysteme auf eine geringere Dimension eingeschränkt werden können und der Lösungsaufwand entsprechend reduziert wird.

Günstig anwendbar ist dieser Ansatz gemäß [99] vor allem auf Modelle, die Komponenten mit Dynamiken unterschiedlicher Zeitskalen besitzen. Wie in Abschnitt 2.1.5 beschrieben, trifft dies auf mechatronische Systeme durchaus zu. Daher ist ein Einsatz in diesem Kontext grundsätzlich geeignet. Eine Verallgemeinerung hiervon stellt eine Form des sogenannten Sparsings dar, welches im Rahmen dieses Abschnitts beschrieben wird.

Sparsing-Methoden

Sparsing-Methoden zielen zunächst auf den Rechenaufwand bei der Gleichungslösung. Da sie wichtige Ideen beinhalten, die im Rahmen der Berechnung beziehungsweise Approximation von Jacobi-Matrizen essentiell sind, werden sie bereits im Folgenden vorgestellt.

Die Grundidee von Sparsing-Methoden besteht darin, den Rechenaufwand zur Integration einer steifen Differentialgleichung dadurch zu reduzieren, dass als Integrator eine W-Methode gewählt und der Aufwand zur Lösung der auftretenden linearen Gleichungssysteme verringert wird, indem ihre Struktur künstlich in einer Weise ausgedünnt wird, dass dies für die Dynamik des Systems nur einen geringen Einfluss hat. Diese Ausdünnung wird durch Lösungsverfahren für dünn besetzte Gleichungen in eine Reduktion der Rechenzeit umgemünzt. Sie kann einerseits in jedem Zeitschritt neu vorgenommen werden. Dies wird als dynamisches Sparsing bezeichnet. Andererseits kann die Ausdünnung im Kontext von Echtzeitsimulationen auch bereits vor der Implementierung durchgeführt werden, so dass während der Laufzeit keine Berechnung von Sparsing-Kriterien notwendig wird.

Dynamisches Sparsing

Dynamisches Sparsing wurde in [79] eingeführt. Als Ausdünnungskriterium dient ein Schwellwert, bei dessen betragsmäßiger Unterschreitung ein Eintrag der Jacobi-Matrix $J_f\left(t_n, x_n\right)$ Null gesetzt wird. Dieser Ansatz zeigt daher insbesondere im Fall von Jacobi-Matrizen mit vielen betragsmäßig kleinen Termen Wirkung. Zum Ausdünnen von betragsmäßig großen Jacobi-Matrix-Einträgen, die auf die Dynamik des Systems einen geringen Einfluss haben, ist dieser Ansatz nicht geeignet. Zu beachten ist, dass, wie in [97] beschrieben, sich der Einfluss eines Jacobi-Matrix-Eintrags auf die Dynamik des Systems allein durch seinen Betrag bei steifen Systemen nicht charakterisieren lässt.

Das Sparsing gemäß [79] ist für den Einsatz bei Echtzeitsimulationen nicht ideal geeignet. Denn dadurch, dass die Ausdünnung weniger einen physikalischen Bezug hat und stattdessen eher auf rein quantitativen Aspekten beruht, bestehen zwei potentielle Schwachpunkte: Einerseits besteht die Gefahr unerwünschter Effekte auf die Dynamik, falls der Schwellwert für die Ausdünnung zu groß gewählt wird. Andererseits kommt der Aspekt zum Tragen, dass weniger ausgedünnt wird als möglich, falls der Schwellwert zu klein gewählt wird. Eine Berücksichtigung der Eigenschaften der Dynamik des Systems führt im Vergleich hierzu zu einer anpassungsfähigeren und differenzierteren Form der Ausdünnung.

Sparsing für Echtzeitsimulationen

Neben dem dynamischen Sparsing ist auch das Sparsing gemäß [98] und [97] eine wichtige Form der Ausdünnung. Dieses ist gezielt für Echtzeitsimulationen konzipiert, wobei keine Steuergeräte, sondern PC-Hardware als Zielhardware genutzt und zudem differentiell-algebraische Systeme berücksichtigt werden. Bei diesem Ansatz wird die linearisierte Gleichung des Integrators zunächst mit dem QZ-Algorithmus auf Blockdiagonalform mit Dreiecksmatrizen auf der Diagonalen transformiert. Für jeden Eintrag der entstehenden Matrix wird dann anhand eines Kriteriums ermittelt, ob dieser Null gesetzt werden kann, ohne dass hierdurch die Dynamik des Systems wesentlich verändert wird. Hierdurch wird die Struktur des zu lösenden linearen Gleichungssystem ausgedünnt. Durch den Einsatz strukturausnutzender Gleichungslöser kann damit die Rechenzeit zur Lösung des linearen Gleichungssystems reduziert werden. Die Pointe des Kriteriums ist unter anderem, dass der Rechenaufwand für einen einzelnen Eintrag im Wesentlichen lediglich in der Berechnung eines Skalarproduktes besteht.

Das Sparsing wird in [98] zur Laufzeit betrieben, während es in [97] nicht zur Laufzeit, sondern bereits vorab im Rahmen einer Systemanalyse durchgeführt wird, so dass zur Laufzeit kein zusätzlicher Rechenaufwand durch das Sparsing verursacht wird. Da im Rahmen der Analyse, die nicht zur Laufzeit stattfindet, reichlich Rechenzeit zur Verfügung steht, kann dort ein aufwändiger Optimierungsalgorithmus zur Wahl des Sparsings eingesetzt werden. Ansonsten sind die Ansätze in beiden Arbeiten weitgehend deckungsgleich.

Als Indikator dafür, ob das Nullsetzen eines Eintrags einen nennenswerten Einfluss auf die Dynamik des Systems hat, werden die durch die Ausdünnung entstehenden Änderungen der Eigenwerte des durch die linearisierte Differenzengleichung

$$(I - \Delta t A)\, x_{n+1} = (I + \Delta t\, (J_f - A))\, x_n \tag{4.1}$$

induzierten Matrixpaares

$$(B, C) = (I - \Delta t A, I + \Delta t\, (J_f - A))$$

verwendet. Hierbei steht J_f für die exakte Jacobi-Matrizen und A für die jeweils verwendete Approximation. Die diskrete Evolution in Gleichung 4.1 ist durch die verallgemeinerten Eigenwerte von (B, C) nur zum Teil beschrieben, da weder die verwendete Basis noch die jeweiligen Eigenräume erfasst sind. Dennoch erscheint die Wahl der Eigenwerte als Indikator zweckmäßig: Einerseits ist er noch praktikabel zugänglich, andererseits dürfte sich im Fall, dass das Nullsetzen eines Jacobi-Matrix-Eintrags einen wesentlichen Einfluss auf die diskrete Evolution nach sich zieht, dies in vielen Fällen auch auf die Eigenwerte niederschlagen.

4.2.2 Ein Ausdünnungskriterium

Das Kriterium, das in [98] entwickelt wird, erlaubt es, die Summe $\sum_i \left(\tilde{\lambda}_i - \lambda_i\right)$ der Abweichungen der Eigenwerte $\tilde{\lambda}_i$ nach dem Ausdünnen gegenüber den Eigenwerten λ_i vor dem Ausdünnen zu approximieren, ohne hierzu die Eigenwerte selbst oder die Eigenvektoren zu berechnen. Das entsprechende Theorem 4 aus [98] lautet:

Theorem 1. *Sei (B, C) ein diagonalisierbares Matrixpaar mit Eigenwerten λ_i und Linkseigenvektoren y_i sowie Rechtseigenvektoren x_i. Alle Eigenwerte werden als algebraisch simpel angenommen sowie C als regulär. Sei E ein hinreichend kleine $\mathcal{O}(\epsilon)$ Störmatrix. Seien $\tilde{\lambda}_i$ die Eigenwerte des gestörten Matrixpaares $(B + E, C + E)$. Dann ist eine Approximation erster Ordnung für die Summe der Differenzen zwischen gestörten Eigenwerte und den originalen Eigenwerten gegeben durch*

$$\sum_i \left(\tilde{\lambda}_i - \lambda_i\right) = \mathit{Spur}\left(C^{-1} E \left(I - C^{-1} B\right)\right) + \mathcal{O}\left(\epsilon^2\right).$$

Als Korollar ergibt sich für das Matrixpaar $(B, C) = ((I - \Delta t A), I + \Delta t (J_f - A))$, sofern die Voraussetzungen erfüllt sind, folgende Approximation:

$$\left| \sum_i \left(\tilde{\lambda}_i - \lambda_i \right) \right| \approx \left| \text{Spur} \left((I - \Delta t J_f)^{-1} \Delta t \Delta J \left(I - (I - \Delta t J_f)^{-1} (I + \Delta t J_f) \right) \right) \right|. \tag{4.2}$$

Hierbei ist $\Delta J = A - J_f$. Diese Approximation bildet die Grundlage der Bestimmung einer Ausdünnung.

Eigenschaften Die Approximation 4.2 nähert die Summe der Veränderungen der Eigenwerte der diskreten Evolution. Ist die Veränderung jedes einzelnen Eigenwerts hinreichend klein, so bleibt die Stabilität der Integration, bei der keine Ausdünnung vorgenommen wurde, erhalten. Zudem wird dann die Dynamik des Systems durch das Sparsing unwesentlich verändert. Die Auswirkung auf die Genauigkeit der Integration wird im Rahmen der Untersuchungen in [98] folgendermaßen beschrieben: Dadurch dass als Integrator eine W Methode eingesetzt wird, führt das Verwenden einer inexakten Jacobi-Matrix nicht zu einer Verringerung der Konsistenz- oder, im Falle einer stabilen Integration, Konvergenzordnung. Für hinreichend kleine Zeitschrittweite ist auch beim Einsatz eines Sparsings daher eine ausreichende Genauigkeit zu erwarten.

Eine Voraussetzung dafür, dass die Approximation 4.2 aussagekräftig ist, ist, dass die nichtlinearen Anteile der Störterme, die durch die Veränderung der Jacobi-Matrix induziert werden, gering sind. Um dies in möglichst vielen Fällen zu erreichen, wird in [98] vorgeschlagen, die Wahl der Diagonalblöcke im Rahmen der Blockdiagonalisierung so zu wählen, dass nicht in verschiedenen Diagonalblöcken nahe beieinander liegende Eigenwerte enthalten sind. Wie in [97] beschrieben, wird hierbei zunächst eine Triangularisierung der Matrix der linearisierten diskreten Evolution vorgenommen. Anschließend werden mittels Givens-Rotationen eine geeignete Gruppierung der Eigenwerte durchgeführt. Danach wird eine Blockdiagonalisierung entsprechend der Gruppierung der Eigenwerte vollzogen.

Auslöschungseffekte In [98] ist beschrieben, dass durch Auslöschungseffekte bei der Approximation 4.2 im Rahmen eines Sparsings die Gefahr besteht, dass Jacobi-Matrix-Einträge ausgedünnt werden, die einen relevanten Einfluss auf die Dynamik des Systems haben. Dies wird in denjenigen Fällen nicht durch die Approximation 4.2 erfasst, bei denen sich die Abweichungen bei den verschiedenen Eigenwerten auslöschen. Das Risiko hierzu besteht grundsätzlich und wird [98] insbesondere bei oszillierenden Moden als beträchtlich gesehen.

Eine Erweiterung der Approximation, bei der keine Auslöschungseffekte auftreten, da anstelle von

$$\sum_i \left(\tilde{\lambda}_i - \lambda_i \right)$$

der Term

$$\sum_i \left| \tilde{\lambda}_i - \lambda_i \right|^2$$

approximiert wird, wird in Abschnitt 4.2.3 vorgestellt.

Resultate In [98] wird am Beispiel einer Echtzeitsimulation eines mathematischen Modells auf einem Industrieroboter Sparsing dargestellt. Hierbei wird durch das Sparsing eine erhebliche Vereinfachung bei der Struktur des linearen Gleichungssystems erzielt, die zu einer Reduktion der zur Lösung des auftretenden linearen Gleichungssystems benötigten Rechenzeit um über 65 % führt.

Sparsing als Verallgemeinerung der Mixed-Mode Integration Sparsing umfasst als Spezialfall die Mixed-Mode Integration: Sei I die Menge der Indizes von denjenigen Komponenten eines stei-

fen mathematischen Modells, die im Rahmen einer Mixed-Mode Integration mit einem expliziten Integrator stabil gelöst werden. Dann führt das Nullsetzen aller Spalten der Jacobi-Matrizen dieses mathematischen Modells mit einem Index $i \in I$ sowie aller Zeilen mit einem Index $i \in I$ gerade auf ein Gleichungssystem, das äquivalent zur Mixed-Mode Integration ist.

4.2.3 Erweiterung des Ausdünnungskriteriums

Das Ausdünnen der Jacobi-Matrix gemäß Schiela und Bornemann, wie es in Abschnitt 4.2.1 und Abschnitt 4.2.2 beschrieben ist, ist dediziert für Echtzeitsimulationen konzipiert. Der Einsatz eines Sparsings nicht während der Laufzeit, sondern bereits vorab wie in [97] dargestellt, ist wie in Abschnitt 4.3.1 ausgeführt nutzbringend einsetzbar bei der Echtzeitsimulation eines steifen Modells auf einem Fahrzeug-Steuergerät. Das Kriterium basierend auf Gleichung 4.2 dient hierbei dazu, zum Sparsing geeignete Einträge der Jacobi-Matrizen nahe zu legen. Die letztliche Überprüfung, ob eine gewählte Menge an Einträgen geeignet ist, um auf Null gesetzt zu werden, wird anhand der Resultate bei der Referenztrajektorie vorgenommen.

Die Konzepte, die in [98] erläutert werden, sind in vorliegender Arbeit auch über das Sparsing hinaus anwendbar. Sie bilden das Fundament, um in Abschnitt 4.3.2 geeignete Vergröberungen der Jacobi-Matrix zu finden. Aus diesem Grund wird im Folgenden die theoretische Grundlage des Sparsings dargestellt. Hierbei wird der Aufbau gegenüber [98] leicht verändert, da in vorliegender Arbeit keine differentiell-algebraische Gleichungssysteme betrachtet werden und die Modifikation in Abschnitt 4.3.1 günstig sein wird. Darüber hinaus wird das Ausdünnungskriterium basierend auf Gleichung 4.2 in einer Weise erweitert werden, dass eines seiner zentralen Defizite behoben wird:

Wie in 4.2.2 beschrieben, sind Auslöschungseffekte eine Ursache, die dazu führen kann, dass der Wert $\left|\sum_i \left(\tilde{\lambda}_i - \lambda_i\right)\right|$ in Gleichung 4.2 eine geringere Beeinflussung der Dynamik des Simulationsverlaufs vorhersagt, als tatsächlich eintritt. Im weiteren Verlaufs dieses Abschnitts wird eine Variation von Gleichung 4.2 beschrieben, die es ermöglicht, derartige Auslöschungseffekte zu verhindern. Die folgenden Darstellungen orientieren sich wie erwähnt bis auf kleine Modifikationen an [98].

Die Integrationsgleichung für einen Zeitschritt hat die Form:

$$x_{n+1} = \left(I + (I - \Delta t J_f)^{-1} \Delta t J_f\right) x_n = F x_n. \tag{4.3}$$

Für F wird zunächst die Schur Form mittels eines QR-Algorithmus berechnet: Es sei $T = U^H F U$ mit einer unitären Matrix U und einer Dreiecksmatrix T. Auf der Diagonalen stehen die Eigenwerte von F. Nun wird eine Umordnung der Eigenwerte derart vorgenommen, dass die Eigenwerte in T zusammen gruppiert sind entsprechend ihrer Größe. Der Zweck der Gruppierung der Eigenwerte wird im weiteren Verlauf des Abschnitts deutlich. Hieraus ergibt sich dann auch eine Präzisierung der Anforderung an eine derartige Gruppierung in Cluster S_i. Zur Umordnung der Eigenwerte entsprechend der Wahl der Gruppierung werden Givens-Rotationen eingesetzt. Ohne Beschränkung der Allgemeinheit kann im Folgenden auch diese umgeordnete Schur-Form mit $T = U^H F U$ bezeichnet werden.

Nun wird eine Block-Diagonalisierung durchgeführt. Dies wird in einer Weise vollzogen, dass die Teilmengen von Spalten, die zu den verschiedenen Clustern S_i von Eigenwerten gehören, orthogonal sind. Hierzu wird folgender Algorithmus aus [97] gewählt:

Im Folgenden bezeichne $P_{i,j}$ die kanonische Projektion auf den Raum, der durch die Einheitsvektoren $\{e_k | \, i \leq k \leq j\}$ erzeugt wird, also

$$P_{i,j} e_k = \begin{cases} e_k & \text{, falls } i \leq k \leq j, \\ 0 & \text{sonst.} \end{cases}$$

Seien $T_1 = U_1^H F U_1$ und $T_2 = U_2^H F U_2$ zwei Schur Formen von F, wobei in T_1 alle Eigenwerte eines Clusters S_{i_0} in den ersten k Diagonaleinträgen sind, während in T_2 alle Eigenwerte des Clusters S_{i_0} in derselben Reihenfolge in den letzten k Diagonaleinträgen sind. Sei

$$\begin{aligned} Y_2 &= P_{k+1,n} U_1, \\ Y_1 &= P_{n-k+1,n} U_2, \\ X_1 &= P_{1,k} U_1, \\ X_2 &= P_{1,n-k} U_2. \end{aligned}$$

Die Menge der Spalten ist bei jeder dieser Matrizen orthogonal. Desweiteren gilt $Y_1^H F X_2 = 0$ sowie $Y_2^H F X_1 = 0$. Hierdurch folgt:

$$\begin{pmatrix} Y_1^H \\ Y_2^H \end{pmatrix} F \begin{pmatrix} X_1 & X_2 \end{pmatrix} = \begin{pmatrix} T^{(1)} & 0 \\ 0 & T^{(2)} \end{pmatrix}.$$

Entsprechend lässt sich $T^{(2)}$ für die weiteren Cluster $S_i \neq S_{i_0}$ umformen. Durch iteratives Vorgehen lässt sich hierdurch eine Block-Diagonalform mit Dreiecksmatrizen auf den Diagonalblöcken entsprechend der Wahl der Gruppierung der Eigenwerte erhalten. Falls die Matrix F diagonalisierbar ist, lassen sich dementsprechend Matrizen X und Y erzielen, so dass $Y^H F X$ eine Diagonalmatrix ist.

Nun werden die Auswirkungen einer Störung ϵM der Matrix F auf die Eigenwerte charakterisiert. Hierbei wird F als ohne mehrfache Eigenwerte angenommen. Insbesondere ist F also diagonalisierbar. Sei

$$\begin{pmatrix} Y_1^H \\ Y_2^H \end{pmatrix} M \begin{pmatrix} X_1 & X_2 \end{pmatrix} = \begin{pmatrix} M_{11} & M_{12} \\ M_{21} & M_{22} \end{pmatrix}.$$

Dann gibt es gemäß Theorem 2.8 in [108] Transformationsmatrizen $\tilde{X}, \tilde{Y}$ mit

$$\tilde{Y}^H (F + \epsilon M) \tilde{X} = \begin{pmatrix} T^{(1)} + \epsilon M_{11} + \epsilon M_{12} P & 0 \\ 0 & T^{(2)} + \epsilon M_{22} + P \epsilon M_{12} \end{pmatrix},$$

wobei $\tilde{X}_1 = X_1 + P X_2$ und $\tilde{Y}_2 = Y_2 - Y_1 P^H$ ist mit

$$\|P\| \leq 2 \frac{\|\epsilon M_{21}\|}{\operatorname{sep}\left(T^{(1)}, T^{(2)}\right) - \|\epsilon M_{22}\| - \|\epsilon M_{22}\|} \tag{4.4}$$

und

$$\operatorname{sep}\left(T^{(1)}, T^{(2)}\right) = \left(\inf_{\|Q\|=1} \left\| Q T^{(1)} - T^{(2)} Q \right\| \right) > 0. \tag{4.5}$$

Ferner bilden die Spaltenvektoren von $\tilde{X}_1$ beziehungsweise von $\tilde{Y}_2$ Basen für die einfachen rechtsbeziehungsweise links-invarianten Unterräume von $F + \epsilon M$.

Aus Gleichung 4.4 und Gleichung 4.5 folgt $\|P\| = \mathcal{O}(\epsilon)$ und daher

$$\tilde{X}_1 = X_1 + \mathcal{O}(\epsilon) \frac{P}{\|P\|} X_2 \tag{4.6}$$

und

$$\tilde{Y}_2 = Y_2 - Y_1 \mathcal{O}(\epsilon) \frac{P^H}{\|P^H\|}. \tag{4.7}$$

Mit analoger Argumentation lässt sich auch

$$\tilde{X}_2 = X_2 + \mathcal{O}(\epsilon) \frac{\hat{P}}{\left\|\hat{P}\right\|} X_1 \tag{4.8}$$

und

$$\tilde{Y}_1 = Y_1 - Y_2 \mathcal{O}(\epsilon) \frac{\hat{P}^H}{\left\|\hat{P}^H\right\|} \tag{4.9}$$

folgern. Insgesamt folgt weiter

$$\tilde{Y}^H (F + \epsilon M) \tilde{X} = \begin{pmatrix} T^{(1)} + \epsilon M_{11} + \mathcal{O}\left(\epsilon^2\right) & \mathcal{O}\left(\epsilon^2\right) \\ \mathcal{O}\left(\epsilon^2\right) & T^{(2)} + \epsilon M_{22} + \mathcal{O}\left(\epsilon^2\right) \end{pmatrix}.$$

An dieser Stelle kommt die Wahl der Gruppierung der Eigenwerte zum Tragen. Dadurch, dass die Eigenwerte entsprechend ihres Wertes in verschiedenen Diagonalblöcken gruppiert sind und insbesondere nicht in verschiedenen Diagonalblöcken nahe beieinander liegende Eigenwerte auftreten, approximieren bereits die linearen Anteile der Störterme die Störungen gut und die nichtlinearen Anteile der Störterme können vernachlässigt werden (vgl. [98, 108]). Umgekehrt würden nahe beieinander liegende Eigenwerte in verschiedenen Diagonalblöcken zu großen nichtlinearen Anteilen bei den Störtermen führen. Dadurch, dass die nichtlinearen Störterme vernachlässigbar sind, gilt:

$$\tilde{Y} (F + \epsilon M) \tilde{X} \approx \begin{pmatrix} T^{(1)} + \epsilon M_{11} & 0 \\ 0 & T^{(2)} + \epsilon M_{22} \end{pmatrix}.$$

Zur Untersuchung der Auswirkungen der Störung auf die Eigenwerte genügt es insbesondere also, jeweils die Auswirkungen auf die Eigenwerte des jeweiligen Diagonalblocks zu untersuchen.

Im Fall, dass F diagonalisierbar und ohne mehrfache Eigenwerte ist, führt eine Clusterung der Eigenwerte, bei der jeder Eigenwert ein eigenes Cluster bildet, dazu, dass die Diagonalblöcke jeweils einzelne Einträge sind. Hier lassen sich besonders griffige Aussagen für die Auswirkung einer Störung ϵM der Matrix F auf die Eigenwerte angeben.

Lemma 2. *Sei die Matrix F ohne mehrfache Eigenwerte. Seien Y und X Matrizen, die Links- und Rechts-Eigenvektoren als Spaltenvektoren enthalten, so dass $Y^H F X$ sowie $Y^H X$ Diagonalmatrizen sind. Dann gelten für hinreichend kleine $\epsilon > 0$ und eine Matrix $\tilde{\Lambda}$ der Eigenwerte $\tilde{\lambda}_i$ der gestörten Matrix $F + \epsilon M$ folgende Aussagen:*

1. $\tilde{\Lambda} = X^{-1} (F + \epsilon M) X + \mathcal{O}\left(\epsilon^2\right)$. Insbesondere ist $X^{-1} (F + \epsilon M) X$ bis auf $\mathcal{O}\left(\epsilon^2\right)$ Terme eine Diagonalmatrix.
2. $\tilde{\Lambda} - \Lambda = X^{-1} \epsilon M X + \mathcal{O}\left(\epsilon^2\right)$ für hinreichend kleine $\epsilon > 0$.
3. $\sum_i \left(\tilde{\lambda}_i - \lambda_i\right) = \mathrm{Spur}(\epsilon M) + \mathcal{O}\left(\epsilon^2\right)$.

Beweis. Behauptung 1 folgt aus Korollar 2 in [98] angewandt auf das Matrixpaar $(A, B) = (F, I)$ mit $(E, F) = (\epsilon M, 0)$.

Behauptung 2 ergibt sich aus Behauptung 1 auf Grund des Umstandes, dass die Einträge von $\tilde{\Lambda}$ stetig von ϵ abhängen, wie beispielsweise aus Korollar 2.4 in [108] hervorgeht, und somit die Reihenfolge der Eigenwerte bei $\tilde{\Lambda}$ und Λ für hinreichend kleine ϵ übereinstimmen. Ferner gilt

$$\Lambda = \lim_{\epsilon \to 0} \tilde{\Lambda} = \lim_{\epsilon \to 0} \left(X^{-1} (F + \epsilon M) X + \mathcal{O}\left(\epsilon^2\right)\right) = X^{-1} F X.$$

Daher folgt:

$$\begin{aligned}\tilde{\Lambda} - \Lambda &= X^{-1}\left(F + \epsilon M\right)X + \mathcal{O}\left(\epsilon^2\right) - X^{-1}FX, \\ &= X^{-1}\epsilon MX + \mathcal{O}\left(\epsilon^2\right).\end{aligned}$$

Wegen

$$\sum_i \left(\tilde{\lambda}_i - \lambda_i\right) = \text{Spur}\left(\tilde{\Lambda} - \Lambda\right) = \text{Spur}\left(\epsilon M\right) + \mathcal{O}\left(\epsilon^2\right)$$

ergibt sich hiermit Behauptung 3 unmittelbar. □

Wie in Abschnitt 4.2.2 beschrieben ist, beinhaltet diese Approximation die Gefahr, wesentliche Veränderungen der Eigenwerte auf Grund von Auslöschung nicht zu erfassen. Nachstehende Aussage ermöglicht eine Approximation, die bei reellen Störungen verhindert, dass derartige Auslöschungseffekte zum Tragen kommen.

Lemma 3. *Sei die reelle Matrix F ohne mehrfache Eigenwerte. Dann gilt für hinreichend kleine $\epsilon > 0$ und eine Matrix $\tilde{\Lambda}$ der Eigenwerte $\tilde{\lambda}_i$ der gestörten Matrix $F + \epsilon M$ mit $\epsilon M \in \mathbb{R}^n$:*

$$\sum_{i=1}^{n} \left|\tilde{\lambda}_i - \lambda_i\right|^2 \leq \mathit{Spur}\left(\epsilon^2 M^2\right) + \mathcal{O}\left(\epsilon^3\right).$$

Beweis. Wähle Matrizen X und Y wie auf Seite 53 beschrieben, so dass $Y^H F X$ eine Diagonalmatrix ist. Auf Grund von $F \in \mathbb{R}^n$ ist auch die Matrix U im Rahmen der Wahl von X und Y reell. Somit sind auch X und Y reelle Matrizen. Außerdem ist X invertierbar. Wähle $\tilde{X}$ und $\tilde{Y}$ wie auf Seite 53 beschrieben, so dass auch $\tilde{Y}^H\left(F + \epsilon M\right)\tilde{X}$ eine Diagonalmatrix ist. Dann erfüllen $\tilde{X}$ und $\tilde{Y}$ die Voraussetzungen von Lemma 2. Folglich gilt für hinreichend kleine $\epsilon > 0$

$$\tilde{\Lambda} - \Lambda = \tilde{X}^{-1}\epsilon M\tilde{X} + \mathcal{O}\left(\epsilon^2\right).$$

Durch komplexe Konjugation der gesamten Rahmensituation folgt für hinreichend kleine $\epsilon > 0$

$$\overline{\tilde{\Lambda} - \Lambda} = \overline{\tilde{X}^{-1}\epsilon M\tilde{X} + \mathcal{O}\left(\epsilon^2\right)}.$$

Auf Grund der Gleichungen 4.6 und 4.8 gilt $\tilde{X} = X + \mathcal{O}\left(\epsilon\right)$. Da X invertierbar ist, gilt dies für hinreichend kleine $\epsilon > 0$ auch für $\tilde{X}$. Zudem ist $\tilde{X}^{-1} = X^{-1} + \mathcal{O}\left(\epsilon\right)$ und wegen $X \in \mathbb{R}^n$ daher

$$\tilde{X}\overline{\tilde{X}^{-1}} = \left(X + \mathcal{O}\left(\epsilon\right)\right)\overline{\left(X^{-1} + \mathcal{O}\left(\epsilon\right)\right)} = I + \mathcal{O}\left(\epsilon\right).$$

Somit folgt für hinreichend kleine $\epsilon > 0$

$$\begin{aligned}\sum_i^n \left|\tilde{\lambda}_i - \lambda_i\right|^2 &= \text{Spur}\left(\left(\tilde{\Lambda} - \Lambda\right)\overline{\left(\tilde{\Lambda} - \Lambda\right)}\right), \\ &= \text{Spur}\left(\left(\tilde{X}^{-1}\epsilon M\tilde{X} + \mathcal{O}\left(\epsilon^2\right)\right)\left(\overline{\tilde{X}^{-1}\epsilon M\tilde{X}} + \mathcal{O}\left(\epsilon^2\right)\right)\right), \\ &= \text{Spur}\left(\tilde{X}^{-1}\epsilon M\tilde{X}\overline{\tilde{X}^{-1}\epsilon M\tilde{X}} + \mathcal{O}\left(\epsilon^3\right)\right), \\ &= \text{Spur}\left(\epsilon^2 M^2\right) + \mathcal{O}\left(\epsilon^3\right).\end{aligned}$$

□

Im Folgenden werden die vorangehenden Resultate eingesetzt, um ein Kriterium für die Eignung einer Vergröberung der Jacobi-Matrix bei der Integration mit einem linear-impliziten Euler-Verfahren zu erhalten.

Bei einem linear-impliziten Euler-Verfahren, bei dem Sparsing angewandt wird, tritt an die Stelle der linearisierten Integrationsgleichung 4.3 die Gleichung

$$x_{n+1} = \left(I + \Delta t \left(I - \Delta t A\right)^{-1} J_f\right) x_n.$$

Diese lässt sich auffassen als linearisierte Integrationsgleichung 4.1, bei der F gestört wird mit der Matrix ϵM mit

$$\begin{aligned} \epsilon &= \left\| \Delta t \left(\left(I - \Delta t A\right)^{-1} - \left(I - \Delta t J_f\right)^{-1} \right) J_f \right\|, \\ M &= \frac{\Delta t \left(\left(I - \Delta t A\right)^{-1} - \left(I - \Delta t J_f\right)^{-1} \right) J_f}{\epsilon}. \end{aligned} \tag{4.10}$$

Es ist nämlich

$$\begin{aligned} F + \epsilon M &= I + \left(I - \Delta t J_f\right)^{-1} \Delta t J_f + \Delta t \left(\left(I - \Delta t A\right)^{-1} - \left(I - \Delta t J_f\right)^{-1} \right) J_f, \\ &= I + \left(I - \Delta t A\right)^{-1} \Delta t J_f. \end{aligned}$$

Zu bemerken ist

$$\lim_{A \to J_f} \epsilon = 0,$$

also dass für geringe Änderungen der Jacobi-Matrix ϵ klein wird. Es gilt nachfolgende Aussage.

Theorem 4. *Seien λ_i die Eigenwerte der Matrix $F = I + \left(I - \Delta t J_f\right)^{-1} \Delta t J_f$ und $\tilde{\lambda}_i$ die Eigenwerte der Matrix $I + \left(I - \Delta t A\right)^{-1} \Delta t J_f$. Sei F ohne mehrfache Eigenwerte. Weiter seien A und J_f reell sowie ϵ definiert gemäß Gleichung 4.10. Ist ϵ hinreichend klein, so gilt:*

1. $\sum_i \left(\tilde{\lambda}_i - \lambda_i\right) = \text{Spur}\left(\Delta t \left(\left(I - \Delta t A\right)^{-1} - \left(I - \Delta t J_f\right)^{-1} \right) J_f\right) + \mathcal{O}\left(\epsilon^2\right).$
2. $\sum_i \left|\tilde{\lambda}_i - \lambda_i\right|^2 = \text{Spur}\left(\left(\Delta t \left(\left(I - \Delta t A\right)^{-1} - \left(I - \Delta t J_f\right)^{-1} \right) J_f \right)^2 \right) + \mathcal{O}\left(\epsilon^3\right).$

Beweis. Wähle Matrizen $\tilde{X}$ und $\tilde{Y}$ wie auf Seite 53 beschrieben. Auf Grund ihrer Existenz folgt Behauptung 1 aus Lemma 2 und Behauptung 2 aus Lemma 3 jeweils angewandt auf

$$F = I + \left(I - \Delta t J_f\right)^{-1} \Delta t J_f$$

und

$$\epsilon M = \Delta t \left(\left(I - \Delta t A\right)^{-1} - \left(I - \Delta t J_f\right)^{-1} \right) J_f.$$

□

Zu vorangehendem Theorem sind einige Anmerkungen anzufügen:
Aussage 1 aus Theorem 4 stellt eine leicht variierte Übertragung des Sparsing-Kriteriums beruhend auf Theorem 4 in [98] dar. Hierbei werden die Auswirkungen der Störung der Jacobi-Matrix durch ein Sparsing, also über die Formulierung eines verallgemeinerten Eigenwertproblems angegangen. Fasst man die linearisierte Integrationsgleichung 4.1 in diesem Sinne auf, so ist dieses Theorem nicht unmittelbar anwendbar, um eine zu Theorem 4 analoge Aussage zu liefern, da das hierzu nötige Matrixpaar (E, F) der Störungen zu wählen wäre als $\left(I + \left(I - \Delta t A\right)^{-1} \Delta t J_f, 0\right)$. In Theorem 1 sind aber nur Matrixpaare der Form (E, E) zulässig. Die Approximation für ein Sparsing-Kriterium, das sich aus diesem Theorem ableiten lässt, ist in Gleichung 4.2 dargestellt. Durch die Approximation von $\sum_i \left|\tilde{\lambda}_i - \lambda_i\right|^2$ ist es möglich, auch dann festzustellen, dass große Abweichungen $\tilde{\lambda}_i - \lambda_i$ auftreten, wenn sich diese nicht in großen Werten von $\sum_i \left(\tilde{\lambda}_i - \lambda_i\right)$ niederschlagen. Diese Detektionsmöglichkeit ist wertvoll. Für einen zuverlässigen Einsatz ist zu

berücksichtigen, dass die nichtlinearen Auswirkungen der Störmatrix vernachlässigbar sein müssen. Dies gilt, wenn ϵ hinreichend klein ist. Darüber hinaus kann es aber auch für große ϵ gelten, sofern sich die Eigenwerte geeignet gruppieren lassen. Daher ist die Anwendung dieses Kriteriums nicht auf betragsmäßig kleine Jacobi-Matrix-Einträge beschränkt, wenn die Eigenwerte der linearisierten Integrationsgleichung des linear-impliziten Euler-Verfahrens geeignet verteilt sind.

Wahl der Schranken

Bisher wurde gezeigt, wie sich die Auswirkungen einer Jacobi-Matrix-Vergröberung auf die Eigenwerte der linearisierten Integrationsgleichung approximieren lassen. Dies dient als Indikator dafür, wie die Dynamik des Simulationsverlaufs durch eine Jacobi-Matrix-Ausdünnung beeinflusst wird. Eine Frage, die sich stellt, ist, welche Veränderungen der Eigenwerte der linearisierten Integrationsgleichung als zulässig anzusehen sind. Im Hinblick auf ein Ausdünnungskriterium entspricht dies der Wahl der Schranken C_1 und C_2, für die gilt: Erfüllt eine Jacobi-Matrix-Ausdünnung die Ungleichungen

$$\begin{aligned} \sum_i \left(\tilde{\lambda}_i - \lambda_i\right) &\leq C_1, \\ \sum_i \left|\tilde{\lambda}_i - \lambda_i\right|^2 &\leq C_2, \end{aligned}$$

so ist das Sparsing geeignet.

Grundsätzlich sind die Schranken C_1 und C_2 selbstverständlich abhängig von Modell und Anforderungen an das Integrationsresultat. Da sich wie in Abschnitt 3.2.2 beschrieben ein stabiler Integrationsverlauf im Kontext vorliegender Arbeit typischerweise als Anforderung annehmen lässt, ergibt sich folgende heuristische Abschätzung von C_1 und C_2:

Die Stabilität des Integrationsverlaufs im Sinne der Definition aus Abschnitt 3.2.2 legt nahe, dass für die Eigenwerte $\tilde{\lambda}_i$ der linearisierten Integrationsgleichung 4.3 zumeist gilt:

$$\tilde{\lambda}_i \leq 1. \tag{4.11}$$

Aus diesem Grund ist es naheliegend, die Schranken C_1 und C_2 derart zu wählen, dass Gleichung 4.11 nahe gelegt wird. Sei $r_i = d\left(\lambda_i, \partial B_1\left(0\right)\right)$. Dann ist durch

$$C_2 = \min_i \left(r_i^2\right)$$

eine entsprechende Schranke gegeben, die im Falle der Gültigkeit des Theorems 4 dies gewährleisten kann, die also zu einer Beschränktheit der Zustände über den Simulationsverlauf hinweg führt. Für C_1 ergibt sich mit einer analogen Überlegung der Wert

$$C_1 = \min_i \left(r_i\right),$$

wobei das entsprechende Kriterium auf Grund möglicher Kürzungseffekte als nicht gesichert anzusehen ist.

Der Begriff der Stabilität gemäß Abschnitt 3.2.2 fordert außerdem, dass durch die Integration numerisch bedingte Oszillationen gering zu sein haben. Oszillationen spiegeln sich bei den Eigenwerten $\tilde{\lambda}_j = \left|\tilde{\lambda}_j\right| e^{i\varphi_j}$, wobei $0 \leq \varphi_j \leq 2\pi$ sei, der linearisierten Integrationsgleichung in deren Argument φ_j wieder: Ist $|\pi - \varphi_j|$ klein, so ist dies ein Indikator für ein oszillierendes Verhalten des Eigenwerts $\tilde{\lambda}_j$. Demzufolge ist im Falle ungewollter Oszillationen darauf zu achten, dass die Eigenwerte $\tilde{\lambda}_j$ der linearisierten Integrationsgleichung nicht nur betragsmäßig kleiner oder gleich 1 sind, sondern auch hinsichtlich ihres Arguments nahe der positiven reellen Halbachse liegen.

4.2.4 Jacobi-Matrix-Vergröberung

Wie in Abschnitt 4.1.3 beschrieben, liegt der Hauptaufwand bei der Lösung des steifen Anfangswertproblems bei der Leitanwendung in der Berechnung der Jacobi-Matrix. Daher besteht zunächst das Ziel, den Rechenaufwand für diesen Teil zu verringern.

In Abschnitt 4.2.1 wurde eine Übersicht über verschiedene numerische Methoden gegeben, die zur Rechenzeitreduktion für die Ermittlung einer Jacobi-Matrix-Approximation dienlich sein können. In diesem Abschnitt wird nun ein eigener Ansatz beschrieben, der in dieser Arbeit genutzt wird. Sowohl dieser als auch die Methoden aus Abschnitt 4.2.1 werden in Abschnitt 4.4 zur Rechenzeitreduktion im Rahmen einer Echtzeitsimulation angewandt.

Wie vorangehend beschrieben ist das Sparsing gemäß [98] eine Methode zur Reduktion des Rechenaufwands zur Lösung der linearen Gleichungssysteme im Rahmen der Integration eines steifen Anfangswertproblems. Eine Verallgemeinerung davon wird im Folgenden vor allem dazu genutzt, um den Aufwand zur Berechnung einer Jacobi-Matrix zu verringern. Mit Jacobi-Matrix wird hierbei nicht nur die exakte Jacobi-Matrix der rechten Seite bezeichnet, sondern auch jede Approximation und Ausdünnung hiervon, die für den Einsatz im Rahmen eines linear-impliziten Euler-Verfahrens geeignet ist.

Idee des Ansatzes Die Idee der Jacobi-Matrix-Vergröberung ist folgende: Durch die Verwendung einer W Methode als Integrator ist auch mit einer inexakten Jacobi-Matrix ein guter Simulationsverlauf für ein steifes Modell möglich. Dies wird genutzt, um den Rechenaufwand für die Bestimmung einer Jacobi-Matrix zur Laufzeit zu reduzieren: Hierzu wird nicht zur Laufzeit, sondern vorab im Rahmen einer Untersuchung der Differentialgleichung eine geeignete Wahl für eine inexakte Jacobi-Matrix bestimmt, die einerseits zu einem guten Simulationsverlauf führt und andererseits mit einem geringen Rechenaufwand ermittelt werden kann. Hierdurch wird der Rechenaufwand zur Bestimmung der Jacobi-Matrix reduziert. Solche inexakte Jacobi-Matrizen werden im Folgenden als Vergröberungen der Jacobi-Matrizen bezeichnet.

Der Ansatz enthält zahlreiche Freiheitsgrade: Eine solche Vergröberung umfasst als Spezialfälle beispielsweise das Festhalten der Jacobi-Matrix über mehrere Zeitschritte hinweg, die Mixed-Mode Integration und das Sparsing. Das Ermitteln einer Jacobi-Matrix durch numerische Differenziation lässt sich ebenfalls als eine Form der Vergröberung der exakten Jacobi-Matrix auffassen.

Eine naheliegende Art der Vergröberung der Jacobi-Matrix besteht darin, eine rechenaufwändige Funktion aus einem Eintrag der exakten Jacobi-Matrix durch eine mit geringem Aufwand berechenbare, beispielsweise polynomielle Näherung im entsprechenden Eintrag der inexakten Jacobi-Matrix zu ersetzen. Bei Funktionen mit einer physikalischen Bedeutung ist hier die Kombination mit einem datenbasierten Ansatz wie einem Kennfeld geeignet. Eine weitere vielversprechende Art, eine Vergröberung zu wählen, ist, bei der Berechnung der inexakten Jacobi-Matrix Synergien zwischen den verschiedenen Einträge der Jacobi-Matrix zu schaffen. Insbesondere besteht es eine Möglichkeit darin, verschiedene Einträge der exakten Jacobi-Matrix im Rahmen der Vergröberung gleich zu setzen, so dass bei der inexakten Jacobi-Matrix bereits die Berechnung eines einzelnen Eintrags genügt, um mehrere Einträge zu erhalten. Die Varianten werden im weiteren Verlauf des Kapitels bei der Leitanwendung eingesetzt.

Auswirkung auf die Genauigkeit der Lösung der Differentialgleichung Eine wichtige Beobachtung beim Einsatz einer Jacobi-Matrix-Vergröberung bei einer Integration mit dem linear-impliziten Euler-Verfahren ist, dass die Vergröberung in der Jacobi-Matrix keineswegs eine Vergröberung im Resultat der Integration im Sinne einer entsprechenden Ungenauigkeit nach sich ziehen muss. Ein Grund hierfür ist, dass bei

$$x_{n+1} = x_n + \Delta t \left(I - \Delta t J_f\right)^{-1} f\left(x_n\right)$$

lediglich die Jacobi-Matrix J_f, nicht aber die rechte Seite f verändert wird. Der Term $(I - \Delta t J_f)^{-1}$ hat eine stabilisierende Wirkung. Eine Veränderung der Matrix J_f hin zur Matrix $J_f + \Delta J = A$ führt nicht unbedingt zu einem größeren Fehler. Beispielsweise führt die Wahl der Jacobi-Matrix-Vergröberung $A = 0$ dazu, dass die entstehenden Integrationsgleichungen äquivalent mit denen des expliziten Euler-Verfahrens sind:

$$\begin{aligned} x_{n+1} &= x_n + \Delta t \left(I - \Delta t A\right)^{-1} f\left(x_n\right), \\ &= x_n + \Delta t f\left(x_n\right). \end{aligned}$$

Eine Verwendung dieser Jacobi-Matrix-Vergröberung ist somit im Hinblick auf die Genauigkeit nicht schlechter als das linear-implizite Euler-Verfahren. Folglich wirkt sich eine Veränderung in der Jacobi-Matrix nicht unmittelbar negativ auf die Genauigkeit des Integrationsverlaufs aus.

Allerdings ist die Jacobi-Matrix-Vergröberung durchaus mit Bedacht zu wählen: Da das linear-implizite Euler-Verfahren eine W Methode ist, führt zwar jede Wahl für die vergröberte Jacobi-Matrix zu einer Integration mit Konsistenzordnung 1. Bei vorgegebener Zeitschrittweite Δt ist es aber durchaus möglich, selbst wenn ein stabiler Integrationsverlauf vorausgesetzt wird, den Integrationsverlauf durch eine entsprechende Wahl der Jacobi-Matrix-Vergröberung weitreichend zu verfälschen.

Überprüfung anhand der Referenztrajektorien Bei der Vergröberung der Jacobi-Matrix handelt es sich um einen Ansatz, der ein auf ein Modell zugeschnittenes Verfahren liefert. Im Rahmen der verschiedenen Schritte der Bestimmung dieses maßgeschneiderten Verfahrens werden eine oder mehrere Referenztrajektorien vorausgesetzt, um die Eignung einer Vergröberung zu untersuchen. Aus diesem Grund ist es essentiell, dass die Referenztrajektorien umfassend sind und das erwünschte Systemverhalten umfänglich widerspiegeln. Andernfalls können im Rahmen der Anpassung durch die Jacobi-Matrix-Vergröberung unzulässige Vereinfachungen oder Veränderungen eingeführt werden.

Es ist möglich, dass ein dementsprechend umfassender Satz an Referenztrajektorien nicht basierend auf Messungen verfügbar ist. In diesem Fall ist es möglich, basierend auf dem gegebenen mathematischen Modell durch Vorgabe entsprechender Eingangsverläufe und Parameter simulativ weitere Referenztrajektorien zu erzeugen. Erst danach ist eine Vergröberung der Jacobi-Matrix zu bestimmen. Hierbei dient dann der gegebenenfalls simulativ erweiterte Satz an Referenztrajektorien zur Überprüfung, ob durch verschiedene Maßnahmen im Rahmen der Vergröberung der Jacobi-Matrix eine unzulässige Veränderung des Systemverhaltens verursacht wird.

In vorliegender Arbeit wird die in Abschnitt 4.1.2 beschriebene Referenztrajektorie als umfassend angenommen. Darüber hinaus werden dementsprechend keine weiteren Trajektorien durch Simulation des Rohrmodells erzeugt.

Vorgehen bei der Bestimmung einer Vergröberung der Jacobi-Matrix Wie vorangehend beschrieben, wird im Folgenden die Entscheidung, ob eine Vergröberung einer Jacobi-Matrix für den Einsatz im Rahmen der Integration eines mathematischen Modells mit dem linear-impliziten Euler-Verfahren geeignet ist, folgendermaßen getroffen: Führt eine Vergröberung bei einem Satz von Referenztrajektorien zu stabilen und hinreichend akkuraten Simulationsverläufen, so wird sie als grundsätzlich geeignet angesehen. In diesem Fall ist der Rechenaufwand zur Bestimmung der vergröberten Jacobi-Matrizen das nächste Kriterium für die Eignung einer untersuchten Vergröberung. Eine Konsequenz hiervon ist folgende: Die Vergröberung einer Jacobi-Matrix ist in Abhängigkeit vom mathematischen Modell und den gegebenen Anforderungen zu ermitteln. Ist eine Vergröberung gefunden, die den Rechenaufwand zur Jacobi-Matrix-Ermittlung reduziert, so ist für ihren Einsatz nur zu überprüfen, ob sie zu hinreichend guten Simulationsverläufen bei den Referenztrajektorien führt. Dies eröffnet insbesondere die Möglichkeit, die Vergröberung der Jacobi-Matrix basierend auf Systemexpertise zu ermitteln. Liegt für ein mathematisches Modell ein tiefgreifendes Systemverständnis vor, so ist dies eine sehr gute Möglichkeit, die Vergröberung

zu wählen, da die Auswirkungen auf das Systemverhalten in diesem Fall unter Umständen gut eingeschätzt werden können.

Die Vorgehensweise zur Bestimmung einer geeigneten Jacobi-Matrix-Vergröberung beruht unter anderem auf den Überlegungen aus Abschnitt 4.2.3. Zu beachten ist, dass sich die theoretischen Überlegungen aus Abschnitt 4.2.3 unmittelbar vom Sparsing auf die Jacobi-Matrix-Vergröberung übertragen lassen: In Abschnitt 4.2.3 wird nicht verwendet, dass die Einflüsse von $\Delta J = A - J_f$ nur zu einem Nullsetzen von Jacobi-Matrix-Einträge verwendet werden. Ebenso ist jede andere Veränderung von A gegenüber J_f in der Untersuchung ebenfalls umfasst.

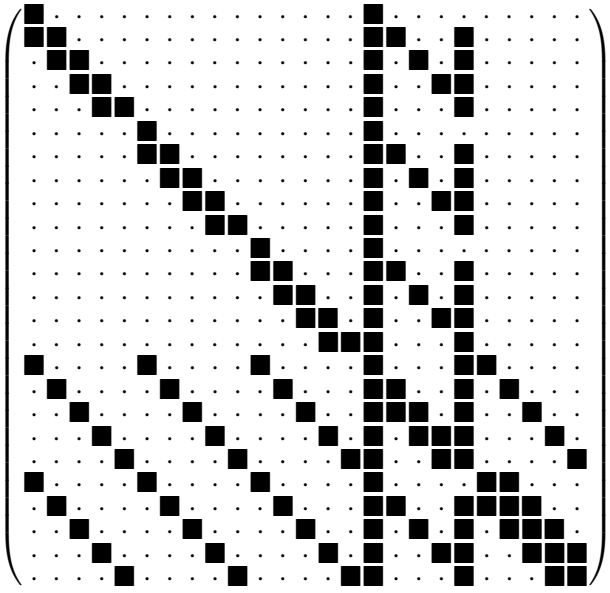
4.3 Anwendung auf das Rohrmodell

Im Folgenden wird am Beispiel des Rohrmodells eine Bestimmung geeigneter Vergröberungen der Jacobi-Matrix durchgeführt. Hierbei wird das Rohrmodell mit 5 Diskretisierungszellen verwendet. Die Bestimmung wird in mehreren Schritten vorgenommen. Den ersten Schritt bildet das Nullsetzen von Einträgen, also ein Sparsing. Im Anschluss werden die weitere Vergröberungsprinzipien betrachtet.

4.3.1 Bestimmung einer Ausdünnung

Zur Bestimmung eines Sparsings ist die Jacobi-Matrix J_f der rechten Seite f des Rohrmodells zu untersuchen.Hierzu kann grundsätzlich eine symbolische Darstellung der Jacobi-Matrix herangezogen werden. Im Fall des Rohrmodells ergibt sich auf Grund der zustandsabhängigen Koeffizienten ein überaus komplizierter symbolischer Ausdruck, der zwar mit einer geeigneten Software wie beispielsweise MuPAD 6.0.0 (vgl. [86]) ermittelt werden kann, dessen Darstellung allerdings teilweise über Tausend Terme je Eintrag umfasst. Daher eröffnet das Arbeiten mit dem symbolischen Ausdruck nur begrenzte Verständnismöglichkeiten für die prinzipielle Gestalt der Jacobi-Matrizen. Zudem ist ein Auswerten der symbolischen Ausdrücke weit jenseits des Rechenaufwands, der für einen Zeitschritt der Simulation des Modells im Rahmen einer Software-Funktion auf einem Fahrzeug-Steuergerät zur Verfügung steht, so dass ein direktes Verwenden der symbolischen Jacobi-Matrix zur Laufzeit nicht möglich ist. Eine wertvolle Information, die aus dem symbolischen Ausdruck der Jacobi-Matrix gewonnen werden kann, ist die Strukturmatrix S (vgl. Seite 47). Diese ist

$$S = \left(\right), \tag{4.12}$$



wobei ein Punkt für einen stets verschwindenden Eintrag steht und ein Kästchen für einen potentiell nicht verschwindenden Eintrag.

Da der symbolische Ausdruck für die Jacobi-Matrix zu unhandlich und kompliziert ist, wird zur Motivation der weiteren Schritte eine Jacobi-Matrix aus der Referenztrajektorie herangezogen. Die Überprüfung der Eignung einer Vergröberung der Jacobi-Matrix wird jeweils an der vollen Referenztrajektorie überprüft.

Als Jacobi-Matrix wird die Matrix J_n verwendet, deren Einträge im Anhang notiert sind. Zu erkennen ist, dass bei der Jacobi-Matrix J_n nahezu alle Einträge, die durch die symbolische Strukturmatrix S als potentiell nicht verschwindend eingestuft sind, auch tatsächlich nicht verschwinden. Nun besteht die Aufgabe eine geeignete Menge an Einträgen zu wählen, die im Rahmen der Ausdünnung auf Null gesetzt werden. Hierbei ist zu beachten, dass die betragsmäßige Größe der Einträge nicht ihren Einfluss auf die Dynamik des Systems widerspiegeln: Beispielsweise wird sich zeigen, dass der Eintrag $j_{16,1} \approx 2320.0$ ausgedünnt werden kann, während der Eintrag $j_{16,16} \approx -1014.0$ nicht auf Null gesetzt werden sollte. Zur Wahl des Ausdünnung wird folgendermaßen vorgegangen:

Zunächst wird eine Transformation betrachtet, die zum Verständnis der Relevanz der verschiedenen Einträge beiträgt: Wie in Abschnitt 4.2.3 erläutert, gibt es eine Ähnlichkeitstransformation mit der Matrix U, die J_n auf eine geeignete Blockdiagonalform mit Dreiecksmatrizen als Diagonalblöcke bringt. Sei $Z_{k,l} = (z_{k,l})$ mit

$$z_{\bar{k},\bar{l}} = \begin{cases} 0 & \text{, falls } \left(\bar{k},\bar{l}\right) = (k,l)\,, \\ j_{\bar{k},\bar{l}} & \text{sonst.} \end{cases}$$

Die Matrix $Z_{k,l}$ entspricht also der Jacobi-Matrix, bei der der Eintrag $j_{k,l}$ auf Null gesetzt wird. Dann gibt $\tilde{Z}_{k,l} = U^{-1} Z_{k,l} U$ Aufschluss über den Einfluss des Sparsings des Jacobi-Matrix-Eintrags $j_{k,l}$ auf die Eigenwerte der Matrix. Die Betrachtung der Matrizen $\tilde{Z}_{k,l}$ ist instruktiv und kann zum Verständnis der Auswirkungen einer Ausdünnung eines Eintrags beitragen.

Sei I eine Menge von Indizes von nicht stets verschwindenden Einträgen $j_{k,l}$ und $Y_I = (y_{k,l})_{k,l}$ eine Matrix mit

$$y_{k,l} = \begin{cases} -j_{k,l} & \text{, falls } (k,l) \in I, \\ 0 & \text{sonst.} \end{cases}$$

Die Matrix Y_I führt also durch Addition auf J_n dazu, dass die Einträge mit den Indizes aus I auf Null gesetzt werden. Sofern die Jacobi-Matrizen eines Modells den Voraussetzungen aus Lemma 3 genügen, lässt sich für jede Menge an nicht stets verschwindenden Einträgen mit geringem Rechenaufwand eine Approximation $d^{(2)}_{J,k,l}$ für den Term $\sum_i \left|\tilde{\lambda}_i - \lambda_i\right|^2$ gemäß Lemma 3 durch

$$d^{(2)}_{J,k,l} \approx \text{Spur}\left(Y_I^2\right)$$

ermitteln.

Für die Eigenwerte Λ_J von J_n gilt

$$\begin{aligned} \Lambda_J \approx \big\{ & -1014.0, -1012.0, -1012.0, -1005.0, -1005.0 \\ & -0.03519, -0.03904, -0.04664, -0.05416, -0.05727, \\ & -748.5, -748.5, -748.5, -756.0, -756.0, \\ & -756.0, -750.4, -750.4, -750.4, -754.1, \\ & -754.1, -754.1, -752.2, -752.2, -752.2 \big\}. \end{aligned} \tag{4.13}$$

Die vorliegende Matrix J_n ist für eine derartige Approximation dadurch nicht notwendigerweise geeignet, dass sie sehr eng beieinander liegende und eventuell sogar mehrfache Eigenwerte besitzen könnte. Aus diesem Grund ist der Term $\sum_i \left|\tilde{\lambda}_i - \lambda_i\right|^2$ im vorliegenden Fall auf eine andere Weise zu bestimmen: Hierzu wird durch eine Blockdiagonalisierung, wie in Abschnitt 4.2.3 erläutert, die Matrix $\tilde{Y}_I = U^{-1} Y_I U$ ermittelt, auf deren Diagonale die Eigenwerte $\tilde{\lambda}_i$ stehen, wodurch

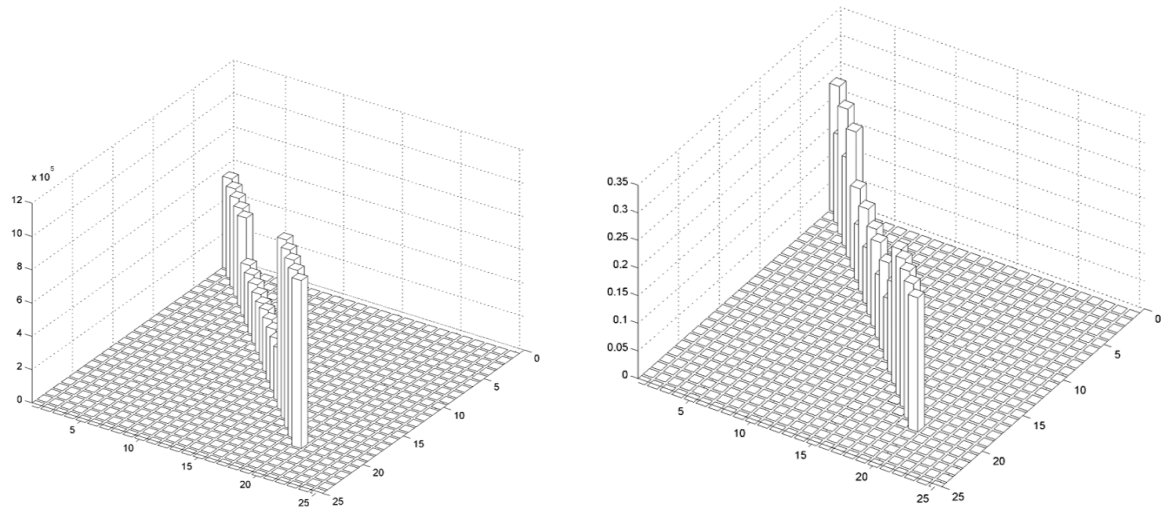


Abbildung 4.4: Darstellungen der Matrizen $D_J^{(2)}$ und $D_F^{(2)}$ als Säulendiagramm.

$d_{J,k,l}^{(2)}$ dann leicht errechnet werden kann.

Analog zur Matrix J_n wird auch mit der Matrix $F_n = F = I + (I - \Delta t J_n)^{-1} \Delta t J_n$ vorgegangen. Es ist

$$\begin{aligned}\Lambda_F \quad = \quad & \big\{0.9965, 0.9961, 0.9954, 0.9946, 0.9943, \\ & 0.009768, 0.009788 - 1.724 \cdot 10^{-5}\,\mathrm{i}, 0.009788 + 1.724 \cdot 10^{-5}\,\mathrm{i}, \\ & 0.009851 - 1.776 \cdot 10^{-5}\,\mathrm{i}, 0.009851 + 1.776 \cdot 10^{-5}\,\mathrm{i} \\ & 0.01318, 0.01305, 0.01315, 0.01318, 0.01318 \\ & \ 0.01315, 0.01315, 0.01305, 0.01305, 0.01312 \\ & 0.01312, 0.01312, 0.01309, 0.01309, 0.01309\big\}.\end{aligned}$$

Sei $D_F^{(2)} = \left(d_{F,k,l}^{(2)}\right)_{k,l}$ analog zu $D_J^{(2)}$ definiert. Die Matrizen sind in Abbildung 4.4 in Form von Säulendiagrammen dargestellt.

Aus Abbildung 4.4 geht hervor, dass bei der untersuchten Approximationsweise zu erwarten ist, dass vorrangig bei den Diagonaleinträgen ein Nullsetzen zu einem nennenswerten Einfluss auf die Eigenwerte von J_n und F führt. Alle übrigen Einträge sind somit grundsätzliche Kandidaten, um auf Null gesetzt zu werden.

Gemäß Abbildung 4.4 ist ein geringer Einfluss $\sum_i \left|\tilde{\lambda}_{F,i} - \lambda_{F,i}\right|^2$ einer Ausdünnung eines Wertes der ersten Subdiagonalen auf die Eigenwerte zu erwarten. Die erste Subdiagonale ist allerdings zur Ausdünnung dennoch nicht geeignet. Dies liegt daran, dass Eigenwerte die Beschaffenheit eines Systems nicht vollumfänglich charakterisieren. Dies wird im Folgenden am Beispiel eines Teilsystems erläutert:

Zur Veranschaulichung wird das Teilsystem betrachtet, das durch die kanonische Projektion des betrachteten Systems auf die ersten fünf Zustände erhalten wird. Die entsprechenden Größen werden mit einem Hütchen notiert. Dann ergibt sich für das Teilsystem im Zeitschritt t_n, in dem das originale System als Jacobi-Matrix der rechten Seite also die Matrix J_n hat, folgende Gleichung durch das linear-implizite Euler-Verfahren:

$$\left(I - \Delta t \hat{J}_n\right) \Delta \hat{x} = \hat{f}\left(\hat{x}_n\right)$$

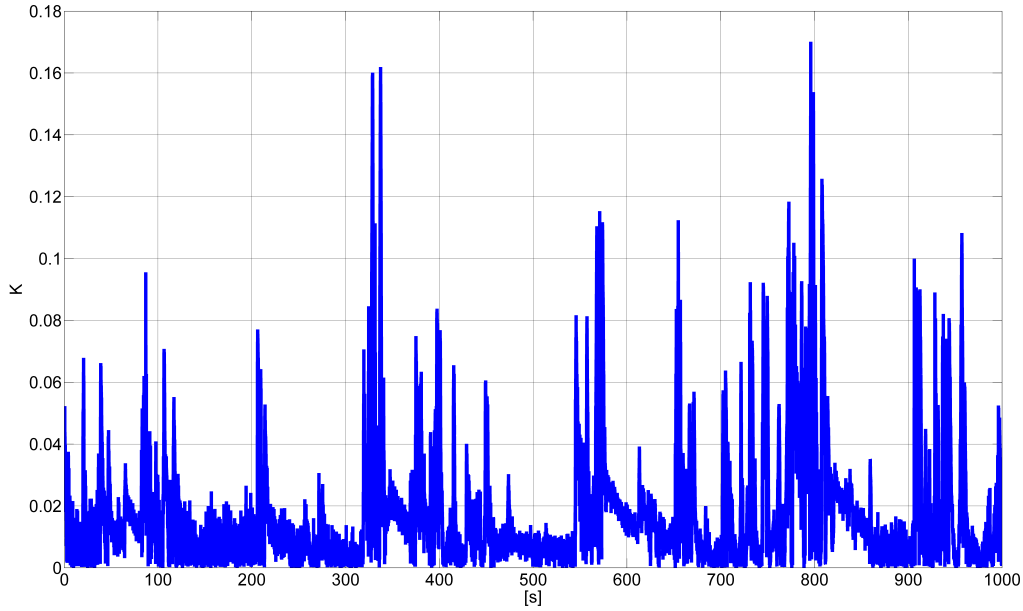


Abbildung 4.5: Betrag der Abweichung des Simulationsverlaufs der Gastemperatur bei Verwendung einer Ausdünnung der Jacobi-Matrix in Bidiagonalform gegenüber der Verwendung des originalen linear impliziten Euler-Verfahren mit einer Schrittweite von $\Delta t = 100\,\mathrm{ms}$ bei der Referenztrajektorie.

mit

$$\hat{J}_n = \begin{pmatrix} -756.0 & 0 & 0 & 0 & 0 \\ 754.1 & -754.1 & 0 & 0 & 0 \\ 0 & 752.2 & -752.2 & 0 & 0 \\ 0 & 0 & 750.4 & -750.4 & 0 \\ 0 & 0 & 0 & 748.5 & -748.5 \end{pmatrix}.$$

Offensichtlich ändert das Nullsetzen der ersten Subdiagonale die Eigenwerte von $\hat{J}_n$ nicht, aber es verändert die Beschaffenheit des Systems weitreichend: Seien $\hat{f}_i$ die Komponentenfunktionen von $\hat{f}$. Dann gilt bei der Integration mit einem linear-impliziten Euler-Verfahren ohne Ausdünnung

$$\hat{x}_i\left(t_{n+1}\right) - \hat{x}_i\left(t_n\right) = \frac{\Delta t}{1 - \Delta t j_{ii}} \left(\hat{f}_i\left(\hat{x}\left(t_n\right)\right) - \frac{-\Delta t j_{i,i-1}}{1 - \Delta t j_{i-1,i-1}} \hat{f}_{i-1}\left(\hat{x}\left(t_n\right)\right) \right),$$

$$\hat{x}_i\left(t_{n+1}\right) - \hat{x}_i\left(t_n\right) = \frac{\hat{f}_i\left(\hat{x}\left(t_n\right)\right) - c_i \hat{f}_{i-1}\left(\hat{x}\left(t_n\right)\right)}{1 + \Delta t j_{ii}}, \quad i = 2,3,4,5 \tag{4.14}$$

mit

$$c_i = \frac{-\Delta t j_{i,i-1}}{1 - \Delta t j_{i-1,i-1}} \approx 1.$$

Bei einem linear-impliziten Euler-Verfahren mit Ausdünnung ergibt sich hingegen

$$\hat{x}_{\mathrm{sparse},i}\left(t_{n+1}\right) - \hat{x}_{\mathrm{sparse},i}\left(t_n\right) \approx \frac{\hat{f}_{\mathrm{sparse},i}\left(\hat{x}_{\mathrm{sparse}}\left(t_n\right)\right)}{1 - \Delta t j_{ii}}, \quad i = 2,3,4,5. \tag{4.15}$$

Dass die beiden Fälle trotz übereinstimmender Eigenwerte bei $\hat{J}_n$ und $\hat{J}_{\mathrm{sparse},n}$ zu einem signifikant unterschiedlichen Systemverhalten führen können, ist offensichtlich: Während in Gleichung 4.15 lediglich $\hat{f}_{\mathrm{sparse}}\left(\hat{x}_{\mathrm{sparse}}\left(t_n\right)\right) = 0$ zu einem stationären Verhalten von $\hat{x}_{\mathrm{sparse}}$ führt, gilt dies

$$S = \left(\right) . \tag{4.16}$$

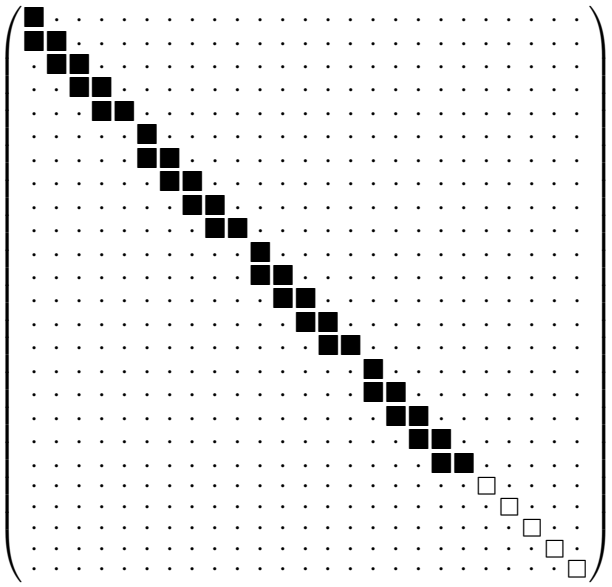


Abbildung 4.6: Resultierende Struktur aus Sparsing

bei Gleichung 4.14 auch, wenn $\hat{f}_{\text{sparse}}\left(\hat{x}_{\text{sparse},i}\left(t_n\right)\right) = c_i \hat{f}_{\text{sparse}}\left(\hat{x}_{\text{sparse},i-1}\left(t_n\right)\right)$ gilt.

Das vorangehende Beispiel illustriert die Situation bei der Ausdünnung der Jacobi-Matrix des Rohrmodells: Auch hier führt das Nullsetzen der Einträge der ersten Subdiagonale zu einer wesentlichen Veränderung des Systemverhaltens, ohne dass sich dies in den Eigenwerte der ausgedünnten Jacobi-Matrix niederschlägt. Bei der Referenztrajektorie führt ein Ausdünnen der ersten Subdiagonale zu einem vorzeitigen Simulationsabbruch auf Grund eines Verlassen der zulässigen Wertebereiche. Dies bestätigt, dass eine solche Ausdünnung nicht geeignet ist. Die Subdiagonaleinträge der Jacobi-Matrix werden daher nicht allesamt auf Null gesetzt.

Wie durch Abbildung 4.4 nahe gelegt, sind diejenigen Jacobi-Matrix-Einträge, die einen unmittelbaren Bezug zu den Zuständen $x_{21}, \ldots, x_{25}$, also den Rohrtemperaturen haben in dem Sinne, dass sie in den Spalten 21 bis 25 der Jacobi-Matrix stehen, Kandidaten, um auf Null gesetzt zu werden. Dies deckt sich mit der Erwartung, die sich bei Betrachtung der Eigenwerte von Λ_J ergibt: Die Jacobi-Matrix J_n hat einige Eigenwerte λ_i mit $\left|\lambda_i\right| < 1$. Mittels Gerschgorin-Kreise wird der enge Zusammenhang zwischen diesen Eigenwerten und den Zuständen $x_{21}, \ldots, x_{25}$ unmittelbar deutlich. Aus diesem Grund würde auch ein Ansatz wie die Mixed-Mode Integration (vgl. Abschnitt 4.2.1) vorschlagen, diese Zustände mit einem expliziten Integrator zu berechnen und nur auf die restlichen Zustände ein implizites Integrationsverfahren anzuwenden. Die Betrachtung des entsprechenden Fehlerverlaufs zeigt, dass dies möglich ist. Da der Fehlerverlauf visuell nicht von dem in Abbildung 4.5 zu unterscheiden ist, wird ein entsprechender Plot nicht dargestellt und an dieser Stelle auf Abbildung 4.5 verwiesen.

Insgesamt ergibt sich also die in Abbildung 4.6 dargestellte Sparsing-Struktur.

Ein leeres Kästchen steht hierbei für einen Eintrag, der bei der Jacobi-Matrix auf Null gesetzt wird, aber beim resultierenden linearen Gleichungssystem im Rahmen des linear-impliziten Euler-Verfahrens durch die Addition der Einheitsmatrix nicht verschwindet.

4.3.2 Bestimmung einer weitergehenden Vergröberung

Basierend auf der vorangehend ermittelten Ausdünnungsstruktur werden nachfolgend weitere Ansätze zur Vergröberung der Jacobi-Matrix auf das Rohrmodell angewandt. Wie auf Seite 58 beschrieben, kommen hierfür folgende Ansätze in Frage:

1. Approximation rechenaufwändiger Funktionen durch schnell berechenbare Funktionen in Kombination mit einem Kennfeld

2. Synergiebildung zwischen der Berechnungen verschiedener Einträge
3. Symbolische Approximationsterme für Einträge der Jacobi-Matrix

Die Eignung der verschiedenen Ansätze ist stark vom betrachteten Modell abhängig. Beim Rohrmodell stellt es sich folgendermaßen dar:

1. Beim Rohrmodell nimmt die Berechnung physikalischer Koeffizienten, insbesondere der Wärmeübergangskoeffizienten und der spezifischen Wärmekapazitäten, einen großen Anteil der Rechenzeit ein. Diese lassen sich im Rahmen der Jacobi-Matrix-Berechnung durch polynomielle Interpolation eines Kennfelds adäquat ersetzen.
2. Synergien lassen sich bei der Berechnung verschiedener Einträge schaffen. Diese werden sich in der Möglichkeit äußern, einen Großteil der Diagonaleinträge als identisch anzunehmen.
3. Für einige Einträge der Jacobi-Matrix lassen sich geeignete symbolische Approximationsausdrücke angeben. Diese tragen dazu bei, die Anzahl an Richtungsableitungen reduzieren, die zur Bestimmung der vergröberten Jacobi-Matrix nötig ist.

Kennfeldbasierte polynomielle Näherung für physikalische Koeffizienten

Die Berechnungsweise einiger physikalischen Koeffizienten des Rohrmodells wird basierend auf physikalischen Abhängigkeiten vorgenommen (vgl. Abschnitt 2.2). Es erscheint naheliegend, dass der daraus resultierende Rechenaufwand im Rahmen der Ermittlung einer inexakten Jacobi-Matrix verringert werden kann, ohne dass dies zu relevanten negativen Auswirkungen auf den resultierenden Simulationsverlauf führt. Grundsätzlich kommt eine Vielzahl an datenbasierten Modellierungsmethoden in Frage. Im Folgenden wird eine einfache Polynomapproximation eines Kennfelds betrachtet, die beim der Leitanwendung bereits zu guten Resultaten führt. Dies wird am Beispiel der Wärmeübergangskoeffizienten $\alpha_{B,C,i}$ von der Gasphase zum Rohr gezeigt. Hierzu wird folgendermaßen vorgegangen[1]:

Zunächst wird basierend auf der Referenztrajektorie ein Kennfeld für die Werte $\alpha_{B,C,i}$ in Abhängigkeit relevanter Parameter, Eingänge und Zustände erstellt. Innerhalb der während der Referenztrajektorie auftretenden Bereiche der Parameter, Eingänge und Zustände wird anschließend ein Approximationspolynom ermittelt. Für die Referenztrajektorie führt bereits für ein Polynom in zwei Variablen zu guten Ergebnissen:

$$\tilde{\alpha}_{B,C,i} = \sum_{j,k \leq 2} c_{j,k} \dot{m}^j T_{B,i}^k. \tag{4.17}$$

Für den relativen Fehler gilt bei der Referenztrajektorie

$$\left| \frac{\tilde{\alpha}_{B,C,i} - \alpha_{B,C,i}}{\alpha_{B,C,i}} \right| < 1\,\%.$$

Sei $\tilde{f}$ eine Variante der rechten Seite, bei der die Berechnung von $\alpha_{B,C,i}$ durch $\tilde{\alpha}_{B,C,i}$ für $i = 1, \ldots, 5$ ersetzt ist. Werden für die Berechnung der Jacobi-Matrix Richtungsableitungen der Funktion $\tilde{f}$ verwendet, so ergibt sich die in Abbildung 4.7 dargestellte betragsmäßige Abweichung gegenüber dem unveränderten linear-impliziten Euler-Verfahren mit einer Schrittweite von $\Delta t = 100\,\text{ms}$. Diese Abweichung ist unkritisch und der Einsatz eines vergröberten Jacobi-Matrix, die die Wärmeübergangskoeffizienten in der oben beschriebenen Weise ermittelt, somit möglich.

Synergiebildung

Die folgende Synergiebildung bei der Berechnung von Jacobi-Matrix-Einträgen beim Rohrmodell besteht aus zwei Stufen: Zunächst werden die nicht verschwindenden Subdiagonaleinträge gleich-

[1] Die Idee zum Approximationspolynom und das Polynom selbst stammen von Christian Potthast (Robert Bosch GmbH).

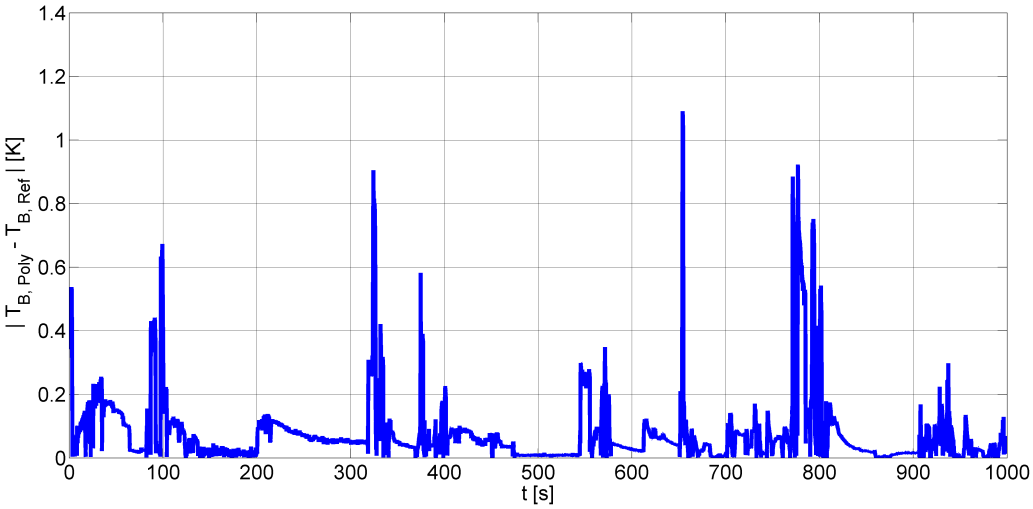


Abbildung 4.7: Betrag der Abweichung eines linear-impliziten Euler Verfahrens mit einer vergröberten Jacobi-Matrix, bei der zur Berechnung der Wärmeübergangskoeffizienten $\alpha_{B,C,i}$ eine Polynomapproximation eingesetzt wird, gegenüber dem originalen linear-impliziten Euler-Verfahren bei der Referenztrajektorie mit einer Zeitschrittweite von $\Delta t = 100\,\mathrm{ms}$.

gesetzt mit den Diagonaleinträgen aus der jeweils selben Zeile der Jacobi-Matrix. Anschließend wird die Möglichkeit zur Gleichsetzung einer Reihe von an sich unterschiedlichen Diagonalwerten untersucht.

Subdiagonaleinträge Wie in Abschnitt 2.2 beschrieben, entsteht die gewöhnliche Differentialgleichung des Modells durch Semidiskretisierung einer partiellen Differentialgleichung mittels Rückwärtsdifferenzen. Dies zeigt sich der rechten Seite beispielsweise folgendermaßen:

$$f_{15}\left(x\right) = -\frac{\dot{m} R x_{20}\left(x_{14} - x_{15}\right)}{A_B m_{\mathrm{mol}} p_1 \mathrm{d}z}.$$

Entsprechend gilt

$$j_{k,k-1} = -j_{k,k} \text{ für } k \in \{2,\ldots,15\} \setminus \{6,11\}.$$

Bei den Gastemperaturen besteht keine Gleichheit zwischen den Diagonal- und Subdiagonaleinträgen, allerdings unterscheiden sich die Terme jeweils in geringem Maß, sodass insgesamt im Rahmen einer Vergröberung folgende Gleichheit angenommen wird:

$$j_{k,k-1} = -j_{k,k} \text{ für } k \in \{2,\ldots,20\} \setminus \{6,11,16\}. \tag{4.18}$$

Hierdurch lassen sich die nicht auf Null gesetzten Subdiagonaleinträge aus den Diagonaleinträgen ermitteln gemäß Gleichung 4.18 und müssen nicht separat berechnet werden.

Diagonaleinträge Wie in Gleichung 4.13 zu erkennen, bilden die Eigenwerte von J_n drei Gruppen: eine Gruppe korrespondiert mit dem Massenstrom der verschiedenen Gasfraktionen, eine Gruppe korrespondiert mit dem Enthalpiestrom der Gasphase und eine Gruppe korrespondiert mit dem Enthalpiestrom im Rohrmantel. Letztere wurde im Rahmen der Ausdünnung auf Null gesetzt (vgl. Seite 64). Das Ziel ist nun, Synergien bei der Berechnung der korrespondierenden Diagonaleinträge der Jacobi-Matrix-Einträge zu schaffen. Die radikalste Form der Synergiebildung ist das Gleichsetzen. Aus diesem Grund wird das Ergebnis untersucht für den Fall, dass

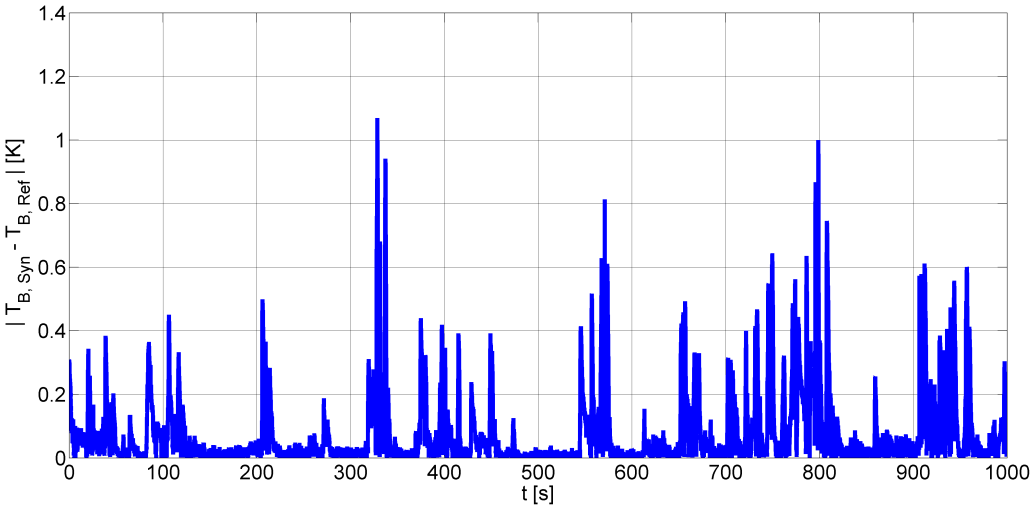


Abbildung 4.8: Betrag der Abweichung des Simulationsverlaufs mit einer Vergröberung mit Synergiebildung gegenüber dem originalen linear impliziten Euler-Verfahren mit einer Schrittweite von $\Delta t = 100\,\mathrm{ms}$ bei der Referenztrajektorie.

gilt:

$$j_{i,i} = \begin{cases} j_{1,1} & \text{, falls } i \leq 15, \\ j_{19,19} & \text{, falls } 15 < i \leq 20, \\ 0 & \text{sonst.} \end{cases}$$

Die Ergebnisse für die Referenztrajektorie sind in Abbildung 4.8 dargestellt. Diese zeigen, dass der Simulationsverlauf durch obige Gleichsetzung nicht im relevanten Maß beeinflusst wird und die Gleichsetzung daher gerechtfertigt ist.

Ein Aspekt, weshalb für die mit den Gastemperaturen korrespondierenden Diagonalwerte der Jacobi-Matrix die Wahl auf $j_{19,19}$ fällt, ist, dass die Jacobi-Matrix eine Färbung besitzt, so dass $j_{1,1}$ und $j_{19,19}$ simultan ermittelt werden können. Falls zur Differenziation die numerische Differenziation mit Absicherung in Form der Bestimmung eines weiteren Differenzenquotients mit jeweils halber Auslenkung eingesetzt wird, so sind über eine ohnehin nötige Auswertung der rechten Seite hinaus also nur zwei weitere Auswertungen zur Bestimmung der vergröberten Jacobi-Matrix ausreichend. Falls die numerische Differenziation ohne eine Absicherung eingesetzt wird, was wie in Abschnitt 3.2.3 beschrieben, im Allgemeinen als anfällig anzusehen ist, so ist eine zusätzliche Auswertung ausreichend. Da bei der Referenztrajektorie in keinem Zeitschritt die Absicherung der numerischen Differenziation aktiv wird, also die Richtungsableitungen zum selben Richtungsvektoren mit unterschiedlichen Auslenkungen jeweils um weniger als 10 % voneinander abweichen, wird es als geeignet angesehen, im Rahmen dieser Variante der Vergröberung der Jacobi-Matrix keine numerische Absicherung bei der Differenziation zu implementieren.

Symbolische Approximation für den Enthalpiestrom

Im Folgenden wird untersucht, ob eine symbolische Approximation einer der beiden Diagonalwerte $j_{1,1}$ oder $j_{19,19}$ möglich ist. Es gilt

$$f_1(x) = -\frac{\dot{m} R x_{16} (x_1 - u_1)}{A_B m_{\mathrm{mol}} p_1 \mathrm{d}z}$$

sowie

$$f_{19}(x) = -\frac{x_{19} \left(\alpha_{B,C,4} l_B (x_{19} - x_{24}) - (\dot{m} c_{p,B,4} (x_{18} - x_{19}))\, \mathrm{d}z^{-1}\right)}{A_B p_4 (m_{\mathrm{mol}} c_{p,B,4} R^{-1} - 1)}.$$

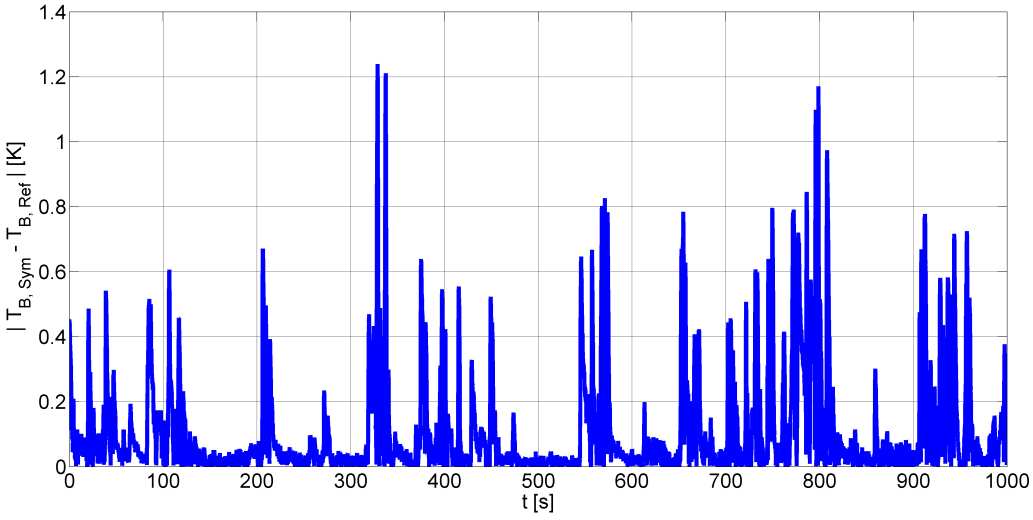


Abbildung 4.9: Betrag der Abweichung des Simulationsverlaufs mit einer Vergröberung mittels symbolischer Approximation gegenüber dem originalen linear impliziten Euler-Verfahren bei der Referenztrajektorie.

Im Folgenden wird die Approximation untersucht, die sich ergibt, wenn die Zustandsabhängigkeit der beteiligten physikalischen Koeffizienten vernachlässigt wird. Dies entspricht folgenden Approximationen für die beiden Werte:

$$\tilde{j}_{1,1} = -\frac{\dot{m} R x_{16}}{A_B m_{\mathrm{mol}} \tilde{p}_1 \mathrm{d}z}$$

und

$$\tilde{j}_{19,19} = -\frac{\tilde{\alpha}_{B,C,4} l_B \left(x_{19} - x_{24}\right) - \left(\dot{m} \tilde{c}_{p,B,4} \left(x_{18} - x_{19}\right)\right) \mathrm{d}z^{-1} + x_{19} \left(\tilde{\alpha}_{B,C} l_B + \dot{m} \tilde{c}_{p,B,4} \mathrm{d}z^{-1}\right)}{A_B \tilde{p}_4 \left(m_{\mathrm{mol}} \tilde{c}_{p,B,4} R^{-1} - 1\right)},$$

wobei für $\tilde{\alpha}_{B,C,4}$, $\tilde{p}_1$, $\tilde{p}_4$ und $\tilde{c}_{p,B,4}$ jeweils die Werte herangezogen werden, die im Rahmen der im Rahmen des linear-impliziten Euler-Verfahrens ohnehin nötigen Auswertung der rechten Seite ermittelt werden.

Die Ergebnisse mit dieser Variante sind in Abbildung 4.9 dargestellt. Sie führen zu einem zufrieden stellenden Simulationsverlauf.

Bei dieser Variante der Vergröberung der Jacobi-Matrix ist im Rahmen eines linear-impliziten Euler-Verfahrens zur Ermittlung der vergröberten Jacobi-Matrix keine zusätzliche Auswertung der rechten Seite f über die ohnehin nötige Auswertung $f\left(x_n\right)$ hinaus notwendig.

Zu beachten ist, dass diese und die vorangehenden Variante auf die Referenztrajektorie zugeschnitten sind und anhand dieser überprüft wurden. Eine Anwendung in einem anderen Kontext bedarf einer erneuten Überprüfung an einem geeigneten Satz von Referenztrajektorien (vgl. Seite 59).

Ausblick: Physikalisch motivierte Vergröberung

Die Vergröberung der Jacobi-Matrix kann physikalisch motiviert sein. Hierzu wird zunächst folgendes Beispiel ohne physikalischen Bezug betrachtet:

Sei ein Modell der Form

$$\dot{x}\left(t\right) = Lx + N\left(f\left(x\right)\right)$$

gegeben mit einer Matrix L und einer nichtlinearen Funktion N, die eine kleine Lipschitz-

Konstante L_N besitzt. Sei $L_N \ll \|L\|_2$. Die Steifheit des Modells werde also vor allem durch den Anteil Lx verursacht. Dann kann es möglich sein, N im Rahmen einer Jacobi-Matrix-Vergröberung nicht berücksichtigen zu müssen, also die Jacobi-Matrix-Approximation A zu wählen als $A = L$. Dies führt je nach Beschaffenheit der Jacobi-Matrix der Funktion N zu einer wesentlichen Reduktion des Berechnungsaufwandes der Jacobi-Matrix-Approximation. Es führt aber wie auf Seite 58 beschrieben nicht dazu, dass der Einfluss durch N auf das Systemverhalten vernachlässigt wird.

Das Prinzip dieses Beispiels lässt sich folgendermaßen bei mathematischen Modellen verwenden:

Sind bei einem mathematischen Modell eine Reihe von physikalischen Effekten involviert, von denen nur manche zur Steifheit des Systems beitragen, so kann es gut geeignet sein, im Rahmen der Berechnung der vergröberten Jacobi-Matrix A die Gleichungen eines Ersatzmodells heranzuziehen, bei dem nur diese zur Steifheit führenden physikalischen Effekte berücksichtigt sind. Derartige Ersatzmodelle können deutlich weniger komplex als die Originalmodelle ausfallen.

Im Fall des Rohrmodells liegt beispielsweise die Situation vor, dass die Dynamik der Rohrtemperaturen deutlich langsamer ist als es die Temperaturänderungen in der Gasphase sind. Daher ist es im Sinne vorangehender Überlegung, ein Ersatzmodell des Rohrmodells im Rahmen der Berechnung der Jacobi-Matrix heranzuziehen, bei dem die Rohrtemperaturen als invariant innerhalb eines Zeitschrittes angenommen werden.

Dieses Vorgehen ist bei einer Anwendung bei der Referenztrajektorie geeignet: Es entspricht einer Mixed-Mode Integration (vgl. 4.2.1), bei der die Rohrtemperaturen explizit gerechnet werden, und hat zur Folge, dass die entsprechenden Einträge der vergröberten Jacobi-Matrix zu Null werden. Die Legitimität dieses Vorgehens wird durch die Untersuchungen zum Sparsing gerechtfertigt.

Ausblick: Anwendung von Intervallarithmetik und Uncertainty Quantification

Damit die Vergröberung der Jacobi-Matrix noch schärfer auf einen gegebenen Satz an Referenztrajektorien abgestimmt werden kann, ist folgendes Vorgehen denkbar: Für die beteiligten Zustände, Eingänge und Parameter werden jeweils Intervalle erfasst, in denen sie enthalten sein müssen. Für die physikalischen Größen des Rohrmodells wie beispielsweise die Temperaturen von Gas, Rohr und Umgebung und den beteiligten den Drücken ergeben sich allein dadurch natürliche Intervallgrenzen, dass das Modell nur für gewisse Temperaturbereiche ausgelegt ist und sich Schranken für beteiligten Drücke abschätzen lassen. Anstelle von Intervallen sind auch Wahrscheinlichkeitsverteilungen interessant. Im Fall von Intervallen lässt sich Intervallarithmetik (vgl. [7]) und im Fall von Wahrscheinlichkeitsverteilungen lassen sich Methoden der Uncertainty Quantification (vgl. [68]) einsetzen, um für auftretende Zwischengrößen oder die Jacobi-Matrix-Einträge selbst Intervalle abzuschätzen, in denen sie mit sehr großer Wahrscheinlichkeit liegen. Durch Verwendung von speziell für diese Intervalle gültigen Approximationen eröffnen sich weiter reichende Approximationsmöglichkeiten im Rahmen der Jacobi-Matrix-Vergröberung.

4.4 Implementierung

In diesem Abschnitt wird untersucht, wie sich die vorangehend bestimmten Vergröberungen der Jacobi-Matrix im Rahmen eines linear-impliziten Euler-Verfahrens in eine Reduktion der Rechenzeit auf dem Steuergerät ummünzen lassen. Hierzu werden folgende fünf Varianten betrachtet:

1. Berechnung der Jacobi-Matrix mittels numerischer Differenziation ohne weitere Anpassungen.
2. Berechnung der Jacobi-Matrix mit Einsatz der in Abschnitt 4.2.1 beschriebenen Ansätze Mixed-Mode Integration und Färbung der Jacobi-Matrix.
3. Berechnung einer vergröberten Jacobi-Matrix mittels einer Polynominterpolation eines Kennfelds gemäß Seite 66.

4. Berechnung einer vergröberten Jacobi-Matrix mit einer Gleichsetzung verschiedener Einträge gemäß Seite 65.
5. Berechnung einer vergröberten Jacobi-Matrix mit Verwendung eines symbolischen Approximationsausdrucks gemäß Seite 67.

Die Implementierung erfolgt in C. Als Entwicklungsumgebung wird ECU.WorX 2014.1.2 der *DGS Toolbase* der Robert Bosch GmbH eingesetzt. Dieses ist das gängige Tool für die Programmierung der MDG1-Steuergeräte-Software. Für die Lösung des linearen Gleichungssystems wird, sofern nicht anders angegeben, eine Gauß Elimination mit Spaltenpivotisierung eingesetzt. Der Aspekt der Lösung linearer Gleichungssysteme steht bei den folgenden Varianten nicht im Fokus. Eine ausführliche Untersuchung zur Rechenzeitreduktion innerhalb der Lösung linearer Gleichungssysteme findet sich in Kapitel 5.

Variante 1 In dieser Variante wird numerische Differenziation ohne weitere Anpassungen eingesetzt. Wie in Abschnitt 3.2.3 dargestellt ist der Einsatz der numerischen Differenziation ohne Absicherungsmaßnahme im Allgemeinen anfällig, da Nichtlinearitäten zu erheblichen Fehlern in den erhaltenen Einträgen der Jacobi-Matrix führen können. Aus diesem Grund und da die Erkenntnisse aus den Untersuchungen im Rahmen der Bestimmung einer Vergröberung der Jacobi-Matrix in dieser Variante nicht berücksichtigt werden, wird bei dieser Variante für jede numerische Richtungsableitung

$$D_{x+\Delta x^{(i)}} f \approx \delta^{-1} \left(f\left(x_n + \Delta x^{(i)}\right) - f\left(x_n\right) \right)$$

eine zusätzliche Berechnung

$$D_{x+\Delta x^{(i)}} f \approx \left(\frac{\delta}{2}\right)^{-1} \left(f\left(x_n + \frac{\Delta x^{(i)}}{2}\right) - f\left(x_n\right) \right)$$

zur Überprüfung gebildet, um unzureichende Resultate der numerischen Differenziation durch eine große Abweichung zwischen beiden Berechnungsweisen zu detektieren.

Variante 2 Zunächst werden Zustände identifiziert, die auch bei einer expliziten Integration zu einem stabilen Simulationsverlauf führen. In Gleichung 4.13 sind die Eigenwerte einer Jacobi-Matrix aus der Referenztrajektorie angegeben. Diese lassen sich in offensichtlicher Weise in Gruppen von Eigenwerten von unterschiedlichen Größenordnungen einteilen, wobei die Eigenwerte der niedrigen Größenordnungen zu Kandidaten von Zuständen führen, die bei einer expliziten Integration mit einer Zeitschrittweite von $\Delta t = 0.1\,\mathrm{s}$ einen stabilen Verlauf nach sich ziehen. Wie auf Seite 64 erläutert, stehen diese betragsmäßig kleinen Eigenwerte in einem direkten Bezug zu den Zuständen $T_{C,i}$ mit $i = 1, \ldots, 5$. Dieser Umstand liegt für alle Jacobi-Matrizen der Referenztrajektorie vor. Daher sind die Rohrtemperaturen $T_{C,i}$ potentiell für eine explizite Berechnung im Rahmen einer Mixed Mode Integration geeignet. Dass dies tatsächlich der Fall ist, wird auch dadurch gestützt, dass die Sparsing-Struktur aus Gleichung 4.16 eine letztlich explizite Berechnung der Rohrtemperaturen umfasst. Die Umsetzung der Mixed Mode Integration wird in der Weise durchgeführt, dass die zu den Zuständen $T_{C,i}$ mit $i = 1, \ldots, 5$ korrespondierenden Zeilen und Spalten der Jacobi-Matrizen als Null angenommen und lediglich die übrigen Einträge der Jacobi-Matrizen ermittelt werden.

Die Mixed Mode Integration wird in vorliegender Variante kombiniert mit einer Färbung der Jacobi-Matrix. Die Strukturmatrix $\tilde{S}$ der Teilmatrix $\tilde{J}$ der Jacobi-Matrix J_f, bei der die durch die Mixed Mode Integration entstehenden Nullzeilen und -spalten nicht enthalten sind, ist folgende:

$$\tilde{S} = \left(\begin{array}{cccccccccccccccccccccccc}
\blacksquare & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \blacksquare & \cdot & \cdot & \cdot & \cdot \\
\blacksquare & \blacksquare & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \blacksquare & \blacksquare & \cdot & \cdot & \blacksquare \\
\cdot & \blacksquare & \blacksquare & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \blacksquare & \cdot & \blacksquare & \cdot & \blacksquare \\
\cdot & \cdot & \blacksquare & \blacksquare & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \blacksquare & \cdot & \cdot & \blacksquare & \blacksquare \\
\cdot & \cdot & \cdot & \blacksquare & \blacksquare & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \blacksquare & \cdot & \cdot & \cdot & \blacksquare \\
\cdot & \cdot & \cdot & \cdot & \cdot & \blacksquare & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \blacksquare & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \blacksquare & \blacksquare & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \blacksquare & \blacksquare & \cdot & \cdot & \blacksquare \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \blacksquare & \blacksquare & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \blacksquare & \cdot & \blacksquare & \cdot & \blacksquare \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \blacksquare & \blacksquare & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \blacksquare & \cdot & \cdot & \blacksquare & \blacksquare \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \blacksquare & \blacksquare & \cdot & \cdot & \cdot & \cdot & \cdot & \blacksquare & \cdot & \cdot & \cdot & \blacksquare \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \blacksquare & \blacksquare & \cdot & \cdot & \cdot & \cdot & \blacksquare & \blacksquare & \cdot & \cdot & \blacksquare \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \blacksquare & \blacksquare & \cdot & \cdot & \cdot & \blacksquare & \cdot & \blacksquare & \cdot & \blacksquare \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \blacksquare & \blacksquare & \cdot & \cdot & \blacksquare & \cdot & \cdot & \blacksquare & \blacksquare \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \blacksquare & \blacksquare & \cdot & \blacksquare & \cdot & \cdot & \cdot & \blacksquare \\
\blacksquare & \cdot & \cdot & \cdot & \cdot & \blacksquare & \cdot & \cdot & \cdot & \blacksquare & \cdot & \cdot & \cdot & \blacksquare & \cdot & \blacksquare & \cdot & \cdot & \cdot & \cdot \\
\cdot & \blacksquare & \cdot & \cdot & \cdot & \cdot & \blacksquare & \cdot & \cdot & \cdot & \blacksquare & \cdot & \cdot & \cdot & \cdot & \blacksquare & \blacksquare & \cdot & \cdot & \blacksquare \\
\cdot & \cdot & \blacksquare & \cdot & \cdot & \cdot & \cdot & \blacksquare & \cdot & \cdot & \cdot & \blacksquare & \cdot & \cdot & \cdot & \blacksquare & \blacksquare & \blacksquare & \cdot & \blacksquare \\
\cdot & \cdot & \cdot & \blacksquare & \cdot & \cdot & \cdot & \cdot & \blacksquare & \cdot & \cdot & \cdot & \blacksquare & \cdot & \cdot & \blacksquare & \cdot & \blacksquare & \blacksquare & \blacksquare \\
\cdot & \cdot & \cdot & \cdot & \blacksquare & \cdot & \cdot & \cdot & \cdot & \blacksquare & \cdot & \cdot & \cdot & \blacksquare & \blacksquare & \cdot & \cdot & \blacksquare & \blacksquare & \blacksquare
\end{array} \right) . \tag{4.19}$$

Für $\tilde{S}$ ist eine Färbung zu bestimmen, durch die möglichst wenig Richtungsableitungen zur Ermittlung von $\tilde{J}$ ausreichen. In einer Zeile treten bis zu sieben nicht verschwindende Einträge auf. Aus diesem Grund sind neben $f(x_n)$ mindestens sieben weitere Richtungsableitungen zur Bestimmung von $\tilde{J}$ nötig. Folgende Wahl für die Richtungsvektoren der Richtungsableitungen ist geeignet, mit sieben weiteren Richtungsableitungen tatsächlich $\tilde{J}$ ermitteln zu können.

$$\begin{aligned}
\Delta x^{(1)} &= \delta e_{16}, \\
\Delta x^{(2)} &= \delta e_{20}, \\
\Delta x^{(3)} &= \delta \left(e_{17} + e_{19} \right), \\
\Delta x^{(4)} &= \delta \left(e_{18} \right), \\
\Delta x^{(5)} &= \delta \left(e_1 + e_3 + e_5 + e_7 + e_9 \right), \\
\Delta x^{(6)} &= \delta \left(e_6 + e_8 + e_{10} + e_{12} + e_{14} \right), \\
\Delta x^{(7)} &= \delta \left(e_2 + e_4 + e_{11} + e_{13} + e_{15} \right).
\end{aligned}$$

Hierbei ist e_i jeweils der i-te Einheitsvektor der Standardbasis von $\mathbb{R}^{25}$. Sukzessive werden die Komponenten aus den Richtungsableitungen $D_{x_n + \Delta x^{(i)}} f$ den entsprechenden Werte der Jacobimatrix $\tilde{J}$ zugewiesen. In der folgenden Matrix ist für jeden nicht verschwindenden Eintrag angegeben, aus welcher Richtungsableitung er abgelesen wird:

$$\tilde{P} = \left(\begin{array}{cccccccccccccccccccccccc}
5 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 1 & \cdot & \cdot & \cdot & \cdot \\
5 & 7 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 1 & 3 & \cdot & \cdot & 2 \\
\cdot & 7 & 5 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 1 & \cdot & 4 & \cdot & 2 \\
\cdot & \cdot & 5 & 7 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 1 & \cdot & \cdot & 3 & 2 \\
\cdot & \cdot & \cdot & 7 & 5 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 1 & \cdot & \cdot & \cdot & 2 \\
\cdot & \cdot & \cdot & \cdot & \cdot & 6 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 1 & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & 6 & 5 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 1 & 3 & \cdot & \cdot & 2 \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 5 & 6 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 1 & \cdot & 4 & \cdot & 2 \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 6 & 5 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 1 & \cdot & \cdot & 3 & 2 \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 5 & 6 & \cdot & \cdot & \cdot & \cdot & \cdot & 1 & \cdot & \cdot & \cdot & 2 \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 7 & \cdot & \cdot & \cdot & \cdot & 1 & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 7 & 6 & \cdot & \cdot & \cdot & 1 & 3 & \cdot & \cdot & 2 \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 6 & 7 & \cdot & \cdot & 1 & \cdot & 4 & \cdot & 2 \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 7 & 6 & \cdot & 1 & \cdot & \cdot & 3 & 2 \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 6 & 7 & 1 & \cdot & \cdot & \cdot & 2 \\
5 & \cdot & \cdot & \cdot & \cdot & 6 & \cdot & \cdot & \cdot & \cdot & 7 & \cdot & \cdot & \cdot & \cdot & 1 & \cdot & \cdot & \cdot & 2 \\
\cdot & 7 & \cdot & \cdot & \cdot & \cdot & 5 & \cdot & \cdot & \cdot & \cdot & 6 & \cdot & \cdot & \cdot & 1 & 3 & \cdot & \cdot & 2 \\
\cdot & \cdot & 5 & \cdot & \cdot & \cdot & \cdot & 6 & \cdot & \cdot & \cdot & \cdot & 7 & \cdot & \cdot & 1 & 3 & 4 & \cdot & 2 \\
\cdot & \cdot & \cdot & 7 & \cdot & \cdot & \cdot & \cdot & 5 & \cdot & \cdot & \cdot & \cdot & 6 & \cdot & 1 & \cdot & 4 & 3 & 2 \\
\cdot & \cdot & \cdot & \cdot & 5 & \cdot & \cdot & \cdot & \cdot & 6 & \cdot & \cdot & \cdot & \cdot & 7 & 1 & \cdot & \cdot & 3 & 2
\end{array} \right) . \tag{4.20}$$

Somit wird der Eintrag $j_{16,1}$ beispielsweise aus der Komponente 16 der Richtungsableitung $D_{x_n + \Delta x^{(5)}} f$ abgelesen. Für die Lösung des linearen Gleichungssystems im Rahmen des linear-

impliziten Euler-Verfahrens wird eine Gauß Elimination mit Spaltenpivotisierung eingesetzt. Diese wird auf das gesamte lineare Gleichungssystem angewandt. Hierdurch werden die im Rahmen der Mixed Mode Integration entstehenden Nulleinträge der Matrix des Gleichungssystems nicht in eine adäquate Reduktion der Rechenzeit umgemünzt. Dies würde beispielsweise erreicht, indem lediglich das durch die führende Teilmatrix $\tilde{J}$ induzierte lineare Gleichungssystem mit einer Gauß Elimination gelöst wird und die Zuständen $T_{C,i}$ in expliziter Form aus der Auswertung der rechten Seite bestimmt werden.

Variante 3 Diese Variante unterscheidet sich von der Ausgangsfassung in der Berechnung der Werte $\alpha_{B,C,i}$, $i = 1, \ldots, 5$. Anstatt einer Implementierung der physikalischen Beziehungen aus Abschnitt 2.2.2 wird hierzu eine Polynomapproximation gemäß Gleichung 4.17 eingesetzt. Im Sinne einer Vergröberung der Jacobi-Matrix wird bei dieser Variante auf einen Absicherungsmechanismus bei der numerischen Differenziation verzichtet, da die Eignung der verwendeten Differenziation im Rahmen einer Überprüfung anhand einer repräsentativen Referenztrajektorie festgestellt wird.

Variante 4 Hierbei wird eine zusätzliche Auswertung der rechten Seite $f\left(x_n + (e_1 + e_{19})\,\delta\right)$ berechnet, wobei e_i die entsprechenden Standardeinheitsvektoren und δ eine geeignete Auslenkungsweite berechne. Hiermit lässt sich j_{11} berechnen durch

$$j_{11} = \frac{f_1\left(x_n + (e_1 + e_{19})\,\delta\right) - f_1\left(x_n\right)}{\delta},$$

da wegen der Struktur von J_n (vgl. Gleichung 4.12)

$$\frac{f_1\left(x_n + (e_1 + e_{19})\,\delta\right) - f_1\left(x_n\right)}{\delta} = \frac{f_1\left(x_n + e_1\delta\right) - f_1\left(x_n\right)}{\delta}$$

gilt. Analog wird j_{19} berechnet durch

$$j_{19,19} = \frac{f_{19}\left(x_n + (e_1 + e_{19})\,\delta\right) - f_{19}\left(x_n\right)}{\delta}.$$

Das im Rahmen des linear-impliziten Euler-Verfahrens entstehende Gleichungssystem hat Bidiagonalform und wird durch eine entsprechend angepasste Vorwärtssubstitution gelöst.

Variante 5 Die Implementierung dieser Variante ergibt sich unmittelbar aus den Gleichungen auf Seite 67. Das im Rahmen des linear-impliziten Euler-Verfahrens entstehende Gleichungssystem hat Bidiagonalform und wird durch eine entsprechend angepasste Vorwärtssubstitution gelöst.

In den vorangehenden Varianten ist das Festhalten der Jacobi-Matrix (vgl. Abschnitt 4.2.1) nicht beinhaltet. Es wird im Folgenden vor dem Hintergrund einer Echtzeitanwendung beleuchtet.

Festhalten der Jacobi-Matrix Wie in Abschnitt 3.2.3 erläutert führt dieser Ansatz nicht unmittelbar zu einer Reduktion der Rechenzeit, wenn wie in Abschnitt 3.2.2 beschrieben die Rechenzeit in den ungünstigsten praktisch auftretenden Zeitschritte als Maß herangezogen wird. Dies liegt daran, dass in den Zeitschritten, in denen die Jacobi-Matrix erneuert wird, die Rechenzeit einer vollen Jacobi-Matrix-Berechnung nötig ist. Aus diesem Grund ist das Festhalten der Jacobi-Matrix in einer variierten Form anzuwenden, falls eine Rechenzeitreduktion in einer für Echtzeitanwendungen relevanten Weise erzielt werden soll. In [65] wird hierzu folgendes Vorgehen eingesetzt:

Sei ein Festhalten der Jacobi-Matrix für eine betrachtete Differentialgleichung über $2k$ Zeitschritte hinweg möglich und sei n die Anzahl der Richtungsableitungen die zur Ermittlung der

Rechenzeit [µs]	5 Zellen	10 Zellen	15 Zellen
Variante 1	3692	14244	31594
Variante 2	1211	2585	3784
Variante 3	1241	4350	9344
Variante 4	152	292	437
Variante 5	78	153	226

Tabelle 4.2: Rechenzeiten der verschiedenen Varianten der Approximation der Jacobi-Matrix samt Auswertung der rechten Seite.

Jacobi-Matrix verwendet werden. Dann werden zwei Speicherungen einer Jacobi-Matrix J und $\tilde{J}$ sowie des Zustandsvektors x und $\tilde{x}$ eingesetzt. In jedem k-ten Zeitschritt wird der Zustandsvektor x in $\tilde{x}$ gesichert. Sei $\tilde{J}$ weniger aktuell als J. Dann wird $\tilde{J} = \frac{\partial}{\partial x} f(\tilde{x})$ berechnet, indem über k Zeitschritte hinweg je Zeitschritt $\left\lceil \frac{n}{k} \right\rceil$ Richtungsableitungen bestimmt werden. Hierdurch wird der Rechenaufwand zur Ermittlung von $\tilde{J}$ auf k Zeitschritte verteilt. Nachdem $\tilde{J}$ erneuert ist, tauschen J und $\tilde{J}$ ihre Rollen: $\tilde{J}$ wird nun für k Zeitschritte als Jacobi-Matrix eingesetzt, während J erneuert wird.

Insgesamt wird bei diesem Vorgehen die Rechenzeit auf etwa den Faktor

$$\eta \approx \frac{\left\lceil \frac{n}{k} \right\rceil}{n}$$

der regulären Rechenzeit reduziert. Der Speicherbedarf für die Jacobi-Matrix wird verdoppelt. Für eine geeignete Initialisierung ist es gegebenenfalls nötig, die Simulation mit einer k Zeitschritte andauernden Wartezeit zu starten, um in dieser Zeit die erste Jacobi-Matrix zu ermitteln.

Die Anpassung des Festhaltens der Jacobi-Matrix wurde in [66] auf das Rohrmodell aus Abschnitt 2.2 angewandt und bei der Referenztrajektorie aus Abschnitt 4.1.2 die resultierenden Fehler im Falle einer Simulation auf einem PC ermittelt. Es zeigt sich, dass in diesem Fall ein Festhalten über sechs Zeitschritte der Schrittweite $\Delta t = 100\,\mathrm{ms}$ mit einer geringen Fehlerzunahme möglich ist, während bei einem Festhalten über mehr als acht Zeitschritte der Schrittweite $\Delta t = 100\,\mathrm{ms}$ der Fehler in inakzeptabler Weise zunimmt. Dementsprechend ist, sofern sich die Ergebnisse auf die Anwendung auf einem Steuergerät übertragen lassen, eine Reduktion der Rechenzeit auf etwa

$$\eta = \frac{\left\lceil \frac{25}{\frac{6}{2}} \right\rceil}{25} = 36\,\%$$

der Ausgangsrechenzeit zu erwarten, wenn keine Färbung der Jacobi-Matrix eingesetzt wird.

4.5 Numerische Experimente

Im Folgenden werden Rechenzeitmessungen für den Einsatz der vorangehend beschriebenen Varianten der Approximation der Jacobi-Matrix sowie der zugehörigen Varianten des linear-impliziten Euler-Verfahrens auf dem Steuergerät MDG1C Device 4 von Bosch durchgeführt (vgl. Abschnitt 3.1.1). Hierzu wird das Programm SMART 2.1.1 der *DGS Toolbase* von Bosch eingesetzt. Die Laufzeitmessungen werden für das Rohrmodell mit den Anfangswerten, Eingangsdaten und Parameter eines Ausschnittes der Referenztrajektorie von 20 Zeitschritten der Schrittweite 100 ms durchgeführt. Durch SMART 2.1.1 werden jeweils die minimale, maximale und die durchschnittliche Rechenzeit der Funktion zur Jacobi-Matrix-Berechnung angegeben. Hiervon wird im Folgenden jeweils der Wert zur maximalen Rechenzeit verwendet.

Um die Aussagekraft der Ergebnisse zu erhöhen, werden auch für eine Semidiskretisierung des Rohrmodells mit zehn Zellen sowie mit 15 Zellen jeweils analoge Varianten gemessen.

In Tabelle 4.2 sind Rechenzeiten für die Ermittlung der Jacobi-Matrix samt der Auswertung der rechten Seite aufgeführt. Bei Variante 1 sind die Proportionen zwischen den Rechenzeiten bei fünf, zehn und 15 Diskretisierungszellen $1 : 3.86 : 8.56$. Somit skalieren diese Rechenzeiten in einer quadratischen Abhängigkeit von der Anzahl der Zustände. Dies ist dadurch naheliegend, dass die Anzahl der Auswertungen der rechten Seite proportional zur Anzahl der Zustände ist und für die Rechenzeit einer einzelnen Auswertung der rechten Seite ein lineares Skalierungsverhalten in der Anzahl der Zustände plausibel ist.

Die Kombination aus Mixed Mode Integration und der Verwendung einer Färbung der Jacobi-Matrix führt bei Variante 2 zu einer Reduktion der für die Ermittlung der Jacobi-Matrix sowie die Auswertung der rechten Seite benötigten Rechenzeit um $67,2\,\%$ bei fünf Diskretisierungszellen und um $88,0\,\%$ bei 15 Diskretisierungszellen. Dass sich die Rechenzeitreduktion für die verschiedenen Anzahlen an Diskretisierungszellen in diesem Maß unterscheidet, ist darauf zurückzuführen, dass bei Variante 2 die Rechenzeiten aus Tabelle 4.2 ein Skalierungsverhalten zeigen, das linear ist in der Anzahl der Zustände mit den Proportionen $1 : 2.13 : 3.12$. Der Grund hierfür ist, dass die Färbung der Jacobi-Matrix auch bei zehn und 15 Diskretisierungszellen mit sieben Farben analog zu Gleichung 4.20 möglich ist und sich die Anzahl der benötigten Auswertungen der rechten Seite bei den verschiedenen Anzahlen an Diskretisierungszellen nicht unterscheidet. Aus diesem Grund ist insbesondere die Methode der Färbung der Jacobi-Matrix bei der Struktur der Jacobi-Matrix des Rohrmodells bei einer großen Zahl an Diskretisierungszellen gut geeignet.

Bei Variante 3 liegt die Rechenzeitreduktion gegenüber Variante 1 bei $66,4\,\%$ bei fünf Diskretisierungszellen, bei $69,5\,\%$ bei zehn Diskretisierungszellen und bei $70,4\,\%$ bei 15 Diskretisierungszellen und ist somit für die verschiedenen Anzahlen an Diskretisierungszellen ähnlich. Dies ist zu erwarten, da sich Variante 3 gegenüber Variante 1 zum Einen darin unterscheidet, dass kein Absicherungsmechanismus bei der numerischen Differenziation eingesetzt wird und zum Anderen in der Berechnung der physikalischen Koeffizienten während der Auswertungen der rechten Seite im Rahmen der Jacobi-Matrix-Bildung. Somit ist es naheliegend, dass der Quotient der Rechenzeiten etwa den doppelten Wert des Quotienten der Rechenzeit einer Auswertung der rechten Seite und der Rechenzeit einer Auswertung der vergröberten rechten Seite besitzt.

Variante 4 und Variante 5 besitzen gegenüber den vorangehenden Varianten signifikant geringere Werte in Tabelle 4.2. Im Fall von fünf Diskretisierungszellen beträgt die Reduktion der Rechenzeit gegenüber der Ausgangsvariante $95,9\,\%$ bei Variante 4 und $97,9\,\%$ bei Variante 5. Im Fall von 15 Diskretisierungszellen beträgt der Anteil der Rechenzeitreduktion $98,4\,\%$ bei Variante 4 und $99,3\,\%$ bei Variante 5. Die Rechenzeiten von Variante 5 sind bei fünf Diskretisierungszellen um $48.7\,\%$ geringer als die entsprechenden Rechenzeiten von Variante 4, bei zehn Diskretisierungszellen um $48.6\,\%$ und bei 15 Diskretisierungszellen um $48,3\,\%$. Es ist festzustellen, dass die Proportionen der Anzahl der Auswertungen der rechten Seite im Rahmen eines Zeitschrittes von Variante 1, Variante 4 und Variante 5 eine gute Näherung für die Proportionen der Rechenzeiten darstellen: Bei fünf Diskretisierungszellen verwendet Variante 5 eine Auswertung der rechten Seite, Variante 4 zwei Auswertungen und Variante 1 verwendet 50 Auswertungen. Die Proportionen der entsprechenden Rechenzeiten aus Tabelle 4.2 sind $78 : 152 : 3692 \approx 1 : 1.95 : 47.3$. Da bei Variante 1 bei zehn Diskretisierungszellen 100 Auswertungen der rechten Seite und bei 15 Diskretisierungszellen 150 Auswertungen der rechten Seite berechnet werden und bei Variante 4 respektive Variante 5 unabhängig von der Anzahl der Diskretisierungszellen stets zwei respektive eine Auswertung der rechten Seite verwendet werden, wird durch diese Beobachtung erklärt, weshalb die Rechenzeitreduktion durch Variante 4 und Variante 5 mit zunehmender Anzahl an Diskretisierungszellen zunimmt.

Wie in Abschnitt 4.1.3 beschrieben, dominiert der Rechenaufwand für die Bildung der Jacobi-Matrix den Rechenaufwand eines Zeitschritt des linear-impliziten Euler-Verfahrens im Fall der Leitanwendung. Durch die in diesem Teilkapitel beschriebenen Ansätze können wie vorangehend dargestellt die Rechenzeiten zur Ermittlung der Jacobi-Matrix samt der Auswertung der rechten Seite erheblich reduziert werden. Gemäß den Beobachtungen aus Abschnitt 4.1.3 ist somit zu erwarten, dass hierdurch die Lösung der linearen Gleichungssysteme im Rahmen des linear-

Rechenzeit [µs]	5 Zellen	10 Zellen	15 Zellen
Variante 1	4485	14104	52049
Variante 2	2045	7701	24006
Variante 3	2068	10582	29847
Variante 4	157	302	451
Variante 5	84	163	240

Tabelle 4.3: Rechenzeiten eines Zeitschritts verschiedener Varianten des linear-impliziten Euler-Verfahrens.

impliziten Euler-Verfahrens zum dominierenden Teil der Rechenzeit eines Zeitschritts führt. In Tabelle 4.3 sind die Rechenzeiten der verschiedenen Varianten für einen Zeitschritt aufgeführt. Durch Vergleich mit Tabelle 4.2 wird hieraus ersichtlich, dass bei den Variante 1 bis 3 die Lösung des linearen Gleichungssystems den dominierenden Anteil der Rechenzeit ausmacht. Bei diesen Varianten können die Ansätze aus Kapitel 5 zur Reduktion der Rechenzeit bei der Lösung linearer Gleichungssysteme eingesetzt werden. Auf Grund der in Variante 4 und Variante 5 entstehenden Bidiagonalform des linearen Gleichungssystems, die sich durch das zu Grunde liegende Sparsing gemäß Abschnitt 4.3.1 ergibt, ist bei diesen Varianten die Lösung des linearen Gleichungssystems mit einem sehr geringen Rechenaufwand verbunden. Die Rechenzeiten für einen Zeitschritt bei diesen Varianten ist um Größenordnungen geringer als die Rechenzeit bei der Ausgangsvariante. Dementsprechend ist die Zielsetzung der Reduktion der Rechenzeit des linear-impliziten Euler-Verfahrens bei der Referenztrajektorie des Rohrmodells gemäß Abschnitt 4.1.3 mit diesen beiden Varianten bereits erreicht.

5 Lösung dünn besetzter linearer Gleichungssysteme

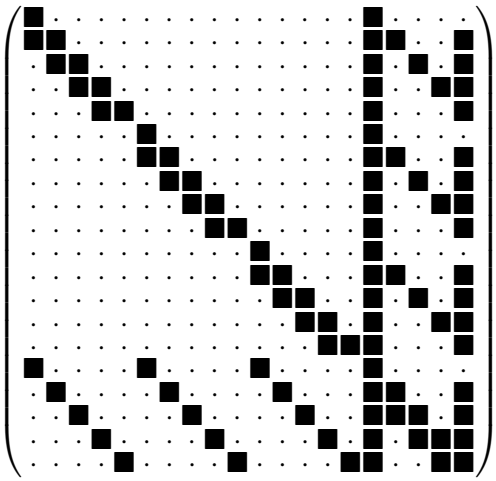
Im Abschnitt 4.2 wurde untersucht, wie die Rechenzeit für einen Zeitschritt einer Echtzeitsimulation des Rohrmodells aus Abschnitt 2.2 reduziert werden kann. Wie im Abschnitt 4.1.3 dargestellt ist, ist hierzu eine Reduktion der Rechenzeit für die Jacobi-Matrix-Ermittlung vorrangig. Das Ziel einer Rechenzeitreduktion bei der Jacobi-Matrix-Approximation wird in Abschnitt 4.2 verfolgt und erreicht.

In diesem Kapitel wird nun untersucht, wie der Rechenzeitanteil der Gleichungslösung in einem linear-impliziten Euler-Verfahren reduziert werden. Wie in Abschnitt 2.1.7 beschrieben, sind die Gleichungssysteme der betrachteten Problemklasse dünn besetzt. Zudem wird durch den Einsatz eines Sparsings im Rahmen der Jacobi-Matrix-Vergröberung die Dünnbesetztheit sogar noch gesteigert. Aus diesem Grund wird im Folgenden der Ansatz verfolgt, dass die Dünnbesetztheit der Systeme zur Reduktion der Rechenzeit genutzt wird.

Die Untersuchungen werden am Beispiel von linearen Gleichungssystemen mit der Struktur des Rohrmodells vorgenommen. Um einen geeigneten Untersuchungsgegenstand zu haben, werden hierbei die durch ein Sparsing gemäß Abschnitt 4.3.1 erzielbare Bidiagonalstruktur nicht berücksichtigt. Stattdessen wird untersucht, welche Rechenzeitreduktion erzielt werden kann, sofern keine Vergröberung der Jacobi-Matrix zum Einsatz kommt. Daher wird die Struktur, die sich im Rahmen einer Mixed-Mode Integration ergibt, betrachtet. In Kombination mit dem Einsatz einer Färbung der Jacobi-Matrix gemäß Abschnitt 4.4 lässt sich hierdurch eine optimierte Variante erzielen, die keine Jacobi-Matrix-Vergröberung umfasst.

Die untersuchte Struktur ist also folgende:

$$\tilde{S} = \left(\right) . \tag{5.1}$$



Wie bei der Berechnung der Jacobi-Matrix wird auch im Folgenden die Situation genutzt, dass im Rahmen einer entsprechenden Untersuchung eines mathematischen Modells bereits vor der Implementierung die Struktur der linearen Gleichungssysteme bekannt ist, die im Rahmen einer linear-impliziten Integration des mathematischen Modells auftritt.

5.1 Kondition

Im Rahmen einer Software-Funktion auf einem Motorsteuergerät steht zur Speicherung einer Gleitkomma-Zahl standardmäßig keine andere Größe zur Verfügung als 32 Bit (vgl. Abschnitt 3.1.1). Aus diesem Grund wird im Folgenden untersucht, ob durch diese begrenzte Präzision zusätzliche Anforderungen an die Lösungsverfahren gestellt werden müssen. Hierzu ist die Kondition der Gleichungssysteme eine wesentliche Eigenschaft (vgl. [53, 59, 30]). Dies beruht wesentlich auf den Zusammenhängen aus folgendem Paragraphen.

Fehlerabschätzung bei einer Gauß Elimination in Gleitkomma-Arithmetik Bei der Lösung eines linearen Gleichungssystem $Ax = b$ führt eine Gleitkomma-Arithmetik zunächst dazu, dass die beteiligten Größen nicht exakt, sondern in einer entsprechend beschränkten Genauigkeit gespeichert werden, so dass anstelle des Gleichungssystems $Ax = b$ das gestörte Problem $(A + \Delta A)\, y = b + \Delta b$ vorliegt. Das folgende Resultat gibt eine Schranke an den Fehler, der aus der nicht exakten Speicherung der beteiligten Größen resultiert. Hierbei ist die Kondition $\kappa(A)$ der Matrix A ein entscheidender Faktor für die Auswirkung der Störung auf die Fehlerschranke.

Satz 5. *Sei $Ax = b$ sowie $(A + \Delta A)\, y = b + \Delta b$ mit $A, \Delta A \in \mathbb{R}^{n\times n}$, $b \in \mathbb{R}^n \setminus \{0\}$ und $\Delta b \in \mathbb{R}^n$. Weiter sei $\|\Delta A\| \leq \epsilon \|A\|$ und $\|\Delta b\| \leq \epsilon \|b\|$ sowie $\epsilon\kappa(A) < 1$. Dann ist $A + \Delta A$ nicht-singulär und es gilt*

$$\frac{\|y - x\|}{\|x\|} \leq \frac{2\epsilon\kappa(A)}{1 - \epsilon\kappa(A)}.$$

Beweis. Theorem 2.6.2. in [53]. □

Das vorangehende Resultat beschreibt eine Fehlerschranke für eine Störung des linearen Gleichungssystems, wie sie durch die Speicherung im Gleitkommaformat dargestellt wird. Für die Jacobi-Matrizen J der Referenztrajektorie gilt die Abschätzung $\kappa_\infty(I - \Delta t J) < 2516$. Im IEEE754-Format ist bei 32 Bit Gleitkommavariablen ist die resultierende Störung durch $\epsilon < 10^{-7}$ beschränkt. Hiermit ergibt sich im Fall der Referenztrajektorie beim Rohrmodell die Abschätzung

$$\frac{\|y - x\|_\infty}{\|x\|_\infty} \leq \frac{2\epsilon\kappa_\infty(A)}{1 - \epsilon\kappa_\infty(A)} < \frac{2 \cdot 10^{-7} \cdot 2516}{1 - 10^{-7} \cdot 2516} < 5.04 \cdot 10^{-4}.$$

Somit liegen etwa drei gültige Ziffern vor.

Im Folgenden wird nun eine Aussage aufgeführt, die es ermöglicht, den Fehler im Rahmen einer Elimination bei einem linearen Gleichungssystem mit Gleitkomma-Arithmetik einzuschätzen. Die nachfolgende Aussage ist eine Heuristik gemäß [53].

Heuristik Liegt eine Maschinengenauigkeit von $u \approx 10^{-d}$ sowie eine Kondition $\kappa_\infty(A) \approx 10^q$, so liefert eine Gauß Elimination eine Lösung y mit etwa $d - q$ korrekten Dezimalstellen.

Im Fall des Rohrmodells ist $d = 7$ und $q = 3$, so dass anhand dieser Heuristik mindestens etwa $d - q = 4$ korrekte Dezimalstellen zu erwarten sind. Durch die Gauß Elimination in Maschinenarithmetik wird also die Fehlerschranke gegenüber einer Speicherung im Gleitkommaformat und einer Elimination in exakter Arithmetik nicht vergrößert. Insgesamt ist sind etwa drei gültige Dezimalstellen zu erwarten. Dies ist für den Einsatzzweck ausreichend. Insbesondere ist der Einsatz von Lösungsverfahren basierend auf orthogonalen Transformationen wie beispielsweise ein QR-Verfahren nicht notwendig. Diese sind für schlecht konditionierte lineare Gleichungssysteme besser geeignet wie eine Gauß Elimination, aber sind im Vergleich zur Gauß Elimination rechenaufwändiger.

Einfluss der Mixed-Mode Integration auf die Kondition Sei J_m die Teilmatrix von J_n, die sich im Rahmen einer Mixed-Mode Integration ergibt, bei der also die mit den Rohrtemperaturen korrespondierenden Zeilen und Spalten gestrichen werden. Für sie gilt $\kappa_\infty (I - \Delta t J_m) \approx 596$. Demgegenüber ist $\kappa_\infty (I - \Delta t J_n) \approx 704$. Es ist zu beobachten, dass die Mixed-Mode Integration zu einer zusätzlichen Verbesserung der Kondition der Gleichungssystems führt. Dies ist darauf zurückzuführen, dass die Rohrtemperaturen im Vergleich zu den anderen Zuständen relativ insensitiv sind, so dass das Streichen der entsprechen Matrixzeilen und -spalten die Kondition verbessert.

Pivotisierung Bei einer Gauß Elimination ermöglicht eine Pivotisierung ein numerisch stabiles Lösen gut konditionierter linearer Gleichungssysteme (vgl. [53]). Wird keine Pivotisierung eingesetzt, so ist die Durchführung einer Gauß Elimination beim Auftreten einer Null als Pivotelement nicht möglich. Falls dieser Fall nicht eintritt, so kann auch für gut konditionierte Systeme die Lösung mittels einer Gauß Elimination ohne Pivotisierung numerisch instabil sein. Im Allgemeinen ist aus diesem Grund eine Gauß Elimination mit Pivotisierung wie beispielsweise eine Spaltenpivotisierung einer Gauß Elimination ohne Pivotisierung vorzuziehen.

Im Fall der linearen Gleichungssysteme, die im Rahmen eines linear-impliziten Euler-Verfahrens bei der Referenztrajektorie der Leitanwendung auftreten, ist auch eine Gauß Elimination ohne Pivotisierung möglich und numerisch stabil. Dies liegt daran, dass für die auftretenden Diagonaleinträge a_{ii} stets $a_{ii} \neq 0$ sowie $|a_{ii}| > 0.25\,|a_{ij}|$ für $i > j$ und zudem $|a_{ij}| < 10^{-5}\,|a_{ii}|$ für $i < j$ gilt. Dies führt dazu, dass sich der Wachstumsfaktor

$$\rho_n = \frac{\max_{i,j,k} \left| a_{ij}^{(k)} \right|}{\max_{i,j} |a_{ij}|}$$

einer Gauß Elimination ohne Pivotisierung, wobei $a_{ij}^{(k)}$ die Einträge der Matrix im k-ten Eliminationsschritt seien, abschätzen lässt durch

$$\rho_n < 1 + n \frac{1}{0.25} 10^{-5} < 2.$$

Gemäß [44] gilt die Fehlerabschätzung

$$\frac{\|y - x\|_\infty}{\|x\|_\infty} \leq \gamma \kappa_\infty (A)\, \rho_n \epsilon,$$

wobei γ ein Polynom in n von geringem Grad und nicht groß ist und ϵ die Maschinengenauigkeit ist. Insgesamt folgt wegen $\kappa_\infty (A) < 1000$ und $\epsilon < 10^{-7}$ hieraus, dass die Gauß Elimination ohne Pivotisierung im Fall der Leitanwendung numerisch stabil ist.

5.2 Literaturübersicht

Die Dünnbesetztheit eines linearen Gleichungssystems führt auch bei einer Gauß Elimination, die ein Sparse-Format verwendet, keineswegs unmittelbar zu einem geringerem Rechenaufwand bei dessen Lösung. Dies lässt sich am bekannten Beispiel von Pfeilmatrizen veranschaulichen:

$$A_l = \begin{pmatrix} \blacksquare & \blacksquare & \blacksquare & \blacksquare & \blacksquare & \blacksquare \\ \blacksquare & \blacksquare & \cdot & \cdot & \cdot & \cdot \\ \blacksquare & \cdot & \blacksquare & \cdot & \cdot & \cdot \\ \blacksquare & \cdot & \cdot & \blacksquare & \cdot & \cdot \\ \blacksquare & \cdot & \cdot & \cdot & \blacksquare & \cdot \\ \blacksquare & \cdot & \cdot & \cdot & \cdot & \blacksquare \end{pmatrix} \quad A_r = \begin{pmatrix} \blacksquare & \cdot & \cdot & \cdot & \cdot & \blacksquare \\ \cdot & \blacksquare & \cdot & \cdot & \cdot & \blacksquare \\ \cdot & \cdot & \blacksquare & \cdot & \cdot & \blacksquare \\ \cdot & \cdot & \cdot & \blacksquare & \cdot & \blacksquare \\ \cdot & \cdot & \cdot & \cdot & \blacksquare & \blacksquare \\ \blacksquare & \blacksquare & \blacksquare & \blacksquare & \blacksquare & \blacksquare \end{pmatrix}. \tag{5.2}$$

Beide Matrizen sind dünn besetzt. Im Rahmen einer Gauß Elimination ohne Pivotisierung führt A_l zu einer voll besetzten Matrix, da die anfänglichen Nulleinträgen im Rahmen der Elimination sukzessive aufgefüllt werden, während bei A_r kein derartiges Auffüllen auftritt, das als *Fill-in* bezeichnet wird. Sofern bei der Gauß Elimination ohne Pivotisierung ein Sparse-Format für die Speicherung der Matrix verwendet und somit die Struktur des Gleichungssystems berücksichtigt wird, führt die Lösung eines linearen Gleichungssystems mit der Matrix A_r zu einem Rechenaufwand, der signifikant geringer ist als der Rechenaufwand für die Lösung eines linearen Gleichungssystems mit der Matrix A_l. Zu beachten ist, dass das Gleichungssystem $A_l x = b$ durch eine Permutation, die x_1 und x_6 vertauscht, in ein Gleichungssystem mit einer Matrixstruktur wie A_r überführt werden kann.

Dies ist ein einfaches Beispiel für den Mehrwert, der durch eine geeignete Umordnung eines Gleichungssystems erzielt werden kann. Im Folgenden werden Umordnungen und andere Möglichkeiten dargestellt, um auch für allgemeine dünn besetzte Matrizen bei der Rechenzeit einen Vorteil aus der Struktur zu ziehen. Im Gebiet direkter Lösungsmethoden für dünn besetzte Matrizen bestehen etablierte heuristische Ansätze.

Zunächst werden verschiedene Sparse-Formate knapp vorgestellt. Im Anschluss sind die folgenden verschiedenen Klassen von Ansätzen erläutert:

1. Permutationen, die die Matrix A des Gleichungssystems auf Blockdreiecksform zu transformieren
2. Lokale Methoden zur Minimierung des Fill-in
3. Globale Methoden

Auf die dritte Klasse wird detaillierter eingegangen, da sie besonders relevant für den weiteren Abschnitt ist. Als fundamentale Quellen für die Fragestellungen dieses Kapitels dienen [39, 28, 96, 53].

5.2.1 Speicherung dünn besetzter Matrizen

Um die Dünnbesetztheit einer Matrix in einen Rechenzeitvorteil bei der Lösung eines zugehörigen Gleichungssystems umzuwandeln, ist die Möglichkeit wichtig, die Dünnbesetztheit adäquat zu erfassen. Aus diesem Grund werden in diesem Abschnitt drei Beispiele von Formaten vorgestellt, die zur Speicherung dünn besetzter Matrizen eingesetzt werden. Ihr Einsatzzweck liegt in vorliegender Arbeit nicht vorrangig in der Reduktion des Speicherbedarfs, sondern in der Unterstützung bei der Rechenzeitreduktion.

Wie in [39] erwähnt, ist bei der Speicherung dünn besetzter Matrizen zu unterscheiden, ob die Speicherstruktur statisch, also festgelegt oder ob sie dynamisch und somit veränderlich gewählt wird. Entsprechend der Speicherverwaltung auf einem Fahrzeug-Steuergerät wie in Abschnitt 3.1.2 beschrieben, sind statische Strukturen passend. Dynamische Speicherformate bedürfen einer besonderen Sorgfalt, um vorab ihren maximalen Speicherbedarf abzuschätzen und Laufzeitfehler zu vermeiden.

Zentrale Anforderung an die verwendeten Sparse-Formate ist, dass sie zu einer Reduktion der Rechenzeit der Lösung eines linearen Gleichungssystems mittels einer Elimination beitragen. Hierzu ist der effiziente Zugriff auf Spaltenvektoren sowie auf Zeilenvektoren der Matrix hilfreich. Aus diesem Grund werden im Folgenden Formate untersucht, die für den Einsatz bei der Gleichungslösung konzipiert sind. Hierzu werden das CRS Format sowie das FEM Format betrachtet. Darüber hinaus ist das sehr intuitive Koordinatenformat kurz beschrieben. Die verschiedenen Formate bieten jeweils Vor- und Nachteile. Ein Urteil, welches Format das Beste ist, hängt von der Struktur der Gleichung ab. Eine derartige Untersuchung ist nicht im Fokus dieser Arbeit. Allerdings wird eine einfache Möglichkeit gezeigt, die Gegebenheiten bei einer Echtzeitsimulation eines mathematischen Modells auf einem Fahrzeug-Steuergerät beim CRS Format gewinnbringend zu nutzen.

Mit *Eintrag* wird im Folgenden jeder nicht durchgehend verschwindende Matrixeintrag bezeichnet.

Koordinatenformat Bei diesem Format wird jeder Eintrag durch ein Tripel $(a_{i,j}, i, j)$ repräsentiert, wobei $a_{i,j}$ den Wert des Eintrags, i den Zeilenindex und j den Spaltenindex angibt. Dementsprechend ist für i und j eine unsignierte Ganzzahlvariable zur Speicherung zweckmäßig, während für $a_{i,j}$ eine Gleitkommavariable sinnvoll ist.

Um einen Spaltenvektor $(a_{i,j_0})_i$ zu ermitteln, ist bei diesem Format eine Indexsuche in den Tripeln nötig. Entsprechendes gilt für Zeilenvektoren. Bei zeilenweiser Sortierung der Tripel wird der Aufwand zur Ermittlung von Zeilenvektoren gering gehalten.

CRS Format Beim *Compressed Row Storage* (CRS) Format, das auch *Compressed Sparse Row* (CSR) Format genannt wird, wird jede Zeile einer Matrix in komprimierter Form gespeichert. Durch das Mitführen von Zeigern auf die führenden Einträge der Zeilen können diese unabhängig voneinander gespeichert werden. Jeder Eintrag $a_{i,j}$ der Matrix wird durch ein Tripel $(a_{i,j}, j, n_i)$ repräsentiert. Hierbei gibt $a_{i,j}$ den Wert des Eintrags an, j den Spaltenindex und n_i einen Zeiger auf das erste Tripel der Zeile i.

Die Matrix

$$A = \begin{pmatrix} 1 & 0 & 0 & 0 & 2 \\ 3 & 4 & 0 & 0 & 5 \\ 6 & 0 & 7 & 0 & 8 \\ 9 & 0 & 0 & 10 & 11 \\ 12 & 0 & 0 & 0 & 13 \end{pmatrix}$$

lässt sich im CRS Format folgendermaßen darstellen, wobei hierbei die Zeilen der Matrix unmittelbar nacheinander gespeichert werden und für jede Zeile der Speicher von fünf Tripeln vorgesehen wird, um beispielsweise beim Auftreten von Fill-in im Rahmen der Lösung eines entsprechenden Gleichungssystems für jede Zeile genügend Speicher zu haben, um die auftretenden Einträge darin zu erfassen:

$$\begin{array}{llllllll} ((1,1,1), & (2,5,1), & (3,1,6), & (4,2,6), & (5,5,6), & (6,1,11), & (7,3,11), \\ (8,5,11), & (9,1,16), & (10,4,16), & (11,5,16), & (12,1,21), & (13,5,21)). & \end{array}$$

Der Zugriff auf die Einträge einer Zeile der Matrix ist im CRS Format effizient möglich. Die Ermittlung der Einträge einer gegebenen Spalte erfordert eine Indexsuche in den Spaltenindizes der Tripel.

FEM Format Bei der in [39] als FEM Format oder auch Clique Format bezeichneten Weise, eine dünn besetzte Matrix zu speichern, wird eine Matrix A als Summe von kleinen, nicht dünn besetzten Teilmatrizen $A^{(k)}$ dargestellt:

$$A = \sum_k A^{(k)}.$$

Beispielsweise kann die Matrix

$$A = \begin{pmatrix} 1 & 0 & 1 & 0 & 1 \\ 0 & 2 & 0 & 2 & 0 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 2 & 0 & 2 & 3 \\ 1 & 0 & 1 & 3 & 1 \end{pmatrix}$$

durch

$$A^{(1,3,5)} = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}, \quad A^{(2,4)} = \begin{pmatrix} 2 & 2 \\ 2 & 2 \end{pmatrix}, \quad A^{(4,5)} = \begin{pmatrix} 0 & 3 \\ 3 & 0 \end{pmatrix}$$

repräsentiert werden. Für jede Matrix gibt es eine entsprechende Überdeckung durch kleine nicht dünn besetzte Teilmatrizen, da die einzelnen Einträge als voll besetzte Teilmatrizen gewählt werden können.

Sofern eine Matrix eine entsprechende Struktur besitzt, kann das FEM Format eine gute Möglichkeit darstellen, die Struktur effizient zu repräsentieren.

5.2.2 Transformation auf Blockdreiecksform

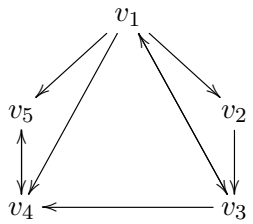
In diesem Abschnitt wird der Ansatz betrachtet, die Matrix A, die im Rahmen der Lösung einer gewöhnlichen Differenzialgleichung auftritt, durch Permutationsmatrizen auf Blockdreiecksform zu transformieren.

Hierbei sind Graphen ein verbreitetes und nützliches Hilfsmittel. Der Adjazenzgraph G_A der quadratischen Matrix A ist folgendermaßen definiert: Ist A eine $n \times n$ Matrix, so ist G_A ein gerichteter Graph mit n Knoten. Ist $a_{ij} \neq 0$, so besitzt G_A die Kante (i, j), andernfalls ist (i, j) keine Kante von G_A. In vorliegendem Kapitel wird die Annahme zu Grunde gelegt, dass stets sämtliche Diagonaleinträge einer Matrix ungleich Null sind. Diese Annahme ist darin begründet, dass die Matrizen, die im Rahmen eines linearen Gleichungssystems bei einem linear-impliziten Euler-Verfahren auftreten, die Form $I - \Delta t J_f$ haben und somit, falls J_f nicht von Δt abhängt, bei zufälliger Wahl von Δt fast immer ungleich Null sind. Beim Adjazenzgraph G_A werden in vorliegender Arbeit die zu den Diagonaleinträgen korrespondierenden Schleifen nicht dargestellt.

Ist beispielsweise

$$A = \begin{pmatrix} 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 1 \end{pmatrix}, \tag{5.3}$$

so ist der zugehörige Adjazenzgraph



Für die Bestimmung einer Permutation, die eine Matrix auf Blockdreiecksform zu transformiert, hat der Begriff der starken Zusammenhangskomponente eines gerichteten Graphen eine wesentliche Bedeutung. Aus diesem Grund werden im Folgenden verschiedene relevante Begriffe aus der Graphentheorie definiert:

Ein gerichteter Graph $G = (V, E)$ heißt stark zusammenhängend, falls es für jedes Paar von Knoten (v_i, v_j) mit $v_i, v_j \in V$ einen gerichteten Weg von v_i nach v_j gibt. Ein gerichteter Graph heißt schwach zusammenhängend, wenn der zugehörige ungerichtete Graph $\tilde{G} = \left(V, \tilde{E}\right)$ zusammenhängend ist, es also für jedes Knotenpaar (v_i, v_j) mit $v_i, v_j \in V$ einen Weg von v_i nach v_j gibt. Ein Teilgraph U eines gerichteten Graphen G heißt starke Zusammenhangskomponente, falls U stark zusammenhängend ist und nicht zu einem größeren Teilgraphen von G erweitert werden kann, der ebenfalls stark zusammenhängend ist.

Bei G_A wird beispielsweise von den Knoten $\{v_1, v_2, v_3\}$ eine starke Zusammenhangskomponente induziert.

Eine Matrix, die eine Blockdreiecksform besitzt, die aus mehr als einem Diagonalblock besteht, wird reduzibel genannt. Die Reduzibilität einer Matrix A ist äquivalent dazu, dass der Adjazenzgraph G_A nicht stark zusammenhängend ist. Zur Bestimmung einer Blockdiagonalform kann folgendermaßen vorgegangen werden:

Zunächst werden die schwachen Zusammenhangskomponenten des Adjazenzgraphen einer Matrix ermittelt. Dies liefert eine Permutation auf Blockdiagonalform, wenn die Diagonaleinträge entsprechend der Zusammenhangskomponenten gruppiert werden. Anschließend werden in jeder Zusammenhangskomponente die starken Zusammenhangskomponenten bestimmt. Diese entsprechen den Diagonalblöcken einer Blockdreiecksform innerhalb der Diagonalblöcke der Blockdiagonalform. Zur Bestimmung von starken Zusammenhangskomponenten ist der Algorithmus von Tarjan verbreitet (vgl. [113]).

Der Vorteil einer Blockdreiecksform besteht darin, dass lediglich die Diagonalblöcke invertiert werden anstelle der ganzen Matrix. Da die Gleichungslösung mittels einer Elimination die Komplexität $\mathcal{O}\left(n^3\right)$ besitzt, ist durch das Zurückführen der Lösung auf kleinere Gleichungssysteme wie bei der Blockdreiecksform eine signifikante Reduktion des nötigen Rechenaufwands zu erwarten. Beispielsweise kann die Lösung der Gleichung $Ax = b$ mit der Matrix A wie in Gleichung 5.3 mit der Partitionierung

$$A = \begin{pmatrix} A_{11} & 0 \\ A_{21} & A_{22} \end{pmatrix} \quad x = \begin{pmatrix} u \\ v \end{pmatrix} \quad b = \begin{pmatrix} f \\ g \end{pmatrix}$$

zurückgeführt werden auf die Lösung von

$$A_{11}u = f \quad A_{22}v = g - A_{21}u \ .$$

In [80] werden Transformationen auf Blockdreiecksform im Kontext von Simulationen mathematischer Modelle eingesetzt. Dort werden sie im Rahmen von der Lösung nichtlinearer Gleichungssysteme untersucht, was zu einem leicht variierten Vorgehen führt, wie beispielsweise die Betrachtung eines Inzidenzgraphen des Gleichungssystems anstelle des Adjazenzgraphen der Matrix verwendet wird.

5.2.3 Lokale Methoden zur Minimierung des Fill-in

Das Problem der Minimierung der Anzahl der Fülleinträge beschreibt Aufgaben folgender Art: Bei gegebener Matrix A sind eine Zeilen- sowie Spaltenpermutationsmatrix P bzw. Q derart zu bestimmen, dass die Anzahl der strukturell verschwindenden Einträge in einer Zerlegung von PAQ maximiert oder aber der Aufwand zur Ermittlung der Zerlegung minimiert wird. (vgl. [28])

Bereits vereinfachte Versionen dieses Problems sind NP-vollständig (vgl. [92]). Daher liegen keine praxistauglichen Algorithmen zur exakten Lösung dieses Problems vor. Methoden zur Minimierung des Fill-in sind aus diesem Grund heuristisch und in der Regel Greedy Ansätze.

Im Folgenden werden zwei wichtige Ansätze vorgestellt: der Reverse Cuthill McKee Algorithmus und der Approximate Minimum Degree Algorithmus. Beide Ansätze sind für struktursymmetrische Matrizen konzipiert. Indem der gerichtete Adjazenzgraph einer unsymmetrischen Matrix durch den zugehörigen ungerichteten Graphen ersetzt wird, was einem Auffüllen der Strukturmatrix bis zur Symmetrie entspricht, lassen sich diese Ansätze auch für unsymmetrische Matrizen einsetzen.

Reverse Cuthill McKee Ein sehr verbreiteter Algorithmus im Kontext von symmetrischen Strukturmatrizen ist der von Cuthill und McKee [27]. Seine zu Grunde liegende Intention, ist

eine Bandstruktur mit möglichst geringer Bandbreite herzustellen. Das Prinzip des Verfahrens ist folgendermaßen:

Ausgehend von einem Startknoten m_1 des ungerichteten Adjazenzgraphen einer Matrix sei $M_2 = \{m_j | j = 1, \ldots, r\}$ eine geordnete Menge der r Nachbarknoten von m_1. Nun werde sukzessive für die verschiedenen Knoten $m_j \in M_2$ die Knotenmenge $M_{2,j}$ definiert als die Menge der jeweiligen Nachbarknoten des Knoten m_j, die noch nicht in einer der vorangehend definierten Mengen enthalten sind. Die Verkettung der geordneten Menge $M_3 = M_{2,1} \circ \cdots M_{2,r}$ definiert die nächste Menge einer Hierarchie. Die Mengen M_i werden als *Level* Sets bezeichnet. Durch analoges Fortsetzen dieser Vorgehensweise ergibt sich eine baumartige Struktur für den Ausgangsgraphen. Diese liefert eine Permutation der Variablen, die zu einer Block-Tridiagonalform für die untersuchte Matrix führt.

Zur Reduktion von Fülleinträgen ist der ähnliche Reverse Cuthill McKee Algorithmus günstig, dessen Unterschied gegenüber dem Originalalgorithmus darin besteht, die Reihenfolge der Variablen gegenüber der sich beim Cuthill McKee Algorithmus ergebenden Reihenfolge der Variablen umzukehren. Dies hat folgende heuristische Begründung (vgl. [39]): Während beim Originalalgorithmus die Kanten zu Nachbarknoten spaltenweise Einträge in der Block-Tridiagonalform nach sich ziehen, treten die entsprechenden Einträge bei der umgekehrten Reihenfolge in der Block-Tridiagonalform gerade zeilenweise auf. Sofern die Blöcke nicht voll besetzt sind, tritt nun derselbe Effekt auf wie bei Pfeilmatrizen (vgl. Gleichung 5.2), so dass die Struktur, die sich durch den Reverse Cuthill McKee Algorithmus ergibt, typischerweise zu einem geringeren Fill-in führt.

Die Wahl der Startknoten hat einen nennenswerten Einfluss auf die resultierende Bandbreite. Um eine günstige Bandstruktur zu erhalten, ist der Gibbs Poole Stockmeyer Algorithmus zur Bestimmung geeigneter Startknoten nützlich (vgl. [52]) .

Approximate Minimum Degree Da eine globale Minimierung des Fill-in im Rahmen einer Elimination nicht bestimmbar ist, werden lokale Heuristiken hierzu betrachtet. Bei *Minimum Degree* Algorithmen wird derjenige Knoten des ungerichteten Adjazenzgraphen gewählt, der den geringsten Grad, also die geringste Anzahl an Kanten hat.

Eine wichtige Rolle bei dieser Klasse von Algorithmen nimmt das Markowitz-Kriterium ein: Sei $A^{(k)}$ die aktive Teilmatrix im Rahmen einer Elimination im k -ten Eliminationsschritt sowie $r_i^{(k)}$ die Anzahl an Einträgen in der i-ten Zeile von $A^{(k)}$ und $s_j^{(k)}$ entsprechend die Anzahl an Einträgen in der j-ten Spalte von $A^{(k)}$. Dann ist derjenige Eintrag a_{ij} als Pivoteintrag zu wählen, für den

$$\left(r_i^{(k)} - 1\right)\left(s_j^{(k)} - 1\right)$$

minimiert wird. Diese Heuristik zielt darauf ab, in jedem Eliminationsschritt möglichst wenig Einträge zu verändern, um hierdurch möglichst auch wenig Fülleinträge zu erzeugen.

Das Vorgehen ist beim Approximate Minimum Degree Algorithmus folgendes: Zunächst wird strukturell die Elimination eines Systems vollständig durchgeführt. Aus dieser ergibt sich eine geeignete Permutation, die zu einem geringen Fill-in führt. Die Reihenfolge durch diese Permutation wird dann im Rahmen der tatsächlichen Elimination genutzt.

Hierbei hat die Wahl der Startknoten sowie der Tie-breaking Strategie, also der Entscheidungsfindung bei Einträgen, die hinsichtlich des Kriteriums zu gleichen Werten führen, einen erheblichen Einfluss auf die Qualität der Resultate (vgl. [39]).

Eine Übersicht über Entwicklungen und Erweiterungen hierzu ist in [51] zu finden.

5.2.4 Globale Methoden

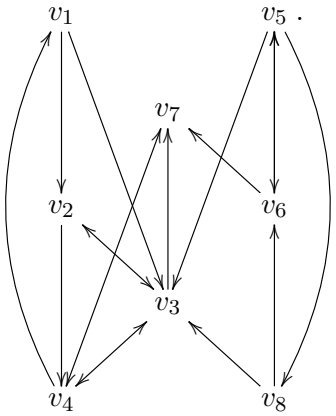
Ein positiver Aspekt von globalen Methoden ist, dass von vornherein die Struktur der Ausgangsmatrix ganzheitlich berücksichtigt wird (vgl. [39]). Die Ansätze sind Heuristiken, die durch

Gebietszerlegungsmethoden motiviert sind. Analog zu einer Gebietszerlegung bei einer partiellen Differentialgleichung wird der Adjazenzgraph der Matrix untergliedert in mehrere Teile. Die Knoten, die die Grenze zwischen den Teilen bilden, ergeben den Separator. Zur Bestimmung eines Separators können wie beim Reverse Cuthill McKee Ansatz ausgehend von einem Startknoten die Level Sets des Adjazenzgraphen ermittelt werden. Ein Level Set, dessen Entfernung aus dem Graphen zu zwei ähnlich großen Zusammenhangskomponenten führt, ist eine günstige Wahl als Separator. Eine derartige Unterteilung wird als *One-way Dissection* bezeichnet. Bei *Nested Dissection* wird der Ansatz mehrfach angewandt. Die Matrixform, die durch den Einsatz dieser Methoden erzielt wird, ist eine doppelte berandete Blockdiagonalform oder eine doppelt berandete Blockdreiecksform abhängig davon, ob die Teilgraphen schwache oder starke Zusammenhangskomponenten des Restgraphen bilden.

Dies wird an einem einfachen Beispiel illustriert. Sei

$$A = \begin{pmatrix} 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \end{pmatrix}.$$

Dann ist der zugehörige Adjazenzgraph



Wird als Separator $S = \{v_3, v_7\}$ gewählt, so ist der Restgraph nach Entfernen des Separators samt der von ihm ausgehenden Kanten nicht zusammenhängend. Hierdurch ergibt sich also eine doppelt berandete Blockdiagonalform, wenn die Reihenfolge der Diagonaleinträge von A entsprechend der Zusammenhangskomponenten des Restgraphen gewählt wird und die dem Separator entsprechenden Einträge zuletzt auf der Diagonale auftreten:

$$\tilde{A} = \left(\begin{array}{cccccc|cc} 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ \hline 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 \end{array}\right) = \begin{pmatrix} A_{11} & 0 & A_{13} \\ 0 & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \end{pmatrix}.$$

Die horizontale und vertikale Linie in der obigen Darstellung von $\tilde{A}$ sind zur besseren Übersicht über die Struktur von $\tilde{A}$ eingefügt. Da das Resultat von der Wahl der Startknoten erheblich

beeinflusst wird, ist die vorangehende Bestimmung geeigneter Startknoten, sogenannter pseudoperipherer Knoten, sinnvoll. Diese weisen Eigenschaften auf, wie sie von Knoten vom Rand eines Graphen erwartet werden (vgl. [39]). Wie bereits erwähnt ist der Gibbs Poole Stockmayer Algorithmus hierzu ein bewährter Ansatz (vgl. [52]).

Eine große Ähnlichkeit mit dieser Vorgehensweise besitzt das sogenannte Tearing (vgl. [39]). Hierbei wird der Zusammenhang eines Graph dadurch aufgehoben, dass Knoten samt ihrer Kanten oder lediglich Kanten aus dem Graphen entfernt werden. Ersteres wird als *Branch Tearing*, letzteres als *Node Tearing* bezeichnet. Die resultierenden Zusammenhangskomponenten werden als Unterteilung des Graphen verwendet.

Der Einsatz von Tearing bei Simulationen mathematischer Modelle ist in [80, 23, 41, 22] beschrieben. Dort wird es bei der Lösung nichtlinearer Gleichungssysteme eingesetzt und dient insbesondere dem Zweck, das Lösen eines solchen Gleichungssystems auf das Lösen kleinerer, idealerweise skalarer nichtlinearer Gleichungen zurück zu führen.

Eine wesentliche Rolle bei diesen Methoden kommt dem Schur-Komplement zu. Daher sind wichtige Eigenschaften des Schur-Komplements im Folgenden zusammengestellt. Anschließend wird das Vorgehen bei einer Schur-Komplement basierten Lösung linearer Gleichungssysteme beschrieben.

Eigenschaften des Schur-Komplements

Sei eine Partitionierung der Matrix A gegeben durch

$$A = \begin{pmatrix} B & E \\ F & C \end{pmatrix}.$$

Ist B nicht-singulär, so ist gemäß [96] das Schur-Komplement $S_{A/B}$ beziehungsweise $S\,(A/B)$ von B in A, das, falls die Matrix A aus dem Kontext klar ist, auch als S_B geschrieben werde, definiert als

$$S_B = C - FB^{-1}E.$$

Entsprechend ist für ein nicht-singuläres C das Schur-Komplement S_C definiert als

$$S_C = B - EC^{-1}F.$$

Das Schur-Komplement besitzt eine Reihe interessanter Eigenschaften. Beispielsweise besteht folgende Block-LU-Zerlegung der Matrix A (vgl. [96]):

$$\begin{pmatrix} B & E \\ F & C \end{pmatrix} = \begin{pmatrix} I & O \\ FB^{-1} & I \end{pmatrix} \begin{pmatrix} B & E \\ O & S_B \end{pmatrix}.$$

Wichtige Eigenschaften des Schur-Komplements sind in [43] übersichtlich zusammengestellt. Hierin finden sich die folgenden drei Eigenschaften. Zum einen ist die Determinante von A multiplikativ bezüglich des Schur-Komplements:

$$\det A = \det B \det S_B. \tag{5.4}$$

Außerdem gilt die Additivität des Ranges:

$$\operatorname{rang} A = \operatorname{rang} B + \operatorname{rang} S_B.$$

Für die Partitionierung

$$A = \begin{pmatrix} B_{11} & B_{12} & E_1 \\ B_{21} & B_{22} & E_2 \\ F_1 & F_2 & C \end{pmatrix}$$

und

$$B = \begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix}$$

lässt sich das Schur-Komplement in eingebetteter Weise wählen, also beispielsweise das Schur-Komplement $S_{B/B_{11}}$. In diesem Fall gilt folgende Quotienteneigenschaft:

$$S_{A/B} = S\left(S_{A/B_{11}}/S_{B/B_{11}}\right).$$

Für die Konditionszahl $\kappa(S)$ gilt gemäß Lemma 3.4 aus [31] die Abschätzung

$$\kappa(S) \leq \rho_n \kappa(A),$$

wobei

$$\rho_n = \frac{\max_{i,j,k}\left|a_{ij}^{(k)}\right|}{\max_{i,j}|a_{ij}|}$$

der Wachstumsfaktor bei einer Gauß Elimination ohne Pivotisierung sei und $a_{ij}^{(k)}$ die Einträge der Matrix im k-ten Eliminationsschritt seien. Sofern die Matrix A gut konditioniert ist und die Gauß Elimination ohne Pivotisierung stabil ist, also der Wachstumsfaktor ρ_n nicht groß ist, ist folglich auch das Schur-Komplement S gut konditioniert.

Block-Gauß Elimination

Im Folgenden wird eine Möglichkeit zur Lösung des Gleichungssystems

$$Ax = b \Leftrightarrow \begin{pmatrix} B & E \\ F & C \end{pmatrix}\begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} f \\ g \end{pmatrix}$$

beschrieben. Das Gleichungssystem ist äquivalent zu $Bu+Ev = f \wedge Fu+Cv = g$. Durch Auflösen der ersten Gleichung nach u und Einsetzen in die zweite Gleichung ergibt sich das sogenannte *reduzierte System* $\left(C - FB^{-1}E\right)v = g - FB^{-1}f$. Mit $S = C - FB^{-1}E$ und $\bar{g} = g - FB^{-1}f$ stellt sich das reduzierte System dar als $Sv = \bar{g}$. Die Lösung des Ausgangssystems erfolgt bei Verwendung des Schur-Komplements nun dadurch, dass zunächst das reduzierte System gebildet und gelöst wird und mit dem erhaltenen Wert für v anschließend $Bu = f - Eg$ gelöst wird (vgl. [96]).

Ein geschicktes Vorgehen erlaubt es, den Aufwand der Lösung des Rücksubstitutionssystems zu vermeiden (vgl. [96]): Im Rahmen der Bildung des Schur-Komplements werden die Matrix $\bar{E} := B^{-1}E$ sowie der Vektor $\bar{f} := B^{-1}f$ gespeichert. Mit diesen ergibt sich dann $u = \bar{f} - \bar{E}v$.

Der resultierende Algorithmus ist eine Block-Gauß Elimination. Sie untergliedert sich also in vier Schritte:

1. Lösen von $B\bar{E} = E$ und $B\bar{f} = f$.
2. Berechnen des Schur-Komplements $S = C - F\bar{E}$ und von $\bar{g} = g - F\bar{f}$.
3. Lösen des reduzierten Systems $Sv = \bar{g}$.
4. Berechnen von $u = \bar{f} - \bar{E}v$.

Im Folgenden wird die numerische Stabilität der Block-Gauß Elimination betrachtet:

Aus Gleichung 5.4 folgt, dass bei einer nicht-singulären Matrix A auch die Teilmatrix B sowie das Schur-Komplement S nicht-singulär sind. Sei A gut konditioniert und der Wachstumsfaktor ρ_n im Rahmen der Lösung von $Ax = b$ mit einer Gauß Elimination nicht groß, also die Gauß Elimination numerisch stabil. Wie auf Seite 86 beschrieben, ist dann auch das Schur-Komplement S auf Grund von $\kappa(S) \leq \rho_n \kappa(A)$ gut konditioniert. Gemäß der Heuristik von Seite 77 ist in diesem Fall die Lösung des reduzierten Systems $Sv = \bar{g}$ mit einer Gauß Elimination mit einer Spaltenpivotisierung numerisch stabil.

In Abschnitt 5.1 wird erläutert, dass für die Gleichungen aus der Referenztrajektorie der Leitanwendung eine Gauß Elimination ohne Pivotisierung numerisch stabil ist und die Wachstumsfaktoren nicht groß sind. Folglich ist die Lösung der Gleichungen $B\bar{E} = E$ und $B\bar{f} = f$ mit einer Gauß Elimination mit Spaltenpivotisierung numerisch stabil. Für das reduzierte System erscheint es in einer solchen Konstellation dennoch im Allgemeinen sinnvoll, eine Gauß Elimination mit Pivotisierung zur Lösung einzusetzen.

5.3 Rechenzeitreduktion für dünn besetzte lineare Gleichungen bei der Leitanwendung

Im Folgenden wird untersucht, wie die Rechenzeit zur Lösung eines dünn besetzten linearen Gleichungssystems reduziert werden kann, das im Rahmen eines linear-impliziten Euler-Verfahrens bei der Simulation eines steifen Modells auf einem Fahrzeug-Steuergerät auftritt. Hierbei wird der Ansatz verfolgt, dass die Struktur des Gleichungssystems vor der Implementierung analysiert und das Lösungsverfahren zugeschnitten wird auf die vorliegende Struktur. Im Fokus steht dabei insbesondere ein Schur-Komplement basiertes Lösen der Gleichungssysteme, da es als besonders geeignet erachtet wird für die Struktur der Gleichungssysteme, wie sie bei der Leitanwendung sowie bei Modellen der betrachteten Problemklasse auftreten (vgl. 2.1.7).

Darüber hinaus werden der Einsatz von lokalen Methoden zur Reduzierung des Fill-in sowie Möglichkeiten zur Rechenzeitreduktion im Rahmen des Sparse-Formats betrachtet. Der Ansatz einer Transformation auf Blockdreiecksform wird nicht eigenständig eingesetzt, da bei der Leitanwendung keine Transformation auf eine nicht-triviale Blockdreiecksform möglich ist. Im Rahmen der Lösung des Gleichungssystems mittels einer Block-Gauß Elimination bildet die Transformation auf Blockdreiecksform allerdings einen wesentlichen Bestandteil.

5.3.1 Motivation der Anwendung einer Block-Gauß Elimination

In diesem Abschnitt wird die der Einsatz einer Block-Gauß Elimination zur Lösung von dünn besetzten linearen Gleichungssystemen von mathematischen Modellen auf Fahrzeug-Steuergeräten motiviert.

Einsatz bei aus Submodellen bestehenden mathematischen Modellen Bei Differentialgleichungen bestehen verschiedene Faktoren, die den Einsatz von Gebietszerlegungsmethoden motivieren. In [96] werden hierfür unter anderem folgende beide Aspekte angegeben:

- Eine Gebietszerlegung begünstigt eine Parallelisierung der Rechnung
- Durch die Unterteilung in Teilgebiete mit vorteilhafter Geometrie kann die Problemstellung vereinfacht oder der Einsatz von spezialisierten Verfahren ermöglicht werden.

Grundsätzlich besteht eine große Ähnlichkeit zwischen Abhängigkeiten zwischen räumlich benachbarten Teilgebieten und den Abhängigkeiten zwischen Teilmodellen, die durch Wirkzusammenhänge im Sinne von direkten physikalischen Abhängigkeiten stehen. Aus diesem Grund bilden die Konzepte der Gebietszerlegung eine interessante Möglichkeit zur Anwendung bei mathematischen Modellen, die sich in natürlicher Weise in Submodelle untergliedern lassen. Wie in Abschnitt 2.1.7 beschrieben, sind die Modelle der in vorliegender Arbeit untersuchten Problemklasse in natürlicher Weise in Bestandteile unterteilbar, die nur in geringerem Maß Querabhängigkeiten aufweisen. Daher ist der Einsatz von Zerlegungen analog zu Gebietszerlegungen bei diesen Modellen naheliegend: Sie ermöglichen, dass die Gleichungsstruktur vereinfacht oder die Lösung des Gleichungssystems auf die Lösung kleinerer Systeme in gewinnbringender Weise zurückgeführt wird und begünstigen eine Parallelisierung der Rechnung.

Einsatz bei allgemeinen dünn besetzten linearen Gleichungssystemen

Sei

$$Ax = b \Leftrightarrow \begin{pmatrix} B & E \\ F & C \end{pmatrix} \begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} f \\ g \end{pmatrix}.$$

Der Mehrwert der auf Seite 86 beschriebenen Lösung linearer Gleichungssysteme mittels einer Block-Gauß Elimination tritt unter anderem auf, wenn die Invertierung von B günstig und C von geringer Dimension ist. Beispielsweise ist eine solche Wahl des Schur-Komplements S_B günstig, bei der die Matrix B Blockdreiecksform oder Blockdiagonalform hat. Beide können zu einer Strukturausnützung eines linearen Gleichungssystems dienlich sein. Hat B Blockdiagonalform hat, so zerfällt das Lösen von $B\bar{E} = E$ und $B\bar{f} = f$ in unabhängige Teilgleichungen, die parallel gelöst werden können.

Wie in Abschnitt 3.2.5 beschrieben, ist die grundsätzliche Eignung eines Algorithmus zur Parallelisierung im Hinblick auf die Tauglichkeit für zukünftige Steuergeräte eine kollateraler Aspekt für die Eignung der betrachteten Algorithmen. In dieser Hinsicht ist die Block-Gauß Elimination gut geeignet.

Insgesamt ist die Block-Gauß Elimination also sowohl für die Strukturausnützung als auch für die Parallelisierung geeignet.

5.3.2 Einsatz der Schur-Komplements basierten Lösung bei der Leitanwendung

Im Folgenden wird das Schur-Komplement basierte Lösen für die Leitanwendung eingesetzt. Hierzu wird die Struktur von Gleichung 5.1 betrachtet. Der Adjazenzgraph $G_A = (V_A, E_A)$ der betrachteten Matrix ist in Abbildung 5.1 dargestellt.

Zur weiteren Vorgehensweise werden verschiedene Begriffe aus der Graphentheorie verwendet, die nachfolgend definiert sind. Weitergehende Inhalte der Graphentheorie sind beispielsweise in [115, 37, 19] beschrieben.

Definition. Ein gerichteter Graph $G = (V, E)$ heißt k-fach schwach respektive stark zusammenhängend, wenn für jede Knotenmenge $\tilde{V} \subset V$ mit $\left|\tilde{V}\right| < k$ der durch $V \backslash \tilde{V}$ induzierte Teilgraph von G schwach respektive stark zusammenhängend ist. Die minimale Knotenzahl, zu der es eine Knotenmenge $\tilde{V} \subset V$ gibt, durch die $V \backslash \tilde{V}$ einen nicht schwach zusammenhängenden respektive nicht stark zusammenhängenden Teilgraph induziert, wird als schwache Zusammenhangszahl $\kappa(G)$ respektive starke Zusammenhangszahl $\sigma(G)$ bezeichnet. Eine Knotenmenge $S \subset V$ werde als schwacher respektive starker Separator in G bezeichnet, wenn durch $V \backslash S$ ein nicht schwach zusammenhängender respektive ein nicht stark zusammenhängender Teilgraph induziert wird.

Die Zusammenhangszahl $\kappa(G)$ und die starke Zusammenhangszahl $\sigma(G)$ des Graphen G sind

$$\kappa(G) = \sigma(G) = 1.$$

Dies bedeutet hinsichtlich der zugehörigen Matrixstruktur, dass eine Zeile und Spalte eine Permutation auf eine Blockdiagonalform und auf eine Blockdreiecksform verhindern. Der Knoten T_{B_1} bildet einen Separator $S_1 = \{T_{B_1}\}$, durch den ein nicht zusammenhängender Teilgraph $G[V \backslash S]$ mit zwei Zusammenhangskomponenten $G_1 = (V_1, E_1)$ und $G_2 = (V_2, E_2)$ induziert wird. Hierbei ist

$$\begin{aligned} V_1 &= \{w_{11}, w_{21}, w_{31}\}, \\ V_2 &= \{w_{12}, w_{22}, w_{32}, T_{B_2}, w_{13}, w_{23}, w_{33}, T_{B_3}, w_{14}, w_{24}, w_{34}, T_{B_4}, w_{15}, w_{25}, w_{35}, T_{B_5}\}. \end{aligned}$$

Insbesondere ist $|V_1| = 3$ und $|V_2| = 16$. Hinsichtlich der zugehörigen Matrixstruktur bedeutet dies, dass es bei Entfernen der mit T_{B_1} korrespondierenden Zeile und Spalte eine Permutation auf Blockdiagonalform mit zwei Diagonalblöcken gibt, die drei beziehungsweise 16 Diagonaleinträge

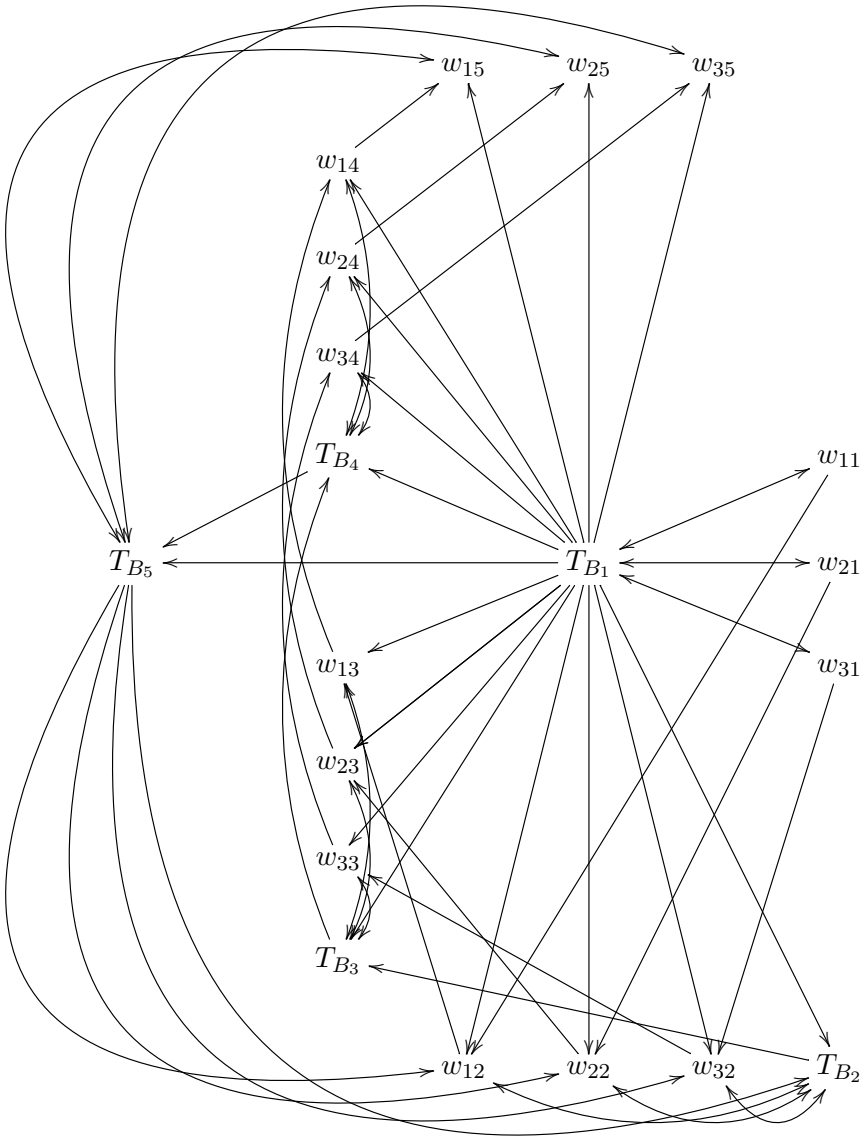


Abbildung 5.1: Adjazenzgraph der Strukturmatrix des Rohrmodells mit fünf Diskretisierungszellen.

besitzen. Um eine gute Strukturausnützung zu erzielen, ist daher eine weitere Unterteilung von G_2 nötig. Es ist

$$\kappa\left(G_2\right) = 4 \quad \sigma\left(G_2\right) = 1.$$

Folglich sind für eine Permutation auf eine feinere Blockdiagonalform vier Zeilen und Spalten zu entfernen, während es bei Entfernen einer Zeile und Spalte eine Permutation auf Blockdreiecksform gibt. Die Knotenmenge $S_2 = \{T_{B_5}\}$ ist ein Separator des gerichteten Graphen G_2. Die Entfernung des Separators führt zu dem in Abbildung dargestellten Teilgraph G_3. Dieser gerichtete Graph ist 4-fach schwach zusammenhängend, besitzt aber neun starke Zusammenhangskomponenten

$$\begin{array}{lll} W_1 = \{w_{11}\}, & W_4 = \{w_{12},\, w_{22},\, w_{22},\, T_{B_2}\}, & W_7 = \{w_{15}\}, \\ W_2 = \{w_{21}\}, & W_5 = \{w_{12},\, w_{22},\, w_{22},\, T_{B_2}\}, & W_8 = \{w_{25}\}, \\ W_3 = \{w_{31}\}, & W_6 = \{w_{12},\, w_{22},\, w_{22},\, T_{B_2}\}, & W_9 = \{w_{35}\}. \end{array}$$

Dies entspricht einer Blockdreiecksform mit Diagonalblöcken, die höchstens vier Diagonaleinträge besitzen. Das entsprechende Schur-Komplement korrespondiert mit den beiden Zuständen T_{B_1} und T_{B_5} und besitzt daher zwei Diagonaleinträge. Daher ist die Lösung des reduzierten Systems $Sv = \bar{g}$ im Rahmen der Schur-Komplement basierten Lösung des linearen Gleichungssystems mit

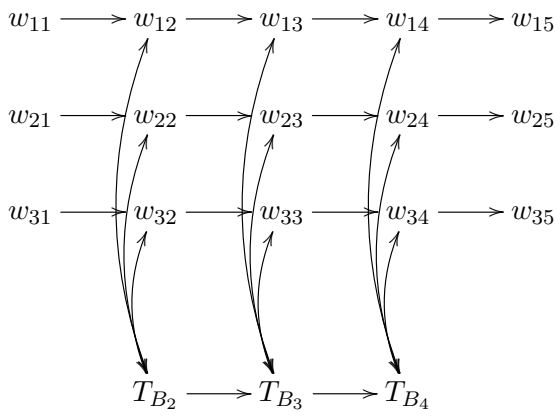


Abbildung 5.2: Teilgraph G_3 des Graphen aus Abbildung 5.1.

geringem Aufwand verbunden.
Die zugehörige Permutation des Zustandsvektors x ist

$$P : (x) \quad \mapsto \quad (w_{11},\, w_{21},\, w_{21},\, w_{12},\, w_{22},\, w_{32},\, T_{B_2},\, w_{13},\, w_{23},\, w_{33},\, T_{B_3},\\ w_{14},\, w_{24},\, w_{34},\, T_{B_4},\, w_{15},\, w_{25},\, w_{25},\, T_{B_5},\, T_{B_1})\,.$$

Hiermit resultiert folgende Struktur der Matrix:

$$S_P = \left(\begin{array}{cccccccccccccccccc}
\blacksquare & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \blacksquare \\
\cdot & \blacksquare & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \blacksquare \\
\cdot & \cdot & \blacksquare & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \blacksquare \\
\cdot & \cdot & \cdot & \blacksquare & \cdot & \blacksquare & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \blacksquare & \blacksquare \\
\blacksquare & \cdot & \cdot & \cdot & \blacksquare & \cdot & \blacksquare & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \blacksquare & \blacksquare \\
\cdot & \blacksquare & \cdot & \cdot & \cdot & \blacksquare & \blacksquare & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \blacksquare & \blacksquare \\
\cdot & \cdot & \blacksquare & \blacksquare & \blacksquare & \blacksquare & \blacksquare & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \blacksquare & \blacksquare \\
\cdot & \cdot & \cdot & \blacksquare & \cdot & \cdot & \cdot & \blacksquare & \cdot & \blacksquare & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \blacksquare & \blacksquare \\
\cdot & \cdot & \cdot & \cdot & \blacksquare & \cdot & \cdot & \cdot & \blacksquare & \blacksquare & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \blacksquare & \blacksquare \\
\cdot & \cdot & \cdot & \cdot & \cdot & \blacksquare & \blacksquare & \blacksquare & \blacksquare & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \blacksquare & \blacksquare \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \blacksquare & \cdot & \cdot & \cdot & \blacksquare & \cdot & \blacksquare & \cdot & \cdot & \cdot & \blacksquare & \blacksquare \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \blacksquare & \cdot & \cdot & \cdot & \blacksquare & \blacksquare & \cdot & \cdot & \cdot & \blacksquare & \blacksquare \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \blacksquare & \blacksquare & \blacksquare & \blacksquare & \cdot & \cdot & \cdot & \cdot & \blacksquare & \blacksquare \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \blacksquare & \cdot & \cdot & \cdot & \blacksquare & \cdot & \cdot & \blacksquare & \blacksquare \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \blacksquare & \cdot & \cdot & \cdot & \blacksquare & \cdot & \blacksquare & \blacksquare \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \blacksquare & \cdot & \cdot & \cdot & \blacksquare & \blacksquare & \blacksquare \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \blacksquare & \blacksquare & \blacksquare & \blacksquare & \blacksquare & \blacksquare \\
\blacksquare & \blacksquare & \blacksquare & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \blacksquare & \blacksquare
\end{array}\right). \tag{5.5}$$

Für die Lösung des resultierenden permutierten Gleichungssystems $PAP^{-1}\,(Px) = Pb$ wird eine Block-Gauß Elimination eingesetzt, wie auf Seite 86 beschrieben. Hierbei wird A entsprechend der Wahl des Separators $S_1 \cup S_2$ partitioniert, so dass das Schur-Komplement S_B also eine 2×2 Matrix und B die führende Teilmatrix von A in Blockdreiecksform ist.

Es ist zweckmäßig, die Permutation P hierbei nicht zur Laufzeit anzuwenden. Stattdessen ist die Reihenfolge der Zustände in der Formulierung des Modells anzupassen. Hierdurch treten im Rahmen der Lösung der Differenzialgleichung ohne weiteren Rechenaufwand die Matrizen in der gewünschten Struktur auf. Für ein komfortables Arbeiten mit derartigen Permutationen im Rahmen der Implementierung ist der Einsatz von Compiler-Direktiven zur Bezeichnung der Zustände möglich. Hierdurch kann durch die Änderung der Direktiven für den Zustandsvektor eine konsistente und handliche Änderung über alle Programmteile hinweg ohne zusätzlichen Rechenaufwand erreicht werden.

Eine Alternative zur Lösung der linearen Gleichungssysteme mittels einer Block-Gauß Elimination mit einer Struktur wie S_P in Gleichung 5.5, ist, die vorangehenden Überlegungen zur Bestimmung einer Permutation zu nutzen, um das Gleichungssystem mittels einer Gauß Elimi-

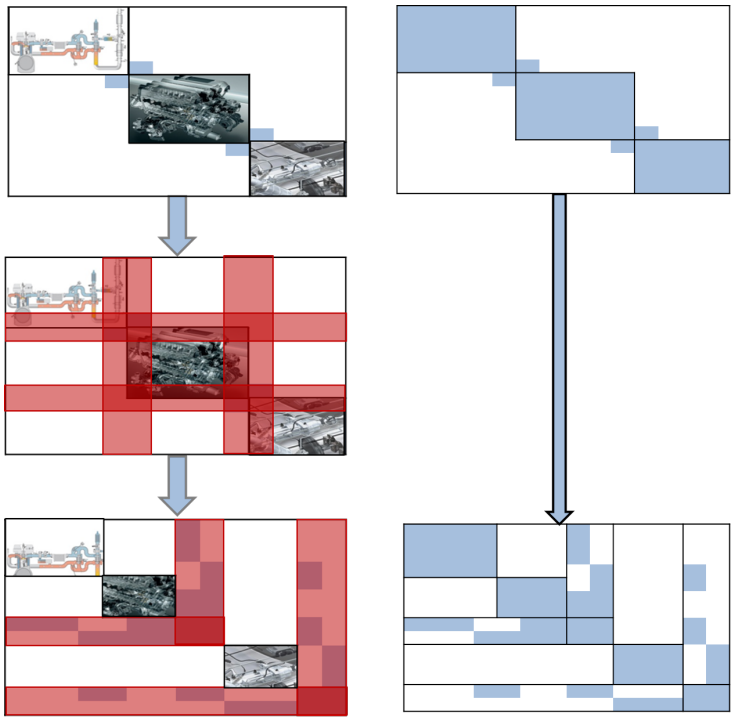


Abbildung 5.3: Schematische Strukturabbildungen (links) sowie Strukturmatrizen (rechts) eines schematischen Fahrzeugmodells. Oben: Ausgangsstruktur. Mitte: Schema der Struktur mit Kennzeichnung der zur Wahl des Separators korrespondierenden Zeilen und Spalten (rote Balken). Unten: Resultierende Struktur bei Anwendung der Schur-Komplement basierten Lösung.

nation zu lösen. Dies wird im folgenden Abschnitt betrachtet.

Ausblick: Anwendung bei weiteren mathematischen Modellen

Bei mathematischen Modellen, die aus Submodellen bestehen, ermöglicht das Schur Komplement basierte Lösen eine Modellzerlegung. Somit ist bei derartigen Modellen zu erwarten, dass der Einsatz dieser Lösungsweise in vielen Fällen von Nutzen ist.

In der folgenden Betrachtung wird eine solche Modellzerlegung schematisch illustriert und damit erläutert, weshalb das Lösen linearer Gleichungssysteme mittels einer Block-Gauß Elimination bei Modellen aus der Problemklasse gemäß Abschnitt 2.1.7 in vielen Fällen geeignet anwendbar sein dürfte und wie das Schur-Komplement hierbei günstig zu wählen ist. Dies wird am Beispiel eines Schemas eines Fahrzeugmodells veranschaulicht. Dieses Modell bestehe aus einem Luftsystem-Submodell, einem Motor-Submodell und einem Submodell für den Antriebsstrang samt Kurbelwelle. Außerdem wird angenommen, dass es nur relativ gering ausgeprägte Abhängigkeiten zwischen diesen Submodellen aufweist.

Die Ein- und Auslassventile sowie die Kurbelwelle werden in nachfolgender Erläuterung als sinnbildliche Repräsentanten der physikalischen Größen verwendet, die an den Kopplungen zwischen den Submodellen beteiligt sind, also beispielsweise Drücke und Temperaturen bei den Ventilen sowie Drehzahl und anliegendes Drehmoment bei der Kurbelwelle.

In Abbildung 5.3 ist in zwei Spalten die Struktur des zu lösenden linearen Gleichungssystems in verschiedenen Varianten gezeigt, wobei links die Submodelle und Kopplungsterme bei einer entsprechenden Anordnung der Zustände illustriert sind und rechts allein die zugehörige Strukturmatrix dargestellt ist. In der linken Spalte ist oben die Ausgangssituation abgebildet, bei der eine Anordnung der Zustände zu Grunde gelegt wird, bei der die blau dargestellten Kopplungsterme die gezeigte Position einnehmen. Auf Grund der gegebenen Abhängigkeiten zwischen

	5 Zellen	10 Zellen	15 Zellen
Originale Anordnung	114	339	664
Reverse Cuthill McKee	160	590	1320
Approximate Minimum Degree	108	286	448
Anordnung gemäß Zellen	102	227	352
Maßgeschneiderte Anordnung	102	227	352

Tabelle 5.1: Anzahl an Einträge samt Fill-in bei verschiedenen Anordnungen.

den Submodellen ist eine derartige Anordnung möglich, wobei das Verhältnis von Anzahl der Kopplungsterme und Anzahl der Zustände des Submodells lediglich als Beispiel dient.

Nun wird eine Zerlegung im Sinne einer Nested Dissection angewandt, bei der die Zustände, die jeweils an den Kopplungen zwischen Submodellen beteiligt sind, als Separator gewählt werden. Im Sinne der vorangehend beschriebenen sinnbildlichen Repräsentation werden also die Ein- und Auslassventile sowie die Kurbelwelle als Separator gewählt. Die mit den Zuständen des Separators korrespondierenden Zeilen und Spalten sind im mittleren der drei schematischen Strukturabbildungen durch rote Balken markiert. Im Rahmen der Zerlegung werden sie permutiert. Die resultierende schematische Strukturabbildung ist im unteren der drei schematischen Strukturabbildungen gezeigt. Rechts daneben ist die resultierende Gleichungsstruktur dargestellt. Diese ermöglicht eine effiziente Lösung, da anstelle der gesamten Matrix bereits eine Elimination innerhalb der Diagonalblöcke bei dieser Struktur ausreicht.

5.3.3 Lokale Methoden

Im Rahmen der vorangehenden Analyse zur Lösung eines Gleichungssystems mittels einer Block-Gauß Elimination mit einer Struktur wie in Gleichung 5.1 wird eine Permutation ermittelt, durch die die Matrix auf eine Struktur wie in Gleichung 5.5 transformiert wird. Statt einer Block-Gauß Elimination kann diese transformierte Gleichung mittels einer Gauß Elimination gelöst werden. In diesem Fall wird durch die Analyse der Gleichungsstruktur also ein *Ordering* bzw. eine Anordnung manuell bestimmt. Dies stellt eine Alternative zu den lokalen Methoden aus Abschnitt 5.2.3 dar. Im Folgenden wie die durch die Analyse aus Abschnitt 5.3.2 gewonnene maßgeschneiderte Anordnung gegenübergestellt zu den Permutationen des Reverse Cuthill McKee Algorithmus und des Approximate Minimum Degree Algorithmus sowie zwei natürlichen physikalisch motivierten Anordnungen.

Bei der originalen Anordnung sind jeweils korrespondierende Massenanteile der Gasfraktionen $w_{j,i}$ sowie die Temperaturen T_{B_i} unterschiedlicher Zellen zusammen gruppiert. Eine andere natürliche physikalisch motivierte Anordnung ist, dass jeweils sämtliche Zustände einer Zelle i zusammen gruppiert werden. Innerhalb einer Zelle wird die Temperatur T_{B_i} jeweils zuletzt angeordnet, da hierdurch die Pfeilmatrizen einer Zelle nach rechts weisen und für den Fill-in günstig sind (vgl. Gleichung 5.2). Die Zustände der verschiedenen Diskretisierungszellen werden umgekehrt zur Reihenfolge der Zellen angeordnet, da dies ebenfalls zu einem geringeren Fill-in führt, weil hierdurch die Kopplungsterme zwischen den Diskretisierungszellen weniger Fill-in verursachen.

In Tabelle 5.1 sind die Anzahl an Einträgen des Fill-in in verschiedenen Konstellationen aufgeführt. Die Anordnung gemäß den Diskretisierungszellen und die maßgeschneiderte Anordnung aus Abschnitt 5.3.2 führen auf dieselben Werte für den Fill-in, sind aber nicht äquivalent. Im Fall von fünf Diskretisierungszellen führen die verschiedenen Anordnungen mit Ausnahme der Anordnung durch den Reverse Cuthill McKee Algorithmus zu Werten, die sich um höchstens 12 % voneinander unterscheiden. Die Reduktion des Fill-in durch die verschiedenen Anordnungen außer derjenigen durch den Reverse Cuthill McKee Algorithmus gegenüber der originalen Anordnung steigt mit der Anzahl an verwendeten Diskretisierungszellen an. Im Fall von zehn Diskretisierungszellen wird eine Reduktion des Fill-in um etwa 33.0% durch die Anordnung ge-

mäß Zellen und die maßgeschneiderte Anordnung erzielt. Bei 15 Diskretisierungszellen liegt die entsprechende Reduktion bei 47.0 %. Die Anordnung basierend auf dem Reverse Cuthill McKee Algorithmus führt zu einer beträchtlichen Vergrößerung des Fill-in, die zwischen etwa 40.3 % und 98.8 % liegt. Somit ist diese Anordnung für das vorliegende Gleichungssystem ungünstig. Die Anordnung basierend auf dem Approximate Minimum Degree Algorithmus führt zu einer Reduktion des Fill-ins zwischen etwa 5.3 % und 32.5 %. Somit ist diese Anordnung für das vorliegende Gleichungssystem gut geeignet. Die Anordnung gemäß Zellen sowie die maßgeschneiderte Anordnung sind der Anordnung basierend auf dem Approximate Minimum Degree Algorithmus im vorliegenden Gleichungssystem aber überlegen.

Numerische Stabilität bei Umordnungen Wird bei einem Gleichungssystem eine Umordnung vorgenommen, um eine im Hinblick auf den Fill-in günstigere Struktur zu erzielen, ist darauf zu achten, dass dies in einer Weise geschieht, dass die numerische Stabilität bei einer Lösung des Gleichungssystems erhalten bleibt. Hierzu sind geeignete Pivoteinträge sicherzustellen. In [39] wird empfohlen, die numerische Stabilität dadurch zu wahren, dass die Pivoteinträge $a_{kk}^{(k)}$ die Relation $\left|a_{kk}^{(k)}\right| \geq u \left|a_{ik}^{(k)}\right|$ mit $i > k$ für ein problemabhängiges $u \in \mathbb{R}^+$ erfüllen. Ein in [39] zitierter Beispielwert für u basierend auf eine Reihe von Experimenten aus [25] liegt bei $u = 0.25$. Die entsprechende Relation für die Pivoteinträge ist im Fall der Referenztrajektorie beim Rohrmodell für jede der Umordnungen erfüllt, so dass im Falle der Leitanwendung die Umordnungen als unkritisch im Hinblick auf die numerische Stabilität eingestuft werden.

Die linearen Gleichungssysteme, die im Rahmen eines linear-impliziten Euler-Verfahrens bei der Referenztrajektorie der Leitanwendung auftreten, sind wie in Abschnitt 5.1 beschrieben wenig anfällig für eine numerisch instabile Lösung im Rahmen einer Gauß Elimination. Bei anderen Anwendungen kann das Finden einer guten Balance zwischen einer numerisch stabilen Umordnung und einer im Hinblick auf den Fill-in günstigen Umordnung eine sorgfältige Untersuchung erfordern.

5.3.4 Nutzen von Strukturwissen im Sparse-Format

In der vorliegenden Arbeit besteht ein zentrales Motiv zur Rechenzeitreduktion bei der Gleichungslösung darin, die Struktur des Gleichungssystems bereits vor der Implementierung zu analysieren und die damit gewonnenen Erkenntnisse im Rahmen des Lösungsalgorithmus zu nutzen. Die verfügbaren Kenntnisse lassen sich zum Teil auch im Rahmen der Speicherung der dünn besetzten Matrizen ausnutzen. Dies wird für das CRS Format gezeigt. Zu berücksichtigen ist, dass, wie in Abschnitt 3.1.2 beschrieben, die Speicheradressen der Variablen im RAM-Speicher in Fahrzeug-Steuergeräten typischerweise fest reserviert sind. Hierdurch sind statische Speicherstrukturen für dünn besetzte Matrix günstig. Im Folgenden wird eine statische Speicherstruktur beschrieben, die Vorteile hinsichtlich der Rechenzeit mit sich bringt und zugleich im Hinblick auf den Speicherbedarf gut geeignet ist:

Im Rahmen einer Analyse vor der Implementierung wird die betrachtete Matrixstruktur samt des im Rahmen der Elimination auftretenden Fill-in ermittelt. Genau diese Einträge werden bei der Speicherung der Matrix erfasst. Sie werden dabei in jedem der Umformungsschritte unabhängig davon gespeichert, ob sie Null oder ungleich Null sind. Hierdurch entsteht eine Speicherstruktur der Matrix, die über die Umformungsschritte hinweg statisch bleibt, da im Rahmen der Elimination kein Fill-in bei Matrixelementen auftritt, die nicht vornherein erfasst werden. Folglich lässt sich der zur Speicherung der Matrix benötigte Speicherplatzbedarf präzise angeben und es ist nicht nötig darüber hinaus Speicher für potentielle weitere Einträge zu reservieren. Dies verringert den benötigten Speicher. Die statische und vorab ermittelbare Struktur kann im Rahmen des CRS Format weitergehend genutzt werden:

Die Spaltenindizes der Tripel der Einträge sind bekannt. Diese müssen also nicht mehr im Rahmen der Umformung von A mitgeführt werden, sondern können statisch abgelegt werden. Darüber

hinaus kann die Indexsuche bei der Ermittlung von Einträgen einer gegebenen Spalte eingespart werden kann: Die Indizes der Einträge jeder einzelnen Spalte werden hierzu bereits vor der Implementierung bestimmt und statisch abgelegt.

Diese Art der Speicherung ermöglicht eine Reduktion der Rechenzeit im Rahmen der Lösung eines linearen Gleichungssystems. Dies ist am Beispiel des Rohrmodells in Abschnitt 5.5 belegt. Hinsichtlich des Speicherbedarfs stellt sich die Situation wie nachfolgend beschrieben dar.

Speicherplatzbedarf Je Eintrag wird eine 32 Bit Gleitkommavariable für den Wert des Eintrags sowie zwei Ganzzahlvariablen wie bei der Speicherstruktur im Rahmen des CRS Formats benötigt sowie jeweils eine zusätzliche Ganzzahlvariable für die Erfassung des Eintrags im Rahmen des zugehörigen Spaltenvektors. Wie in Tabelle 5.1 aufgeführt, beträgt die die maximale Anzahl an Einträgen einschließlich des während der Elimination auftretenden Fill-in im Fall der originalen Anordnung beim Rohrmodell mit fünf Diskretisierungszellen 114 Einträge. Somit ergibt sich ein Speicherplatzbedarf von etwa

$$M_{\text{maß}} \approx 114\,(4\,\text{Byte} + 1\,\text{Byte} + 1\,\text{Byte} + 1\,\text{Byte}) = 798\,\text{Byte}$$

zuzüglich des Speicherplatzbedarfs für einzelne nützliche Hilfsgrößen wie beispielsweise die Gesamtanzahl der Einträge. Im Fall einer Speicherung ohne Sparse-Format beträgt der Speicherplatzbedarf

$$M_{\text{voll}} = 25^2\,(4\,\text{Byte}) = 2500\,\text{Byte}.$$

Bei Verwendung der üblichen Speicherstruktur im Rahmen eines CRS Formats hängt der Speicherplatzbedarf wesentlich davon ab, wie viel Speicher für den auftretenden Fill-in reserviert wird. Wird keine Voruntersuchung der Struktur vorgenommen, so dass der Fill-in je Matrixzeile nicht abgeschätzt wird, ist für jede Matrixzeile Speicher für einen Fill-in von bis zu 25 Einträge je Zeile zu reservieren und der Speicherplatzbedarf somit etwa

$$M_{\text{CRS,max}} \approx 25^2\,(4\,\text{Byte} + 1\,\text{Byte} + 1\,\text{Byte}) = 3800\,\text{Byte}.$$

Wird eine Voruntersuchung des Fill-in verwendet und die Speicherreservierung dementsprechend angepasst, so beträgt der Speicherplatzbedarf etwa

$$M_{\text{CRS,min}} \approx 114\,(4\,\text{Byte} + 1\,\text{Byte} + 1\,\text{Byte}) = 684\,\text{Byte}.$$

Somit liegt der Speicherplatzbedarf im Rahmen der maßgeschneiderten Speicherstruktur im Rahmen eines CRS Formats etwa 17 % über dem Speicherplatzbedarf bei einem CRS Format mit einer Voruntersuchung des Fill-in und im Fall des Rohrmodells signifikant unter dem Bedarf im Fall einer Speicherung ohne Sparse-Format oder mit CRS Format ohne Voruntersuchung des Fill-in.

Insgesamt ist durch die beschriebene Vorgehensweise zur Speicherung dünn besetzter Matrizen im Falle der Leitanwendung eine Reduktion der Rechenzeit im Vergleich zu einer Elimination mit einem CRS Format in gängiger Weise zu erzielen sowie ein Speicherplatzbedarf, der geringfügig über dem Bedarf im Fall der Verwendung eines CRS Formats mit Voruntersuchung des Fill-in und deutlich unter dem Bedarf im Fall einer Speicherung ohne Sparse-Format liegt.

5.4 Implementierung

Im Abschnitt 5.5 werden fünf Varianten zur Lösung von Gleichungssystemen der Struktur wie in Gleichung 5.1 beziehungsweise einer entsprechenden Struktur im Fall von zehn und 15 Diskretisierungszellen beim Rohrmodell untersucht. Diese werden im Folgenden beschrieben.

Variante 1: Gauß Elimination Bei dieser Variante wird eine Gauß Elimination mit Spaltenpivotisierung eingesetzt. Sie repräsentiert ein numerisches Standardverfahren zur Lösung eines linearen Gleichungssystems. Die dünn besetzte Struktur wird hierbei nicht berücksichtigt.

Variante 2: Elimination mit Sparse-Format Hierbei wird das CRS Format zur Speicherung der Matrix eingesetzt (vgl. 5.2.1). In dieser sowie den nachfolgenden Varianten wird eine Elimination ohne Pivotisierung eingesetzt. Für jede Zeile der Matrix $A \in \mathbb{R}^{n \times n}$ wird der Speicherplatz für n Einträge reserviert, um für Fill-in im Rahmen der Elimination ausreichend zusammenhängenden Speicherplatz zu haben. Innerhalb einer Zeile sind die Einträge entsprechend ihrer Spaltenindizes sortiert.

Variante 3: Elimination mit maßgeschneidertem Sparse-Format Zur Speicherung der Matrix wird die in Abschnitt 5.3.4 beschriebene maßgeschneiderte Anpassung des CRS Formats eingesetzt. Die Struktur des im Rahmen der Elimination auftretenden Fill-in sowie die Spaltenindizes der Einträge und die Indizes der Einträge innerhalb der verschiedenen Spalten werden vorab erzeugt.

Variante 4: Elimination mit maßgeschneidertem Sparse-Format und manuellem Ordering Die Variante unterscheidet sich von Variante 3 genau darin, dass eine Anordnung gewählt wird, so dass die Matrix des Gleichungssystems eine Struktur entsprechend zur Gleichung 5.5 besitzt. Dies lässt sich durch die in Abschnitt 5.3.2 beschriebene Umordnung der Zustände

$$\begin{aligned} P : (x) \quad \mapsto \quad & (w_{11},\, w_{21},\, w_{21},\, w_{12},\, w_{22},\, w_{32},\, T_{B_2},\, w_{13},\, w_{23},\, w_{33},\, T_{B_3}, \\ & w_{14},\, w_{24},\, w_{34},\, T_{B_4},\, w_{15},\, w_{25},\, w_{25},\, T_{B_5},\, T_{B_1})\,. \end{aligned}$$

erzielen.

Variante 5: Maßgeschneiderte Block-Gauß Elimination Diese Variante setzt die Überlegungen aus Abschnitt 5.3.2 um. Hierzu wird das Gleichungssystem basierend auf einer wie dort beschriebenen Wahl des Schur-Komplements gemäß des Algorithmus aus Abschnitt 5.2.4 gelöst. Das reduzierte System hat hierbei die Dimension zwei. Der Hauptaufwand liegt somit bei Schritt 1 des Algorithmus. Für diesen wird keine Elimination im eigentlichen Sinn eingesetzt. Um diesen möglichst effizient zu gestalten, wird eine Anordnung gemäß 5.3.2 zu Grunde gelegt, so dass das Teilsystem Blockdreiecksstruktur hat. Dementsprechend ergibt sich die Lösung des Teilsystems dann blockweise und sukzessive aus den Lösungen von Gleichungen basierend auf den Diagonalblöcken. Die Diagonalblöcke besitzen jeweils vier Diagonaleinträge sowie eine nach rechts unten weisende Pfeilstruktur. Zur Optimierung der Rechenzeit ist für derartige Blöcke eine Routine implementiert, die zur Lösung von Gleichungssystemen dieser speziellen Form in naheliegender Weise eine explizite Folge von Rechenschritten enthält.

5.5 Numerische Experimente

Im Folgenden wird dargestellt, wie sich die verschiedenen in diesem Kapitel beschriebenen Ansätze zur Lösung eines linearen Gleichungssystems mit einer Struktur analog zur Gleichung 5.1 auf die Rechenzeiten auswirken. Hierzu werden die Ansätze jeweils auch auf die Strukturen angewandt, die sich im Fall von zehn und 15 Zellen bei der Semidiskretisierung des Rohrmodells ergeben.

In Tabelle 5.2 sind die Rechenzeiten der verschiedenen Varianten aus Abschnitt 5.4 dargestellt. Diese werden im Folgenden diskutiert:

Die gemessenen Rechenzeiten bei der Gauß Elimination skalieren etwa kubisch, so dass also eine Verdopplung respektive einer Verdreifachung der Anzahl der Variablen etwa zur achtfachen

Rechenzeit [µs]	5 Zellen	10 Zellen	15 Zellen
Gauß Elimination	799	6024	19917
Elimination mit Sparse-Format	287	1030	2158
Elimination mit maßgeschneidertem Sparse-Format	146	538	1180
Elimination mit maßgeschneidertem Sparse-Format und manuellem Ordering	134	306	480
Maßgeschneiderte Block-Gauß Elimination	78	222	433

Tabelle 5.2: Rechenzeit zur Lösung eines linearen Gleichungssystems in Mikrosekunden für verschiedene Anzahlen an Diskretisierungszellen und Varianten des Lösungsverfahrens.

Rechenzeit respektive zur 27-fachen Rechenzeit führt, wobei die gemessenen Rechenzeiten dabei bis zu etwa 7.7 % unterhalb der Werte liegen, die bei einer exakt kubischen Skalierung erhalten würden. Das Verhältnis der Rechenzeiten liegt bei etwa $1 : 7.54 : 24.9$.

Durch die Verwendung eines Sparse-Formats werden die Rechenzeiten zur Lösung der Gleichungssysteme zwischen 64.1 % und 89.2 % reduziert. Es ist zu beobachten, dass sich zudem das Skalierungsverhalten der Rechenzeiten wesentlich von denen im Fall der Gauß Elimination ohne Sparse-Format unterscheidet: Das Verhältnis der Rechenzeiten liegt bei etwa $1 : 3.59 : 7.52$. Somit wird die Rechenzeit nicht durch einen kubisch skalierenden Anteil dominiert. Ein Zusammenhang mit der Skalierung der Anzahl der Einträge der Matrix ist naheliegend. Für diese bestehen gemäß Tabelle 5.1 für die verwendete originale Anordnung die Proportionen von etwa $1 : 2.97 : 5.82$. Aus dem gegenüber der Gauß Elimination mit Spaltenpivotisierung ohne Sparse-Format wesentlich veränderte Skalierungsverhalten lässt sich schließen, dass die Reduktion der Rechenzeit nicht vorrangig darauf zurückzuführen ist, dass keine Pivotisierung eingesetzt wird, da in diesem Fall das Skalierungsverhalten einer Gauß Elimination vorzufinden wäre.

Durch eine Elimination mit dem maßgeschneiderten Sparse-Format, wie es in Abschnitt 5.3.4 beschrieben ist, wird die Rechenzeit gegenüber Variante 2 zwischen etwa 49.1 % und 45.3 % reduziert. Die Proportionen zwischen den Rechenzeiten für die verschiedenen Anzahlen an Diskretisierungszellen liegen nahe an denen der Elimination mit Sparse-Format: $1 : 3.68 : 8.08$.

Bei Variante 4 wird neben dem Sparse-Format aus Abschnitt 5.3.4 die Anordnung der Zustände eingesetzt, wie sie in Abschnitt 5.3.2 bestimmt wird. Hierdurch wird die Rechenzeit um etwa 8.2 % bei fünf Diskretisierungszellen, um etwa 43.1 % bei zehn Diskretisierungszellen sowie um etwa 59.3 % bei 15 Diskretisierungszellen reduziert. Wie aus Tabelle 5.1 hervorgeht, ist die Anzahl der Einträge bei der manuellen Anordnung gegenüber der originalen Anordnung um etwa 10.5 % bei fünf Diskretisierungszellen, um etwa 33.0 % bei zehn Diskretisierungszellen sowie um etwa 47.0 % bei 15 Diskretisierungszellen reduziert. Somit liefert der Einsatz des manuellen Orderings eine Rechenzeitreduktion, die die Reduktion der Anzahl der Einträge der Matrizen im moderaten Maß übertrifft.

Bei der maßgeschneiderten Block-Gauß Elimination liegt der Hauptaufwand in Schritt 1 des in Abschnitt 5.2.4 beschriebenen Algorithmus. Es ist naheliegend, dass der Aufwand der Lösung des entsprechenden Teilsystems durch die nichttrivialen Diagonalblöcke dominiert wird. Im Fall von fünf Diskretisierungszellen sind drei dieser nichttrivialen Diagonalblöcke enthalten, im Fall von zehn Diskretisierungszellen treten acht dieser Blöcke auf und im Fall von 15 Diskretisierungszellen sind es 13 dieser Blöcke. Aus diesem Grund sind etwa die Proportionen $3 : 8 : 13$ zwischen den Rechenzeiten der verschiedenen Gleichungssysteme zu erwarten. Die gemessenen Laufzeiten besitzen die Proportionen $3 : 8.54 : 16.7$. Die Rechenzeit im Fall von 15 Diskretisierungszellen liegt somit etwas höher als erwartet. Als eine Ursache hierfür ist denkbar, dass das Einlesen der Teilmatrizen nicht wie bei Variante 4 elementweise anhand eines vorab ermittelten Arrays erfolgt, sondern bei der betrachteten Implementierung stattdessen alle Einträge oberhalb einer Nebendiagonalen durchlaufen werden. Gegenüber den Rechenzeiten der verschiedenen Varianten der Elimination fallen die Rechenzeiten bei der maßgeschneiderten Block-Gauß

Elimination geringer aus, wobei der Unterschied gegenüber einer Elimination mit einem Sparse-Format gemäß Abschnitt 5.3.4 und einer Anordnung der Zustände gemäß Abschnitt 5.3.2 bei fünf Diskretisierungszellen signifikant ist und mit zunehmender Anzahl an Diskretisierungszellen geringer ausfällt.

Insgesamt wird deutlich, dass die Verwendung eines Sparse-Formats im Rahmen der Elimination eine wesentliche Rechenzeitreduktion bewirkt. Durch eine zusätzliche Anpassung an eine gegebene Gleichungsstruktur wie beispielsweise durch die in Abschnitt 5.3.4 beschriebene Variante eines Sparse-Formats sowie der Umordnung der Zustände gemäß Abschnitt 5.3.2 ist jeweils nochmals eine weitere signifikante Reduktion der Rechenzeit zu erzielen. Darüber hinaus ist festzustellen, dass durch Variante 5 ein Verfahren, das, wie in Abschnitt 5.3.2 ausgeführt ist, sehr spezifisch auf die untersuchte Gleichungsstruktur zugeschnitten ist, praktikabel umsetzbar ist und zu sehr guten Rechenzeiten führt, die unterhalb der Rechenzeiten der übrigen Varianten liegt.

6 Das Parareal Verfahren für Anwendungen im Automotive Bereich

Bei zukünftigen Generationen von Fahrzeug-Steuergeräten ist wie in Abschnitt 3.2.5 beschrieben von einer Zunahme der Anzahl an Rechenkernen auszugehen. Im Hinblick auf den Einsatz von Simulationen mathematischer Modelle auf Fahrzeug-Steuergeräten stellt sich aus diesem Grund die Frage nach einer Parallelisierungsmöglichkeit im Rahmen der Lösung der entsprechenden Differentialgleichungen. Eine Möglichkeit hierzu stellt der Parareal Algorithmus dar. Dieser wird im Folgenden vorgestellt und im weiteren Kapitel untersucht.

6.1 Beschreibung der Verfahrens

Das Parareal Verfahren wurde in [69] eingeführt und ist ein Schema zur Integration von Differentialgleichungen, das eine Parallelisierung in der Zeit ermöglicht. Die Historie der Entwicklung derartiger Integrationsansätze zur Parallelisierung in der Integrationszeit ist in [47] beschrieben und geht bis auf [78] zurück. Diese Klasse von Verfahren zeichnet sich dadurch aus, dass die Zeitschritte im Rahmen der Lösung einer Differentialgleichung nicht streng sequentiell ermittelt werden, sondern trotz ihrer natürlich vorgegebenen temporalen und damit in gewisser Hinsicht auch kausalen Reihenfolge zum Teil parallelisiert werden.

6.1.1 Definition

Um einen Simulationszeitschritt mit der Schrittweite ΔT zu berechnen, wird beim Parareal Verfahren das Zeitintervall $[T_n; T_n + \Delta T]$ in kleinere Intervalle unterteilt. Diese müssen nicht äquidistant sein , allerdings werden in vorliegender Arbeit lediglich äquidistante Unterteilungen mit $\Delta t = \Delta T/m$ betrachtet. Zeitschritte mit der Schrittweite ΔT werden im Folgenden als Makrozeitschritte und die zugehörigen Zeitintervalle als Makrozeitintervalle bezeichnet und Zeitschritte mit der Schrittweite Δt werden Mikrozeitschritte und die zugehörigen Zeitintervalle $[T_n + i\Delta t; T_n + (i+1)\,\Delta t]$ Mikrozeitintervalle genannt. Eine Übersicht über die im Kontext des Parareal Verfahrens verwendete Notation ist im Anhang zu finden.

Im Rahmen des Parareal Verfahrens werden zwei unterschiedliche Integratoren eingesetzt. Diese werden als grobe und feine Stufe und auf Englisch mit *coarse stage* und *fine stage* bezeichnet und im Weiteren als $G(\cdot,\cdot,\cdot)$ beziehungsweise $F(\cdot,\cdot,\cdot)$ notiert, wobei das erste Argument den Endzeitpunkt der Integration, das zweite Argument den Startzeitpunkt und das letzte Argument den zugehörigen Anfangswert angibt. Die grobe Stufe kommt in jedem Makrozeitschritt auf jedem der Kerne etliche Male zum Einsatz. Dementsprechend ist für die grobe Stufe ein geringer Rechenaufwand eine wesentliche Anforderung. Die feine Stufe wird je Makrozeitschritt und je Kern lediglich einmal eingesetzt. Daher kann der Rechenaufwand der feinen Stufe entsprechend höher sein als bei der groben Stufe. Damit das Parareal Verfahren zweckmäßig arbeiten kann, muss die feine Stufe eine höhere Genauigkeit als die grobe Stufe aufweisen.

Die Berechnung eines Zeitschrittes der Lösung des Anfangswertproblems

$$\dot{x}(t) = f(t, x), \quad x(T_n) = x_n$$

mit dem Parareal Verfahren läuft folgendermaßen ab: Im Rahmen des ersten Durchlaufs werden mit der groben Stufe ausgehend vom aktuellen Zeitpunkt T_n die Anfangswerte $x_{n,i}^{(0)} =$

$G\left(t_{n,i}, t_n, x_n\right)$ für die Mikrozeitschritte zu den Zeitpunkten $t_{n,i} = T_n + i\Delta t$ mit $i = 1, \ldots, m$ ermittelt. Generell wird $x_{n,0}^{(k)} = x_n$ für alle Iterationen k gesetzt. Nun wird in der Iteration $k = 1$ die feine Stufe auf jeden der Mikrozeitschritte angewandt:

$$\tilde{x}_{n,i+1}^{(k)} = F\left(t_{n,i+1}, t_{n,i}, x_{n,i}^{(k-1)}\right).$$

Dies kann parallel durchgeführt werden. Hierdurch werden für die Intervalle $[t_{n,i}; t_{n,i+1}]$ also relativ akkurate Lösungen berechnet, deren zu Grunde liegende Startwerte $x_{n,i}^{(k-1)}$ allerdings im Allgemeinen nicht diejenigen sind, die sich mit der einer sequentiellen Rechnung mit der feinen Stufe ergeben. Insbesondere gilt im Allgemeinen

$$S_{n,i+1}^{(k)} = \tilde{x}_{n,i+1}^{(k)} - x_{n,i+1}^{(k-1)} \neq 0 \text{ für } 1 \leq i \leq m-2,$$

so dass zwischen den berechneten Verläufen der feinen Stufe auf den Mikrozeitintervallen gewissermaßen Sprünge an den Grenzen $t_{n,i}$ der Mikrozeitintervalle vorliegen. Mit der groben Stufe werden diese Sprünge nun über die Mikrozeitschritte hinweg propagiert, um hierdurch sukzessive eine Korrektur für die Resultate der feinen Stufe der verschiedenen Mikrozeitschritte zu berechnen. Dies wird mittels $\delta_{n,0}^{(k)} = 0$ und

$$\delta_{n,i+1}^{(k)} = G\left(t_{n,i+1}, t_{n,i}, x_{n,i}^{(k)} + S_{n,i}^{(k)} + \delta_{n,i}^{(k)}\right) - G\left(t_{n,i+1}, t_{n,i}, x_{n,i}^{(k-1)}\right) \text{ für } i = 0, \ldots, m-1$$

sowie

$$x_{n,i+1}^{(k)} = \tilde{x}_{n,i+1}^{(k)} + \delta_{n,i+1}^{(k)} \text{ für } i = 0, \ldots, m-1.$$

erreicht. Auf Grund von $\tilde{x}_{n,i+1}^{(k)} = x_{n,i+1}^{(k)} + S_{n,i}^{(k)}$ werden die Werte $S_{n,i}^{(k)}$ in einer Implementierung nicht explizit ermittelt. Dieses Vorgehen wird nun über k iteriert.

Insgesamt ergibt sich also im n-ten Integrationsschritt mit dem Parareal Verfahren die Iterationsvorschrift (vgl. [49]): Für $i = 0, \ldots, m-1$ und $k \geq 0$ ist

$$x_{n,i+1}^{(k+1)} = F\left(t_{n,i+1}, t_{n,i}, x_i^{(k)}\right) + G\left(t_{n,i+1}, t_{n,i}, x_{n,i}^{(k+1)}\right) - G\left(t_{n,i+1}, t_{n,i}, x_{n,i}^{(k)}\right). \tag{6.1}$$

Ein Pseudo-Code des Parareal Verfahrens ist im Algorithmus 6.1 aufgeführt.

Algorithm 6.1 Pseudo-Code des Parareal Algorithmus .

```
for i = 1 to m do
    x_i^{G,f} = G(t_i, t_{i-1}, x_{i-1}^{G,f})
    x_i = x_i^{G,f}
end for
while Iterationskriterium do
    x_i^{G,a} = x_i^{G,f}
    parfor i = 1 to m do
        x̃_i = F(t_i, t_{i-1}, x_{i-1})
    end parfor
    for i = 1 to m do
        x_i^{G,f} = G(t_i, t_{i-1}, x_{i-1})
        δ_i = x_i^{G,f} − x_i^{G,a}
        x_i = x̃_i + δ_i
    end for
end while
```

In vorliegender Arbeit wird das Parareal Verfahren wie in [69, 73, 14] als Defektkorrekturverfahren aufgefasst. Eine alternative Betrachtungsweise wird in [49] dargestellt. Hierbei wird die

Möglichkeit gezeigt, das Parareal Verfahren als ein Mehrfachschießverfahren zu interpretieren.

6.1.2 Eigenschaften

Im Folgenden werden die grundlegenden Eigenschaften des Parareal Verfahrens zusammengefasst. Hierbei wird die Notation von Abschnitt 6.1.1 weiterhin verwendet. Zunächst wird eine sehr einfache Eigenschaft angegeben.

Lemma 6. *Für $i \leq k$ gilt $x_{n,i}^{(k)} = F\left(t_{n,i}, t_n, x_n\right)$. Wird als feine Stufe ein exakter Integrator verwendet, so sind in der k-ten Iteration des Parareal Verfahrens die Werte $x_{n,i}^{(k)}$ der Mikrozeitintervalle $[t_{n,i-1}; t_{n,i}]$ für $1 \leq i \leq k$ exakt.*

Beweis. Satz 3.1. in [93]. □

Eine Fehlerschranke für das Parareal Verfahren bietet die folgende Aussage:

Theorem 7. *Betrachte*

$$\dot{x}\left(t\right) = f\left(x\right), \quad x\left(t_n\right) = x_n.$$

Sei F der exakte Integrator und G ein Integrator der Konsistenzordnung p, für dessen lokalen Diskretisierungsfehler τ gelte $\|\tau\| \leq C_1 \Delta t^{p+1}$. G erfülle zudem folgende Lipschitz-Bedingung

$$\left\|G\left(t + \Delta t, t, x\right) - G\left(t + \Delta t, t, y\right)\right\| \leq \left(1 + C_2 \Delta t\right) \left\|x - y\right\|.$$

Außerdem gelte

$$F\left(t_{n,i}, t_{n,i-1}, x\right) - G\left(t_{n,i}, t_{n,i-1}, x\right) = c_{p+1}\left(x\right) \Delta t^{p+1} + c_{p+2}\left(x\right) \Delta t^{p+2} + \cdots,$$

wobei c_j mit $j = p+1, p+2, \ldots$ stetig differenzierbar seien. Dann gilt

$$\max_{1 \leq i \leq m} \left\| x\left(t_{n,i}\right) - x_{n,i}^{(k)} \right\| \leq \frac{\left(C_1 \Delta T\right)^k}{k!} e^{C_2\left(\Delta T - (k+1)\Delta t\right)} \left(\Delta t\right)^{pk} \max_{1 \leq i \leq m} \left\| x\left(t_{n,i}\right) - x_{n,i}^{(0)} \right\|$$

Beweis. Der Beweis ist zu finden bei Theorem 1 aus [48], wobei die Formulierung der Aussage entsprechend Theorem 2 aus [47] gewählt ist. □

Ein signifikanter Aspekt an vorangehender Abschätzung ist, dass die Konstanten weder von den Zeitschrittweiten Δt und ΔT noch von der Iterationszahl des Parareal Verfahrens k abhängen. Hierdurch folgt aus vorangehender Aussage zum einen, sofern die Voraussetzungen des Satzes erfüllt sind, dass für ein Parareal Verfahren, dessen grobe Stufe die Konsistenzordnung p hat und dessen feine Stufe der exakte Integrator ist, nach k Iterationen mindestens die Konsistenzordnung pk besitzt. Zum anderen ermöglicht die Fehlerschranke eine Aussage darüber, wie $x_{n,i}^{(k)}$ in Abhängigkeit von der Anzahl k an Parareal Iterationen konvergiert: Auf beschränkten Intervallen $[T_n; T_{n+1}]$ liegt eine superlineare Konvergenz vor (vgl. [48]).
In vorliegender Arbeit wird das Parareal Verfahren derart eingesetzt, dass die Anzahl m an Mikrozeitschritten je Makrozeitschritt als fest angenommen wird. In diesem Fall ist eine Konvergenzaussage in Abhängigkeit von ΔT also ebenfalls nützlich.

Korollar 8. *Seien die Voraussetzungen gemäß Satz 7 gegeben. Ferner gelte für die grobe Stufe G des Parareal Verfahrens, dass sie die Zustände konstant auf dem Anfangswert der verschiedenen Mikrozeitintervalle hält, so dass also stets $G\left(t + \Delta t, t, x\right) = x$ gilt. Die rechte Seite erfülle zudem eine Lipschitz-Bedingung zur Konstante L. Dann besitzt ein Parareal Verfahren mit $k_0 = 1$ Iteration, für das $\Delta T \leq C_3$ und $\Delta t \leq C_3$ gilt, die Konsistenzordnung 1.*

Beweis. Die grobe Stufe G ist konsistent mit der Ordnung $p = 0$. Gemäß Satz 7 gilt

$$\max_{1\leq i\leq m}\left\|x\left(t_{n,i}\right)-x_{n,i}^{(k_0)}\right\| \leq \frac{\left(C_1\Delta T\right)^{k_0}}{k_0!}e^{C_2(\Delta T-(k_0+1)\Delta t)}\max_{1\leq i\leq m}\left\|x\left(t_{n,i}\right)-x_{n,i}^{(0)}\right\|,$$

woraus die Behauptung auf Grund von

$$\max_{1\leq i\leq m}\left\|x\left(t_{n,i}\right)-x_{n,i}^{(0)}\right\| \leq L\Delta TC_4$$

mit

$$C_4 = \max_{1\leq i\leq m}\left\|x\left(t_{n,i}\right)-x_{n,i}^{(0)}\right\|\max_{\Delta T,\Delta t\in[0;C_3]}e^{C_2(\Delta T-(k_0+1)\Delta t)}$$

folgt. □

Für das in Korollar 8 beschriebene Parareal Verfahren ist die Vermutung naheliegend, dass sich auch eine Konvergenzaussage erzielen lässt, die auch für eine konstante Makrozeitschrittweite ΔT eine Konvergenz des Parareal Verfahrens bei $\Delta t \to 0$ der Ordnung 1 liefert. Für die vorliegende Arbeit ist ein solches Resultat nicht nötig und wird daher nicht untersucht.
In [105] wird folgende Stabilitätsaussage für das Parareal Verfahren beschrieben.

Theorem 9. *Sei das Anfangswertproblem*

$$\dot{x}\left(t\right) = \mu x, \quad x\left(t_0\right) = x_0$$

sowie ein Parareal Verfahren mit grober Stufe G und feiner Stufe F gegeben. Sei sowohl G als auch F ein Einschrittverfahren und sowohl die grobe Stufe G als auch die feine Stufe F seien stabil für das gegebene Anfangswertproblem.
Dann ist das Parareal Verfahren mit einer fest vorgegebenen Anzahl von k_0 Iterationen für einen hinreichend große Anzahl m an Mikrozeitintervallen je Makrozeitintervall für das gegebene Anfangswertproblem stabil.

Beweis. Sei δT die Zeitschrittweite des Einschrittverfahrens der groben Stufe G und δt die Zeitschrittweite des Einschrittverfahrens der feinen Stufe F. Es bezeichne $R\left(z\right)$ die Stabilitätsfunktion der groben Stufe und $r\left(z\right)$ die Stabilitätsfunktion der feinen Stufe. Ferner sei $\bar{R} = R\left(\mu\delta T\right)^{N_G}$ mit $N_G = \frac{\Delta t}{\delta T}$ sowie $\bar{r} = r\left(\mu\delta t\right)^N$ mit $N = \frac{\Delta t}{\delta t}$. Auf Grund der Annahme der Stabilität der groben Stufe G und der feinen Stufe F gilt $\left|\bar{R}\right| \leq 1$ und $\left|\bar{r}\right| \leq 1$. Die Aussage folgt hiermit unmittelbar aus Theorem 1 aus [105]. □

Darüber hinaus besteht ein weiterer Stabilitätsbegriff. Dieser ist insbesondere nützlich, wenn die Anzahl N an Zeitschritten der feinen Stufe je Mikrozeitintervall fest ist oder auch wenn die Anzahl k an Parareal Iterationen nicht fest ist.

Definition 10. Seien ein Anfangswertproblem sowie ein Parareal Verfahren gegeben. Es bezeichne $N = \frac{\Delta T}{\Delta t}$ die Anzahl an Mikrointervallen je Makrozeitintervall. Gilt für das Parareal-Verfahren, dass für alle $N \in \mathbb{N}$ und $1 \leq k \leq N$ das Parareal Verfahren mit m Mikrozeitintervallen und k Iterationen stabil ist, so wird es für das Anfangswertproblem als *stark stabil* bezeichnet (vgl. [105]).
Ein hinreichendes Kriterium für starke Stabilität gibt der folgende Satz an.

Theorem 11. *Sei das Anfangswertproblem*

$$\dot{x}\left(t\right) = \mu x, \quad x\left(t_0\right) = x_0$$

mit $Re\left(\mu\right) \leq 0$ sowie ein Parareal Verfahren mit grober Stufe G und feiner Stufe F gegeben. Sei G ein Einschrittverfahren mit Zeitschrittweite δT und F ein Einschrittverfahren mit Zeitschrittweite δt. Sowohl die grobe Stufe G als auch die feine Stufe F seien stabil für das gegebene

Anfangswertproblem. Es bezeichne $R(z)$ die Stabilitätsfunktion der groben Stufe und $r(z)$ die Stabilitätsfunktion der feinen Stufe. Ferner sei $\bar{R} = R(\mu\delta T)^{N_G}$ mit $N_G = \frac{\Delta t}{\delta T}$ und $\bar{R} = \rho_R e^{i(\varphi+\sigma)}$ mit $\rho_R \in \mathbb{R}$ sowie $\bar{r} = r(\mu\delta t)^N$ mit $N = \frac{\Delta t}{\delta t}$ und $\bar{r} = x_r e^{i\varphi}$ mit $x_R \in \mathbb{R}$.

Dann ist das Parareal Verfahren für das gegebene Anfangswertproblem stabil, wenn

$$\rho_R \leq \frac{1-\rho_r}{2-2\rho_r \cos\sigma}$$

gilt.

Beweis. Theorem 3 in [105]. □

Zur Diskussion paralleler Verfahren sind die beiden Begriffe aus folgender Definition von Belang.

Definition 12. Der Speedup S eines parallelen Verfahrens wird definiert als

$$S = \frac{L_{ser}}{L_{par}},$$

wobei L_{par} die Rechenzeit des parallelen Verfahrens und L_{ser} die Rechenzeit des Verfahrens bei serieller Ausführung bezeichnet.

Die parallele Effizienz eines parallelen Verfahrens, das c Rechenkerne nutzt, sei definiert als

$$E_c = \frac{S_c}{c},$$

wobei S_n der zugehörige Speedup ist.

Im Hinblick auf den Einsatz des Parareal Verfahrens auf einem Fahrzeug-Steuergerät ist wie in Abschnitt 3.2.2 beschrieben der effiziente Umgang mit der durch einen Algorithmus in Anspruch genommenen Rechenkapazität eine Kernanforderung. Dies betrifft auch parallele Verfahren. Vor diesem Hintergrund ist es eine Anforderung an einen Algorithmus, dass die parallele Effizienz nahe bei 1 liegt.

Wie in Abschnitt 3.2.5 beschrieben, ist bei Fahrzeug-Steuergeräten eine Zunahme der Anzahl an Rechenkernen zu erwarten. Dies kann zu einer Verringerung der Knappheit der Rechenressource führen. Je weniger Bedeutung die Knappheit der Rechenressource besitzt, desto höher wird der Stellenwert des Speedups. Der Speedup kann bei einer Echtzeitsimulation dazu führen, dass ein Algorithmus, der bei serieller Ausführung nicht echtzeitfähig ist, bei einer parallelen Ausführung für den Echtzeiteinsatz geeignet ist. In vorliegender Arbeit wird die Auffassung vertreten, dass nach derzeitigem Stand die Anforderung an die Effizienz als vorrangig gegenüber dem möglichen Speedup zu erachten ist.

Es bezeichne L_G die Rechenzeit der groben Stufe G für die Integration über ein Mikrozeitintervall und L_F die Rechenzeit der feinen Stufe für die Integration über ein Mikrozeitintervall. Diese Rechenzeiten L_G und L_F werden im Folgenden als konstant angenommen. Außerdem werden die Kommunikationszeiten bei folgender Betrachtung nicht berücksichtigt. In [15, 13, 93] finden sich jeweils eine Betrachtung des Speedups des Parareal Verfahrens. Wie auch in diesen Quellen wird die Integration mittels der feinen Stufe F als das zum Parareal Verfahren zugehörige serielle Verfahren angesehen. Somit gilt bei einem Parareal Verfahren mit m Mikrozeitintervallen

$$\begin{aligned} S &= \frac{mL_F}{mL_G + k(L_F + mL_G)}, \\ S &= \frac{mL_F}{kL_F + (k+1)mL_G}. \end{aligned}$$

Hieraus ergibt sich wegen $k \geq 1$ die Abschätzung

$$S \leq \frac{mL_F}{2L_G + L_F},$$

die im Fall von nur einer Parareal Iteration scharf ist. Wird für die grobe und feine Stufe dasselbe Integrationsverfahren mit den Zeitschrittweiten $\delta T = \Delta t$ bei der groben Stufe und $\delta t = \Delta t/N$ eingesetzt, so ist die Annahme $L_F = L_G N$ mit $N = \frac{\Delta t}{\delta t}$ naheliegend. Hieraus folgt $S \leq N$. Wird ein Parareal Verfahren auf c Rechenkernen gerechnet, so ist es natürlich, $m = c$ Mikrozeitintervalle je Makrozeitintervall zu verwenden. In diesem Fall gilt für die parallele Effizienz

$$\begin{aligned} E_c &= \frac{S_c}{c}, \\ E_c &= \frac{mL_G N}{kL_G N + (k+1)\, mL_G} \frac{1}{c}, \\ E_c &= \frac{N}{kN + c\,(k+1)}. \end{aligned} \tag{6.2}$$

Es ist erkennbar, dass eine geringe Anzahl an Parareal Iterationen für eine hohe parallele Effizienz nötig ist. Die Effizienz lässt sich unter den gegebenen Bedingungen abschätzen durch

$$E_c = \frac{N}{kN + c\,(k+1)} < \frac{1}{k}. \tag{6.3}$$

Im Hinblick auf den Einsatz eines Parareal Verfahrens auf einem Fahrzeug-Steuergerät unter den Anforderungen wie in Abschnitt 3.2.2 beschrieben, ist somit eine geringe Anzahl an Rechenkernen und wegen $m = c$ dementsprechend auch eine geringe Anzahl an Mikrozeitintervallen je Makrozeitintervall nötig.

6.2 Wahl der Verfahrensparameter

Das Parareal Verfahren umfasst etliche Einstellmöglichkeiten wie beispielsweise die Anzahl N an Zeitschritten des feinen Verfahrens je Mikrozeitintervall, die maximale Anzahl k_0 an durchgeführten Parareal Iterationen und auch die Anzahl der Rechenkerne, die im Rahmen der Rechnung verwendet werden. In diesen Abschnitt wird die Wahl dieser Größen in einer einfach aufgebauten Untersuchung beispielhaft betrachtet.

6.2.1 Aufbau der Untersuchung

Folgende Voraussetzungen werden im Rahmen der Untersuchung angenommen: Sämtliche Intervallunterteilungen seien äquidistant. Weiter sollen sowohl die grobe als auch die feine Stufe des Parareal Verfahrens ein Einschrittverfahren erster Ordnung verwenden. Für die Zeitschrittweite δT der groben Stufe wird $\delta T = \Delta t$ angenommen, so dass die grobe Stufe je Mikrozeitintervall also genau einen Zeitschritt durchführt. Desweiteren entspreche die Anzahl m an Mikrozeitintervallen der Anzahl an verwendeten Rechenkernen, so dass also $m = c$ ist. Außerdem wird die Makrozeitschrittweite ΔT als gegeben angenommen. Dann verbleiben die drei oben beschriebenen Einstellmöglichkeiten.

Die folgenden Untersuchungen werden am Beispiel des Rohrmodells durchgeführt. Um eine geeignete Einstellung der Parameter des Verfahren zu ermitteln, wird der lokale Diskretisierungsfehler in Abhängigkeit des Rechenaufwandes bestimmt. Im Gegensatz zum Simulationsverlauf kommt beim lokalen Diskretisierungsfehler die Abhängigkeit der Lösung von den Eingangsdaten weniger zum Tragen.

In der Untersuchung wird derjenige Zeitschritt des Referenzverlaufs betrachtet, bei dem die Abweichung von der Referenztrajektorie bei einer Integration mit dem linear-impliziten Euler-Verfahren und einer Zeitschrittweite von 100 ms das Maximum annimmt.

Festhalten der Jacobi-Matrix Bei der folgenden Untersuchung wird die Jacobi-Matrix jeweils innerhalb des Makrozeitintervalls als konstant angenommen. Wird bei jedem Zeitschritt der groben und feinen Stufe eine Approximation der Jacobi-Matrix gebildet, so führt dies dazu, dass im Hinblick auf einen vertretbaren Rechenaufwand die Werte für die Anzahl N an Zeitschritten der feinen Stufe je Mikrozeitintervall sehr gering zu wählen sind. Durch ein Festhalten der Jacobi-Matrix fällt der Rechenaufwand wesentlich geringer aus und es sind größere Werte für die Anzahl N an Zeitschritten der feinen Stufe je Mikrozeitintervall möglich.

Fehlermaß Im Folgenden wird der relative Fehler in der Gastemperatur am Rohrausgang betrachtet. Als Referenzlösung dient hierbei eine Lösung, die mit einem linear-impliziten Euler-Verfahren und einer Zeitschrittweite von 0.0125 ms berechnet wird.

Maß für die Rechenzeit Die Verhältnisse der Rechenzeiten werden im vorliegenden Abschnitt beruhend auf heuristischen Überlegung rechnerisch ermittelt. Hierzu wird einerseits angenommen, dass die Zeit für die Kommunikation zwischen den Kernen im Rahmen des Parareal Verfahrens vernachlässigbar ist gegenüber der Rechenzeit eines Zeitschrittes mit dem verwendeten Einschrittverfahren. Andererseits wird angenommen, dass für die Rechenzeit L_F der Integration über ein Mikrozeitintervall mit der feinen Stufe F gegenüber der Rechenzeit L_G der Integration über ein Mikrozeitintervall mit der groben Stufe G gilt: $L_F = L_G N$.

Unter diesen Annahmen lassen sich die Rechenzeiten des Parareal Verfahrens in verschiedenen Einstellungen und für verschiedene Anzahlen an verwendeten Rechenkernen relativ zur Rechenzeit L_G angeben. Für die einmalige Bildung der Approximation der Jacobi-Matrix wird der Wert L_J berücksichtigt. Für den Fall, dass keine Vergröberung der Jacobi-Matrix oder ähnliche Methode zum Einsatz kommen, ist im Fall des Rohrmodells gemäß Abschnitt 4.1.3 die Approximation $L_J = 4L_G$ realistisch, da auf Grund des Festhaltens der Jacobi-Matrix bei einem Zeitschritt der groben Stufe lediglich eine Auswertung der rechten Seite sowie die Lösung eines linearen Gleichungssystems durchzuführen. Als Rechenaufwand oder Rechenzeit wird hierbei die Summe der Rechenzeiten des Parareal Verfahrens auf den einzelnen Rechenkernen bezeichnet. Die maximale Rechenzeit auf einem einzelnen Rechenkern wird Antwortzeit genannt.

6.2.2 Resultate und Diskussion

Wie in Abschnitt 6.1.1 beschrieben, ist im Hinblick auf eine hohe parallele Effizienz eine geringe Anzahl k_0 an Parareal Iterationen notwendig. In Abbildung 6.1 ist der Fehler in Abhängigkeit der Anzahl an Parareal Iterationen für verschiedene Anzahlen N an Zeitschritten der feinen Stufe je Mikrozeitintervall dargestellt. Hierbei ist die Anzahl Mikrozeitintervalle $m = 4$. Aus der Abbildung ist zu erkennen, dass im betrachteten Szenario für verschiedene Werte für die Anzahl N der Zeitschritte der feinen Stufe je Mikrozeitintervall bereits nach einer Parareal Iteration ein Fehler erreicht wird, der durch die weiteren Parareal Iterationen nicht weiter verringert wird. In Abbildung 6.2 ist der relative Fehler in Abhängigkeit der Anzahl an Parareal Iterationen für verschiedene Anzahlen m an Mikrozeitintervallen dargestellt. Hierbei ist $N = 10$ gewählt. Auch in diesen Fällen ist bereits nach einer Parareal Iteration ein Fehler erreicht, der durch die weiteren Iterationen nicht wesentlich verringert wird.

Insgesamt ist daher die Wahl von $k_0 = 1$, so dass also genau eine Parareal Iteration durchgeführt wird, im vorliegenden Fall sinnvoll. Hierdurch ergibt sich für das Parareal Verfahren in diesem

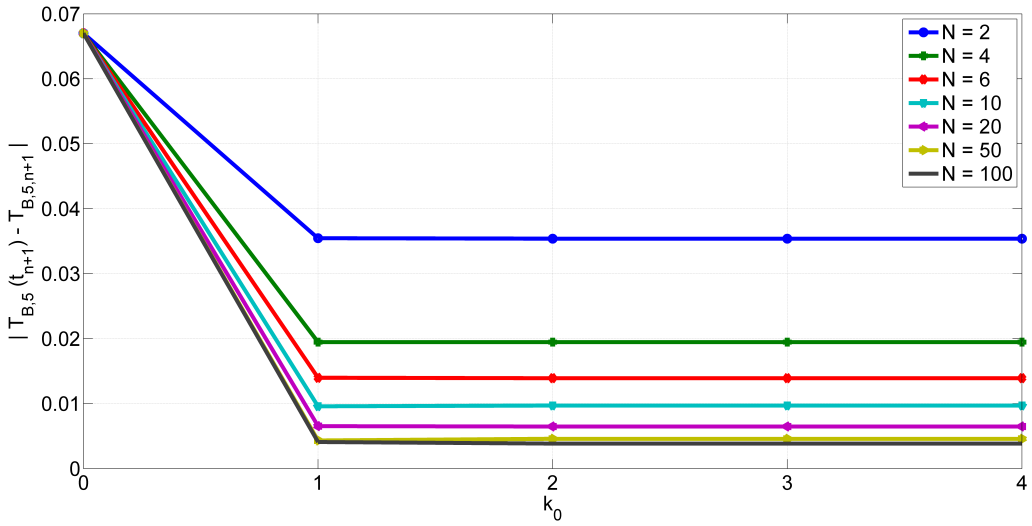


Abbildung 6.1: Absoluter Fehler eines Parareal Verfahrens in Abhängigkeit der Anzahl k_0 an Parareal Iterationen für verschiedene Anzahlen N von Zeitschritten der feinen Stufe je Mikrozeitintervall.

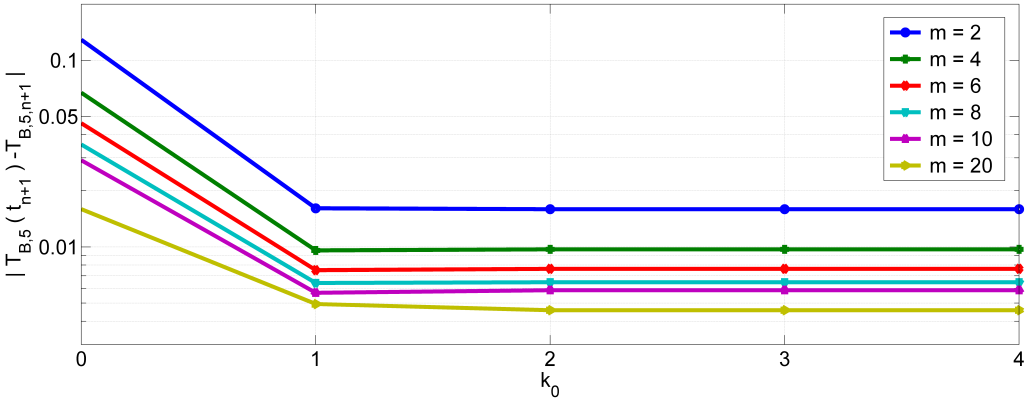


Abbildung 6.2: Absoluter Fehler eines Parareal Verfahrens in Abhängigkeit der Anzahl k_0 an Parareal Iterationen für verschiedene Anzahlen m an Mikrozeitintervallen je Makrozeitintervall.

Beispiel ein hoher Wert für die parallele Effizienz. Gemäß Gleichung 6.2 ist

$$E_c = \frac{N}{kN + c\,(k+1)} = \frac{100}{100 + 2c}.$$

Somit lässt sich für Werte von $c \leq 16$ für die Anzahl an Rechenkernen die parallele Effizienz abschätzen durch $0.75 < E_c < 1$. Werden wie in Abbildung 6.1 $m = 4$ Mikrozeitintervalle verwendet und gemäß der Annahme $m = c$ auf Seite 103 somit $c = 4$ Rechenkerne eingesetzt, so beträgt die parallele Effizienz etwa $E_c \approx 0.93$.

Aus Abbildung 6.1 ergibt sich zudem, dass auch kleine Werte für die Anzahl N an Zeitschritten der feinen Stufe je Mikrozeitintervall sinnvoll einsetzbar sind, da die resultierenden Werte für den Fehler des Parareal Verfahrens in verhältnismäßigem Maße korrelieren mit dem Wert von N. Bei dem Wert $N = 100$ ist in der vorliegenden Untersuchung hingegen keine angemessene Reduktion des Fehlers des Parareal Verfahrens gegeben gegenüber einem Parareal Verfahren, bei $N = 50$ verwendet wird.

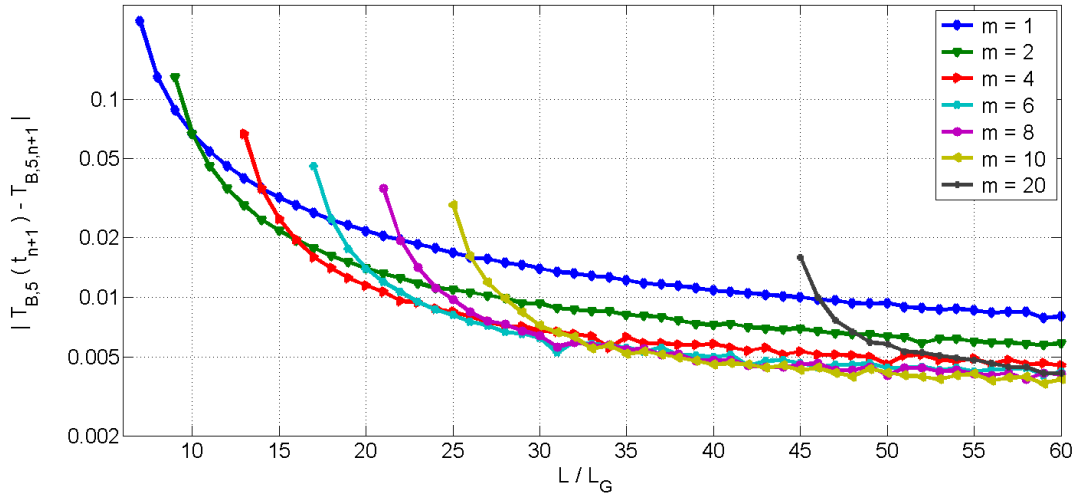


Abbildung 6.3: Absoluter Fehler des Parareal Verfahrens in Relation zur benötigten Antwortzeit L für verschiedene Anzahlen c an verwendeten Rechenkernen.

Wie auf Seite 102 beschrieben, ist beim Parareal Verfahren die Anforderung an die Effizienz vorrangig. Sofern diese geeignet ist, kann das Parareal Verfahren allerdings zu dem Zweck eingesetzt werden, die Antwortzeit eines Berechnungsaufgabe zu reduzieren. Im Hinblick auf diesen Einsatzzweck ist es von Interesse, die Parameter des Parareal Verfahrens einzustellen, dass sich eine optimale Antwortzeit ergibt.

In Abbildung 6.3 ist der Fehler in Abhängigkeit der benötigten Antwortzeit für verschiedene Werte für die Anzahl c an Rechenkernen gezeigt. Hierbei wird jeweils genau eine Parareal Iteration durchgeführt. Aus der Abbildung ist zu erkennen, dass je nach Anforderung an den Fehler ein unterschiedlicher Wert für die Anzahl m an Mikrozeitintervallen und auf Grund von $m = c$ somit auch an die verwendeten Rechenkerne beim Parareal Verfahren zweckmäßig ist. Desweiteren ist zu erkennen, dass die Genauigkeit bei einem Fehler von etwa $|T_{B,5}(t_{n+1}) - T_{B,5,n+1}| \approx 0.005$ mit zunehmender Antwortzeit nur noch sehr langsam zunimmt. Bemerkenswert ist, dass somit in dem Fehlerbereich, der bei genau einer Parareal Iteration erzielt werden kann, ohne dass eine sehr langsame Verringerung des Fehlers mit einer zunehmenden Antwortzeit vorliegt, ein Wert von $c \leq 10$ für die Anzahl an verwendeten Rechenkernen im vorliegenden Beispiel optimal ist.

Der Fall $m = 1$ ist sehr ähnlich zu einem linear-impliziten Euler-Verfahren dar. Es ist zu erkennen, dass die Konfigurationen des Parareal Verfahrens mit $m > 1$ eine deutliche Reduktion demgegenüber ermöglichen. Beispielsweise ist beim Fehlerniveau von $|T_{B,5}(t_{n+1}) - T_{B,5,n+1}| \approx 0.01$ durch die Wahl $m = 4$ eine Reduktion der Antwortzeit von $45\,L_G$ auf etwa $22\,L_G$ und somit um etwa die Hälfte möglich.

Aus Abbildung 6.3 wird insgesamt auch deutlich, dass das Parareal Verfahren auch bei einer geringen Anzahl Mikrozeitintervallen und Rechenkernen im vorliegenden Beispiel zweckmäßig einsetzbar ist, da eine angemessene Korrelation zwischen der Antwortzeit und der Verringerung des Fehlers besteht.

In Abbildung 6.4 ist die Antwortzeit des Parareal Verfahrens dargestellt, die nötig ist, um verschiedene vorgegebene Werte für den Fehler zu erfüllen, in Abhängigkeit der Anzahl m an verwendeten Mikrozeitintervallen. Hieraus ist zu erkennen, dass es für die betrachteten Vorgaben an den Fehler jeweils eine Anzahl m an Mikrozeitintervallen und auf Grund von $m = c$ somit auch an verwendeten Rechenkernen im Bereich $2 \leq m \leq 16$ gibt, die zu einem Minimum bei der Antwortzeit führt. Insbesondere ist zu erkennen, dass für große Werte für c die Antwortzeit zunehmend größer wird. Insgesamt geht aus der Abbildung hervor, dass auch bei einer geringen Anzahl an Rechenkernen bereits ein effizienter Einsatz des Parareal Verfahrens möglich ist.

Aus den vorangehenden Betrachtungen ergeben sich folgende Schlussfolgerungen:

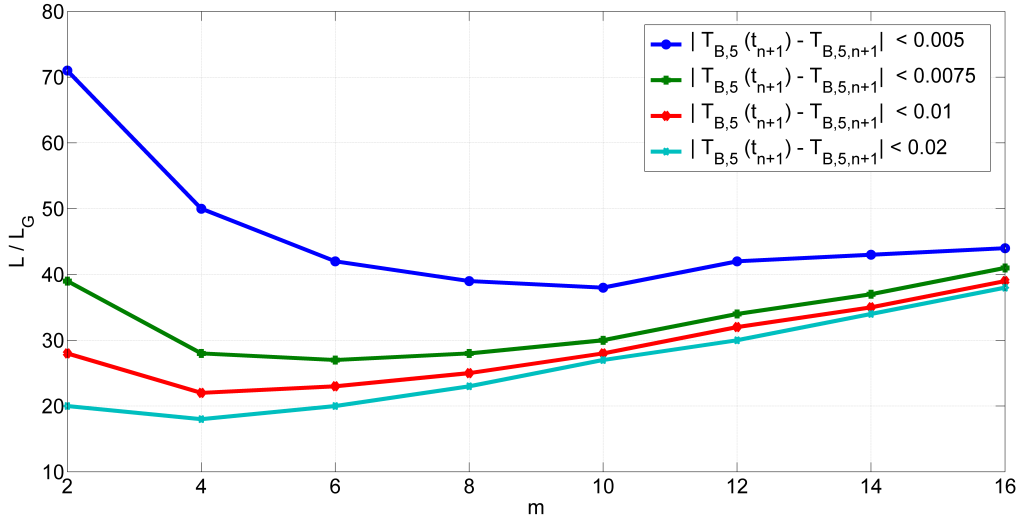


Abbildung 6.4: Antwortzeit L des Parareal Verfahren mit m Mikrozeitintervallen je Makrozeitintervall , bei dem die Anzahl N an Zeitschritten der feinen Stufe je Mikrozeitintervall minimal ist, so dass verschiedene vorgegebene Fehlerschranken eingehalten werden.

Im vorliegenden Beispiel führt bereits der Wert $k_0 = 1$ für die Anzahl an Parareal Iterationen zu geringen Werten für den Fehler und höhere Werte für k_0 verringern die Fehler nicht wesentlich. Somit liegt ein Beispiel dafür vor, dass das Parareal Verfahren bereits mit einer Parareal Iteration geeignete Resultate liefern und das Parareal Verfahren mit einem hohen Wert $E \approx 0.93$ für die parallele Effizienz eingesetzt werden kann.

Für die Anzahl N an Zeitschritten der feinen Stufe je Mikrozeitintervall führen in vorliegender Untersuchung kleine Werte zu einer verhältnismäßigen Korrelation zwischen dem Rechenaufwand und dem erzielten Fehler. Für Werte $N > 50$ verringert sich der Fehler hingegen nicht mehr in zweckmäßiger Weise. Somit ist im betrachteten Beispiel eine Wahl von $2 \leq N \leq 50$ sinnvoll.

Desweiteren ist festzustellen, dass sich für die Antwortzeit bei Vorgabe eines Fehlerschranke jeweils eine im Hinblick auf die Rechenzeit optimale Anzahl an Mikrozeitintervallen und dementsprechend auch eine optimale Anzahl an durch das Parareal Verfahren zu verwendenden Rechenkernen erzielen lässt, die im vorliegender Untersuchung bei $2 \leq m \leq 16$ liegt. Somit wird deutlich, dass das Parareal Verfahren auch bereits mit einer geringen Anzahl an Rechenkernen zweckmäßig eingesetzt werden kann.

Außerdem ist festzustellen, dass das Parareal Verfahren zu einer signifikanten Reduktion der Antwortzeit im betrachteten Beispiel führen kann.

6.3 Kombination des Parareal Verfahrens mit weiteren Ansätzen

Wie auf Seite 102 erwähnt, ist ein geringer Rechenaufwand im Sinne einer geringen Summe von Rechenzeiten auf den verschiedenen Kernen als eine wesentliche Anforderung an ein Integrationsverfahren vor dem Hintergrund einer limitierten Rechenkapazität eines Steuergeräts anzusehen. Wie im Abschnitt 6.2 beschrieben, besitzt das Parareal Verfahren insbesondere im Fall, dass nur eine Parareal Iteration durchgeführt wird, eine deutlich höhere parallele Effizienz als im Fall $k_0 > 1$. Im Sinne der Effizienz ist es zudem eine Anforderung an ein Verfahren, das wie das Parareal Verfahren eine Parallelisierung ermöglicht, dass der Rechenaufwand im Fall einer parallelisierten Berechnung nur geringfügig höher ist als eine Berechnung ohne Parallelisierung. Unter den gegebenen Rahmenbedingungen besteht das Ziel nun darin, dass durch eine Parallelisierung

der Integration durch das Parareal Verfahren der Gesamtrechenaufwand nur geringfügig höher ist als bei der Integration ohne das Parareal Verfahren.

Ein wesentlicher Faktor, der zu einer Vergrößerung des Gesamtrechenaufwands bei einem Parareal Verfahren mit einer Parareal Iteration führt, ist der Rechenaufwand der groben Stufe. Um diesen zu verringern, werden im Folgenden zwei Ansätze beschrieben, im Rahmen der groben Stufe Rechenzeitreduktionen vorzunehmen.

6.3.1 Kombination mit einer vergröberten Jacobi-Matrix

Bei Verwendung einer W Methode wie dem linear-impliziten Euler-Verfahren als Integrator ist das Parareal Verfahren gut kombinierbar mit der in Kapitel 4 beschriebenen Methode der Vergröberung der Jacobi-Matrix. Diese ist insbesondere für den Einsatz in der groben Stufe prädestiniert. Hinsichtlich der Konvergenzordnung der groben Stufe führt dies wie bereits in Kapitel 4 beschrieben dadurch, dass es sich um eine W Methode handelt, zu keiner Reduktion. Aus Satz 7 geht hervor, dass hierdurch auch die Konsistenzordnung des Parareal Verfahrens nicht vermindert wird.

Festhalten der Jacobi-Matrix Wie in Kapitel 4 beschrieben, stellt das sogenannte Festhalten der Jacobi-Matrix einen Spezialfall einer Vergröberung einer Jacobi-Matrix dar. Eine naheliegende Anwendung auf das Parareal Verfahren besteht darin, die Jacobi-Matrix jeweils innerhalb eines Makrozeitschrittes festzuhalten, wie es auch bereits in Abschnitt 6.2 getan wurde. Durch die kurze Dauer eines Zeitschrittes ist es plausibel, dass dies zu geeigneten Resultaten führt. Da die Approximation der Jacobi-Matrix beim Rohrmodell einen wesentlichen Anteil des Rechenaufwands eines Zeitschrittes verursacht, wird durch das Festhalten der Jacobi-Matrix der Rechenaufwand eines Parareal Verfahrens signifikant reduziert. Die Eignung dieses Vorgehen wurde in Abschnitt 6.2 betrachtet und bestätigt.

Vergröberung ohne Festhalten der Jacobi-Matrix Der Vollständigkeit halber wird zudem der Fall betrachtet, dass ein Festhalten der Jacobi-Matrix nicht eingesetzt wird. In diesem Fall ist somit für jeden Zeitschritt der groben und feinen Stufe die Bildung einer Approximation der Jacobi-Matrix nötig. Folglich ist im Hinblick auf den Rechenaufwand die Anzahl an Zeitschritten der groben und feinen Stufe je Makrozeitschritt zu wählen.

Die Bestimmung einer geeigneten Jacobi-Matrix-Vergröberung für das Rohrmodell wurde in Kapitel 4 durchgeführt. In Abbildung 6.5 ist die betragsmäßige Abweichung in der Gastemperatur am Rohrausgang bei der Referenztrajektorie für Parareal Verfahren dargestellt, die für die grobe Stufe die in Kapitel 4 erarbeitete Jacobi-Matrix-Vergröberung verwenden, gegenüber Parareal Verfahren, die für grobe und die feine Stufe keine Jacobi-Matrix-Vergröberung einsetzen.

Hierbei gilt für die Anzahl N an Integrationsschritten der feinen Stufe je Mikrozeitintervall $N = 1$ oder $N = 4$. Es werden $m = 6$ Mikrozeitintervalle bei der Simulation verwendet, so dass das Parareal Verfahren für den Einsatz auf sechs Rechenkernen geeignet ist. Im Fall $N = 1$ wird somit in einem Simulationszeitschritt sechs Integrationszeitschritte mit der feinen Stufe durchgeführt. Für die Anwendung auf einem Fahrzeug-Steuergerät ist diese Anzahl im Hinblick auf einen geringen Rechenaufwands klein zu wählen. Der Fall $N = 1$ ist hierdurch eine günstige Wahl. Bei $N = 4$ werden bereits $mN = 24$ Integrationszeitschritte der feinen Stufe je Simulationszeitschritt durchgeführt, was im Hinblick auf den Rechenaufwand als hoch einzustufen ist.

In Abbildung 6.5 ist zu erkennen, dass in beiden Fällen der Fehler beim Simulationsverlauf in sehr geringem Maß beeinflusst wird. Die Änderung des Fehlers ist bei $N = 4$ deutlich höher als bei $N = 1$, so dass denkbar ist, dass bei größeren Werten für N das Verwenden der Jacobi-Matrix-Vergröberung in der groben Stufe relevanten Fehlern führen kann. Für die relevanten Werte von N ist allerdings der Einsatz einer Jacobi-Matrix-Vergröberung in der groben Stufe des Parareal Verfahrens im Falle des Rohrmodells gut geeignet.

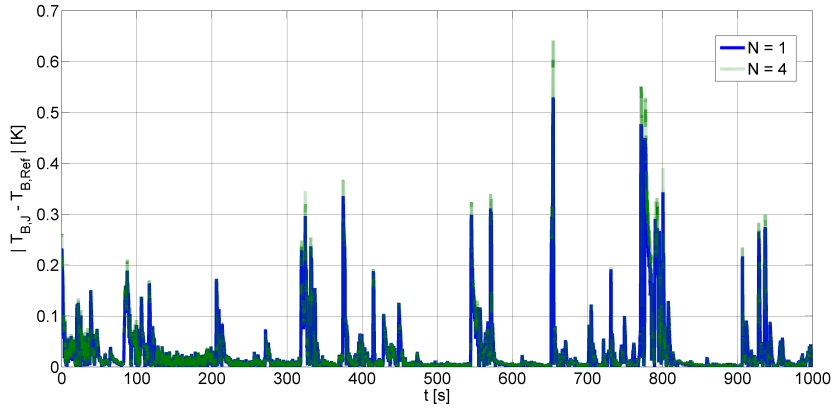


Abbildung 6.5: Verlauf der betragsmäßigen Abweichung der Gastemperatur am Rohrausgang durch Verwendung eines Parareal Verfahrens mit $m = 6$ Mikrozeitintervallen je Makrozeitschritt und einer Jacobi-Matrix-Vergröberung gemäß Variante 5 in Abschnitt 4.4 gegenüber einem linear-impliziten Euler-Verfahren mit Schrittweite 1 ms.

6.3.2 Kombination mit einer Reduktion der Linearisierung der rechten Seite

In [72] ist beschrieben, wie für die grobe Stufe eines Parareal Verfahrens ein reduziertes Modell eingesetzt wird. Am Beispiel eines steifen linearen Modells einer chemischen Reaktion wird dort eine in [18] beschriebene Reduktion des Modells eingesetzt. Das reduzierte System besitzt gerade die langsamen Eigenwerte des originalen Systems und ist von entsprechend geringerer Größe. Durch einen algebraischen Term wird nach der Lösung des reduzierten Systems für die langsam reagierenden Spezien die Auswirkung auf die übrigen Spezien berechnet. Dieser Ansatz wird im Weiteren auf die Leitanwendung vorliegender Arbeit übertragen.

Für das in vorliegender Arbeit untersuchte Rohrmodell gilt, dass sich die Zustände wie in Kapitel 4.2.3 ermittelt in zwei Gruppen unterteilen lassen: Die Gasfraktionen sowie die Gastemperaturen im Rohr sind von den schnellen Anteilen in der Dynamik des Modells betroffen, während auf die Rohrtemperaturen lediglich die langsamen Anteile der Dynamik des Modells wirken. Aus diesem Grund wird der Ansatz verfolgt, im Rahmen der groben Stufe des Parareal Verfahrens das Teilsystem, das lediglich die Zustände umfasst, die durch die schnellen Anteile der Dynamik betroffen sind, zu simulieren. Die langsame Dynamik der übrigen Zustände wird in der groben Stufe vernachlässigt. Hierbei werden die Einflüsse des Verlaufs der Zustände mit der schnellen Dynamik auf die Zustände mit der langsamen Dynamik in einer Weise berücksichtigt, wie es aus Gleichung 6.8 hervorgeht.

Der zu Grunde liegende Gedanke ist, dass über einen kurzen Zeitraum hinweg die schnellen Anteile der Dynamik zu einer adäquaten Beschreibung der Dynamik eines Systems ausreichen und hierdurch der Rechenaufwand für die grobe Stufe des Parareal Verfahrens in geeigneter Weise verringert werden kann.

Hierzu wird für

$$\dot{x}\,(t) = Ax \tag{6.4}$$

folgendermaßen vorgegangen:

Es wird ein Basiswechsel mit der Matrix T verwendet, durch den A Blockdreiecksform erhält:

$$U^{-1}AU = \begin{pmatrix} V & M \\ 0 & W \end{pmatrix}. \tag{6.5}$$

Sei $Ux = \begin{pmatrix} u \\ v \end{pmatrix}$. Hierbei werde U derart gewählt, dass $\mathrm{Re}\lambda_i^{(W)} < \mathrm{Re}\lambda_j^{(V)}$ für alle i, j, wobei

$\left\{\lambda_i^{(V)}\right\}$ und $\left\{\lambda_i^{(W)}\right\}$ die Eigenwerte von V beziehungsweise W seien. Somit beschreibt W die schnellen Anteile der Dynamik des Systems 6.4 und V die langsamen Anteile.

Eine solche Transformation liefert beispielsweise der QR-Algorithmus mit einer entsprechenden Permutation der Diagonaleinträge (vgl. Abschnitt 4.2.3). Die Matrix M stellt hierbei die Abhängigkeiten zwischen den Zuständen mit der schnellen Dynamik und den übrigen Zustände dar.

In der Theorie lässt sich jede Matrix A durch einen entsprechenden Basiswechsel sogar auf eine Blockdiagonalform mit einer Verteilung der Eigenwerte auf die Diagonalblöcke wie oben beschrieben transformieren. Eine solche Transformation kann unter numerischen Gesichtspunkten allerdings sehr ungünstig sein: Beispielsweise ist in vielen Fällen die Berechnung der Jordanschen Normalform, die obige Eigenschaften besitzt, numerisch ungünstig (vgl. [54]). Im Folgenden wird angenommen, dass für eine Matrix A eine numerische stabile Transformation auf eine Form wie in Gleichung 6.5 vorliegt.

Hierbei wird nun der Ansatz betrachtet, dass für die grobe Stufe des Parareal Verfahrens nicht das System 6.4 gelöst wird, sondern stattdessen das einfachere System

$$U\dot{v}\left(t\right) = WUv, \quad v\left(t_n\right) = v_n. \tag{6.6}$$

Folglich ist

$$G\left(t_{n,i+1}, t_{n,i}, x_{n,i}^{(k+1)}\right) = U\begin{pmatrix} u_n + Mv_{n+1} \\ v_{n+1} \end{pmatrix}. \tag{6.7}$$

Letztlich löst die grobe Stufe also das System

$$\dot{x}\left(t\right) = Bx \quad B = U\begin{pmatrix} 0 & M \\ 0 & W \end{pmatrix}U^{-1}. \tag{6.8}$$

Wird jeweils ein linear-implizites Euler-Verfahren in der groben Stufe und der feinen Stufe verwendet, so führt dieses Vorgehen zu einem konsistenten Parareal Verfahren. Auch für Systeme, bei denen $\|M\|$ groß ist und die Abhängigkeit zwischen den Zuständen, die von den schnellen Anteilen der Dynamik des Systems betroffen sind, und den übrigen Zuständen stark ausgeprägt ist, beeinträchtigt die Vernachlässigung der Matrix V in der groben Stufe des Parareal Verfahrens dessen Konvergenzordnung nicht. Dies wird im Folgenden gezeigt. Hierfür wird ein Nachweis gewählt, der zu einem wesentlichen Teil im Nachrechnen besteht und somit einfach ist, aber eine längliche Darstellung besitzt. Hierbei werden die folgenden Lemmata verwendet.

Korollar 13. *Für die Lösung eines Anfangswertproblems wie in Theorem 16 gilt folgende Aussage:*

Sei $r\left(z\right)$ die Stabilitätsfunktion der feinen Stufe und $R\left(z\right)$ die Stabilitätsfunktion der groben Stufe. Ferner sei

$$H\left(k,i,r,R,z_1,z_2\right) = \sum_{j=0}^{k} \sum_{\left\{\left(b_l\right)_{1\leq l\leq n}\in\{0,1\}^n \middle| \sum_l b_l = j\right\}} \prod_{l=1}^{m} \left(r\left(z_1\right)^N - R\left(z_2\right)\right)^{b_l} R\left(z_2\right)^{1-b_l}.$$

Dann ist der Wert $\lambda_{0,i}^k$ des Parareal Verfahrens für den i-ten Mikrozeitschritt in der k-ten Parareal Iteration

$$\lambda_{n,i}^k = H\left(k,i,r,R,\delta tA,\Delta tB\right)x_n.$$

Beweis. Die Behauptung folgt analog zur Herleitung von Gleichung (6) in Abschnitt 3.2 von Paper I in [105], wenn keine Kommutativität angenommen und dementsprechend die Umformung durch die binomische Formel nicht vorgenommen wird. □

Lemma 14. *Für die Matrix T gelte $\|T\| < 1$. Dann konvergiert $\sum_{j=0}^{\infty} T^j$ und es ist*

$$\sum_{j=0}^{\infty} T^j = (I - T)^{-1}.$$

Beweis. Die Behauptung folgt aus Satz 4.4 in [8]. □

Lemma 15. *Seien B, C und E Matrizen sowie B und C regulär. Dann gilt*

$$\begin{pmatrix} B & E \\ 0 & C \end{pmatrix}^{-1} = \begin{pmatrix} B^{-1} & -B^{-1}EC^{-1} \\ 0 & C^{-1} \end{pmatrix}.$$

Beweis. Auf Grund von

$$\begin{pmatrix} B & E \\ 0 & C \end{pmatrix} \begin{pmatrix} B^{-1} & -B^{-1}EC^{-1} \\ 0 & C^{-1} \end{pmatrix} = \begin{pmatrix} I & 0 \\ 0 & I \end{pmatrix}$$

gilt die Behauptung. □

Theorem 16. *Betrachte das stabile Anfangswertproblem*

$$\dot{x}(t) = Ax, \quad x(t_n) = x_n.$$

Sei

$$U^{-1}AU = \begin{pmatrix} V & M \\ 0 & W \end{pmatrix} = W \text{ und } B = U \begin{pmatrix} 0 & M \\ 0 & W \end{pmatrix} U^{-1}.$$

Sei ein Parareal Verfahren gegeben, dessen feine Stufe F das implizite Euler-Verfahren mit der Zeitschrittweite δt verwendet und dessen grobe Stufe G mit dem impliziten Euler-Verfahren mit der Zeitschrittweite Δt das System

$$\dot{x}(t) = Bx, \quad x(t_0) = x_0.$$

integriere. Weiter gelte für die Anzahl der Parareal Iterationen $k_0 \geq 1$ und es seien die Anzahl m der Mikrozeitintervalle sowie die Anzahl N der Zeitschritte der feinen Stufe je Mikrozeitintervall fest gewählt. Es bezeichne $\lambda_{n,i}$ die durch das Parareal Verfahren berechnete Lösung für den Zeitpunkt $T_n + i\Delta t$ nach der k-ten Parareal Iteration. Dann gilt:

$$\lambda_{n+1,m} = (I + \Delta T A)\,\lambda_{n,m} + \mathcal{O}\left(\Delta T^2\right).$$

Insbesondere besitzt das Parareal Verfahren die Konsistenzordnung 1.

Beweis. Sei $r(z)$ die Stabilitätsfunktion der feinen Stufe und $R(z)$ die Stabilitätsfunktion der groben Stufe. Da beide Stufen das implizite Euler-Verfahren verwenden gilt $r(z) = R(z) = (I - z)^{-1}$. Sei $I_{j,m} = \left\{ (b_l)_{1 \leq l \leq m} \in \{0,1\}^m \,\middle|\, \sum_l b_l = j \right\}$. Gemäß Lemma 13 gilt für den berechneten Wert $\lambda_{n,i}^k$ des Parareal Verfahrens für das betrachtete Anfangswertproblem

$$\lambda_{n,i}^k = H(k, i, r, R, \delta tA, \Delta tB)\, x_n$$

mit

$$H(k, i, r, R, \delta tA, \Delta tB) = \sum_{j=0}^{k} \sum_{(b_l)_l \in I_{j,m}} \prod_{l=1}^{m} \left(r(\delta tA)^N - R(\Delta tB) \right)^{b_l} R(\Delta tB)^{1-b_l}.$$

Sei $H_0 = H\left(k, i, r, R, \delta t A, \Delta t B\right)$. Dann ist

$$H_0 = \sum_{j=0}^{k} \sum_{(b_l)_l \in I_{j,m}} \prod_{l=1}^{m} \left((I - \delta t A)^{-N} - (I - \Delta t B)^{-1} \right)^{b_l} \left((I - \Delta t B)^{-1} \right)^{1-b_l}.$$

Weiter gilt auf Grund der Linearität der Konjugation mit U

$$U^{-1} H_0 U = \sum_{j=0}^{k} \sum_{(b_l)_l \in I_{j,m}} \prod_{l=1}^{m} \left(\left(I - \delta t \begin{pmatrix} V & M \\ 0 & W \end{pmatrix} \right)^{-N} - \left(I - \Delta t \begin{pmatrix} 0 & M \\ 0 & W \end{pmatrix} \right)^{-1} \right)^{b_l} \left(I - \Delta t \begin{pmatrix} 0 & M \\ 0 & W \end{pmatrix} \right)^{-(1-b_l)}.$$

Sei $U^{-1} H_0 U = \begin{pmatrix} L_1 & L_2 \\ L_3 & L_4 \end{pmatrix}$ und

$$S = \left(I - \delta t \begin{pmatrix} V & M \\ 0 & W \end{pmatrix} \right)^{-N} - \left(I - \Delta t \begin{pmatrix} 0 & M \\ 0 & W \end{pmatrix} \right)^{-1}$$

sowie $S = \begin{pmatrix} S_1 & S_2 \\ S_3 & S_4 \end{pmatrix}$. Mit Lemma 15 folgt

$$S_1 = (I - \delta t V)^{-N} - I = N \delta t V + \mathcal{O}\left(\delta t^2\right),$$

$$S_3 = 0,$$

$$S_4 = (I - \delta t W)^{-N} - (I - \Delta t W)^{-1} = \mathcal{O}\left(\delta t^2\right).$$

Folglich ist

$$L_1 = \sum_{j=0}^{k} \sum_{(b_l)_l \in I_{j,m}} \prod_{l=1}^{m} \left((I - \delta t V)^{-N} - I \right)^{b_l},$$

$$L_3 = 0,$$

$$L_4 = \sum_{j=0}^{k} \sum_{(b_l)_l \in I_{j,m}} \prod_{l=1}^{m} \left((I - \delta t W)^{-N} - (I - \Delta t W)^{-1} \right)^{b_l} (I - \Delta t W)^{-(1-b_l)}.$$

Mit Lemma 15 folgt außerdem

$$\left(I - \Delta t \begin{pmatrix} 0 & M \\ 0 & W \end{pmatrix} \right)^{-1} = \begin{pmatrix} I & -\Delta t M (I - \Delta t W)^{-1} \\ 0 & (I - \Delta t W)^{-1} \end{pmatrix},$$

$$\left(I - \Delta t \begin{pmatrix} 0 & M \\ 0 & W \end{pmatrix} \right)^{-1} = \begin{pmatrix} I & -\Delta t M \\ 0 & (I - \Delta t W)^{-1} \end{pmatrix} + \mathcal{O}\left(\Delta t^2\right) \tag{6.9}$$

und ebenso

$$\left(I - \delta t \begin{pmatrix} V & M \\ 0 & W \end{pmatrix} \right)^{-1} = \begin{pmatrix} (I - \delta t V)^{-1} & -\delta t (I - \delta t V)^{-1} M (I - \delta t W)^{-1} \\ 0 & (I - \delta t W)^{-1} \end{pmatrix},$$

$$\left(I - \delta t \begin{pmatrix} V & M \\ 0 & W \end{pmatrix} \right)^{-1} = \begin{pmatrix} (I - \delta t V)^{-1} & -\delta t M \\ 0 & (I - \delta t W)^{-1} \end{pmatrix} + \mathcal{O}\left(\delta t^2\right).$$

Aus Lemma 14 folgt

$$\begin{pmatrix} (I-\delta tV)^{-1} & -\delta tM \\ 0 & (I-\delta tW)^{-1} \end{pmatrix}^N = \begin{pmatrix} I+\delta tV & -\delta tM \\ 0 & I+\delta tW \end{pmatrix}^N + \mathcal{O}\left(\delta t^2\right),$$

$$\begin{pmatrix} (I-\delta tV)^{-1} & -\delta tM \\ 0 & (I-\delta tW)^{-1} \end{pmatrix}^N = \begin{pmatrix} (I+\delta tV)^N & -N\delta tM \\ 0 & (I+\delta tW)^N \end{pmatrix}^N + \mathcal{O}\left(\delta t^2\right),$$

$$\begin{pmatrix} (I-\delta tV)^{-1} & -\delta tM \\ 0 & (I-\delta tW)^{-1} \end{pmatrix}^N = \begin{pmatrix} (I-\delta tV)^{-N} & -N\delta tM \\ 0 & (I-\delta tW)^{-N} \end{pmatrix} + \mathcal{O}\left(\delta t^2\right). \quad (6.10)$$

Aus Gleichung 6.9 und Gleichung 6.10 ergibt sich

$$S_2 = -N\delta t\epsilon M_e - (-\Delta t\epsilon M_e) + \mathcal{O}\left(\delta t^2\right),$$

$$S_2 = \mathcal{O}\left(\delta t^2\right).$$

Dann gilt insgesamt

$$U^{-1}H_0U = \sum_{j=0}^{k} \sum_{(b_l)_l \in I_{j,m}} \prod_{l=1}^{m} \begin{pmatrix} N\delta tV & 0 \\ 0 & \mathcal{O}\left(\delta t^2\right) \end{pmatrix}^{b_l} \begin{pmatrix} I & -\Delta tM \\ 0 & (I-\Delta tW)^{-1} \end{pmatrix}^{1-b_l} + \mathcal{O}\left(\delta t^2\right).$$

Es ist

$$\sum_{j=0}^{0} \sum_{(b_l)_l \in I_{j,m}} \prod_{l=1}^{m} \begin{pmatrix} N\delta tV & 0 \\ 0 & \mathcal{O}\left(\delta t^2\right) \end{pmatrix}^{b_l} \begin{pmatrix} I & -\Delta tM \\ 0 & (I-\Delta tW)^{-1} \end{pmatrix}^{1-b_l} = -m\Delta tM$$

sowie

$$\sum_{j=1}^{k} \sum_{(b_l)_l \in I_{j,m}} \prod_{l=1}^{m} \begin{pmatrix} N\delta tV & 0 \\ 0 & \mathcal{O}\left(\delta t^2\right) \end{pmatrix}^{b_l} \begin{pmatrix} I & -\Delta tM \\ 0 & (I-\Delta tW)^{-1} \end{pmatrix}^{1-b_l} = \mathcal{O}\left(\delta t^2\right).$$

Somit ist

$$L_2 = -m\Delta tM + \mathcal{O}\left(\delta t^2\right).$$

Betrachte die Lösung des Anfangswertproblems

$$\dot{y}(t) = Vy, \quad y(t_0) = y_0$$

durch ein Parareal Verfahren, dessen feine Stufe ein implizites Euler-Verfahren mit der Schrittweite δt verwendet und dessen grobe Stufe die Zustände unverändert lässt und die Schrittweite Δt besitzt. Bei diesem Parareal Verfahren ist

$$L_1 = H(k, i, S, R, \delta tV, \Delta tI, \Delta tI).$$

Die Matrix L_1 lässt sich also interpretieren als die Matrix $H(k, i, S, R, \delta tV, \Delta tI, \Delta tI)$ eines Parareal Verfahrens. Auf Grund von Korollar 8 besitzt dieses Parareal Verfahren wegen $k_0 \geq 1$ die Konsistenzordnung $p = 1$. Somit folgt

$$L_1 = I + i\Delta tV + \mathcal{O}\left(\Delta t^2\right),$$

$$L_1 = (I - i\Delta tV)^{-1} + \mathcal{O}\left(\Delta t^2\right).$$

Betrachte die Lösung des Anfangswertproblems

$$\dot{z}(t) = Wz, \quad z(t_0) = z_0$$

durch ein Parareal Verfahren, dessen feine Stufe ein implizites Euler-Verfahren mit der Schrittweite δt und dessen grobe Stufe ein implizites Euler-Verfahren mit der Schrittweite Δt verwendet. Bei diesem Parareal Verfahren ist

$$L_4 = H\left(k, i, S, R, \delta tW, \Delta tW, \Delta tW\right).$$

Die Matrix L_1 lässt sich also interpretieren als die Matrix $H\left(k, i, S, R, \delta tW_2, \Delta tW_2, \Delta tW_2\right)$ eines Parareal Verfahrens. Auf Grund von Satz 7 besitzt dieses Parareal Verfahren eine Konsistenzordnung $p = 1$. Somit folgt

$$L_4 = I + i\Delta tW + \mathcal{O}\left(\Delta t^2\right),$$
$$L_4 = \left(I - i\Delta tW\right)^{-1} + \mathcal{O}\left(\Delta t^2\right).$$

Insgesamt gilt für $i = m$ also

$$\lambda_{n,m} = U^{-1} \begin{pmatrix} \left(I - m\Delta tV\right)^{-1} & -\Delta TM \\ 0 & \left(I - m\Delta tW\right)^{-1} \end{pmatrix} Ux_n + \mathcal{O}\left(\Delta t^2\right),$$

$$\lambda_{n,m} = U^{-1} \begin{pmatrix} I + m\Delta tV & -\Delta TM \\ 0 & I + m\Delta tW \end{pmatrix}^{-1} Ux_n + \mathcal{O}\left(\Delta t^2\right),$$

$$\lambda_{n,m} = \left(I - \Delta TA\right)^{-1} x_n + \mathcal{O}\left(\Delta t^2,\right)$$

$$\lambda_{n,m} = I + \Delta tTAx_n + \mathcal{O}\left(\Delta t^2\right).$$

Hieraus folgt die Behauptung. □

Im Fall eines Anfangswertproblems

$$\dot{x}\left(t\right) = f\left(t, x\right), \quad x\left(t_0\right) = x_0$$

mit einer nichtlinearen rechten Seite f lässt sich der Ansatz auf das System

$$\dot{x}\left(t\right) = J_f\left(t_n, x_n\right)\left(x\right)$$

anwenden. Im Folgenden wird der Ansatz, für die grobe Stufe des Parareal Verfahrens ein System von reduzierter Größe zu lösen, auf das Rohrmodell angewandt. Wie aus Abschnitt 4.3.1 hervorgeht, sind die Rohrtemperaturen von den langsamen Anteile der Dynamik des Modells betroffen, während die übrigen Zustände auch von schnellen Anteilen der Dynamik des Modells betroffen sind.

Das Rohrmodell besitzt eine nichtlineare rechte Seite. Bei einem autonomen linearen System können die Transformationsmatrix T sowie Matrizen W und M aus Gleichung 6.5 vor der Laufzeit ermittelt werden. Im Fall einer nichtlinearen rechten Seite sind die Matrizen M und W nicht über die Zeitschritte hinweg konstant. Darüber hinaus kann es zudem notwendig sein, für verschiedene Zeitschritte unterschiedliche Transformationsmatrizen T_n einzusetzen. Im Fall des Rohrmodells liegt bereits ohne Transformation, also mit $T = I$ eine geeignete Form vor, da die Mixed Mode Integration bei der Referenztrajektorie geeignete Resultate liefert (vgl. Abschnitt 4.3.1) und somit der entsprechende Nebendiagonalblock der Jacobi-Matrizen der Referenztrajektorie vernachlässigt werden kann.

Zur Ermittlung der Matrizen W und M in einem gegebenen Zeitschritt genügt es beim Rohrmodell daher, die entsprechenden Spalten der Jacobi-Matrix zu berechnen, die für die zugehörigen Zeitschritt der feinen Stufe ohnehin zu berechnen sind.

In Abbildung 6.6 ist die betragsmäßige Abweichung gegenüber einem linear-impliziten Euler-Verfahren mit einer Zeitschrittweite von 1 ms in der Gastemperatur am Rohrausgang bei der Referenztrajektorie für Parareal Verfahren dargestellt, die für die grobe Stufe eine entsprechende

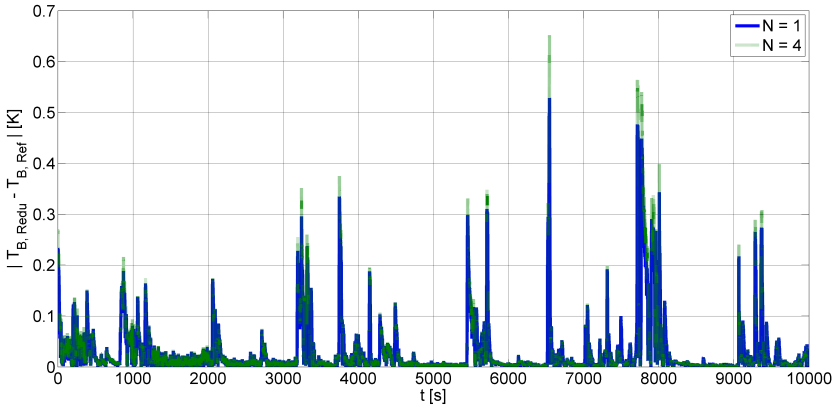


Abbildung 6.6: Verlauf der betragsmäßigen Abweichung der Gastemperatur am Rohrausgang durch Verwendung einer Vereinfachung in der groben Stufe gemäß den Gleichungen 6.6, 6.7 und 6.8 gegenüber einem linear-impliziten Euler-Verfahren mit einer Zeitschrittweite von 1 ms.

Vereinfachung verwenden. Wie in Abschnitt 6.3.1 werden auch hierbei für die Anzahl an Integrationsschritten der feinen Stufe des Parareal Verfahrens je Mikrozeitintervall die Werte $N = 1$ und $N = 4$ betrachtet, wobei $N = 1$ als im Hinblick auf einen geringen Rechenaufwand bei Einsatz des Verfahrens auf einem Fahrzeug-Steuergerät als günstige Wahl einzustufen ist und die Wahl $N = 4$ bereits einen relativ hohen Wert darstellt. Es ist zu erkennen, dass im Fall $N = 1$ die Verwendung des reduzierten Modells ohne signifikanten Einfluss auf den Simulationsverlauf ist. Es ist anzumerken, dass die Gegenüberstellung mit dem Verlauf des linear-impliziten Euler-Verfahrens hierbei nicht dazu dient, den Fehler der Parareal Verfahren zu bestimmen, sondern vielmehr dazu, die Größenordnung der Auswirkung einer Verwendung der betrachteten Vereinfachung im Rahmen eines Parareal Verfahrens auf den Simulationsverlauf einzuordnen.

Insgesamt ist für das Rohrmodell somit der Einsatz eines wie vorangehend beschriebenen reduzierten Modells in der groben Stufe eines Parareal Verfahrens bei einer geringen Anzahl an Mikrozeitintervallen und geringen Werten für $N \leq 4$ gut geeignet.

6.4 Eignung für Anwendungen auf Fahrzeug-Steuergeräten

Mathematische Modelle kommen derzeit vorrangig bei modellbasierter Regelung und Diagnose sowie bei virtuellen Sensoren zum Einsatz (vgl. Abschnitt 2.1). Bei diesen Anwendungen ist nur ein Integrationsschritt je Zeitschritt notwendig. Das Parareal Verfahren erscheint insbesondere zur Lösung von Differentialgleichungen gut geeignet, bei denen die Zeitintegration den Rechenaufwand dominiert. Somit ist der Einsatz vor allem günstig, wenn je Simulationszeitschritt eine Vielzahl an Integrationszeitschritten durchzuführen ist.

Insbesondere im Rahmen von Prädiktionsaufgaben, die auf der Simulation eines mathematischen Modells über eine Vielzahl an Zeitschritte bei jedem Aufruf basiert, erscheint der Einsatz des Parareal Verfahrens somit vielversprechend, da hierbei der Rechenaufwand durch die Zeitintegration dominiert wird.

Bei den derzeitigen Anwendungen mathematischer Modelle auf Fahrzeug-Steuergeräten liegt dies hingegen nicht vor. Um das Parareal Verfahren einsetzen zu können, sind in einem einzelnen Simulationszeitschritt mehrere Integrationszeitschritte durchzuführen. Wie am Beispiel der Leitanwendung gezeigt, sind derartige Zwischenschritte bei der Lösung der gewöhnlichen Differentialgleichung mit einem anderen Verfahren wie beispielsweise einem impliziten Euler-Verfahren nicht notwendig. Daher führt ein Parareal Verfahren im Vergleich zu Einschrittverfahren zunächst zu einem zusätzlichen Rechenaufwand. Die hierdurch erhaltene gegebenenfalls höhere Genauigkeit

ist wie in Abschnitt 3.2.2 beschrieben im Gegensatz zum stabilen Simulationsverlauf und einer geringen Rechenzeit häufig nicht die Anforderung, die sich bei einer Echtzeitsimulation als schwierig zu erreichen darstellt.

Gegenüberstellung zu alternativen Parallelisierungsmöglichkeiten

Sofern es notwendig sein sollte, dass eine Integrationsschrittweite eingesetzt wird, die geringer als die Simulationsschrittweite ist, was insbesondere vor dem Hintergrund, dass die rechte Seite nicht als mehrfach stetig differenzierbar angenommen wird, im Fall, dass eine Erhöhung der Genauigkeit des Simulationsverlaufs nötig sein sollte, naheliegend ist, eröffnet das Parareal Verfahren die Möglichkeit den Rechenaufwand für einen Simulationszeitschritt zu parallelisieren.

Hierzu stehen auch alternative Möglichkeiten zur Verfügung: Die Parallelisierung bei der Simulation eines mathematischen Modells kann durch eine Modellzerlegung ermöglicht. Zudem kann eine Parallelisierung auf Ebene der linearen Algebra erfolgen. Beispielsweise kann die in Kapitel 4 eingesetzte Block-Gauß Elimination dazu eingesetzt werden, ein Restsystem in Blockdiagonalform herzustellen, welches parallelisiert gelöst werden kann. Darüber hinaus gibt es Lösungsverfahren gewöhnlicher Differentialgleichungen, die eine Parallelisierung ermöglichen, wie etwa die Klasse der parallelisierbaren Runge Kutta-Methoden (vgl. [76]).

Insbesondere bei einer Parallelisierung durch eine Modellzerlegung sowie auf Ebene der linearen Algebra ist die Summe des Rechenaufwands auf den verschiedenen Kernen kaum höher als bei der entsprechenden Simulation ohne jegliche Parallelisierung. Insofern sind die Parallelisierungsmöglichkeiten bei Mehrkernrechnern, bei denen die Rechenressourcen als limitiert und knapp angesehen werden, günstig, da es bei diesen nicht wünschenswert ist, dass durch eine Parallelisierung die Summe der Rechenzeiten wesentlich größer ist als die Rechenzeit ohne Parallelisierung.

Das Parareal Verfahren besitzt bei k Parareal Iterationen eine parallele Effizienz von $E < k^{-1}$ (vgl. Gleichung 6.3). Somit ist das Parareal Verfahren bei mehr als einer Iteration für die Parallelisierung der Lösung einer gewöhnlichen Differentialgleichung, bei der je Aufruf ein Zeitschritt zu berechnen ist, nicht im selben Maße geeignet wie vorangehenden Parallelisierungsmöglichkeiten

Das Parareal Verfahren wurde für einen Einsatz bei sehr leistungsfähigen Rechnern konzipiert und auf solchen eingesetzt (vgl. [47, 95, 74]). Aus der Untersuchung in Abschnitt 6.2 geht hervor, dass das Parareal Verfahren auch bei einer geringen Anzahl von Rechenkernen für das dort untersuchte Modell nützlich einsetzbar ist. Außerdem führt es bei diesem Modell bereits mit einer Iteration zu einem relativen Fehler, der akzeptabel ist. Dies ist gerade im Hinblick auf die alternativen Parallelisierungsansätze wesentlich: Im Fall von einer Parareal Iteration besitzt das Verfahren eine hohe parallele Effizienz (vgl. Abschnitt 6.1.2). Hierdurch gewinnt es gegenüber den alternativen Parallelisierungsansätze an Konkurrenzfähigkeit. Sofern der Aspekt der limitierten Rechenkapazität durch die Zunahme der Rechenleistung bei zukünftigen Steuergeräten an Bedeutung verliert, ist der Einsatz eines Parareal Verfahrens auch bei der Simulation von Modellen, bei denen je Simulationszeitschritt mehr als eine Parareal Verfahren nötig sind, eine Option.

Außerdem stellt das Parareal Verfahren in diesem Fall eine interessante Möglichkeit dar, um die Rechenleistung durch weitere Rechenkerne umzumünzen in eine Reduktion der Rechendauer eines Simulationszeitschrittes im Sinne einer Verkürzung der Antwortzeit. Dies kann dazu beitragen, dass bei einer sehr rechenaufwändigen Simulationsaufgabe Echtzeitfähigkeit erreicht wird.

Das Parareal Verfahren mit einer Modellvereinfachung

Unter den Rahmenbedingungen knapper Rechenressourcen besteht die Möglichkeit zur Verwendung eines vereinfachten Modells für die grobe Stufe des Parareal Verfahrens im Falle eines Parareal Verfahrens mit einer Parareal Iteration die Eignung für einen Einsatz weiter zu steigern. Hierdurch kann die Summe der Rechenzeiten auf den verschiedenen Rechenkernen weiter

der Rechenzeit ohne Parallelisierung angenähert werden. Für die Bestimmung einer Vereinfachung im Rahmen der groben Stufe kann einerseits wie in Kapitel 4 ausgeführt vorgegangen werden, um eine Vergröberung der Jacobi-Matrix zu erzielen und diese im Rahmen der groben Stufe einzusetzen. Dies führt im Fall der Leitanwendung nur zu einer geringen Veränderung des Fehlersgegenüber der Referenztrajektorie des Rohrmodells. Somit ist dieser Ansatz gut geeignet für eine Anwendung beim Rohrmodell.

Eine weitere Möglichkeit für eine Vereinfachung im Rahmen der groben Stufe ist die Option, dass bei der Linearisierung der rechten Seite in einem Zeitschritt und gegebenenfalls nach einer geeigneten Transformation lediglich ein Teilsystem gelöst wird, das die schnellen Anteile der Dynamik der Linearisierung der rechten Seite umfasst. Hierbei werden die unmittelbaren Einflüsse der Lösung des Teilsystems auf das übrige System berücksichtigt. Dieser Ansatz ermöglicht also, anstelle des vollen Gleichungssystems lediglich ein Teilsystem lösen zu müssen. Wie in Abschnitt 6.3.2 dargestellt führt dieses Vorgehen bei kleinen Werten für die Anzahl N an Integrationszeitschritte der feinen Stufe je Mikrozeitintervall, wie sie bei einem Einsatz auf einem Fahrzeug-Steuergerät naheliegend sind, zu einer geringen Änderung der Simulationsverlaufs gegenüber dem Fall, dass in der groben Stufe das originale Modell verwendet wird. Dieser Ansatz ist somit für den Einsatz bei dem Rohrmodell ebenfalls gut geeignet.

6.5 Exkurs

Bei glatten Modellen besteht grundsätzlich die Möglichkeit, das Parareal Verfahren, bei dem sowohl in der groben Stufe als auch in der feinen Stufe ein Euler-Verfahren eingesetzt wird, durch eine Gewichtung w_i^k der Terme

$$d_i^k = F\left(t_{n,i+1}, t_{n,i}, x_i^{(k)}\right) - G\left(t_{n,i+1}, t_{n,i}, x_{n,i}^{(k)}\right)$$

ohne zusätzliche Funktionsauswertungen zu einem Verfahren höherer Ordnung modifizieren. Dies wird am Beispiel der Dahlquistschen Testgleichung und für den Fall eines expliziten Euler-Verfahrens illustriert. Hierbei wird $w_i^j = w^j$ für alle i,j angenommen. Sei

$$\dot{x}\left(t\right) = \mu x, \quad x\left(t_n\right) = x_n$$

mit $|\Delta t\mu| < 1$. Ferner seien die Parameter des Parareal Verfahrens $m = c$ sowie N und $k = 1$ fest gewählt. Gemäß Lemma 13 gilt dann für die durch ein Parareal Verfahren berechnete Lösung für den folgenden Zeitschritt

$$x_{n+1} = H\left(k, i, r, R, \delta\mu, \Delta t\mu\right) x_n$$

mit $H_0 = H\left(k, i, r, R, \delta tA, \Delta tB\right)$ sowie

$$\begin{aligned}
H_0 &= \sum_{j=0}^{k} \sum_{(b_l)_l \in I_{j,m}} \prod_{l=1}^{m} w^j \left(r\left(\delta t\mu\right)^N - R\left(\Delta t\mu\right)\right)^{b_l} R\left(\Delta t\mu\right)^{1-b_l}, \\
&= \sum_{j=0}^{k} \binom{m}{j} w^j \left(\left(1+\delta t\mu\right)^N - \left(1+\Delta t\mu\right)\right)^j \left(1+\Delta t\mu\right)^{m-j}, \\
&= \sum_{j=0}^{k} \binom{m}{j} w^j \left(1 + N\delta t\mu + \binom{N}{2}\left(\delta t\mu\right)^2 - \left(1 + \Delta t\mu + \left(\Delta t\mu\right)^2\right)\right)^j \left(1+\Delta t\mu\right)^{m-j} \\
&\quad + \mathcal{O}\left(\Delta t^3\right),
\end{aligned}$$

$$
\begin{aligned}
H_0 &= \binom{m}{0}(1+\Delta t\mu)^m + \\
&\quad \binom{m}{1} w^1 \left(1 + N\delta t\mu + \binom{N}{2}(\delta t\mu)^2 - \left(1 + \Delta t\mu + (\Delta t\mu)^2\right)\right)(1+\Delta t\mu)^{m-1} + \mathcal{O}\left(\Delta t^3\right), \\
&= 1 + m\Delta t\mu + \frac{m(m-1)}{2}(\Delta t\mu)^2 + mw^1\left(\binom{N}{2} - m^2\right)(\Delta t\mu)^2 + \mathcal{O}\left(\Delta t^3\right).
\end{aligned}
$$

Erfüllt w^1 die Gleichung

$$
\left(\frac{m-1}{2m}\right)(m\Delta t\mu)^2 + w^1\left(\frac{N(N-1)}{2m} - 1\right)(m\Delta t\mu)^2 = \frac{(m\Delta t\mu)^2}{2},
$$

also

$$
w^1 = \frac{1}{(N(N-1) - 2m)},
$$

so ist $e^{m\Delta t\mu} - x_{n+1}^k = \mathcal{O}\left(\Delta t^3\right)$.

Eine analoge Möglichkeit ist auch im Fall zu erwarten, dass für die grobe und feine Stufe des Parareal Verfahrens das linear-implizite Euler-Verfahren eingesetzt wird. Für $k_0 > 1$ ergeben sich entsprechend Gleichungen, um die Konvergenzordnung für die betrachtete Anfangswertproblem im Falle $w^k = w_j^k$ für alle $j \in \{1, \ldots, m\}$ je zusätzliche Parareal Iteration gegebenenfalls um eine weitere Ordnung zu reduzieren.

Sofern auf die einschränkende Annahme $w_i^j = w^j$ für alle i,j verzichtet wird, also $w_j^k \neq w_{j_2}^k$ gelten darf, ist es plausibel, dass sich bereits im Fall $k_0 = 1$ wie im Fall linearer Mehrschrittverfahren eine Gewichtung ermitteln lässt, durch die eine signifikant höhere Konvergenzordnung für das betrachtete Anfangswertproblem erreicht wird und dass diese bei zusätzlichen Parareal Iterationen in analoger Weise gesteigert werden kann. Für ein derartig modifiziertes Parareal Verfahren lässt sich analog zum Beweis von Korollar 13 zeigen, dass für die berechnete Lösung

$$
x_{n+1}^k = \left(\sum_{j=0}^{k} \sum_{(b_l)_l \in I_{j,m}} \prod_{l=1}^{m} \left(w_l^j r(\delta t\mu)^N - w_l^j R(\Delta t\mu)\right)^{b_l} R(\Delta t\mu)^{1-b_l}\right) x_n
$$

gilt.

7 Schlusswort

Der Einsatz mathematischer Modelle bei Anwendungen auf Fahrzeug-Steuergeräten findet zunehmend Verbreitung. Im Fall steifer mathematischer Modelle führt die Echtzeitanforderung zu einer wesentlichen numerischen Herausforderung. Am Beispiel eines Leitanwendungsmodell, das den Massenstrom durch ein Rohr beschreibt, wurden in vorliegender Arbeit numerische Methoden untersucht, die eine stabile Echtzeitsimulation eines steifen mathematischen Modells auf einem Fahrzeug-Steuergerät ermöglichen.

Das Leitmotiv der Rechenzeitreduktion stellte das Maßschneidern des Lösungsverfahrens für den Einsatzzweck dar. Zur Lösung der Anfangswertprobleme wurde ein linear-implizites Euler-Verfahren eingesetzt. Um die Rechenzeit zur Bildung einer Approximation der Jacobi-Matrix zu verringern, wurde ein Sparsing eingesetzt. Eine Variation eines bekannten Zusammenhangs, der beim Sparsing dazu genutzt wird, geeignete Einträge zu ermitteln, die durch eine Null ersetzt werden können, wurde dargestellt. Diese Variation verhindert, dass Einträge, die zum Sparsing nicht geeignet sind, zu einem Wert führen, der die Eignung zum Sparsing nahe legt.

Das Sparsing wurde in Form der sogenannten Vergröberung der Jacobi-Matrix in der Hinsicht verallgemeinert, dass die gezielte Störung der Jacobi-Matrix dahingehend gewählt wird, dass die gestörte Matrix mit geringem Rechenaufwand gebildet werden kann und dennoch zu einem geeigneten Simulationsverlauf führt. Dies umfasst solche Störungen der Matrix, die es erlauben, ausgewählte Einträge der Matrix aus anderen Einträgen mit geringem Aufwand zu ermitteln sowie mit geringem Rechenaufwand auswertbare Approximationsausdrücke für Einträge oder Werte, die bei der Berechnung der Einträge benötigt werden.

Durch die Vergröberung der Jacobi-Matrix konnte die Rechenzeit zur Bildung einer Approximation der Jacobi-Matrix um etwa 98 % reduziert werden. Der resultierende Simulationsverlauf besitzt eine Abweichung von einer gemessenen Referenztrajektorie, die sich nur gering unterscheidet von dem Fall, dass keine Vergröberung der Jacobi-Matrix eingesetzt wird. Somit wurde gezeigt, dass der Einsatz einer Vergröberung der Jacobi-Matrix im Fall des Rohrmodells sowie unter Gegebenheiten, wie sie durch die Referenztrajektorie repräsentiert werden, gut geeignet ist.

Durch das in der Vergröberung der Jacobi-Matrix beinhaltete Sparsing lag zudem der Fall vor, dass die Struktur des resultierenden linearen Gleichungssystems stark vereinfacht wurde. Insgesamt wurde hierdurch eine Rechenzeit für einen Zeitschritt des linear-impliziten Euler-Verfahrens erreicht, der gering genug ist, dass ein Einsatz der Simulation des Rohrmodells im Rahmen einer Software-Funktion auf einem Fahrzeug-Steuergerät möglich ist.

Im Hinblick auf die Anwendung bei weiteren Modellen wurde desweiteren die Reduktion der Rechenzeit bei der Lösung eines dünn besetzten linearen Gleichungssystems untersucht. Hierzu wurde die Struktur des Gleichungssystems betrachtet, das sich bei der Simulation des Rohrmodells im Rahmen der Lösung des Anfangswertproblems mit einem linear-impliziten Euler-Verfahren ergibt, wenn anstatt eines Sparsings lediglich eine Mixed Mode Integration eingesetzt wird.

Die Kondition der linearen Gleichungssysteme stellte sich bei der Leitanwendung als gut heraus, so dass die Lösung mittels einer Gauß Elimination mit 32 Bit Gleitkommavariablen unproblematisch ist. Durch eine Abschätzung der Wachstumsfaktoren im Rahmen einer Gauß Elimination ohne Pivotisierung wurde zudem gerechtfertigt, dass auch der Einsatz eines Gauß Elimination ohne Spaltenpivotisierung geeignet ist.

Zur Lösung des dünn besetzten linearen Gleichungssystems wurde eine Modifikation eines Sparse-Formats betrachtet, die darauf ausgelegt, von der bekannten und festgelegten Struktur des Glei-

chungssystems zu profitieren. Hierbei werden alle potentiellen Einträge des Fill-in von vornherein erfasst. Hierdurch lassen sich die Indizes der Einträge einer gegebenen Spalte fest im Speicher ablegen, so dass die entsprechende Indexsuche entfällt. Im Rahmen der Rechenzeitmessungen führt diese Modifikation zu einer spürbaren Reduktion der Rechenzeit.

Im Rahmen einer Gauß Elimination wurde zudem ein Ordering basierend auf einer Untersuchung des Adjazenzgraphen der Matrix des Gleichungssystems bestimmt. Dieses führt zu einem geringeren Fill-in als die initiale Form des Gleichungssystems sowie als die Orderings, die durch bestimmte heuristische Algorithmen bestimmt werden. Gegenüber einem weiteren physikalisch naheliegenden Ordering ist es im Hinblick auf den Fill-in gleichwertig. Es wurde gerechtfertigt, dass bei Verwenden der Orderings die numerische Stabilität nicht beeinträchtigt ist.

Den Fokus der Untersuchungen im Rahmen der Rechenzeitreduktion bei der Lösung linearer Gleichungssysteme bildet ein Vorgehen, das auf einer Nested Dissection basiert. Hierzu wurde basierend auf einer Untersuchung des Adjazenzgraphen der Matrix des Gleichungssystems eine Block-Gauß Elimination verwendet, die in einer Weise durchgeführt wurde, dass die Struktur der Diagonalblöcke möglichst gut geeignet ist zu einer Lösung mit geringen Rechenaufwand. Dies führte zu geringeren Rechenzeiten als bei den übrigen betrachteten Varianten. Basierend auf der Annahme, dass die auftretenden Gleichungssysteme eine Untergliederung in Teilsysteme ermöglichen, die gering ausgeprägte Querabhängigkeiten besitzen, erscheint es vielversprechend, dass sich dieses Vorgehen auf weitere Modelle übertragen lässt.

Im Hinblick auf die mögliche Zunahme der Anzahl an Rechenkernen bei zukünftigen Fahrzeug-Steuergeräten wurde die Eignung des Parareal Verfahrens, das eine Parallelisierung der Zeitintegration ermöglicht, für einen Einsatz auf einem Fahrzeug-Steuergerät untersucht. Hierzu wurde am Beispiel des Rohrmodells eine Untersuchung durchgeführt, wie sich die Parameter des Parareal Verfahrens wählen lassen, um ein gutes Verhältnis zwischen lokalem Diskretisierungsfehler und Rechenaufwand zu erhalten. Der Rechenaufwand basierte hierbei auf heuristischen Berechnungen. Es zeigte sich, dass das Parareal Verfahren auch bei einer geringen Anzahl an Rechenkernen grundsätzlich nützlich einsetzbar ist.

Im Hinblick auf die Knappheit der Rechenressourcen eines Fahrzeug-Steuergeräts ist die Anforderung natürlich, dass die Summe der Rechenzeiten, die ein parallelisiertes Verfahren auf den verschiedenen, als vergleichbar angenommenen Rechenkernen in Anspruch nimmt, die Rechenzeit eines vergleichbaren nicht parallelisierten Verfahrens möglichst wenig übersteigen soll. Dementsprechend ist eine hohe parallele Effizienz eines Verfahrens wesentlich. Im Fall des untersuchten Rohrmodells lag die Situation vor, dass das Verfahren bereits mit einer Parareal Iteration gute Resultate liefert. In diesem Fall besitzt das Parareal Verfahren eine hohe parallele Effizienz.

Um die parallele Effizienz in diesem Fall weiter zu erhöhen, wurde untersucht, wie der Rechenaufwand im Rahmen der groben Stufe reduziert werden kann. Zum einen stellte sich der Einsatz einer Vergröberung der Jacobi-Matrix im Rahmen des Parareal Verfahrens als möglich heraus. Darüber hinaus wurde der Ansatz betrachtet, im Rahmen der groben Stufe lediglich ein Teilsystem des im Rahmen einer Lösung mit dem linear-impliziten Euler-Verfahrens auftretenden linearen Gleichungssystems zu lösen. Das Teilsystem wurde hierbei in einer Weise gewählt, dass die schnellen Anteile der Dynamik erfasst werden. Für dieses Vorgehen wurde nachgewiesen, dass die Konsistenzordnung des resultierenden Parareal Verfahrens dieselbe ist wie im Fall, dass keine derartige Reduktion im Rahmen der groben Stufe eingesetzt wird. Der erhaltene Simulationsverlauf stützte dieses theoretische Resultat. Im Fall des Rohrmodells war die Bildung eines geeigneten Teilsystems ohne eine Transformation der Matrix des linearen Gleichungssystems möglich.

Als Fazit der Untersuchungen der Eignung des Parareal Verfahrens für einen Einsatz auf einem Fahrzeug-Steuergerät ergab sich, dass im Fall von Anwendungen wie modellbasierter Diagnose und Regelung sowie bei virtuellen Sensoren, die je Aufruf lediglich die Lösung eines Anfangswertproblems über einen einzelnen Zeitschritt hinweg benötigen, das Parareal Verfahren eine geringere parallele Effizienz hat als es von anderen Parallelisierungsmöglichkeiten wie einer Modellzerlegung oder einer Parallelisierung auf Ebene der linearen Algebra erwartbar scheint. Somit

erscheint es im Allgemeinen nicht zweckmäßig, im Fall einer Knappheit der Rechenressource auf einem Fahrzeug-Steuergerät das Parareal Verfahren diesen weiteren Parallelisierungsmöglichkeiten bei den genannten Anwendungen vorzuziehen. Da das Parareal Verfahren jedoch auch bei einer geringen Anzahl an Rechenkernen und auch mit einer geringen Anzahl Parareal Iterationen eine gute Genauigkeit bei der Lösung eines Anfangswertproblems erzielen konnte, scheint der Einsatz auf Fahrzeug-Steuergeräten grundsätzlich durchaus in geeigneter Weise möglich. Eine Anwendung erscheint insbesondere im Fall von Prädiktionsfunktionen vielversprechend, bei denen eine Vielzahl an Zeitschritten eines Anfangswertproblems je Aufruf zu berechnen ist.

Insgesamt wurde in vorliegender Arbeit aufgezeigt, dass durch den Einsatz geeigneter numerischer Verfahren die Eignung von Echtzeitsimulationen steifer Modelle für den Einsatz bei Anwendungen auf Fahrzeug-Steuergeräten signifikant gesteigert werden kann. Es verbleibt für eine zukünftige Arbeit, die in dieser Arbeit untersuchten numerischen Methoden bei weiteren steifen Modellen im Rahmen von Echtzeitsimulationen auf Fahrzeug-Steuergeräten anzuwenden.

Literaturverzeichnis

[1] Guidelines for the use of the C language in critical systems.

[2] *Entwicklungsmethodik für mechatronische Systeme : VDI 2206 ; VDI-Richtlinien.* Beuth, 2004.

[3] ISO 17356: Straßenfahrzeuge - Offene Software-Schnittstelle für eingebundene Fahrzeuganwendung. 11 2005.

[4] IEEE Standard for Floating-Point Arithmetic. *IEEE Std 754-2008*, pages 1–70, Aug 2008.

[5] Norm DIN 44300. Informationsverarbeitung - Begriffe. Teil 9: Verarbeitungsabläufe. *Berlin Beuth 1988 - 2001 ersatzlos zurückgezogen.*

[6] Ali Abid, Paul Oborowski, Matthias Schultalbers, and Weinhold Nick. Verfahren zum Betrieb einer Verbrennungskraftmaschine. (Patent DE102011014767 A1), 2011.

[7] Götz Alefeld and Jürgen Herzberger. *Einführung in die Intervallrechnung.* Bibliographisches Institut. B.I.-Wissenschaftsverlag, Mannheim/ Wien/ Zürich, 1974.

[8] Jürgen Appell and Martin Väth. *Elemente der Funktionalanalysis Vektorräume, Operatoren und Fixpunktsätze.* 2005.

[9] Martin Arnold. Numerical methods for simulation in applied dynamics. In *Simulation Techniques for Applied Dynamics*, pages 191–246. Springer, 2009.

[10] Martin Arnold, Bernhard Burgermeister, and Alexander Eichberger. Linearly implicit time integration methods in real-time applications: Daes and stiff odes. *Multibody System Dynamics*, 17(2-3):99–117, 2007.

[11] Martin Arnold, Bernhard Burgermeister, Claus Führer, Gerhard Hippmann, and Georg Rill. Numerical methods in vehicle system dynamics: State of the art and current developments. *Vehicle System Dynamics*, 49(7):1159–1207, 2011.

[12] Nils Arnold, Hermann Henning, Immanuel Kutschera, and Michael Bargende. Cylinder charge based injection control. In *12. Internationales Stuttgarter Symposium Automobil- und Motorentechnik*, 2012.

[13] Guillaume Bal. Parallelization in time of (stochastic) ordinary differential equations. www.columbia.edu/ gb2030/PAPERS/paralleltime.pdf., 2003.

[14] Guillaume Bal. On the convergence and the stability of the parareal algorithm to solve partial differential equations. In *Domain decomposition methods in science and engineering*, pages 425–432. Springer, 2005.

[15] Teresa Beck. *In-Time Parallelization Of Atmospheric Chemical Kinetics.* PhD thesis, Universität Heidelberg, 2015.

[16] Christian Bertsch, Jonathan Neudorfer, Elmar Ahle, Siva Sankar Arumugham, Karthikeyan Ramachandran, and Andreas Thuy. FMI for Physical Models on Automotive Embedded Targets. In *International Modelica Conference*, 2015.

[17] Thomas Bleile, Christian Fleck, Slobodanka Lux, and Alexandre Wagner. Method for real time capability simulation of an air system model of an internal combustion engine. (Patent US20110144927 A1), 2008.

[18] Adel Blouza, Frederic Coquel, and Francois Hamel. Reduction of linear kinetic systems with multiple scales. *Combustion Theory and Modeling*, 4(3):339–362, 2000.

[19] John Adrian Bondy and Uppaluri Siva Ramachandra Murty. *Graph theory with applications*, volume 290. Macmillan London, 1976.

[20] Willi Braun, S Gallardo Yances, Kilian Link, and Bernhard Bachmann. Fast simulation of fluid models with colored jacobians. In *Proceedings of the 9th Modelica Conference, Munich, Germany, Modelica Association*, 2012.

[21] Bernhard Burgermeister, Martin Arnold, and Benjamin Esterl. DAE time integration for real-time applications in multi-body dynamics. *ZAMM-Journal of Applied Mathematics and Mechanics/Zeitschrift für Angewandte Mathematik und Mechanik*, 86(10):759–771, 2006.

[22] E Carpanzano and R Girelli. The tearing problem: definition, algorithm and application to generate efficient computational code from dae systems, second mathmod. In *IMACS Symposium on Mathematical Modelling*, pages 5–7, 1997.

[23] François E Cellier and Ernesto Kofman. *Continuous system simulation*. Springer, 2006.

[24] Alan Curtis, Michael Powell, and John Reid. On the estimation of sparse jacobian matrices. *J. Inst. Math. Appl*, 13(1):117–120, 1974.

[25] Alan Curtis and John Reid. The solution of large sparse unsymmetric systems of linear equations. *IMA Journal of Applied Mathematics*, 8(3):344–353, 1971.

[26] C Curtiss and J Hirschfelder. Integration of stiff equations. *Proceedings of the National Academy of Sciences of the United States of America*, 38(3):235, 1952.

[27] Elizabeth Cuthill and James McKee. Reducing the bandwidth of sparse symmetric matrices. *Proceedings of the 24th ACM national conference*, pages 157–172, 1969.

[28] Timothy A. Davis. *Direct methods for sparse linear systems*. Fundamentals of algorithms. Philadelphia : Society for Industrial and Applied Mathematics, 2006.

[29] Luigi Del Re. *Automotive model predictive control : models, methods and applications*. Lecture notes in control and information sciences: 402. Berlin : Springer, 2010.

[30] James W. Demmel. *Applied numerical linear algebra*. SIAM, 1997.

[31] James W. Demmel, Nicholas J. Higham, and Robert S. Schreiber. Stability of block lu factorization. *Numerical Linear Algebra with Applications*, 2(2):173–190, 1995.

[32] P. Deuflhard and F.A. Bornemann. *Numerische Mathematik*. Number Bd. 2 in De Gruyter Lehrbuch. De Gruyter, 2002.

[33] P. Deuflhard and A. Hohmann. *Eine algorithmisch orientierte Einführung*. Numerische Mathematik. De Gruyter, 2008.

[34] Peter Deuflhard. *Newton methods for nonlinear problems: affine invariance and adaptive algorithms*, volume 35. Springer, 2011.

[35] Peter Deuflhard and Martin Weiser. *Numerische Mathematik 3: Adaptive Lösung partieller Differentialgleichungen*, volume 3. Walter de Gruyter, 2011.

[36] Moritz Diehl, Ilknur Uslu, Rolf Findeisen, Stefan Schwarzkopf, Frank Allgöwer, H Georg Bock, Tobias Bürner, Ernst Dieter Gilles, Achim Kienle, Johannes P Schlöder, et al. Real-time optimization for large scale processes: Nonlinear model predictive control of a high purity distillation column. In *Online Optimization of Large Scale Systems*, pages 363–383. Springer, 2001.

[37] Reinhard Diestel. *Graphentheorie*, volume 2. Springer Berlin, Heidelberg, New York, 1996.

[38] Steven X Ding. *Model-based fault diagnosis techniques*, volume 2013. Springer, 2008.

[39] Iain Duff, Albert Erisman, and John Reid. *Direct methods for sparse matrices*. Monographs on numerical analysis. New York : Oxford University Press, 1986.

[40] Hilding Elmqvist, Sven Erik Mattsson, Hans Olsson, Johan Andreasson, Martin Otter, Christian Schweiger, and Dag Bruck. Realtime simulation of detailed vehicle and powertrain dynamics. *SAE SP*, pages 63–76, 2004.

[41] Hilding Elmqvist and Martin Otter. Methods for tearing systems of equations in object-oriented modeling. In *Proceedings ESM*, volume 94, pages 1–3, 1994.

[42] G. Engeln-Müllges, K. Niederdrenk, and R. Wodicka. *Numerik-Algorithmen: Verfahren, Beispiele, Anwendungen*. Springer Berlin Heidelberg, 2010.

[43] Carlos A Felippa. Introduction to finite element methods. *Course Notes, Department of Aerospace Engineeing Sciences, University of Colorado at Boulder*, 2015.

[44] Leslie Foster. The growth factor and efficiency of gaussian elimination with rook pivoting. *Journal of Computational and Applied Mathematics*, 86(1):177–194, 1997.

[45] Dennis Frühwirth. Improvement of diagnostic functions for cylinder pressure sensors and determination of the signal quality status. Master's thesis, Hochschule Esslingen, 2012.

[46] Ana Gallet, Michael Volpp, and Wolfgang Lengerer. Standard development process for physical models used in real time applications based on the example of an exhaust pipe model. Technical report, Robert Bosch GmbH, 2014.

[47] Martin J. Gander. 50 years of time parallel time integration. In *Householder Symposium XIX June 8-13, Spa Belgium*, page 81.

[48] Martin J. Gander and Ernst Hairer. Nonlinear convergence analysis for the parareal algorithm. In *Domain decomposition methods in science and engineering XVII*, pages 45–56. Springer, 2008.

[49] Martin J. Gander and Stefan Vandewalle. Analysis of the parareal time-parallel time-integration method. *SIAM J. Sci. Comput.*, 29(2):556–578, 2007.

[50] Assefaw Hadish Gebremedhin, Fredrik Manne, and Alex Pothen. What color is your jacobian? graph coloring for computing derivatives. *SIAM review*, 47(4):629–705, 2005.

[51] A. George and J.W.H. Liu. The evolution of the minimum degree ordering algorithm. 1987.

[52] Norman E. Gibbs, William G. Poole, and Paul K. Stockmeyer. An algorithm for reducing the bandwidth and profile of a sparse matrix. *SIAM Journal on Numerical Analysis*, (2):236, 1976.

[53] Gene H. Golub and Charles F. van Loan. *Matrix computations (3. ed.)*. Johns Hopkins University Press, 1996.

[54] Gene H Golub and James Hardy Wilkinson. Ill-conditioned eigensystems and the computation of the jordan canonical form. *SIAM review*, 18(4):578–619, 1976.

[55] John L. Gustafson. Reevaluating Amdahl's law. *Commun. ACM*, 31(5):532–533, 1988.

[56] Ernst Hairer, Christian Lubich, and Gerhard Wanner. *Geometric Numerical Integration: Structure-Preserving Algorithms for Ordinary Differential Equations*. Springer Series in Computational Mathematics. Springer-Verlag Berlin and Heidelberg GmbH & Company KG, 2006.

[57] Ernst Hairer, Syvert P. Nørsett, and Gerhard Wanner. *Solving Ordinary Differential Equations I: Nonstiff Problems*. Solving Ordinary Differential Equations. Springer, 1993.

[58] Ernst Hairer and Gerhard Wanner. *Solving Ordinary Differential Equations II: Stiff and Differential-Algebraic Problems*. Lecture Notes in Economic and Mathematical Systems. Springer, 1996.

[59] Nicholas J Higham. *Accuracy and stability of numerical algorithms*. SIAM, Soc. for Industrial and Applied Mathematics, 2002.

[60] Marlis Hochbruck and Alexander Ostermann. Exponential integrators. *Acta Numerica*, 19:209–286, 2010.

[61] Stefan Hoffmann, Michael Schrott, Thorsten Huber, and Thomas Kruse. Modellbasierte Methoden zur Applikation moderner Verbrennungsmotoren. *MTZ: Motortechnische Zeitschrift*, 76(4):46, 2015.

[62] Rolf Isermann, Jochen Schaffnit, and Stefan Sinsel. Hardware-in-the-loop simulation for the design and testing of engine-control systems. *Control Engineering Practice*, 7(5):643 – 653, 1999.

[63] Christian Kanzow. *Numerik linearer Gleichungssysteme: direkte und iterative Verfahren.* Springer-Lehrbuch. Springer, 2010.

[64] Christian Kirches. *Fast numerical methods for mixed-integer nonlinear model-predictive control.* PhD thesis, Universität Heidelberg, 2011.

[65] Michael Kögel. Echtzeitfähige Integrationsverfahren für steife Anfangswertprobleme. Technical report, Robert Bosch GmbH, 2015.

[66] Michael Kögel. Echtzeitfähige Integrationsverfahren für steife Anfangswertprobleme. Master's thesis, Universität Ulm, 2015.

[67] Simon Kramer, Dirk Ziegenbein, and Arne Hamann. Real world automotive benchmarks for free. *27th Euromicro Conference on Real-Time Systems*, 2015.

[68] Olivier P Le Maître and Omar M Knio. *Introduction: Uncertainty Quantification and Propagation.* Springer, 2010.

[69] J-L Lions, Yvon Maday, and Gabriel Turinici. A "parareal" in time discretization of pde's. *Comptes Rendus de l'Academie des Sciences Series I Mathematics*, 332(7):661–668, 2001.

[70] Lichuan Liu, S.M. Kuo, and MengChu Zhou. Virtual sensing techniques and their applications. In *Networking, Sensing and Control, 2009. ICNSC '09. International Conference on*, pages 31–36, March 2009.

[71] Thomas Loderer, Christian Bertsch, and Vincent Heuveline. Using Schur complement for realtime automotive applications. *PAMM*, 15(1):597–598, 2015.

[72] Yvon Maday. Parareal in time algorithm for kinetic systems based on model reduction. *CRM Proceedings and Lecture Notes*, 2008.

[73] Yvon Maday. The parareal in time algorithm. *Sub-Structuring Techniques and Domain Decomposition Methods, F. Magoules, ed., Computational Science, Engineering & Technology, Saxe-Coburg Publications, Stirlingshire, UK*, pages 19–44, 2010.

[74] Yvon Maday and Gabriel Turinici. Parallel in time algorithms for quantum control: Parareal time discretization scheme. *International Journal of Quantum Chemistry*, 93(3):223–228, 2003.

[75] Nils Menager, Rüdiger Kampfmann, Niklas Worschech, and Lars Mikelsons. Suitability of different real-time solvers for a model-based engineering toolchain using industrial rexroth controllers. In *Proceedings of the 11th International Modelica Conference*, 2015.

[76] Willard L. Miranker and Werner Liniger. Parallel methods for the numerical integration of ordinary differential equations. *Mathematics of Computation*, 21(99):303–320, 1967.

[77] Paul Moraal and Ilya Kolmanovsky. Turbocharger modeling for automotive control applications. 1999.

[78] Jürg Nievergelt. Parallel methods for integrating ordinary differential equations. *Communications of the ACM*, 7(12):731–733, 1964.

[79] Ulrich Nowak. Regular article: Dynamic sparsing in stiff extrapolation methods. *IMPACT of Computing in Science and Engineering*, 5:53 – 74, 1993.

[80] Martin Otter. Objektorientierte Modellierung Physikalischer Systeme, Teil 4: Transformationsalgorithmen. *at Automatisierungstechnik*, 47, 1999.

[81] Ralf Pfau and Thomas Schaden. Real-time simulation of extended vehicle drivetrain dynamics. In *Multibody Dynamics*, pages 195–214. Springer, 2011.

[82] R. Pischinger, M. Klell, and T. Sams. *Thermodynamik Der Verbrennungskraftmaschine.* Der Fahrzeugantrieb. Springer Wien, 2009.

[83] S Poledna, Th Mocken, J Schiemann, and Th Beck. ERCOS: an operating system for automotive applications. *Society of Automotive Engineers International Congress and Exposition*, 1996.

[84] Katalin Popovici and Pieter J. Mosterman. *Real-Time Simulation Technologies: Principles, Methodologies, and Applications.* Computational analysis, synthesis, and design of dynamic models series. Taylor & Francis Group, 2012.

[85] Joe Radespiel. Luftsystemplan pkw. *Bosch Mediaspace*, 2015.

[86] Gerd Rapin, Stefan Wiedmann, Stefan Koospal, and Thomas Wassong. MuPAD : Eine Einführung. 2007.

[87] Konrad Reif. *Automobilelektronik : eine Einführung für Ingenieure.* ATZ-MTZ Fachbuch. Vieweg & Teubner, 2012.

[88] Peter Rentrop. Partitioned Runge-Kutta methods with stiffness detection and stepsize control. *Numerische Mathematik*, 47(4):545–564, 1985.

[89] JM Riedel, W Baumann, and K Röpke. Einsatz von RBF-Netzen zur Modellierung am Dieselmotor. In Clemens Gühmann, editor, *Simulation und Test in der Funktions- und Softwareentwicklung für die Autmobilelektronik II*, pages 33–41, Berlin, Deutschland, 2008. expert Verlag.

[90] G Rill and C Chucholowski. Real time simulation of large vehicle systems. In *Conference Proc. of Multibody Dynamics*, 2007.

[91] Georg Rill. A modified implicit euler algorithm for solving vehicle dynamic equations. *Multibody System Dynamics*, 15(1):1–24, 2006.

[92] Donald J Rose and Robert Endre Tarjan. Algorithmic aspects of vertex elimination on directed graphs. *SIAM Journal on Applied Mathematics*, 34(1):176–197, 1978.

[93] Mirja Rötting. Numerische Verfahren bei Echtzeitsimulationen in Hardware-in-the-loop Systemen. Master's thesis, Technische Universität Berlin, 2014.

[94] Johannes-Joerg Rueger, Alexander Wernet, Hasan-Ferit Kececi, and Thomas Thiel. MDG1: The new, scalable, and powerful ECU platform from Bosch. In *Proceedings of the FISITA 2012 World Automotive Congress*, pages 417–425. Springer, 2013.

[95] Daniel Ruprecht. Convergence of parareal with spatial coarsening. In *Proceedings in Applied Mathematics and Mechanics*, volume 14, pages 1031 – 1034, 2014.

[96] Yousef Saad. *Iterative methods for sparse linear systems.* Siam, 2003.

[97] Anton Schiela. *Sparsing in Real Time Simulation.* PhD thesis, Technische Universität München, 2002.

[98] Anton Schiela and Folkmar Bornemann. Sparsing in real time simulation. *ZAMM-Journal of Applied Mathematics and Mechanics/Zeitschrift für Angewandte Mathematik und Mechanik*, 83(10):637–647, 2003.

[99] Anton Schiela and Hans Olsson. Mixed-mode integration for real-time simulation. In *Modelica Workshop Proceedings,*, pages 69–75, 2000.

[100] Herbert Schuette and Peter Waeltermann. Hardware-in-the-loop testing of vehicle dynamics controllers–a technical survey. *SAE technical paper*, pages 01–1660, 2005.

[101] M. Schüler, M. Hafner, and R. Isermann. Einsatz schneller neuronaler Netze zur modellbasierten Optimierung von Verbrennungsmotoren Teil i: Modellbildung des Motor- und Abgasverhaltens. *MTZ*, 61:704 – 711, 2000.

[102] Dieter Schwarzmann. *Nonlinear internal model control with automotive applications.* PhD thesis, Logos Verl. , Berlin, 2007.

[103] Silke Seuling, Haris Hamedovic, Wolfgang Fischer, and Frank Schuerg. Model based engine speed evaluation for single-cylinder engine control. Technical report, SAE Technical Paper, 2012.

[104] S. Soejima and T. Matsuba. Application of mixed mode integration and new implicit inline integration. *International Modelica Conference*, 2002.

[105] Gunnar Andreas Staff. *Numerical computation of initial value problems originated from partial differential equations.* PhD thesis, PhD thesis, Faculty of Mathematics and Natural Sciences, University of Oslo, 2006.

[106] Matthias Stegmaier. *Modular Simulation of Pressure Swing Adsorption for Hydrogen Purification in Compact Units.* Shaker, 2008.

[107] Trond Steihaug and Arne Wolfbrandt. An attempt to avoid exact jacobian and nonlinear equations in the numerical solution of stiff differential equations. *Mathematics of Computation*, 33(146):521–534, 1979.

[108] G.W. Stewart and Ji guang Sun. *Matrix Perturbation Theory.* Computer science and scientific computing. Academic Press, 1990.

[109] ISO/TC 22 Straßenfahrzeuge. ISO 26262: Straßenfahrzeuge - Funktionale Sicherheit. 11 2011.

[110] K. Strehmel, R. Weiner, and H. Podhaisky. *Numerik gewöhnlicher Differentialgleichungen: Nichtsteife, steife und differential-algebraische Gleichungen.* Vieweg & Teubner Verlag, 2012.

[111] Karl Strehmel. *Linear-implizite Runge-Kutta-Methoden und ihre Anwendung.* Teubner-Texte zur Mathematik. 127. Teubner, 1992.

[112] Alexander Sung. *Dynamische Vermessungsmethoden in der Online-Optimierung moderner Verbrennungsmotoren.* Logos Verlag Berlin GmbH, 2009.

[113] R. Tarjan. Depth-first search and linear graph algorithms. *SIAM Journal on Computing*, 1(2):146–160, 1972.

[114] JG Verwer and S Scholz. Rosenbrock methods and time-lagged jacobian matrices. 1980.

[115] Lutz Volkmann. *Fundamente der Graphentheorie.* Springer-Verlag, 2013.

[116] Alexandre Wagner, Thomas Bleile, Slobodanka Lux, and Christian Fleck. Method and device for ascertaining a modeling value for a physical variable in an engine system having an internal combustion engine. (Patent 20130158834 A1), 2011.

[117] Henning Wallentowitz and Konrad Reif. *Handbuch Kraftfahrzeugelektronik: Grundlagen-Komponenten-Systeme-Anwendungen.* Springer-Verlag, 2008.

[118] Peter Wältermann. Hardware-in-the-loop: The technology for testing electronic controls in automotive engineering. In *6th Paderborn Workshop "Designing Mechatronic Systems"Paderborn*, 2009.

Anhang

Nomenklatur Rohrmodell

Bezeichnung	Einheit	Beschreibung
A	m^2	Querschnittsfläche
c	$J\,K^{-1}\,kg^{-1}$	spezifische Wärmekapazität eines Feststoffs
c_p	$J\,K^{-1}\,kg^{-1}$	spezifische Wärmekapazität eines Gases
d_h	m	hydraulischer Durchmesser
f	1	Rohrreibungszahl
h	$J\ kg^{-1}$	spezifische Enthalpie
l	m	Länge
m	kg	Masse
m_{mol}	$kg\,mol^{-1}$	molare Masse
$\dot{m}$	$kg\,s^{-1}$	Massenstrom
Nu	1	Nusselt Zahl
p	Pa	Druck
r	m	Radius
T	K	Temperatur
R	$J\ mol^{-1}\,K^{-1}$	universelle Gaskonstante
Re	1	Reynolds Zahl
Pr	1	Prandtl Zahl
w_i	1	Massenanteil der Spezie i
α	$W\,m^{-2}\,K^{-1}$	Wärmeübergangskoeffizient
ϵ	1	Emissivität
η	1	Wirkungsgrad
λ	$W\,m^{-1}\,K^{-1}$	Wärmeleitfähigkeit
ρ	$kg\,m^{-3}$	Dichte
σ	1	Boltzmann-Konstante

Tiefgestellter Index	Beschreibung
A	Umgebung
B	Gasphase
C	Rohrmantel

Modellgleichungen

In diesem Abschnitt sind die detaillierten Gleichungen einer Semidiskretisierung der Rohrstückmodells mit fünf Zellen aufgeführt in Form einer MuPAD-Implementierung.

Zunächst werden die Größen aufgelistet, die hierbei als Konstanten eingehen:

```
a, b, c, Pi, R, boltzmann, len, r_B, d_C, rho_C, c_C, lambda_C,
epsilon_C, M_mol,  alpha_B_C_fac, alpha_C_A_fac, T_A,
adv, bdv, cdv, Rm, yN2_EXH_TYP, yCO2_EXH_TYP, yH2O_EXH_TYP,
 MW_H2, MW_C, MW_CO, MW_N2, MW_O2, MW_H2O, MW_CO2, MW_EXH,
acp_H6C3, bcp_H6C3, ccp_H6C3, dcp_H6C3,
acp_H8C3, bcp_H8C3, ccp_H8C3, dcp_H8C3,
acp_CHy, bcp_CHy, ccp_CHy, dcp_CHy,
acp_H2, bcp_H2, ccp_H2,   dcp_H2,
acp_CO, bcp_CO, ccp_CO, dcp_CO,
acp_CO2, bcp_CO2, ccp_CO2, dcp_CO2,
acp_N2, bcp_N2, ccp_N2,  dcp_N2,
acp_O2, bcp_O2, ccp_O2, dcp_O2,
acp_NO2, bcp_NO2, ccp_NO2, dcp_NO2,
acp_NO, bcp_NO, ccp_NO,  dcp_NO,
acp_NOx, bcp_NOx, ccp_NOx, dcp_NOx,
acp_H2O, bcp_H2O, ccp_H2O, dcp_H2O,
MW_H2, MW_C, MW_CO, MW_N2, MW_O2, MW_H2O, MW_CO2, MW_CHy, MW_NOx,
atc_B, btc_B, ctc_B;
```

Im Folgenden ist eine exemplarische Implementierung der Gleichungen in MuPAD angegeben.

```
smooth_function:=(x1,x_min,x_max)->
        piecewise([x1<=x_min,x_min],[x1>=x_max,x_max],
        [x_min<x1<x_max,sin((2*(x1-x_min)/
        (x_max-x_min)-1)*pi*0.5)]);
f_ETA_EXH_T:=T->(adv + bdv*T + cdv*T^2)/1000000;
f_Reynolds_Pipe_v01:=(M_dot_feed, T, d)->
        M_dot_feed*d/((pi * d*d / 4.0)*f_ETA_EXH_T(T));
f_FrictionFactor_v01:=(M_dot_feed,T,d)->
        (1.0-smooth_function(f_Reynolds_Pipe_v01(M_dot_feed, T, d),
        2300.0, 3000.0))*(64.0/f_Reynolds_Pipe_v01(M_dot_feed, T, d))
        +smooth_function(f_Reynolds_Pipe_v01(M_dot_feed, T, d),
        2300.0,3000.0)*((1.0-smooth_function(
        f_Reynolds_Pipe_v01(M_dot_feed, T, d),1000000.0,2000000.0))*
        (0.3164 * (f_Reynolds_Pipe_v01(M_dot_feed, T, d)^(-0.25))) +
        smooth_function(f_Reynolds_Pipe_v01(M_dot_feed, T, d),
        1000000.0, 2000000.0) * (0.0054 + 0.3964 *
        (f_Reynolds_Pipe_v01(M_dot_feed, T, d)^(-0.3))));
f_NU_EXH_T_v02:=(eta_exh_T,rho_exh)->(eta_exh_T/rho_exh);
f_Reynolds_ExhPipeFlowGasoline_v01:=(M_dot_feed, p, T, d)->
        (M_dot_feed/(Pi/4.0)/(d*d)/(p*MW_EXH/Rm/T))*d/
        (f_NU_EXH_T_v02(f_ETA_EXH_T(T),p*MW_EXH/Rm/T));
f_cpj_mass:=(aj, bj, cj, dj, MWj, T)->
        (aj + bj*T + cj*T^(-2) + dj*T^2)/MWj;
f_cp_gas_mass:=(T, w_CHy, w_CO, w_NOx, w_CO2, w_O2, w_H2, w_H2O, w_N2)
        ->w_CHy*f_cpj_mass(acp_CHy,bcp_CHy,ccp_CHy,dcp_CHy,MW_CHy,T)+
        w_CO*f_cpj_mass(acp_CO,bcp_CO,ccp_CO,dcp_CO,MW_CO,T)+
        w_NOx*f_cpj_mass(acp_NOx,bcp_NOx,ccp_NOx,dcp_NOx,MW_NOx,T)+
```

```
        w_CO2*f_cpj_mass(acp_CO2,bcp_CO2,ccp_CO2,dcp_CO2,MW_CO2,T)+
        w_O2*f_cpj_mass(acp_O2,bcp_O2,ccp_O2,dcp_O2,MW_O2,T)+
        w_H2*f_cpj_mass(acp_H2,bcp_H2,ccp_H2,dcp_H2,MW_H2,T)+
        w_H2O*f_cpj_mass(acp_H2O,bcp_H2O,ccp_H2O,dcp_H2O,MW_H2O,T)+
        w_N2*f_cpj_mass(acp_N2,bcp_N2,ccp_N2,dcp_N2,MW_N2,T);
f_TC_EXH_T:=T->((atc_B + btc_B*T + ctc_B*(T^2))/1000.0);
f_Prandtl_v01:=(eta,cp,tc)->eta*cp/tc;
f_Nusselt_v02:=(Rey, Pr, dh, z, L)->
        (1.0-smooth_function(Rey, 2100.0, 2400.0))*
        (((3.66^3)+(0.7^3)+(((1.077*(Rey*Pr*dh/z)^0.3333)-0.7)^3)+
        ((0.5*((2.0/(1.0+22.0*Pr))^0.1667)*
        (Rey*Pr*dh/z)^0.5)^3))^0.333)+
        smooth_function(Rey,2100.0,2400.0)*
        ((1.0-smooth_function(Pr,1.4,1.6))*
        (0.0214 * (Rey^0.8 - 100)*(Pr^0.4)*(1+(dh/L)^(2/3)))+
        smooth_function(Pr,1.4,1.6)*
        (0.012*(Rey^0.87-281.0)*(Pr^0.4)*(1+(dh/L)^(2/3))));
f_alpha_v02:=(M_dot_feed, p, T, d, w_CHy, w_CO, w_NOx, w_CO2, w_O2,
        w_H2, w_H2O, w_N2, z,   L)->
        f_Nusselt_v02(f_Reynolds_ExhPipeFlowGasoline_v01(
        M_dot_feed,p,T,d),f_Prandtl_v01(f_ETA_EXH_T(T),
        f_cp_gas_mass(T,w_CHy,w_CO,w_NOx,w_CO2,w_O2,w_H2,w_H2O,w_N2),
        f_TC_EXH_T(T)),d,z,L)*f_TC_EXH_T(T)/d;
ff := f_FrictionFactor_v01(M_dot_feed, (T_B1 + T_B5) / 2, 2*r_B);
A_B := Pi*(r_B*r_B);
l_B := Pi*r_B*2.0;
A_C := Pi*((d_C+r_B)^2)-Pi*(r_B*r_B);
l_C := Pi*(d_C+r_B)*2.0;
dz := len*(1.0/4.0);
p1 := sqrt(p_feed*p_feed);
p2 := sqrt(p_feed*p_feed-((M_dot_feed*M_dot_feed)*1.0/
        (Pi*Pi)*R*ff*len*1.0/(r_B*r_B*r_B*r_B*r_B)*(T_B1*(1.0/2.0)+
        T_B5*(1.0/2.0))*(1.0/8.0) )/M_mol);
p3:= sqrt(p_feed*p_feed-((M_dot_feed*M_dot_feed)*1.0/
        (Pi*Pi)*R*ff*len*1.0/(r_B*r_B*r_B*r_B*r_B)*(T_B1*(1.0/2.0)+
        T_B5*(1.0/2.0))*(1.0/4.0))/M_mol);
p4 := sqrt(p_feed*p_feed-((M_dot_feed*M_dot_feed)*1.0/
        (Pi*Pi)*R*ff*len*1.0/(r_B*r_B*r_B*r_B*r_B)*(T_B1*(1.0/2.0)+
        T_B5*(1.0/2.0))*(3.0/8.0))/M_mol);
p5 := sqrt(p_feed*p_feed-((M_dot_feed*M_dot_feed)*1.0/
        (Pi*Pi)*R*ff*len*1.0/(r_B*r_B*r_B*r_B*r_B)*(T_B1*(1.0/2.0)+
        T_B5*(1.0/2.0))*(1.0/2.0))/M_mol);
M_dot := M_dot_feed;
alpha_C_A := alpha_C_A_fac*(a*v_vehicle*v_vehicle+b*v_vehicle+c);
alpha_B_C1 := alpha_B_C_fac*
        f_alpha_v02(M_dot_feed, p1, T_B1, 2*r_B, w_11,w_21,w_31,
        0.18,0,0,0.07,0.75,0.01345, 0.1076);
alpha_B_C2 := alpha_B_C_fac*
        f_alpha_v02(M_dot_feed, p2, T_B2, 2*r_B, w_12,w_22,w_32,
        0.18,0,0,0.07,0.75,0.04035, 0.1076);
alpha_B_C3 := alpha_B_C_fac*
        f_alpha_v02(M_dot_feed, p3, T_B3, 2*r_B, w_13,w_23,w_33,
```

```
        0.18,0,0,0.07,0.75,0.06725, 0.1076);
alpha_B_C4 := alpha_B_C_fac*
        f_alpha_v02(M_dot_feed, p4, T_B4, 2*r_B, w_14,w_24,w_34,
        0.18,0,0,0.07,0.75,0.09415, 0.1076);
alpha_B_C5 := alpha_B_C_fac*
        f_alpha_v02(M_dot_feed, p5, T_B5, 2*r_B, w_15,w_25,w_35,
        0.18,0,0,0.07,0.75,0.12105, 0.1076);
c_p_B1 := f_cp_gas_mass(T_B1, w_11,w_21,w_31,0.18,0,0,0.07,0.75);
c_p_B2 := f_cp_gas_mass(T_B2, w_12,w_22,w_32,0.18,0,0,0.07,0.75);
c_p_B3 := f_cp_gas_mass(T_B3, w_13,w_23,w_33,0.18,0,0,0.07,0.75);
c_p_B4 := f_cp_gas_mass(T_B4, w_14,w_24,w_34,0.18,0,0,0.07,0.75);
c_p_B5 := f_cp_gas_mass(T_B5, w_15,w_25,w_35,0.18,0,0,0.07,0.75);
dx0 := -(M_dot*R*T_B1*(w_11-w_1_feed))/(A_B*M_mol*dz*p1);
dx1 := (M_dot*R*T_B2*(w_11-w_12))/(A_B*M_mol*dz*p2);
dx2 := (M_dot*R*T_B3*(w_12-w_13))/(A_B*M_mol*dz*p3);
dx3 := (M_dot*R*T_B4*(w_13-w_14))/(A_B*M_mol*dz*p4);
dx4 := (M_dot*R*T_B5*(w_14-w_15))/(A_B*M_mol*dz*p5);
dx5 := -(M_dot*R*T_B1*(w_21-w_2_feed))/(A_B*M_mol*dz*p1);
dx6 := (M_dot*R*T_B2*(w_21-w_22))/(A_B*M_mol*dz*p2);
dx7 := (M_dot*R*T_B3*(w_22-w_23))/(A_B*M_mol*dz*p3);
dx8 := (M_dot*R*T_B4*(w_23-w_24))/(A_B*M_mol*dz*p4);
dx9 := (M_dot*R*T_B5*(w_24-w_25))/(A_B*M_mol*dz*p5);
dx10 := -(M_dot*R*T_B1*(w_31-w_3_feed))/(A_B*M_mol*dz*p1);
dx11 := (M_dot*R*T_B2*(w_31-w_32))/(A_B*M_mol*dz*p2);
dx12 := (M_dot*R*T_B3*(w_32-w_33))/(A_B*M_mol*dz*p3);
dx13 := (M_dot*R*T_B4*(w_33-w_34))/(A_B*M_mol*dz*p4);
dx14 := (M_dot*R*T_B5*(w_34-w_35))/(A_B*M_mol*dz*p5);
dx15 := -(T_B1*(alpha_B_C1*l_B*(T_B1-T_C1)+
        (M_dot*c_p_B1*(T_B1-T_B_feed))/dz))/
        (A_B*p1*((M_mol*c_p_B1)/R-1.0));
dx16 := -(T_B2*(alpha_B_C2*l_B*(T_B2-T_C2)-
        (M_dot*c_p_B2*(T_B1-T_B2))/dz))/
        (A_B*p2*((M_mol*c_p_B2)/R-1.0));
dx17 := -(T_B3*(alpha_B_C3*l_B*(T_B3-T_C3)-
        (M_dot*c_p_B3*(T_B2-T_B3))/dz))/
        (A_B*p3*((M_mol*c_p_B3)/R-1.0));
dx18 := -(T_B4*(alpha_B_C4*l_B*(T_B4-T_C4)-
        (M_dot*c_p_B4*(T_B3-T_B4))/dz))/
        (A_B*p4*((M_mol*c_p_B4)/R-1.0));
dx19 := -(T_B5*(alpha_B_C5*l_B*(T_B5-T_C5)-
        (M_dot*c_p_B5*(T_B4-T_B5))/dz))/
        (A_B*p5*((M_mol*c_p_B5)/R-1.0));
dx20 := ((alpha_C_A*l_C*(T_A-T_C1)+alpha_B_C1*l_B*(T_B1-T_C1)+
        boltzmann*epsilon_C*l_C*(T_A*T_A*T_A*T_A-T_C1*T_C1*T_C1*T_C1))/
        A_C-1.0/(dz*dz)*lambda_C*(T_C1*2.0-T_C2*2.0))/(c_C*rho_C);
dx21 := ((alpha_C_A*l_C*(T_A-T_C2)+alpha_B_C2*l_B*(T_B2-T_C2)+
        boltzmann*epsilon_C*l_C*(T_A*T_A*T_A*T_A-T_C2*T_C2*T_C2*T_C2))/
        A_C+1.0/(dz*dz)*lambda_C*(T_C1-T_C2*2.0+T_C3))/(c_C*rho_C);
dx22 := ((alpha_C_A*l_C*(T_A-T_C3)+alpha_B_C3*l_B*(T_B3-T_C3)+
        boltzmann*epsilon_C*l_C*(T_A*T_A*T_A*T_A-T_C3*T_C3*T_C3*T_C3))/
        A_C+1.0/(dz*dz)*lambda_C*(T_C2-T_C3*2.0+T_C4))/(c_C*rho_C);
dx23 := ((alpha_C_A*l_C*(T_A-T_C4)+alpha_B_C4*l_B*(T_B4-T_C4)+
```

```
        boltzmann*epsilon_C*l_C*(T_A*T_A*T_A*T_A-T_C4*T_C4*T_C4*T_C4))/
        A_C+1.0/(dz*dz)*lambda_C*(T_C3-T_C4*2.0+T_C5))/(c_C*rho_C);
dx24 := ((alpha_C_A*l_C*(T_A-T_C5)+alpha_B_C5*l_B*(T_B5-T_C5)+
        boltzmann*epsilon_C*l_C*(T_A*T_A*T_A*T_A-T_C5*T_C5*T_C5*T_C5))/
        A_C+1.0/(dz*dz)*lambda_C*(T_C4*2.0-T_C5*2.0))/(c_C*rho_C);
```

Jacobi-Matrix J_n

Es ist $J_n = \begin{pmatrix} J_{1,5} & J_{6,10} & J_{11,15} & J_{16,20} & J_{21,25} \end{pmatrix}$ mit

$$J_{1,5} = \begin{pmatrix} -756.0 & 0 & 0 & 0 & 0 \\ 754.1 & -754.1 & 0 & 0 & 0 \\ 0 & 752.2 & -752.2 & 0 & 0 \\ 0 & 0 & 750.4 & -750.4 & 0 \\ 0 & 0 & 0 & 748.5 & -748.5 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 2320.0 & 0 & 0 & 0 & 0 \\ 0 & 2694.0 & 0 & 0 & 0 \\ 0 & 0 & 2651.0 & 0 & 0 \\ 0 & 0 & 0 & 2608.0 & 0 \\ 0 & 0 & 0 & 0 & 2566.0 \\ 3.735 & 0 & 0 & 0 & 0 \\ 0 & 3.681 & 0 & 0 & 0 \\ 0 & 0 & 3.628 & 0 & 0 \\ 0 & 0 & 0 & 3.575 & 0 \\ 0 & 0 & 0 & 0 & 3.524 \end{pmatrix},$$

$$J_{6,10} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ -756.0 & 0 & 0 & 0 & 0 \\ 754.1 & -754.1 & 0 & 0 & 0 \\ 0 & 752.2 & -752.2 & 0 & 0 \\ 0 & 0 & 750.4 & -750.4 & 0 \\ 0 & 0 & 0 & 748.5 & -748.5 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 921.3 & 0 & 0 & 0 & 0 \\ 0 & 1071.0 & 0 & 0 & 0 \\ 0 & 0 & 1055.0 & 0 & 0 \\ 0 & 0 & 0 & 1040.0 & 0 \\ 0 & 0 & 0 & 0 & 1024.0 \\ 1.483 & 0 & 0 & 0 & 0 \\ 0 & 1.463 & 0 & 0 & 0 \\ 0 & 0 & 1.444 & 0 & 0 \\ 0 & 0 & 0 & 1.425 & 0 \\ 0 & 0 & 0 & 0 & 1.406 \end{pmatrix},$$

$$J_{11,15} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ -756.0 & 0 & 0 & 0 & 0 \\ 754.1 & -754.1 & 0 & 0 & 0 \\ 0 & 752.2 & -752.2 & 0 & 0 \\ 0 & 0 & 750.4 & -750.4 & 0 \\ 0 & 0 & 0 & 748.5 & -748.5 \\ 857.4 & 0 & 0 & 0 & 0 \\ 0 & 996.9 & 0 & 0 & 0 \\ 0 & 0 & 982.0 & 0 & 0 \\ 0 & 0 & 0 & 967.5 & 0 \\ 0 & 0 & 0 & 0 & 953.2 \\ 1.38 & 0 & 0 & 0 & 0 \\ 0 & 1.362 & 0 & 0 & 0 \\ 0 & 0 & 1.344 & 0 & 0 \\ 0 & 0 & 0 & 1.326 & 0 \\ 0 & 0 & 0 & 0 & 1.309 \end{pmatrix},$$

$$J_{16,20} = \begin{pmatrix} -2.223 \cdot 10^{-7} & 0 & 0 & 0 & 0 \\ 3.592 \cdot 10^{-14} & 7.334 \cdot 10^{-9} & 0 & 0 & 3.592 \cdot 10^{-14} \\ 6.926 \cdot 10^{-14} & 0 & 7.087 \cdot 10^{-9} & 0 & 6.926 \cdot 10^{-14} \\ 9.997 \cdot 10^{-14} & 0 & 0 & 6.837 \cdot 10^{-9} & 9.997 \cdot 10^{-14} \\ 1.28 \cdot 10^{-13} & 0 & 0 & 0 & 6.584 \cdot 10^{-9} \\ 2.934 \cdot 10^{-11} & 0 & 0 & 0 & 0 \\ 2.825 \cdot 10^{-16} & 5.768 \cdot 10^{-11} & 0 & 0 & 2.825 \cdot 10^{-16} \\ 8.431 \cdot 10^{-16} & 0 & 8.628 \cdot 10^{-11} & 0 & 8.431 \cdot 10^{-16} \\ 1.706 \cdot 10^{-15} & 0 & 0 & 1.167 \cdot 10^{-10} & 1.706 \cdot 10^{-15} \\ 2.893 \cdot 10^{-15} & 0 & 0 & 0 & 1.487 \cdot 10^{-10} \\ 3.803 \cdot 10^{-11} & 0 & 0 & 0 & 0 \\ 2.832 \cdot 10^{-16} & 5.781 \cdot 10^{-11} & 0 & 0 & 2.832 \cdot 10^{-16} \\ 8.445 \cdot 10^{-16} & 0 & 8.642 \cdot 10^{-11} & 0 & 8.444 \cdot 10^{-16} \\ 1.708 \cdot 10^{-15} & 0 & 0 & 1.168 \cdot 10^{-10} & 1.708 \cdot 10^{-15} \\ 2.895 \cdot 10^{-15} & 0 & 0 & 0 & 1.489 \cdot 10^{-10} \\ -1014.0 & 0 & 0 & 0 & 0 \\ 988.8 & -1012.0 & 0 & 0 & -1.791 \cdot 10^{-7} \\ -3.715 \cdot 10^{-7} & 986.5 & -1010.0 & 0 & -3.715 \cdot 10^{-7} \\ -5.782 \cdot 10^{-7} & 0 & 984.1 & -1007.0 & -5.782 \cdot 10^{-7} \\ -8.002 \cdot 10^{-7} & 0 & 0 & 981.8 & -1005.0 \\ 0.02844 & 0 & 0 & 0 & 0 \\ 0 & 0.02841 & 0 & 0 & 0 \\ -9.021 \cdot 10^{-17} & 0 & 0.02839 & 0 & -9.112 \cdot 10^{-17} \\ 0 & 0 & 0 & 0.02836 & 0 \\ 0 & 0 & 0 & 0 & 0.02833 \end{pmatrix},$$

$$J_{21,25} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 13.2 & 0 & 0 & 0 & 0 \\ 0 & 13.16 & 0 & 0 & 0 \\ 0 & 0 & 13.13 & 0 & 0 \\ 0 & 0 & 0 & 13.09 & 0 \\ 0 & 0 & 0 & 0 & 13.06 \\ -0.04686 & 0.01063 & 0 & 0 & 0 \\ 0.005315 & -0.04684 & 0.005315 & 0 & 0 \\ 0 & 0.005315 & -0.04683 & 0.005315 & 0 \\ 0 & 0 & 0.005315 & -0.04681 & 0.005315 \\ 0 & 0 & 0 & 0.01063 & -0.0468 \end{pmatrix}.$$

Notation Parareal

Symbol	Beschreibung
G	grobe Stufe des Parareal Verfahrens
F	feine Stufe des Parareal Verfahrens
ΔT	Makrozeitschrittweite
Δt	Mikrozeitschrittweite
δT	Schrittweite der groben Stufe G
δt	Schrittweite der feinen Stufe F
m	Anzahl an Mikrozeitintervallen je Makrozeitintervall
N	Anzahl an Zeitschritten der feinen Stufe je Mikrozeitintervall
N_G	Anzahl an Zeitschritten der groben Stufe je Mikrozeitintervall
n	Index des Makrozeitschrittes
i	Index des Mikrozeitschrittes
k	Index der Parareal Iteration
k_0	maximale Anzahl an durchgeführten Parareal Iterationen
c	Anzahl der verwendeten Rechenkerne
L_G	Rechenzeit der Integration über ein Mikrozeitintervall mit der groben Stufe
L_F	Rechenzeit der Integration über ein Mikrozeitintervall mit der feinen Stufe