Fortschritt-Berichte VDI

VDI

Reihe 20

Rechnerunterstützte Verfahren

Nr. 469

Dipl.-Ing. Matthias Jüttner,
Mosbach

Softwareagenten-basierte Berechnung interdisziplinärer gekoppelter Simulationen

Softwareagenten-basierte Berechnung interdisziplinärer gekoppelter Simulationen

Von der Fakultät Informatik, Elektrotechnik und Informationstechnik
der Universität Stuttgart zur Erlangung der Würde eines
Doktor Ingenieurs (Dr.-Ing.) genehmigte Abhandlung

vorgelegt von
Matthias Johannes Helmut Jüttner
aus Mosbach

Hauptberichter:	Prof. Dr. techn. Wolfgang M. Rucker
Mitberichter:	Prof. Dr.-Ing. Dr. h. c. Peter Göhner
	Prof. Dr. sc. techn. Jan Hesselbarth

Tag der mündlichen Prüfung: 13. November 2017

Institut für Theorie der Elektrotechnik
Universität Stuttgart
2017

Fortschritt-Berichte VDI

Reihe 20

Rechnerunterstützte Verfahren

Dipl.-Ing. Matthias Jüttner, Mosbach

Nr. 469

Softwareagenten-basierte Berechnung interdisziplinärer gekoppelter Simulationen

VDI verlag

Jüttner, Matthias
Softwareagenten-basierte Berechnung interdisziplinärer gekoppelter Simulationen
Fortschr.-Ber. VDI Reihe 20 Nr. 469. Düsseldorf: VDI Verlag 2017.
124 Seiten, 49 Bilder, 7 Tabellen.
ISBN 978-3-18-346920-8, ISSN 0178-9473,
€ 48,00/VDI-Mitgliederpreis € 43,20.

Für die Dokumentation: kooperative Feldberechnung – Softwareagenten – gekoppelte Simulation – intelligente Systeme – paralleles Rechnen

Der wachsende Bedarf an interdisziplinären und gekoppelten physikalischen Simulationen auf verteilten und sich dynamisch verändernden Rechnernetzen motiviert den Einsatz des hier vorgestellten Softwareagentensystems. Die Umsetzung entspricht einer neuen Abstraktionsebene oberhalb bestehender Feldberechnungswerkzeuge und ermöglicht deren dynamisches, automatisiertes und leistungsbezogenes Zusammenwirken. Ressourcenabhängig spezialisierte Berechnungseinheiten agieren dabei autonom. Im Kollektiv ermöglichen sie die Beantwortung von Fragen, zu der Einzelne nicht fähig sind. Das System ist einfach erweiterbar und unterstützt Anwender durch intelligente Algorithmen und Erfahrungswissen z. B. bei der Konfiguration des Lösungsverlaufs und bei der Reduktion von Rechenzeit. Die kooperative Feldberechnung gelingt mittels dargestellter Schnittstellen. Herausfordernde Simulationsbeispiele belegen den vorteilhaften Einsatz dieses Systems und den so gewonnenen Mehrwert.

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

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

D 93

Als Manuskript gedruckt. Printed in Germany.
ISSN 0178-9473
ISBN 978-3-18-346920-8

Vorwort

Die vorliegende Arbeit entstand während meiner Tätigkeit als wissenschaftlicher Mitarbeiter am Institut für Theorie der Elektrotechnik der Universität Stuttgart.

Mein besonderer Dank geht an Herrn Prof. Dr. techn. Wolfgang M. Rucker für das Ermöglichen dieser Arbeit, seine vielseitige Unterstützung im Verlauf der Arbeit sowie die Übernahme des Hauptberichts.

Mein Dank gebührt ebenso Herrn Prof. Dr.-Ing. Dr. h. c. Peter Göhner für die vielfältigen Anregungen während des Entstehens dieser Arbeit sowie die Übernahme des Mitberichts.

Herrn Prof. Dr. sc. techn. Jan Hesselbarth sei für das Ermöglichen der abschließenden Arbeiten und den Mitbericht gedankt.

Allen Kolleginnen und Kollegen gilt mein Dank für die gute Zusammenarbeit, die Unterstützung sowie die vielen hilfreichen Diskussionen.

Ebenfalls sei allen Studierenden gedankt, die im Rahmen ihrer studentischen Tätigkeit zum Gelingen der Arbeit beigetragen haben und mich durch ihr Wirken in der Lehre unterstützten.

Abschließend geht mein Dank an die Familie und die vielen Freunde, die mich mit ihrer Fürsorge stets begleitet haben.

„So ist’s ja besser zu
zweien als allein.“

(Kohelet 4,9)

Inhaltsverzeichnis

Formelzeichen und Abkürzungen

Lateinische Buchstaben

$\boldsymbol{A}$	Magnetisches Vektorpotenzial
a	Ordnung der Stetigkeit
$\boldsymbol{B}$	Magnetische Flussdichte
$\tilde{\boldsymbol{C}}$	Adjazenzmatrix einer Kopplung
C^a	Stetigkeit der a. Ordnung
$\boldsymbol{D}$	Elektrische Flussdichte
$\boldsymbol{D_n}$	Diffusionskoeffizient der Elektronenkonzentration
$\boldsymbol{D_p}$	Diffusionskoeffizient der Löcherkonzentration
$\mathcal{D}$	Diskretisierung
$\boldsymbol{E}$	Elektrische Feldstärke
E	Anzahl deformierter Referenzelemente
$\boldsymbol{f}$	FEM-Energiefunktional
$\boldsymbol{G}$	Matrix stationärer iterativer Verfahren
$\boldsymbol{g}$	Vektor stationärer iterativer Verfahren
h	Elementgröße
$\boldsymbol{I}$	Identitätsmatrix
I	Elektrischer Strom
$\boldsymbol{J}$	Volumenstromdichte
$\bar{\boldsymbol{J}}$	Jacobimatrix
j	Komplexe Einheit
$\boldsymbol{K}$	Systemmatrix
$\boldsymbol{L}$	Untere Dreiecksmatrix
$\boldsymbol{M}$	Massenmatrix
N	Fremdatomkonzentration
N_A^-	Akzeptorenkonzentration
N_D^+	Donatorenkonzentration
N_i	Formfunktion
$\boldsymbol{n}$	Normalenvektor
n	Elektronenkonzentration
$\mathfrak{n}$	Frequenztransformierte Elektronenkonzentration
n_i	Intrinsische Ladungsträgerdichte
$\boldsymbol{P}$	Permutationsmatrix
P_e	Péclet-Zahl

p	Löcherkonzentration
$\mathfrak{p}$	Frequenztransformierte Löcherkonzentration
$\boldsymbol{Q}$	Orthogonale bzw. unitäre Matrix
Q	Punktladung
q	Wärmequelldichte
$\boldsymbol{R}$	Obere Dreiecksmatrix
$\boldsymbol{r}$	Lastvektor
$\boldsymbol{S}$	Energieflussdichte
$\hat{\boldsymbol{S}}$	Schurkomplement
T	Temperatur
t	Zeit
$\boldsymbol{U}$	Untere Dreiecksmatrix
U	Elektrische Spannung
$\boldsymbol{u}$	Lösungsvektor
$\mathfrak{u}$	Frequenztransformierter Lösungsvektors
W	Energie
$W_{\boldsymbol{E}}$	Energie des elektrischen Feldes
W_Q	Energie einer Ladungsverteilung
$\boldsymbol{X}$	Suchrichtungsvektor nicht stationärer iterativer Verfahren
$\boldsymbol{x}$	Ortsvektor in kartesischen Koordinaten

Griechische Buchstaben

α	Skalierungsfaktor nicht stationärer iterativer Verfahren
Γ	Rand des Gebiets Ω
Γ_D	Rand mit Dirichlet-Randbedingungen
Γ_N	Rand mit Neumann-Randbedingungen
γ	Häufigkeitsverteilung
δ	Schwellwert zum Berechnungsabbruch
ε_0	Elektrische Feldkonstante
ε_r	Relative Permittivität
ϵ	Fehlerschranke
κ	Elektrische Leitfähigkeit
κ_t	Thermische Leitfähigkeit
Λ	Lagrange-Multiplikator
μ_r	Relative Permeabilität
μ_0	Magnetische Feldkonstante
μ_n	Elektronenbeweglichkeit
μ_p	Löcherbeweglichkeit
ν	Testfunktion

ρ	Volumenladungsdichte
σ	Flächenladungsdichte
τ	Periode eines harmonischen Signals
ϕ	Frequenztransformiertes elektrisches Potenzial
φ	Elektrisches Potenzial
φ_i	Ansatzkoeffizienten des elektrischen Potenzials
Ω	Gebiet
$\bar{\Omega}$	Gebiet außerhalb von Ω
ω	Betrag der Winkelgeschwindigkeit

Symbole und Operatoren

div	Divergenz
grad	Gradient
$*$	Komplexe Konjugation
Δ	Laplace-Operator
∂	Partielle Ableitung
$\Re$	Realteil einer komplexen Zahl
$\top$	Transponierte Matrix
$\cup$	Vereinigungsmenge
$\cap$	Schnittmenge
$\approx$	Approximation
$\emptyset$	Leere Menge
$\circ$	Hadamard-Produkt
$\lvert\,\rvert$	Betrag
$\lVert\,\rVert$	Norm

Abkürzungen

ACG	Agentenbasiertes Rechencluster (Agent-Based Computational Grid)
AMS	Agent Management System
API	Schnittstelle zur Anwendungsprogrammierung
BA	Berechnungsagent
BDDC	Gebietszerlegungsmethode (Balancing Domain Decomposition by Constraints)
BDI	Wissen-Wünsche-Ziele-Verhaltensmodell (Belief-Desires-Intentions behaviour model)
BEM	Randelementmethode
BiCGStab	Lösungsverfahren (Bi-Conjugate Gradient Stabilized Method)
Bitkom	Bundesverband Informationswirtschaft, Telekommunikation und neue Medien e.V.
BOINC	Rechencluster (Berkeley Open Infrastructure for Network Computing)
CG	Lösungsverfahren (Conjugate Gradient Method)
CORBA	Anwendungsarchitektur zur Vermittlung von Objekt-Nachrichten (Common Object Request Broker Architecture)
CPU	Hauptprozessor (Central Processing Unit)
CT	Containertabelle (Container Table)
DA	Dokumentationsagent
DD	Gebietszerlegungsverfahren (Domain Decomposition Method)
DF	Softwareagent (Directory Facilitator)
EMV	Elektromagnetische Verträglichkeit
FEM	Finite Elemente Methode
FETI	Gebietszerlegungsverfahren (Finite Element Tearing and Interconnecting)
FETI-DP	Gebietszerlegungsverfahren (FETI Dual-Primal)
FGMRES	Lösungsverfahren (Flexibel Generalized Minimal Residual Method)
FIPA	Gesellschaft für intelligente physikalische Agenten (Foundation for Intelligent Physical Agent)
FLOPS	Gleitkommaoperationen pro Sekunde (Floating Point Operations Per Second)
GADT	Beschreibung global ausgeführter Agenten (Global Agent Descriptor Table)
GPI	Programmierschnittstelle zur globalen Speicheradressierung (Global Address Space Programming Interface)
GMRES	Lösungsverfahren (Generalized Minimal Residual Method)
JADE	Agentenentwicklungsumgebung (Java Agent DEvelopment Framework)
JVM	Virtuelle Javamaschine (Java Virtual Machine)
KA	Koordinationsagent
KI	Künstliche Intelligenz
LADT	Beschreibung lokal ausgeführter Agenten (Local Agent Descriptor Table)
LGS	Lineares Gleichungssystem

M2M	Maschine zu Maschine (Machine to Machine)
MG	Mehrgitterlösungsverfahren (Multigrid)
MpCCI	Diskretisierungsabhängie Schnittstelle zur Quelltextkopplung (Mesh-based parallel Code Coupling Interface)
MPI	Schnittstelle zum Nachrichtenaustausch (Message Passing Interface)
MUMPS	Lösungsverfahren (MUltifrontal Massively Parallel sparse direct Solver)
OpenMP	Schnittstelle zur parallelen Programmierung (Open Multi-Processing)
PARDISO	Lösungsverfahren (Parallel Sparse Direct and Multi-Recursive Iterative Linear Solvers)
PDE	Partielle Differenzialgleichung (Partial Differential Equation)
RMI	Aufruf entfernter Methoden (Remote Method Invocation)
RPC	Vereinheitlichter Funktions- und Methodenaufruf durch verteilte Systeme (Remote Procedure Call)
SOAP	Protokoll zum Datenaustausch und für Funktionasaufrufe (Simple Object Access Protocol)
SOR	Überrelaxationsverfahren (Successive Over-Relaxation)
SORV	Überrelaxationsverfahren vektoriell
VDE	Verband der Elektrotechnik, Elektronik und Informationstechnik
VDI	Verein Deutscher Ingenieure
VDMA	Verband Deutscher Maschinen- und Anlagenbau e.V.
XML	Erweiterbare Auszeichnungssprache (Extensible Markup Language)
ZVEI	Zentralverband Elektrotechnik- und Elektronikindustrie e.V.

Zusammenfassung

Die Notwendigkeit der Simulation interdisziplinärer und gekoppelter physikalischer Zusammenhänge auf verteilten und sich dynamisch verändernden Rechnernetzen motiviert den Einsatz des im Rahmen dieser Arbeit entwickelten und im Folgenden vorgestellten Softwareagentensystems. Dabei entsprechen interdisziplinär gekoppelte Systeme solchen, deren Berechnung durch ein Zusammenwirken mehrerer spezialisierter Einheiten gelingt. Beispiele stellen verschiedene sich beeinflussende physikalische Zusammenhänge dar, die sich verschiedenen Ingenieursdisziplinen zuordnen lassen. Weitere Beispiele sind sich gegenseitig beeinflussende Teilgebiete oder Zeitabhängigkeiten in einer Simulation. Die Umsetzung des Softwareagentensystems entspricht dabei der Einführung einer neuen Abstraktionsebene oberhalb bestehender Simulationswerkzeuge, die deren Zusammenwirken ermöglicht. Dazu notwendig werde eine Beschreibung der Eigenschaften und Fähigkeiten der gekapselten Simulationswerkzeuge sowie die Umsetzung von Schnittstellen zur numerischen Zusammenarbeit. Die entstehenden spezialisierten Berechnungseinheiten nutzen zur Simulation die ihnen verfügbare Soft- und Hardware und reagieren selbstständig auf bereitgestellte Zwischenergebnisse. Das so realisierte neuartige System ermöglicht es unterschiedlichen und unabhängigen Simulationseinheiten automatisiert zusammenzuarbeiten.

Die kooperative Feldberechnung gelingt durch das Bereitstellen von Randbedingungen und Quellen während der Feldberechnung sowie durch das Auslesen von Ergebnissen an beliebigen Stellen des Modells. Erforderliche Funktionen der Schnittstellen, beispielsweise zur Steuerung und Überwachung von Feldberechnungssoftware, werden hier dargestellt. Jedem Berechnungsagenten wird es so möglich, die globale Simulationsaufgabe entsprechend seiner Vorstellungen eines zielführenden Lösungsbeitrags zu modifizieren. Die entstehenden Teilaufgaben werden dann angepasst auf ihre jeweiligen Anforderungen berechnet, was zu einer schnellen Lösungsbereitstellung führt.

Das vorgestellte Agentensystem ermöglicht die kollektive Bearbeitung von Aufgaben, zu der ein Einzelner nicht fähig ist. Zudem ist das dynamisch entstehende Simulationssystem einfach um Fähigkeiten und Ressourcen erweiterbar. Ein gezieltes Hinzufügen von zusätzlichen Ressourcen mit gleichen Fähigkeiten ermöglicht eine Varianz und Redundanz im Lösungsverlauf. Dabei werden existierende Teilergebnisse umgehend berücksichtigt, sodass das Agentensystem automatisiert leistungsbezogen Lösungswege vergleicht. Dies gestattet die eigenständige Konfiguration des Lösungsablaufs durch beteiligte Berechnungsagenten und den automatischen

Einsatz der zur Berechnung notwendigen Simulationswerkzeuge. Zusätzliche Ressourcen dienen dem Aufbau von Erfahrungswissen, das hier mittels eines lernfähigen Systems verwaltet und zur Reduktion benötigter Rechenzeit verwendet wird.

Bei numerischer Betrachtung entspricht der mittels Softwareagenten umgesetzte Lösungsverlauf einer gestaffelten Berechnung, die für viele gekoppelte numerische Simulationen zielführend ist. Das Einbeziehen von Agenten, deren Berechnungsfähigkeiten zur alleinigen monolithischen Berechnung ausreichen, ermöglicht eine zusätzliche konkurrierende Berechnung sowie weitere Entscheidungsoptionen in Bezug auf die Durchführung der Berechnung. Um die Autonomie der Berechnungsagenten hervorzuheben, wird auf eine zentrale Instanz während der Berechnung verzichtet. Die detaillierte Darstellung des durch Kooperation erreichten numerischen Lösungsprozesses verdeutlicht zusätzlich die Funktionsweise des Agentensystems.

Die Berechnung herausfordernd ausgewählter Simulationsbeispiele belegt, dass die kooperative Berechnung interdisziplinärer gekoppelter Simulationen durch autonome Berechnungseinheiten in verteilten Systemen auch für verschiedene Diskretisierungen gelingt. Zudem veranschaulichen die Beispiele den durch die agentenbasierte Herangehensweise gewonnenen Mehrwert, wie beispielsweise seine Flexibilität und Zuverlässigkeit. Im Hinblick auf die Verwaltung der zur Simulation notwendigen Infrastruktur gelingt die automatisierte Nutzung von auf die betrachtete Simulation angepassten und befähigten Ressourcen. Die durchgeführte Rechenzeitauswertung und die Betrachtung der Ressourcennutzung demonstrieren die Leistungsfähigkeit des realisierten Systems für unterschiedliche Kopplungen.

Das vorgestellte Agentensystem unterstützt dabei Anwender, die Simulationen zur Produktgestaltung verwenden. Beispiele dafür stellen die Konfiguration der Randbedingungen zur Kopplung von Simulationen, die dynamische Anpassung der Lösungssequenz an verfügbare Rechenressourcen oder die intelligente Entscheidung der Softwareagenten für ein Lösungsverfahren dar. Entsprechend ermöglicht es eine Reduktion der zur Durchführung einer Simulation erforderlichen menschlichen Arbeitszeit. Zudem unterstützt es selbst erfahrene Anwender bei getroffenen Entscheidungen bzw. bestätigt diese darin.

Abstract

Simulating interdisciplinary and coupled equations on distributed and dynamically changing computer networks motivates the usage of a software agent system, developed and presented within this thesis. These interdisciplinary and coupled systems mean that a cooperative calculation is possible by multiple specialized units. Examples are different physics that correspond to various engineering disciplines and have an effect on each other. Other examples are different parts of a simulation model or time steps within a simulation model influencing each other. The implementation of the software agent system adds a new abstraction layer above existing simulation tools and enables its cooperation. It requires a description of the features and capabilities of the encapsulated simulation tools as well as the implementation of interfaces used for numeric cooperation. The resulting specialized calculation units' uses own hard- and software for calculation and react independently to provide partial results. The resulting novel system enables individual and independent simulation units to do cooperative work automatically.

The cooperative field calculation get possible by providing boundary conditions and coupling sources during the field calculation and extracting results at arbitrary coordinates within the simulation model. Required interface functionality, for example to the control and monitor the field calculation software, are shown here. Using these, each calculation agent gets capable of modifying the global simulation task, according to its ideas of a target-oriented solution contribution. This allows a calculation of the resulting partial tasks with respect to their needs and consequently a fast deployment of the solutions.

The introduced software agent system enables a cooperative treatment of tasks, which an individual agent is not capable of handling. In addition, the dynamically jointed simulation system is simple to extend by abilities and resources. Pointedly added additional resources with equal capabilities add variance and redundancy in the solution process. Once partial results are provided, they are taken into account by the software agent system. This enables an automatic comparison of the performance of all applied solutions strategies. It allows the involved calculation agents to alter the solution process and to apply the required simulation tools. Additional resources increase the experiential knowledge, managed here by a learning system that reduces calculation time.

The numerical analysis of the software agents based solution process corresponds to a segregated calculation method, suitable for most coupled numerical simulations. Involving agents with sufficient calculation capabilities to monolithically simulate the model add an additional compe-

titive solution strategy as well as another option for calculation. To emphasise the autonomy of the calculation agent a central instance during calculation is forwent. The detailed description of the numerical solution process achieved by the cooperating software agent system additionally highlights its workflow.

The calculation of the selected challenging simulation examples shows that the cooperative calculation of interdisciplinary and coupled simulations works for autonomous calculation units on distributed systems, even for different discretizations. The examples also show the additional benefit obtained by the software agent based approach, such as its flexibility and reliability. Even the management of required simulation infrastructure profits since only proper and versed resources are automatically used for the calculation of the considered simulation. The evaluation of the computation time and the considered resources usage demonstrates the performance of the implemented system for different couplings.

The presented software agent system supports users doing simulations for product design. Examples are the setup of boundary conditions for coupled simulations, the dynamic adjustment of the solution process to available computing resources or the intelligent decision of software agents for solution methods. Accordingly, it allows a reduction of required human working time. It also supports and encourage even trained users while solving simulations.

1 Einleitung

Die Empfehlung der Gründung eines eigenständigen Master- und Bachelorstudiengangs „Simulationswissenschaft“ durch den Wissenschaftsrat der Bundesregierung und der Länder sowie die Empfehlung einer Aufnahme des Moduls „Simulationswissenschaft“ in grundständige Studiengänge verdeutlicht das hohe Potenzial von Simulationen für unsere Zukunft [1]. Bei einer Simulation Zusammenhänge aus verschiedenen Ingenieursdisziplinen und deren Zusammenwirken zu berücksichtigen, erscheint dabei naheliegend [2]. Ingenieursdisziplinen befassen sich hierbei mit Teilgebieten des technischen Wissens, die zur Strukturierung und Beherrschbarkeit eingeführt wurden [3]. Beispiele stellen die Elektrotechnik, der Maschinenbau oder das Bauingenieurwesen dar. Die zur Berechnung einer Simulation notwendige Rechnerinfrastruktur entwickelt sich dabei stetig weiter [4]. Zur Leistungsmessung der zur Verfügung stehenden Rechnerkapazitäten werden im Bereich der Höchstleistungsrechnung speziell auf das System angepasste Algorithmen verwendet [5]. Anwendungen die für den Alltag ausgelegt werden zielen hingegen auf eine möglichst große Verbreitung und dementsprechend eine große Flexibilität in Bezug auf die Nutzungsplattform [6]. Entsprechend verschieden sind die Arten der Programmierung von Anwendungen, die wiederum Einfluss auf die spätere Leistungsfähigkeit der Anwendung haben [7]. Den Aufwand zur Umsetzung der benötigten Funktionalität innerhalb der Anwendung gilt es im Verhältnis zu dessen Leistungsfähigkeit zu sehen [8, 9]. Dabei setzen verschiedene Programmiersprachen auf abstrakte Programmierparadigmen, wovon eines die agentenorientierte Programmierung ist [10]. Die Programmierung eines agentenorientierten Systems erfolgt mittels abgeschlossenen, jedoch zur Kooperation befähigten Softwareagenten [11, 12]. Dabei sind Systeme mit mehreren Softwareagenten ideal zur Bearbeitung von Aufgaben geeignet, für die mehrere Lösungswege oder bzw. und mehrere potenzielle, die Aufgabe bearbeitende Einheiten existieren [13]. Dies trifft beispielsweise auf die Berechnung numerischer Simulationen zu, da hierzu verschieden konfigurierte Rechnersysteme und unterschiedliche Berechnungswege zu deren Lösung eingesetzt werden.

1.1 Motivation

Ausgehend von einer Entwicklungsaufgabe, zu deren Erfüllung es mehrere Ingenieursdisziplinen zu vereinen gilt, stellt sich für einen Arbeitgeber die Frage, wer bearbeitet diese interdisziplinäre Aufgabe? Bringt eine Person genügend interdisziplinäres Wissen und Arbeitsleistung

mit die Aufgabe zu erledigen, so ist die Bearbeitung der Aufgabe durch eine einzelne Person unter Umständen ebenso sinnvoll, wie die durch eine in gleichem Maße befähigte Gruppe. Ansätze zur Aufgabenaufteilung finden sich beispielsweise im Koordinationsmanagement [14, 15]. Je flexibler ein Arbeitgeber dabei auf Anfragen reagieren möchte, desto dynamischer gestaltet er den Arbeitsprozess und die Arbeitsgruppen. Die Flexibilisierung der Arbeitsabläufe stellt dabei eine Kernkomponente von Industrie 4.0 dar [16].

In Bezug auf numerische Simulationen ist eine Vielzahl an Simulationswerkzeugen auf einzelne eigenständige Aufgaben und Fragestellungen spezialisiert und wird hierfür ständig weiterentwickelt. Explizite Schnittstellen für ein gemeinschaftliches Bearbeiten von Aufgaben existieren selten, obwohl gerade die Interoperabilität Vorteile in Bezug auf die Softwarequalität, die Benutzbarkeit und die Wartbarkeit mit sich bringt [17]. Um den Ansprüchen der Anwender nach einer Möglichkeit zur Simulation interdisziplinärer Fragestellungen nachzukommen, war während dieser Arbeit zu beobachten, dass das Portfolio kommerzieller Simulationswerkzeuge durch Zukäufe und die Programmierung neuer Funktionalität innerhalb der Simulationsumgebung erweitert wurde. So entstandene interdisziplinäre Problemlösungsumgebungen ermöglichen es, thematisch angrenzende physikalische Effekte mit zu berücksichtigen. Beispiele solcher Simulationsumgebungen sind in [18–21] beschrieben. Erweiterungen um selten auftretende Themenfelder werden dabei kaum berücksichtigt.

Die Entscheidung für den Einsatz eines geeigneten Simulationswerkzeugs ist angelehnt an [22] möglich, setzt jedoch fundierte Kenntnisse der zu untersuchenden Fragestellung sowie weitreichende Anwendungskenntnisse voraus. Erschwerend kommt aus Anwendersicht hinzu, dass die Notwendigkeit der Berücksichtigung einzelner Disziplinen im Vorfeld der Simulation häufig unbekannt ist und gerade diese durch die Simulation bestimmt werden soll. Dabei wird Simulationssoftware üblicherweise auf leistungsstarken Servern oder homogenen, dedizierten und hochverfügbaren Rechenclustern ausgeführt. Sehr große Fragestellungen werden so beispielsweise auf Höchstleistungsrechenclustern berechnet. Verwendet werden dabei programmiertechnische Parallelisierungsansätze, wie die Schnittstelle zum Nachrichtenaustausch MPI oder bzw. und die zur parallelen Programmierung OpenMP [23], und mathematischen Zerlegungsansätzen, wie Gebiets-, Methoden- und Zeitschrittzerlegungen oder die getrennte Betrachtung der Disziplinen [24, 25], um bereitgehaltene Rechenressourcen sehr gut auszunutzen.

Tabelle 1.1 zeigt beispielhaft Kosten, die durch eine Beschaffung eines weiteren Simulationsrechners entstehen. Zudem sind die Kosten zur Erweiterung des Simulationswerkzeugs durch Programmierarbeiten eines Simulationsingenieurs aufgeführt. Nicht berücksichtigt sind Betriebskosten (Strom, Klimatechnik, ...), Ausfallzeiten sowie die Abwesenheitszeit wegen Krankheit oder Urlaub. Der stark vereinfachte Vergleich der auf eine Stunde bezogenen Kosten zeigt, dass die Bereitstellung von zusätzlicher Rechenleistung wirtschaftlicher ist, als das Optimieren von bestehendem Quelltext. Die Leistungsoptimierung wird so erst für wieder-

kehrende oder sehr große Simulationsaufgaben rentabel, für die eine Anschaffung weniger Simulationsrechner nicht ausreicht.

Tabelle 1.1: Kosten eines Simulationsrechners und Simulationsingenieurs

Simulationsrechner	$15.000\frac{€}{5a} \,/\, 360\frac{\text{Tagen}}{a} \cdot 24\frac{h}{\text{Tag}}$	=	$0{,}35\frac{€}{h}$
Simulationsingenieur	$60.000\frac{€}{a} \,/\, 250\frac{\text{Tagen}}{a} \cdot 8\frac{h}{\text{Tag}}$	=	$23\frac{€}{h}$

Ist der Rechenaufwand einer Simulationsaufgabe nicht absehbar, so ist eine bedarfsorientierte Nutzung von Berechnungsressourcen sinnvoll. Die bedarfsorientierte Rechnernutzung stellt einen Anwendungsfall der Rechnerwolke (Cloud) dar. Dabei besteht die Möglichkeit, die Rechnerinfrastruktur, die Anwendungsplattform und die Software jeweils als Dienstleistung zu beziehen [26, 27]. Obwohl die Cloud prinzipiell eine Alternative zu Höchstleistungsrechnern darstellt, spricht deren Leistungsfähigkeit im Bereich des wissenschaftlichen Rechnens dagegen [28]. Ein Grund hierfür ist, dass Werkzeuge zur Arbeitsaufteilung häufig denen des Rechenclusters entsprechen und dabei die dynamische und tatsächliche Beschaffenheit der Infrastruktur außer Acht bleibt. Deren Eigenschaften, wie die Verfügbarkeit, die Flexibilität, die Skalierbarkeit und die bedarfsorientierten Kosten fördern jedoch Bestrebungen, auch Hochleistungsanwendungen dort auszuführen [29]. Für sehr große Aufgabenstellungen, die sich in kleine voneinander unabhängige Teilaufgaben zerlegen lassen, bietet sich das Verbinden von unabhängigen, verteilten Rechenressourcen zu Rechenclustern (Grids) an. Dabei trägt jeder Clusterknoten die von ihm mögliche Leistung zur Aufgabenbewältigung bei. Ein sehr erfolgreiches System stellt das Rechencluster BOINC dar [30]. Es ist mit 15 PFLOPS und $9 \cdot 10^5$ angeschlossenen Rechnern um zwei Größenordnungen größer als der schnellste Höchstleistungsrechner weltweit (Stand Dezember 2016 [31]). Bei der numerischen Simulation sind Simulationsteile jedoch sehr häufig voneinander abhängig. Ein wirkungsvoller und praxiserprobter Ansatz der Automatisierungstechnik ist es, gekoppelte Aufgaben auf verteilten Systemen mittels agentenorientierten Softwareprogrammen zu bearbeiten [32]. Dabei werden Teilaufgaben von autonom agierenden Softwareagenten erledigt. Die Fähigkeit der Softwareagenten zur Kommunikation ermöglichen deren Kooperation zur Bearbeitung komplexer Aufgaben und eine dynamische Handhabe von Änderungen innerhalb des Systems.

1.2 Zielsetzung

In dieser Arbeit wird der agentenorientierte Programmieransatz angewendet, um die numerische Berechnung interdisziplinärer und gekoppelter Systeme auf verteilten und sich dynamisch verändernden Rechnernetzen zu ermöglichen. Interdisziplinäre und gekoppelte Systeme entsprechen dabei solchen, deren Berechnung durch ein Zusammenwirken mehrerer spezialisierter Simulationseinheiten gelingt. Interdisziplinäre Systeme zeichnen sich durch eine gemeinsame

Betrachtung verschiedener physikalischer Zusammenhänge aus, die sich wiederum unterschiedlichen Ingenieursdisziplinen zuordnen lassen. Gekoppelte Systeme bestehen aus mehreren Teilsimulationen, die gemeinsam betrachtet werden und sich gegenseitig beeinflussen.

Zur Umsetzung des Softwareagentensystems wird eine neue Abstraktionsebene oberhalb bestehender Lösungsumgebungen eingeführt, die eine Erweiterung der Lösungsumgebungen um Schnittstellen für eine agentenorientierte Simulation schafft. Zur Berechnung wird ein neuartiges System aus unabhängigen und individuellen Simulationseinheiten geschaffen, das seine Fähigkeiten angepasst an die zu berechnende Simulation selbstständig einsetzt. Die als Softwareagenten umgesetzten Simulationseinheiten ermöglichen es, dass zur Lösung erforderliche Teilaufgaben angepasst auf deren jeweilige numerische und hardwarebezogene Anforderungen berechnet werden. Dazu werden vernetzte Rechner mit häufig systemgebundenen Simulationslizenzen eingesetzt, wie beispielsweise Arbeitsplatzrechner innerhalb eines Unternehmens nach Feierabend oder bei Krankheit eines Kollegen, um verteilt Simulationsteile zu bearbeiten. Eine auf die unterschiedlichen Software- und Hardwarespezifikationen angepasste Nutzung der Ressourcen ermöglicht es, gemeinschaftlich eine gekoppelte Simulationsaufgabe zu berechnen. Die auf eine zu simulierende Fragestellung bezogen Fähigkeiten einer einzelnen Ressource werden dabei als Blackbox betrachtet. Dies gestattet auch das Einbinden von kommerziellen Simulationswerkzeugen. Die entstehenden spezialisierten Berechnungseinheiten reagieren dabei selbstständig auf bereitgestellte Daten anderer Einheiten und nutzen diese für eine gekoppelte Berechnung. Hierzu notwendig werden Schnittstellen zwischen den Berechnungseinheiten sowie zwischen einer Berechnungseinheit und den eingebundenen Simulationswerkzeugen.

Die von einer Berechnungseinheit zu bewältigende Aufgabe lässt sich als Teilaufgabe eines dynamischen Gesamtsystems betrachten, die je nach Beschaffenheit bereitstehender Ressourcen angepasst bearbeitet wird. Ein optimierter Lösungsablauf einer einzelnen Teilaufgabe trägt dabei zu einem verbesserten Gesamtverhalten bei, was zu einer schnellen Lösungsbereitstellung führt. Die Einführung einer zentralen, an Berechnungen beteiligten Instanz wird dabei bewusst vermieden, um die Eigenschaften des dezentralen Agentenansatzes zu betonen. Das so dynamische entstehende Simulationssystem ist einfach um Fähigkeiten und Ressourcen erweiterbar und berücksichtigt diese automatisiert. Die durch Softwareagenten bearbeiteten, gekapselten Teilaufgaben ermöglichen die Anwendung effizienter und auf die Teilaufgaben angepasster Algorithmen. Diese Kapselung wirkt sich zudem positiv auf die Benutzbarkeit und Wartung des Softwaresystems aus. Auch wird eine Berücksichtigung von Ressourcen mit gleichen Fähigkeiten betrachtet. Für verteilt ausgeführte Softwareagenten ermöglicht dies eine gebietsweise Verteilung der Simulationsaufgabe und ein gezieltes Hinzufügen von Varianz und Redundanz im Lösungsverlauf. Verschiedene Berechnungsmethoden und Lösungsverfahren lassen sich so durch das Berechnungssystem vergleichen. Der automatisierte Einsatz so bestimmter leistungsstarker Methoden und Verfahren ermöglicht zudem eine Reduktion der erforderlichen menschlichen Arbeitszeit zur Durchführung einer numerischen Simulation.

1.3 Gliederung

Der schriftliche Teil der vorliegenden Arbeit gliedert sich in sechs Kapitel. Die Einführung zu Beginn dieses Kapitels stellt die Arbeit in den gesellschaftlichen Kontext. Die angeführte Motivation konkretisiert dies anhand ausgewählter Themenfelder. Aus diesen Themenfeldern ergibt sich die dargelegte Zielsetzung dieser Arbeit.

Ausgehend von dem Paradigma der agentenorientierten Programmierung wird in Kapitel 2 das für diese Arbeit wesentliche Verständnis von Softwareagentensystemen vermittelt. Dazu werden die Eigenschaften der Softwareagenten und das Zusammenwirken mehrerer Softwareagenten in einem Agentensystem dargestellt. Um bestehendes Erfahrungswissen bei der Umsetzung eines Agentensystems angemessen zu berücksichtigen, werden sich daraus ergebende Hindernisse genannt und eine Entwicklungsumgebung für Softwareagentensysteme vorgestellt. Auf deren Grundlage wird im Folgenden das Softwareagentensystem zur gemeinschaftlichen und verteilten Simulationsberechnung entwickelt.

Die Herangehensweise zur Lösung einer gekoppelten interdisziplinären numerischen Simulation auf verteilten Systemen wird in Kapitel 3 behandelt. Dazu wird nach einem Überblick über die Finite Elemente Methode (FEM) auf verschiedene Arten der Simulationsaufteilung und deren anschließende Kopplung eingegangen. Die Möglichkeiten der Berechnung von gekoppelten Simulationsteilen werden anschließend vertieft. Abschließend wird ein Bezug zu bestehenden Berechnungsumgebungen hergestellt und eine Abgrenzung dieser Arbeit davon vorgenommen.

Die Umsetzung des agentenorientierten Programmierparadigmas zur Lösung interdisziplinärer gekoppelter Simulationsaufgaben findet in Kapitel 4 statt. Dazu wird das implementierte Agentensystem und dessen Fähigkeiten mitsamt der Kommunikation über realisierte Schnittstellen beschrieben. Anschließend wird auf die Gestaltung des numerischen Lösungsprozesses und dessen Umsetzung innerhalb des Agentensystems eingegangen. Dabei werden verschiedene Kopplungsarten und der Umgang mit unterschiedlichen Diskretisierungen dargestellt. Eine Betrachtung der Ressourcennutzung rundet dieses Kapitel ab.

Die Auswertung ausgewählter Simulationsbeispiele in Kapitel 5 veranschaulicht die Funktionalität des Agentensystems. Dabei werden zwei elektrotechnische Problemstellungen betrachtet und diese anhand der sich im Lauf des Entwicklungsprozesses ändernden Fragestellungen an eine Simulation ausgewertet. Auch eine sich während der Simulation ändernde Zusammensetzung des Rechnersystems wird hierbei betrachtet. Im Besonderen werden dabei Eigenschaften der Softwareagenten betont, die im Rahmen dieser Arbeit auf die numerische Feldberechnung übertragen wurden und deren Einsatz eine erfolgreiche Berechnung ermöglicht.

Abschließend fasst Kapitel 6 die Arbeit zusammen und zeigt in einem Ausblick weitere Forschungs- und Entwicklungsmöglichkeiten auf.

2 Softwareagentensysteme

Zu Beginn dieses Kapitels wird das Paradigma der agentenorientierten Programmierung näher betrachtet. Die sich daraus ergebenden Eigenschaften von Softwareagenten werden anschließend dargestellt. Ausgehend davon werden Softwareagentensysteme beschrieben, die im weiteren Verlauf der Arbeit zur Berechnung von numerischen Simulationen eingesetzt werden. Auf verschiedene Rollen die ein Softwareagent innerhalb des Agentensystems einnimmt und dessen damit verknüpfte Bearbeitung ausstehender Aufgaben innerhalb des Systems, wird hier ebenfalls eingegangen. Abgerundet wird das Kapitel mit der gesonderten Darstellung von Herausforderungen bei der Umsetzung eines Agentensystems. Ergänzend wird mit JADE, das eine etablierte Entwicklungsumgebung für Softwareagentensysteme darstellt, eine Grundlage zur Umsetzung des numerischen Simulationssystems vorgestellt.

2.1 Agentenorientierte Programmierung

Die agentenorientierte Programmierung ist ein Paradigma zur Softwareentwicklung. Es setzt eine Herangehensweise zur Lösung einer Fragestellung voraus, die es ermöglicht spezifische Eigenschaften zu erzielen. Die agentenorientierte Programmierung ist dabei besonders geeignet, um Aufgaben zu bearbeiten, die auf natürliche Weise in Teilaufgaben mit hohem Zusammenhalt und geringen Beziehungen nach außen zerfallen [33]. Ebenfalls geeignet sind Aufgaben, die sich in eigenständige Teilaufgaben zerlegen und somit gut parallelisieren lassen [34]. Ein Beispiel stellt das automatisierte Vereinbaren von Terminen dar. Ist die Beziehung zwischen einzelnen Teilaufgaben dynamisch oder verschieden strukturiert oder die Anzahl auftretender Fälle durch viele Kombinationsmöglichkeiten komplex und schwer handhabbar, so ist ebenfalls eine Umsetzung als Agentensystem zu prüfen [33].

Zur Umsetzung agentenorientierter Programmierung gilt es eine Modellbeschreibung zu finden, die erforderliche Daten und Prozesse zu interagierenden Objekten zusammenfasst. Diese lassen sich mittels zugehöriger Dienste manipulieren. Dienste werden Agenten zugeordnet, die auf Grundlage von Verhandlungen sich gegenseitig und unabhängig voneinander erforderliche Dienstleistungen bereitstellen. Ein Dienst lässt sich entsprechend als gekapselte Fähigkeit eines Agenten betrachten, über dessen Ausführung der Agent jedoch selbst entscheidet. Abbildung 2.1 ordnet die agentenorientierte Programmierung unter Berücksichtigung der Abstraktion der Modellierung und der Interaktion in den Kontext der Programmierparadigmen für verteilte

Systeme ein. Dabei bedarf es zur agentenorientierten Programmierung einer Steigerung der modellierten Abstraktion und der Interaktion im Vergleich zur Beschreibung von Diensten. Eine weiterführende Abgrenzung zur objektorientierten Programmierung wird in [35] beschrieben.

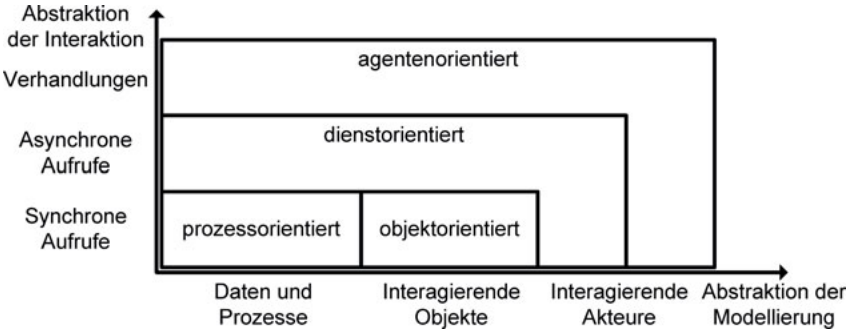

Abbildung 2.1: Paradigmenvergleich für verteilte Systeme [36]

Die Kapselung der Fähigkeiten innerhalb von Agenten ermöglicht es der Implementierung, einzelne Agenten auf unterschiedlichen, über ein Netzwerk verbundenen Systemen zu betreiben. Das Verhalten eines Agentensystems, bestehend aus mehreren Agenten, ergibt sich dynamisch aus dem Zusammenspiel der Agenten. Aus Anwendersicht wirkt das System dabei nicht verteilt, sondern gewohnt lokal. Grund hierfür ist, dass die ablaufenden komplexen Prozesse im Hintergrund zwischen den beteiligten Agenten stattfinden [36].

2.2 Softwareagenten

Die Beschreibung eines Softwareagenten erfolgt ausgehend von dessen Eigenschaften. Die Summe der Eigenschaften stellt dabei ein Kriterium zur Unterscheidung agentenorientiert entwickelter Software mit anderen Programmieransätzen dar. Dabei wird zwischen einer schwachen und einer starken Beschreibung der Agenteneigenschaften unterschieden. Ein einzelner Agent hat entsprechend der schwachen Agentenbeschreibung die folgenden Eigenschaften [37], die es in geeigneter Form für die jeweilige Anwendung umzusetzen gilt.

- Autonomie: Ein Agent agiert selbstständig und ohne direkte Einflussnahme von außen.
- Interaktion: Agenten agieren untereinander sowie mit dem Anwender.
- Reaktionsfreudigkeit: Veränderungen im Umfeld werden vom Agenten wahrgenommen und dieser reagiert darauf.
- Zielgerichtetes Verhalten: Agenten agieren eigenverantwortlich und zielgerichtet, anstatt ausschließlich auf ihre Umgebung zu reagieren.

Die starke Beschreibung eines Agenten erweitert die Eigenschaften um die

- Mobilität: Ein Agent ist in der Lage, selbstständig seinen Aktionsort zu bestimmen.
- Wahrhaftigkeit: Die zwischen den Agenten ausgetauschten Informationen werden nicht absichtlich verfälscht.
- Güte: Ein Agent versucht stets das zu tun, was von ihm erwartet wird.
- Rationalität: Agenten versuchen, das von ihnen angenommene Ziel zu erreichen.

Den beschriebenen Eigenschaften wird durch die Struktur, den Aufbau und das Verhalten der verwendeten Agenten Rechnung getragen. Eine schematische Darstellung eines Agenten stellt Abbildung 2.2 auf der Grundlage des Wissen-Wünsche-Ziele-Verhaltensmodells (BDI) dar [38].

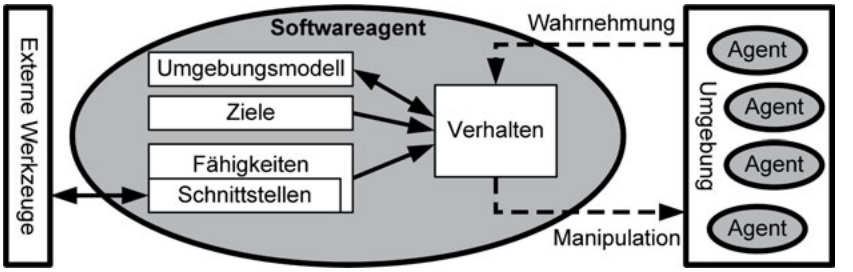

Abbildung 2.2: Aufbau eines Softwareagenten [39]

Ein Agent nimmt dabei Einflüsse aus seiner Umgebung wahr. Seine Wahrnehmungen beeinflussen sein internes Bild seiner Umgebung. Die Ziele des Agenten, das Modell der Umgebung sowie dessen Fähigkeiten bestimmen wiederum das Verhalten des Agenten. Durch sein Verhalten wirkt der Agent dabei auf seine Umgebung. Die Fähigkeiten des Agenten lassen sich durch bereitgestellte Werkzeuge erweitern. Dies setzt voraus, dass eine Schnittstelle zu deren Bedienung existiert und die Werkzeuge nicht Bestandteil der Umgebung sind.

Ein Verhalten stellt eine Abstraktion von Tätigkeiten des Agenten dar und dient zur Kapselung dieser voneinander [40]. Auf Grundlage des Verhaltens lassen sich Agenten als initiativ oder reagierend klassifizieren, wobei kombinierte Ausführungen möglich sind [37]. Reagierende Agenten werden dabei durch ihre Wahrnehmung gesteuert. Initiativ handelnde Agenten ziehen wiederkehrend Schlussfolgerungen aus Ihrem Umgebungsmodell und handeln daraufhin entsprechend verzögert. Vorteile einer initiativ handelnden Software für Prozesse innerhalb der Höchstleistungsrechnung zeigt [41] auf. Ermöglicht wird so beispielsweise eine erhöhte Toleranz der Berechnungsumgebung in Bezug auf Ausfälle. Eine ausführliche Diskussion verschiedener Verhaltensmodelle wird in [36] vorgenommen. Verschiedene Verhalten eines Agenten werden hier sequenziell ausgeführt. Ein Blockieren der Ausführung von Verhalten dient dazu die Hauptprozessoren (CPUs) freizugeben, falls diese temporär nicht benötigt werden.

Die Umsetzung von Multithreading für länger andauernde Aufgaben erlaubt es dem Agenten, immer aktionsfähig zu sein. Möglich wird so auch die Verwaltung von Anfragen nach nur einmal vorhandenen Ressourcen und das Verhindern von Blockierungen (Deadlocks).

Ein Beispiel für ein wesentliches Verhalten der Agenten ist die Kommunikation. Dabei ist zwischen der Kommunikation des Agenten mit dem Menschen als Anwender des Agentensystems, der Kommunikation mit externen Werkzeugen und der Kommunikation der Agenten untereinander zu unterscheiden. Im ersten Fall handelt es sich um eine Mensch-Maschine Kommunikation, die sich mittels entsprechender Anwenderschnittstellen oder Interface-Agenten umsetzen lässt [42]. Im zweiten Fall wird die Umsetzung einer Schnittstelle zur Steuerung und Überwachung des externen Werkzeugs notwendig. Im dritten Fall handelt es sich um eine Maschine zu Maschine (M2M) Kommunikation, die ein Merkmal der agentenorientierten Programmierung ist. Zur Umsetzung der M2M Kommunikation bedarf es einer einheitlichen Ontologie, die sich beispielhaft aus eindeutigen Erkennungsmerkmalen (Identifiern) und einem kommunikativen Handlungstyp zusammensetzt [43]. Der Inhalt der ausgetauschten Nachrichten lässt sich dabei um zusätzliche Informationen erweitern. Serialisierte Objekte stellen hierfür ein Beispiel dar. Allerdings ist auch deren eindeutige Identifizierung durch entsprechende Erkennungsmerkmale notwendig [44].

2.3 Softwareagentensysteme

Die Fähigkeiten eines Agenten, ihm zur Verfügung stehende Werkzeuge, seine Ziele und sein aufgebautes Umgebungsmodell stellen Alleinstellungsmerkmale eines Agenten dar. Die Kommunikation mehrerer individueller Agenten ermöglicht ein System, in dem Agenten gemeinschaftlich Aufgaben erledigen. Agenten fungieren dazu als Gruppe und kooperieren miteinander [13]. Entsprechend ist ein solches Agentensystem in der Lage, Aufgaben zu bewältigen, die ein einzelner Agent nicht handhaben kann. Die erfolgreiche Bearbeitung von Aufgaben durch ein Kollektiv von Softwareagenten stellt dabei ein Teilgebiet der künstlichen Intelligenz (KI) dar [45]. Abbildung 2.3 veranschaulicht den erweiterten Einflussbereich eines kooperierenden Kollektivs innerhalb der Umgebung. Eine Gruppierung der Agenten ist durch entsprechende Organisationsbeziehungen dargestellt.

Agentensysteme haben sich als bewährtes und wirkungsvolles Mittel beispielsweise in der Automatisierungstechnik etabliert [32]. So konnte die Planung von Lieferketten [46], eine flexible Gestaltung von Produktionsstraßen [11] oder die Reaktion auf unvorhergesehene Ereignisse im Produktionsablauf [47] mit reduziertem Aufwand, verglichen zu alternativen Realisierungen, umgesetzt werden. Diese Beispiele verdeutlichen, dass Agentensysteme gut geeignet sind, um komplexe Aufgaben mit vielen Randbedingungen innerhalb eines schwach gekoppelten Systems zu verarbeiten. Die geringe strukturelle Kopplung zwischen den Agenten

wird durch die Autonomie der einzelnen Agenten abgebildet. Diese führt, bezogen auf den Lösungsverlauf der Teilaufgaben, zu einer systembedingt individuellen Bearbeitung durch die Agenten und einer dezentralen Datenhaltung. Im Fall von gleichberechtigten Agenten existiert keine globale Systemkontrolle [36]. Vorteile, die sich daraus ergeben, sind eine hohe Flexibilität und Anpassungsfähigkeit des Agentensystems an dynamische Randbedingungen. Bei einer Umsetzung als offenes System folgt zudem eine gute und einfach umzusetzende Skalierbarkeit des Systems. Damit wird auch das Hinzukommen oder Entfernen von Agenten während der Laufzeit möglich. Dies schafft die Möglichkeit einer einheitsbezogenen Wartung und Weiterentwicklung [48]. Ausgehend von einem initiativen Handeln in einem Agentensystem erhöht sich zudem die Fehlertoleranz für ein Rechencluster, was zu einem robusteren und zuverlässigeren Gesamtsystem führt [49].

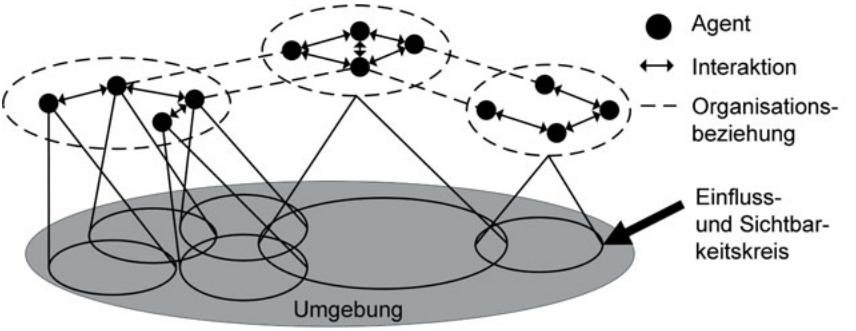

Abbildung 2.3: Zuständigkeitsdarstellung eines Agentensystems [11]

Da Agenten jedoch konzeptbedingt ihre Umwelt nie vollständig erfassen und situationsabhängig verschieden reagieren, ergibt sich ein nicht deterministisches Verhalten des Gesamtsystems [50]. Um sicherzustellen, dass es in Systemen mit autonomen Einheiten auch im Konfliktfall zu einem Ergebnis kommt, gilt es Agentensysteme zu koordinieren [51]. Beispiele für mögliche Konflikte stellen die überlappenden Einflusskreise in Abbildung 2.3 dar. So gilt es in einem gemeinsamen Einflussbereich mehrerer Agenten zu klären, welcher der Agenten eine Aufgabe bearbeitet. Bestehende Ansätze zur Koordination sind die Planung und die Organisation des Ablaufs. Alternativen stellen der Wettbewerb unter den Agenten, deren Verhandlung oder das sich gegenseitige Verpflichten dar [51]. Eine Umsetzung ist agentenunabhängig entweder zentral und hierarchisch, dezentral-kooperativ oder auf Basis eines Wettbewerbs möglich [52]. Abbildung 2.4 veranschaulicht, wo bei der zentralen und dezentralen Planung Entscheidungen getroffen werden. Der Wettbewerb stellt eine Mischform der Entscheidungsprozesse, abhängig von einer Betrachtung der Angebotsabgabe dar. Die zentrale Planung zeigt dabei große Ähnlichkeiten zu bestehenden Parallelisierungsansätzen.

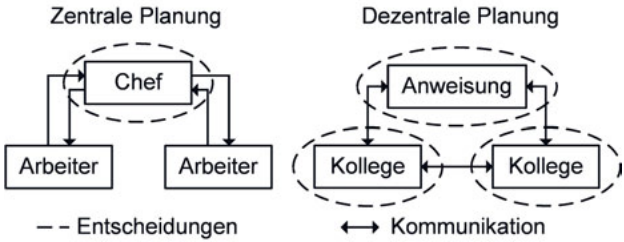

Abbildung 2.4: Hierarchische und kooperative Entscheidungsfindung [53]

Eine Koordination ist dabei vor und nach der Planung sowie während der Durchführung eines Verhaltens möglich [45]. In allen Fällen wird während einer Entscheidungsfindung ein Teil der Autonomie abgetreten [33]. Dabei lassen sich die folgenden Phasen der Koordination unterscheiden [52]. Zu Beginn gilt es einen Zielgraphen einschließlich der Bestimmung und Klassifikation von Abhängigkeiten zu definieren. Anschließend werden definierte Teilbereiche des Graphen durch befähigte Agenten bearbeitet. Es folgt die Steuerung, welche Bereiche des Graphen erkundet werden sollen, wobei die Ergebnisse anschließend bekannt geben werden. Je mehr Beteiligte an einer Aufgabe mitwirken und je mehr individuelle Ziele existieren, desto mehr Kommunikation bedarf es zu dessen Lösung.

Eine Vereinfachung im Hinblick auf die Kommunikation sowie die Koordination ist durch eine weitere Abstraktion entsprechend einer Einführung von Gruppen möglich. Gruppen sind auch im Hinblick auf eine Aufgabenteilung, eine Funktionserweiterung und der damit einhergehenden Kapselung von Agentenverhalten von Vorteil [54]. Dabei wird zwischen funktionellen, strukturellen und deontischen Gruppen unterschieden, wobei letztere Beziehungen zwischen funktionellen und strukturellen Gruppen ermöglichen [55]. Von der Entscheidung, welche Rolle ein Agent in einer Gruppe oder dem Agentensystem annimmt, hängt dessen spätere Tätigkeit ab. Dabei verpflichtet sich ein Agent nur zu Tätigkeiten, zu deren Bewältigung er sich für befähigt hält. Eine Selbstverpflichtung eines Agenten zu einer Rolle oder die Zuteilung einer Rolle zu einem Agenten ist dabei entweder während des Entwurfs oder während der Laufzeit des Agentensystems möglich. Dabei wird zwischen einer statischen, einer adaptiven und einer dynamischen Rolle unterschieden, die ein Agent annehmen kann. Tabelle 2.1 zeigt, wann die Zuteilung einer Rolle an einen Agenten bzw. dessen Auswahl einer Rolle sinnvoll ist.

Tabelle 2.1: Rollenzuteilung und Rollenauswahl eines Agenten [56]

	Zuweisung	Auswahl
statisch	Entwurf	Entwurf
adaptiv	Entwurf	Laufzeit
dynamisch	Laufzeit	Laufzeit

Auf Rollen und Gruppen aufbauende Sicherheitsmechanismen bieten zudem einen gewissen Schutz vor unberechtigtem Zugriff. Eine detaillierte Sicherheitsbetrachtung für Mobile-Agenten-Systeme findet sich in [57].

2.4 Herausforderungen

Durch das Paradigma der agentenorientierten Softwareentwicklung bedingte, sowie durch Erfahrungswerte bestätigte Herausforderungen bei der Umsetzung von Agentensystemen, werden im Folgenden beschrieben [12, 58]. Diese motivieren die gewählte Herangehensweise zur Umsetzung des Agentensystems und beschreiben gleichzeitig zu berücksichtigende Herausforderungen. Diese Herausforderungen finden sich aufgrund:

- der Einstellung zur agentenorientierten Programmierung, wie einem Überschätzen der Fähigkeiten von Agenten oder einem Verabsolutieren des Paradigmas,
- des Einsatzes von Agenten, wie mangelndes Wissen über deren Vor- und Nachteile, Unklarheit über den Mehrwert in Bezug auf die Umsetzung als Agenten und die Überschätzung eines Prototyps im Vergleich zu einem betriebsbereiten System,
- von konzeptionellen Fehlern in Bezug auf das Agentenkonzept und dessen Umsetzung, die den Grenzen der Softwareentwicklung sowie der verteilten Software unterliegen,
- der Analyse und im Entwurf, da der ausschließlich agentenbezogene Anteil verglichen zu anderen Programmteilen klein ist oder der Anteil paralleler Aufgabenteile zu klein ist,
- der Agenten selbst, da der Entwurf einer flexiblen Agentenplattform aufwendig ist und dieser anschließend häufig nur für spezielle Anwendungen funktioniert. Auch der Einsatz von zu viel künstlicher Intelligenz (zum proaktiven Handeln) oder von zu wenig (für eine stückweise Abarbeitung der Aufgaben) ist hinderlich,
- der Interaktionen der Agenten im Fall von zu vielen oder zu wenig Agenten. Weitere Beispiele sind eine Fokussierung auf die Infrastruktur statt auf die Aufgabe oder die Betrachtung von Teilaufgaben deren Zusammengehörigkeit unklar ist bzw. eine unklare Abgrenzung zwischen paralleler Implementierung und verteilten Systemen existiert,
- der Implementierung durch außer Acht lassen von existierenden Strukturen oder Standards, die sich beispielsweise durch eine Schnittstelle weiternutzen lassen.

Zudem lassen sich die Muster und Ergebnisse einer Kooperation grundsätzlich nicht vorhersehen [11]. Das Systemverhalten vorherzusagen ist somit sehr schwierig bis unmöglich, gerade weil das System sich durch seine Entscheidungen weiter entwickelt. Dies ist ein wesentlicher Unterschied zu Ansätzen der Parallelisierung, deren Verhalten als deterministisch zu betrachten sind.

2.5 JADE

Da die Entwicklung einer eigenständigen Agentenumgebung nicht Bestandteil dieser Arbeit ist (Gründe siehe Kapitel 2.4), wird hier auf eine bestehende Entwicklungsumgebung zurückgegriffen. Vergleiche verschiedener bestehender Architekturen finden sich beispielsweise in [48, 59] und [60]. Darin stellt die Java basierte Agentenentwicklungsumgebung JADE eine praxiserprobte, plattformunabhängige, erweiterbare und seit mehreren Jahren stetig weiterentwickelte Entwicklungsumgebung für verteilte Agentensysteme dar [61]. Sie unterstützt die Entwicklung von Systemeinheiten, die anhand vorgegebener Regeln initiativ handeln, eine Kommunikation und Verhandlungen zwischen verteilten gleichberechtigten Einheiten benötigen und komplexe Aufgaben koordiniert und verteilt lösen [62]. Der Aufbau der Agentenplattform ist in Abbildung 2.5 dargestellt.

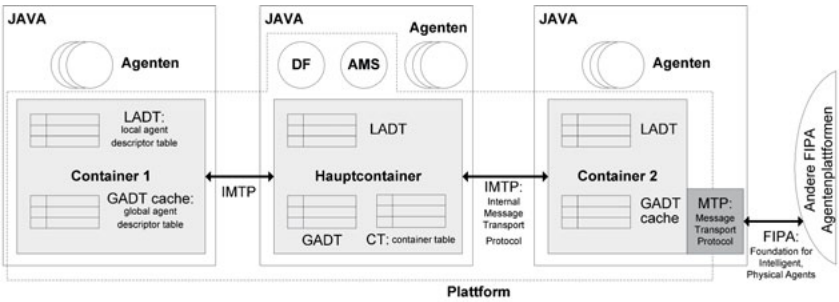

Abbildung 2.5: Aufbau und Zusammenhang der JADE Komponenten [61]

Zum Betrieb ist dabei eine virtuelle Javamaschine (JVM) je eingesetztem Rechner notwendig. Diese beinhaltet mindestens einen Container, der als Ausführungsumgebung für Agenten dient. Zusätzliche Tabellen, wie die Beschreibung lokal ausgeführter Agenten (LADT), die Beschreibung global ausgeführter Agenten (GADT) oder die Containertabelle (CT) dienen zur Verwaltung der Agentenplattform. Sie ermöglichen die verteilte Ausführung der Agenten innerhalb verschiedener JVM. Die Verwaltung der Tabellen erfolgt durch das Agentensystem, falls die innerhalb der Container betriebenen Agenten die dafür notwendigen Verhalten erfolgreich ausführen. Kommt es beispielsweise zu einem abrupten Ausfall eines Containers, so gilt es die Gültigkeit der Tabellen GADT und CT wieder herzustellen [63]. Eine Erweiterung um Agenten mit einer FIPA (Gesellschaft für intelligente physikalische Agenten) konformen Kommunikation ist ebenfalls möglich.

Sonderrollen innerhalb der Plattform nehmen der als Agent Management System (AMS) bezeichnete Agent und der Directory Facilitator (DF) Agent ein. Der AMS Agent stellt dabei die einzige Instanz dar, die Autorität im Agentensystem genießt. Ihm ist es neben dem Anwender

möglich, Verwaltungsaufgaben wie das Starten oder Beenden von Agenten durchzuführen. Der DF Agent stellt Beschreibungen der durch die Agenten angebotenen Dienste zentral bereit. Für jeden Agenten besteht so die Möglichkeit nach Diensten zu suchen, eigene Dienste bereitzustellen, die Dienstbeschreibung zu ändern oder seine Dienste abzumelden. Zusätzliche optionale Werkzeuge der Agentenentwicklungsumgebung erlauben eine Überwachung und Verwaltung der Aktivitäten von Agenten [64].

Einen Einblick in die Entwicklung von Agentensystemen mit JADE bieten [61] und [65]. Untersuchungen in Bezug auf die Effizienz und Leistungsfähigkeit von mit JADE realisierbaren Agentensystemen sind in [66, 67] und für den Nachrichtenaustausch innerhalb von JADE in [68] zu finden. Die Leistungsfähigkeit des Agentenansatzes ist in [69] anhand eines Vergleichs zwischen agentenbezogenem parallelisiertem Java und mittels MPI parallelisiertem C-Quelltext dargestellt. Ein Vergleich zwischen mittels MPI parallelisiertem C-Quelltext und dem Agentenansatz zeigt zudem, dass sich individuelle Softwareeinheiten realisieren lassen, die eine fehlertolerante und dynamische Verwaltung ermöglichen. Eine auf die Cloud angepasste Orchestrierung eines mit JADE entwickelten Agentensystems und dessen dortiger Betrieb schildert [70].

3 Numerische Berechnung gekoppelter Feldsimulationen

Um Softwareagenten zur Berechnung numerischer Simulationen zu entwickeln, wird in diesem Kapitel eine kurze Einführung in eine der Feldberechnungsmethoden, die Methode der finiten Elemente (FEM), gegeben. Die FEM findet in dieser Arbeit aufgrund ihrer vielfältigen Einsatzgebiete in mehreren Ingenieursdiziplinen Anwendung. Dabei wird sie am Beispiel von Fragestellungen der Elektrostatik sowie der Kopplung mit stationären thermischen Simulationen erläutert. Darauf aufbauend werden verschiedene Arten der Kopplung und deren mathematische Darstellungsarten betrachtet. Diese werden in ein Gleichungssystem überführt und dessen Lösung anhand verschiedener Lösungsverfahren dargestellt. Den Ablauf der Berechnung und die Eigenschaften der Lösungsverfahren gilt es dabei, in geeigneter Form bei der Umsetzung in einem Agentensystem zu berücksichtigen. Anschließend werden die Anforderungen an Berechnungsumgebungen zur Lösung gekoppelter Simulationen aufgezeigt. Diese Anforderungen werden abschließend auf eine Auswahl bestehender Berechnungsumgebungen für gekoppelte Simulationen übertragen und sich von diesen abgegrenzt.

3.1 Methode der finiten Elemente im Überblick

Die FEM ist ein in mehreren Ingenieursdisziplinen eingesetztes numerisches Verfahren zur Simulation physikalischer Zusammenhänge. Die Überführung eines als partielle Differenzialgleichung (PDE) formulierten physikalischen Zusammenhangs in ein numerisch berechenbares Gleichungssystem fasst Abbildung 3.1 zusammen.

Abbildung 3.1: Schritte vom physikalischen Zusammenhang zum Gleichungssystem [71]

Die dargestellten Schritte werden im Folgenden am Beispiel der Poisson-Gleichung der Elektrostatik im Vakuum

$$\rho = \operatorname{div}(\boldsymbol{D}) = \varepsilon_0 \operatorname{div}(\boldsymbol{E}) = -\varepsilon_0 \operatorname{div}(\operatorname{grad}(\varphi)) = -\varepsilon_0 \Delta\varphi \tag{3.1}$$

mit dem skalaren elektrischen Potenzial φ und der Raumladungsdichte ρ skizziert [72]. Dabei gilt (3.1) im Inneren eines zusammenhängenden Gebietes Ω, dessen Rand Γ nicht Teil des Gebiets ist. Auf einem Randstück Γ_D gelte die Dirichlet-Randbedingung $\varphi = \varphi_0(\boldsymbol{x})$, wobei $\boldsymbol{x}$ den Ortsvektor eines Punktes darstellt. Eine Ergänzung der Randbedingung ist durch Neumannrandbedingungen $\frac{\partial\varphi}{\partial\boldsymbol{n}}$ auf dem Rand Γ_N möglich. Dabei entspricht $-\frac{\partial\varphi}{\partial\boldsymbol{n}}$ der Normalkomponente der elektrischen Feldstärke $\boldsymbol{E}$ mit $\boldsymbol{n}$ als Normale auf dem Rand Γ_N. Für den Rand gilt $\Gamma_D \cup \Gamma_N = \Gamma$ und $\Gamma_D \cap \Gamma_N = \emptyset$. Die Multiplikation von (3.1) mit einer Testfunktion ν, die auf dem betrachteten Gebietsrand Γ verschwindet und eine anschließende Integration über Ω ergibt

$$\int_\Omega \left(\Delta\varphi + \frac{\rho}{\varepsilon_0}\right)\nu \mathrm{d}\Omega = 0, \tag{3.2}$$

falls φ eine Lösung ist (Methode der gewichteten Residuen). Die partielle Integration mittels des Green'schen Satzes ergibt das schwach formulierte Randwertproblem

$$-\int_\Omega \mathrm{grad}\varphi \cdot \mathrm{grad}\nu \, \mathrm{d}\Omega + \int_{\Gamma_N} \frac{\partial\varphi}{\partial\boldsymbol{n}}\nu \mathrm{d}\Gamma - \int_\Omega \frac{\rho}{\varepsilon_0}\nu \mathrm{d}\Omega = 0. \tag{3.3}$$

Die gesuchte Größe φ kann prinzipiell mit verschiedenen Ansatzverfahren, wie der Kollokation, der Fehlerquadratmethode, dem Rayleigh-Ritz-Ansatz oder der gleichwertigen Galerkin Approximation angenähert werden [73]. Dabei wird eine Linearkombination unabhängiger Formfunktionen N_i mit den Koeffizienten φ_i genutzt, um die gesuchte Größe

$$\varphi \approx \sum_{i=1}^{n} \varphi_i N_i \tag{3.4}$$

zu beschreiben. Unter anderem lassen sich bei der Galerkin-Methode die Formfunktionen N_i auch als Testfunktion ν nutzen. Für eine möglichst genaue Approximation wird das betrachtete Gebiet Ω in eine endliche Anzahl deformierter Referenzelemente Ω_e mit $1 \leq e \leq E$ zerlegt. Abbildung 3.2 zeigt beispielhaft eine die Zerlegung eines zweidimensionalen Gebiets unter Verwendung linearer Dreiecke.

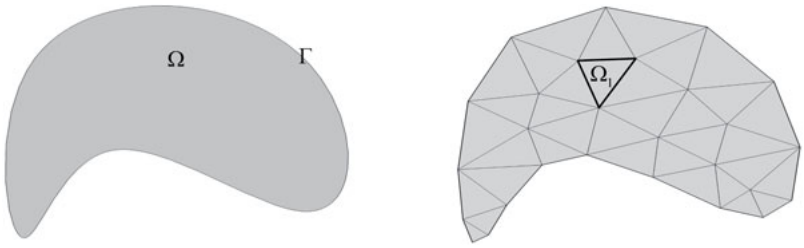

Abbildung 3.2: Modellierung und Diskretisierung einer Geometrie

Dabei ist die hier angewendete Diskretisierung $\mathcal{D} := \Omega_1 \dots \Omega_E$ zulässig, falls Ω_e offene Dreiecke sind, die den Rand nicht beinhalten. Zudem ist Ω_e disjunkt und für $e \neq j$ entsprechend $\Omega_e \cap \Omega_j = \emptyset$ zu wählen. Für $\bar{\Omega} = \bigcup_e \bar{\Omega}_e$ mit $e \neq j$ ist dabei $\bar{\Omega}_e \cap \bar{\Omega}_j$ entweder leer, eine gemeinsame Seite, Kante oder Ecke der Elemente Ω_e, Ω_j. Dabei wird impliziert, dass Ω Polygongestalt hat [74].

Für die erzeugte Diskretisierung $\mathcal{D}$ sind die Formfunktionen N_i für die Referenzelemente zu wählen. Gebräuchlich sind knotenbasierte Formfunktionen für skalare Felder und kantenbasierte Formfunktionen für Vektorfelder. Die gesuchten Koeffizienten werden dabei entweder den Knoten oder den Kanten zugeordnet [75]. Die Formfunktionen gelten als konform, wenn der Übergang zweier Elemente C^a stetig ist. Der Exponent a stellt dabei den Grad der Ableitung dar, für den die Stetigkeit der Lösung sichergestellt ist. Unterscheiden sich das Polynom zur Diskretisierung der Geometrie und die Formfunktion bei der Berechnung, so wird dabei zwischen sub-, iso- oder superparametrischen Elementen unterschieden. Diese Unterscheidung ist bei der Interpretation der berechneten Ergebnisse zu beachten [76]. Beispielsweise für runde Körper entstehen durch die endliche Anzahl E der deformierten Referenzelemente Fehler. Die Qualität einer Diskretisierung lässt sich anhand verschiedener Fehlerindikatoren vor der Lösung und mittels Fehlerschätzern nach erfolgter Berechnung bestimmen [77, 78]. Eine Berücksichtigung der Qualität in Form einer adaptiv angepassten Diskretisierung wirkt sich positiv auf die Robustheit der Lösungsfindung, die Qualität der Lösung und die Effizienz der Berechnung aus [79].

Einsetzen von (3.4) in (3.3) führt bei elementeweiser Betrachtung und gegenseitiger Auslöschung der Randintegrale zu

$$-\sum_{i=1}^{n} \int_{\Omega_e} \varepsilon_0 \operatorname{grad}(N_{ie}) \cdot \operatorname{grad}\left(N_{je}\right) \mathrm{d}\Omega \, \varphi_{ie} - \int_{\Omega_e} \rho N_{je} \mathrm{d}\Omega = 0 \quad \text{mit} \quad j = 1 \dots n. \tag{3.5}$$

Anschaulicher ist die Betrachtung mittels des Ritz-Ansatzes. Dabei wird genutzt, dass die Lösung vieler Randwertaufgaben gleichzeitig Lösung der zugehörigen Variationsaufgabe ist. Das betrachtete Funktional beschreibt die Energie W, die hier die Energie des elektrischen Feldes W_E und der einer Ladungsverteilung W_Q berücksichtigt. Es lässt sich als

$$W = W_E + W_Q = \frac{1}{2\varepsilon_0} \int_{\Omega} \left((\operatorname{grad}\varphi)^2 - 2\rho\varphi \right) \mathrm{d}\Omega \tag{3.6}$$

beschreiben. Die Berechnung des Minimums dieses Energiefunktionals mittels (3.4) führt ebenfalls auf (3.5). Die dabei durchzuführende Integration erfolgt numerisch unter Auswertung der Gaußpunkte, die definierte Punkte innerhalb des Referenzelement Ω_e darstellen. Dies ermöglicht eine Darstellung von (3.5) in Matrixschreibweise als

$$\boldsymbol{K}\boldsymbol{u} = \boldsymbol{r}. \tag{3.7}$$

Vorausgesetzt wird, dass $\boldsymbol{K}$ und $\boldsymbol{r}$ unabhängig von $\boldsymbol{u}$ sind. Dabei entsprechen die abhängigen Variablen $\boldsymbol{u}$ dem gesuchten Potenzial φ an den Stützstellen der Funktion N_i und $\boldsymbol{r}$ der Summe der im Element Ω_e befindlichen Ladung Q. Eine Berücksichtigung zusätzlicher Ladungsquellen führt zu einer Modifikation des Lastvektors $\boldsymbol{r}$. Zur Berücksichtigung einer Polarisation ($\varepsilon = \varepsilon_0 \varepsilon_r$) ist hingegen eine Änderung der Systemmatrix $\boldsymbol{K}$ erforderlich.

Die Systemmatrix $\boldsymbol{K}$ hat dabei die Eigenschaft schwach besetzt zu sein, da jedes finite Element nur wenige direkte Nachbarelemente hat. Zudem ist die Systemmatrix quadratisch, symmetrisch und hat bei entsprechender Nummerierung der Freiheitsgrade eine Bandstruktur, die sich zur effizienten Speicherung nutzen lässt [80]. Die Konditionszahl der Systemmatrix $\boldsymbol{K}$, entsprechend dem Verhältnis zwischen größtem und kleinstem Eigenwert von $\boldsymbol{K}$ ist groß (Erfahrungswerte: $10^2 \ldots 10^6$). Derartige Gleichungssysteme gilt es zu vermeiden, da kleine Fehler in der Systemmatrix zu großen Fehlern im Lösungsvektor führen. Zudem bedeutet eine große Konditionszahl ein schlechtes Konvergenzverhalten bei iterativen Lösungsverfahren [81].

Das Gleichungssystem ist beispielsweise mit dem iterativen Newton-Verfahren

$$\boldsymbol{\varphi}^{[s+1]} = \boldsymbol{\varphi}^{[s]} + \bar{\boldsymbol{J}}^{-1} \left(\boldsymbol{\varphi}^{[s]} \right) \boldsymbol{f} \left(\boldsymbol{\varphi}^{[s]} \right), \tag{3.8}$$

mit $\boldsymbol{f}$ entsprechend (3.5), s als Iterationszähler und $\boldsymbol{\varphi}^{[0]}$ als beliebiger Startwert berechenbar. Die invertierte Jacobimatrix $\bar{\boldsymbol{J}}^{-1}$ beinhaltet dabei die partiellen Ableitungen von $\boldsymbol{f}$ nach den gesuchten Größen φ_m mit $m = 1 \ldots n$. Um die Berechnung von $\bar{\boldsymbol{J}}^{-1}$ zu vermeiden, wird der Rechenschritt $\Delta\boldsymbol{\varphi}^{[s]}$ eingeführt, sodass mit

$$\bar{\boldsymbol{J}} \left(\boldsymbol{\varphi}^{[s]} \right) \Delta\boldsymbol{\varphi}^{[s]} = -\boldsymbol{f} \left(\boldsymbol{\varphi}^{[s]} \right) \tag{3.9}$$

iterativ die Lösung von $\boldsymbol{\varphi}^{[s+1]} = \boldsymbol{\varphi}^{[s]} + \Delta\boldsymbol{\varphi}^{[s]}$ bestimmbar ist. Die Berechnung wird nach k Iterationen beendet, wenn für den Restfehler ϵ

$$\epsilon > \| \boldsymbol{\varphi}^{[k]} - \boldsymbol{\varphi}^{[k-1]} \| \tag{3.10}$$

gilt [82]. Für eine lineare Fragestellung $\boldsymbol{f}$ mit null als Startwert $\boldsymbol{\varphi}^{[0]}$ ergibt sich der Zusammenhang (3.7). Dabei ist die Systemmatrix $\boldsymbol{K}$ gleich der Jacobimatrix $\bar{\boldsymbol{J}}(\boldsymbol{0})$, der Lastvektor $\boldsymbol{r}$ gleich $\boldsymbol{f}(\boldsymbol{0})$ und die abhängigen Variablen $\boldsymbol{u}$ entsprechen dem einzigen Rechenschritt $\Delta\boldsymbol{\varphi}^{[0]}$ an den Stützstellen der Funktion N_i. Die Berücksichtigung nicht linear von φ abhängiger Ladungen $Q(\varphi)$ bedeutet eine Änderung innerhalb von $\boldsymbol{r}$ und somit auch von $\boldsymbol{f}$. Eine Berücksichtigung nicht linearer Materialparameter, wie $\varepsilon_r(\varphi)$ führt zu einer Änderung in $\boldsymbol{K}$. Lässt sich die partielle Ableitung $\frac{\partial \boldsymbol{K}}{\partial \varphi}$ bestimmen, so wird diese Änderung in $\bar{\boldsymbol{J}}$ berücksichtigt. Zudem erfolgt die Berücksichtigung des linearisierten Materialparameters in $\boldsymbol{f}$.

3.2 Gekoppelte Simulationen

Werden in einer Simulation physikalisch- oder berechnungsbedingt mehrere Systeme zusammen untersucht und beeinflussen sich diese gegenseitig, so ist von einer gekoppelten Simulation die Rede. Dabei handelt es sich immer um die Lösung einer Mehrfeldfragestellung, bei der sich einzelne Feldvariablen identifizieren lassen, die unterschiedlich diskretisiert werden können [83]. Die Kopplung zweier Feldsimulationen ist auf mehrere Arten möglich. Angelehnt an [84] teilt Abbildung 3.3 die Kopplungen in zeit- und raumbasierte Kopplungen auf. Zusätzlich sind Kombinationen mehrere dieser Kopplungen gängig.

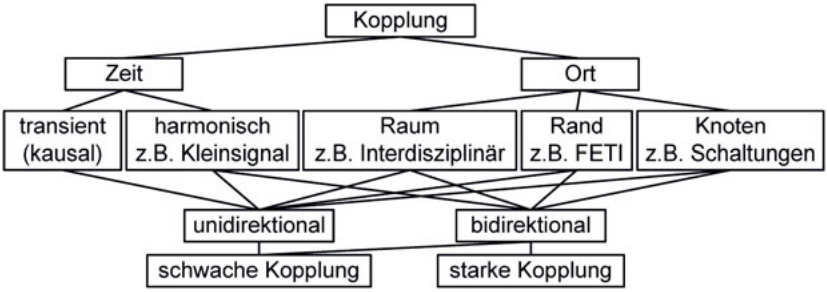

Abbildung 3.3: Umsetzungen von Kopplungen zweier Teilsimulationen (angelehnt an [84])

Zeitbasierte Kopplungen stellen eine Beziehung zwischen einer Simulation zu verschiedenen Zeitpunkten dar. Da es sich bei einer Betrachtung technischer Vorgänge um kausale Zusammenhänge handelt, erfolgt die Kopplung transienter Vorgänge immer unidirektional von der Vergangenheit in die Zukunft. Bei einer Kopplung verschiedener Frequenzen, wie beispielsweise bei einer Kleinsignalanalyse, treten auch bidirektionale Kopplungen auf. Ortsabhängige Kopplungen sind Einflüsse verschiedener Gebiete aufeinander. Dies ist an definierten Knoten, wie beispielsweise bei einer Betrachtung diskreter Bauteile bei einer Schaltungssimulation, möglich. Zudem existieren Kopplungen zwischen aneinander grenzenden Gebietsrändern, wie bei der Berechnung mittels Gebietszerlegung (siehe Kapitel 3.2.4). Auch innerhalb überlappender Gebiete treten Kopplungen auf, wie dies beispielsweise bei interdisziplinären Simulationen der Fall ist. Diese analysieren verschiedene sich gegenseitig beeinflussende physikalische Effekte und deren Wechselwirkung. Die betrachteten Größen sind dabei den jeweiligen Ingenieursdisziplinen zugeordnet. Ein Beispiel stellt die Auswertung von elektro-thermischen Fragestellungen dar, wobei das elektrische Potenzial φ zur Disziplin der Elektrotechnik und die Temperatur T der Thermodynamik zugeordnet werden. Dabei sind allen ortsabhängigen Kopplungen sowohl uni- als auch bidirektionale Kopplungen möglich und lassen sich als gerichtete Graphen entsprechend Abbildung 3.4 darstellen.

Abbildung 3.4: Darstellung der Kopplung als Graph (unidirektional links, bidirektional rechts)

Die Adjazenzmatrix $\tilde{\boldsymbol{C}}$ einer unidirektionalen Kopplung zweier Feldsimulationen ist $\begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}$. Die unidirektionale Kopplung von n Feldsimulationen ist dabei stets eine $1 : n$ Beziehung. Dabei werden Feldgrößen ausschließlich in einer Feldsimulation berechnet und liefern einen Beitrag zur Berechnung der gekoppelten Größen. Die Adjazenzmatrix $\tilde{\boldsymbol{C}}$ einer bidirektionalen Kopplung zweier Feldsimulationen ist $\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$. Kopplungen von drei und mehr Teilsimulationen setzen sich aus den in Abbildung 3.4 gezeigten Kopplungen zusammen, wobei nicht alle Teilsimulationen untereinander gekoppelt sein müssen. Kopplungsabhängig weist der Graph dabei auch hängende Knoten, Kreise oder Schleifen auf. Die in Abbildung 3.3 dargestellten Kopplungen werden im Folgenden näher betrachtet.

3.2.1 Stark oder schwach gekoppelte Simulationen

Bei einer Kopplung zweier Teilsimulationen wird zwischen einer starken oder schwachen Kopplung unterschieden [85]. Diese mögliche Unterscheidung richtet sich nach dem Einfluss der Lösung der beiden Teilsimulationen aufeinander.

- Eine starke Kopplung liegt dann vor, wenn die Lösung beider Teilsimulationen signifikant von der Genauigkeit der anderen abhängt.
- Von einer schwachen Kopplung ist die Rede, falls die Änderung der Lösung durch eine Berücksichtigung der Teillösung gering ist.

Die schwache Kopplung trifft stets auf unidirektional gekoppelte Systeme zu, da eine der Teillösungen keinen Einfluss auf die andere hat. Ein Beispiel hierfür ist die Simulation eines kausalen Systems zu verschiedenen Zeitpunkten. Ein Beispiel für die starke Kopplung stellt die Fluid-Struktur-Kopplung einer schwingenden Wassersäule in einem elastischen Rohr dar [86]. Folglich wird eine Gewichtung der Kopplung unter Berücksichtigung der gewählten abhängigen Variablen, der Materialparameter und den Werten der Kopplungsparameter notwendig. Dass diese Gewichtung subjektiv ist, wird im Fall einer nicht linearen Kopplung deutlich, bei der sich Kopplungsparameter startwertabhängig verändern [83]. Aus Gründen der Vollständigkeit wird darauf hingewiesen, dass es sich im Fall einer Gebietskopplung um eine lineare Kopplung handelt, sodass hier die Unterscheidung in stark und schwach gekoppelte Systeme möglich ist.

In [84] wird daher eine rein numerische Definition der Begriffe vorgeschlagen. Dabei ermöglicht eine numerisch schwache Kopplung im Gegensatz zur starken Kopplung eine sequenzielle Lösung. Im Folgenden wird ausgehend von dieser Definition ein Agentensystem entwickelt, das

schwach gekoppelte Simulationen verteilt berechnet. Stark gekoppelte Simulationen werden von einem einzelnen Softwareagenten bearbeitet.

3.2.2 Zeitliche Kopplungen

Sollten statt der in (3.7) betrachteten stationären Differenzialgleichung auch zeitabhängige Größen, wie beispielsweise die zeitabhängige elektrische Volumenladungsdichte $\rho(t)$ die Verschiebungsstromdichte $\frac{\partial \boldsymbol{D}}{\partial t}$ berücksichtigt werden, ist dies im quasistationären Fall durch Lösung von

$$\boldsymbol{K}\boldsymbol{u} + \boldsymbol{M}\frac{\partial \boldsymbol{u}}{\partial t} = \boldsymbol{r} \tag{3.11}$$

möglich. Dabei beinhaltet die Massenmatrix $\boldsymbol{M}$ die Koeffizienten, die sich bei der Galerkin-Approximation für den zeitabhängigen Anteil $\frac{\partial \boldsymbol{u}}{\partial t}$ ergeben. Analog zu $\boldsymbol{K}$ und $\boldsymbol{r}$ wird auch hier für $\boldsymbol{M}$ vorausgesetzt, dass sie unabhängig von $\boldsymbol{u}$ ist. Zur Vollständigkeit sei die Klasse der steifen Probleme erwähnt, bei der kleine Änderungen innerhalb eines Zeitschritts große Änderungen der zeitabhängigen Lösung bewirken [87, 88]. Eine Besonderheit stellen zeitharmonische Vorgänge dar. Die komplexe Betrachtung der frequenzabhängigen Lösungsvariablen $\mathfrak{u}$ mit

$$\underbrace{(\boldsymbol{K} + \mathrm{j}\omega\boldsymbol{M})}_{\boldsymbol{K}'}\,\mathfrak{u} = \boldsymbol{r} \tag{3.12}$$

ermöglicht eine stationäre Berechnung. Dasselbe gilt für im Frequenzbereich ausgewertete Zusammenhänge, die sich dort durch das Zusammenwirken mehrerer Frequenzen beschreiben lassen. Eine Anwendung dessen stellt die multiharmonische Betrachtung transienter Vorgänge dar [89, 90]. Dazu werden die Lösungsvariablen $\boldsymbol{u}$ durch Fourierreihen approximiert und als

$$\boldsymbol{u} = \Re\left(\sum_{k=0}^{N} \mathfrak{u}_k \cdot e^{\mathrm{j}\omega k t}\right) = \frac{1}{2}\sum_{k=0}^{N} \mathfrak{u}_k \cdot \left(e^{\mathrm{j}\omega k t} + e^{-\mathrm{j}\omega k t}\right) \tag{3.13}$$

mit $\Re$ als Operator zur Berechnung des Realteils dargestellt. Angenommen wird dabei, dass die abhängigen Variablen $\boldsymbol{u}$ reell sind und die Betrachtung von N harmonischen Frequenzen ausreicht [91]. Die Anregung selbst sowie die physikalischen Gleichungen werden mittels Fouriertransformation

$$\mathfrak{u}_k = \frac{n}{\tau}\int_{t=0}^{\tau} \boldsymbol{u} \cdot e^{-\mathrm{j}\omega k t}\,\mathrm{d}t \qquad n = \begin{cases} 1 \text{ für } k = 1 \\ 2 \text{ sonst} \end{cases} \tag{3.14}$$

in den Frequenzbereich überführt und dort berechnet. Die Periode des Signals ist τ mit $\omega = \frac{2\pi}{\tau}$. Die Rücktransformation des frequenzabhängigen Ergebnisses ergibt die Lösung im Zeitbereich. Diese Herangehensweise ermöglicht eine Systemanalyse ausgehend von den N verschiedenen

Frequenzen der Fourierreihenapproximation. Zudem wird für relativ schwach nicht lineare Zusammenhänge der physikalischen Gleichungen eine Ersparnis an Rechenzeit und eine Reduktion des Ressourcenbedarfs erreicht [89, 92].

3.2.3 Raumbasierte Kopplungen

Raumbasierte Kopplungen ermöglichen es, Felder verschiedener ortsabhängiger Variablen $\boldsymbol{u}$ und deren Wechselwirkung in einer Simulation zu betrachten. Sie sind sowohl innerhalb einzelner Disziplinen als auch, bedingt durch angrenzende Effekte, disziplinübergreifend anzutreffen. Am Beispiel eines stationären elektrischen Strömungsfeldes, das aufgrund der Stromwärme q eine Quelle einer thermischen Simulation darstellt, wird die Mehrfeldkopplung im Folgenden exemplarisch erläutert. Unter Berücksichtigung der temperaturabhängigen elektrischen Leitfähigkeit $\kappa(T)$ ergibt sich dabei eine bidirektionale Mehrfeldkopplung. Das Strömungsfeld wird hier durch eine FEM-Simulation des elektrischen Potenzials φ berechnet und mit einer FEM-Simulation der Temperatur gekoppelt. Der physikalische Zusammenhang einer stationären thermischen Betrachtung ist die Wärmeleitungsgleichung

$$\operatorname{div}\left(\kappa_t \operatorname{grad}(T)\right) = q \tag{3.15}$$

mit der thermischen Leitfähigkeit κ_t und der Wärmequelldichte q. Analoges Vorgehen zu Kapitel 3.1 liefert für lineare isotrope Materialien

$$\sum_{i=1}^{n} \int_{\Omega_e} \kappa_t \operatorname{grad}(N_{ie}) \cdot \operatorname{grad}\left(N_{je}\right) \mathrm{d}\Omega \, T_{ie} = \int_{\Omega_e} q N_{je} \mathrm{d}\Omega. \tag{3.16}$$

Die Quelldichte der Energieflussdichte $\operatorname{div}(\boldsymbol{S}) = -\boldsymbol{J} \cdot \boldsymbol{E}$ beschreibt im stationären Fall die Leistungsdichte der Stromwärme [93]. Für die Wärmequelldichte q gilt daher

$$q = \boldsymbol{J} \cdot \boldsymbol{E} = \kappa \boldsymbol{E} \cdot \boldsymbol{E} = \kappa (\operatorname{grad}(\varphi))^2. \tag{3.17}$$

Einsetzen von (3.17) in (3.15) liefert eine nicht linear und unidirektional gekoppelte Beschreibung der Zusammenhänge

$$\operatorname{div}\left(\kappa_t \operatorname{grad}(T)\right) = \kappa (\operatorname{grad}(\varphi))^2 + q_0. \tag{3.18}$$

Der aufgrund der Kopplung in (3.16) zusätzlich zu berücksichtigende Quellterm ist

$$\int_{\Omega_e} \kappa \left(\operatorname{grad}(\varphi)\right)^2 \cdot N_{je} \, \mathrm{d}\Omega. \tag{3.19}$$

Die temperaturabhängige elektrische Leitfähigkeit $\kappa(T)$ beeinflusst lediglich die existierende schwache Formulierung des elektrischen Potenzials φ entsprechend

$$\sum_{i=1}^{n} \int_{\Omega_e} \kappa(T)\, \mathrm{grad}\,(N_{ie}) \cdot \mathrm{grad}\left(N_{je}\right) \mathrm{d}\Omega\, \varphi_{ie} = 0. \tag{3.20}$$

Hierbei wird von einer stationären Stromdichte $\boldsymbol{J}$ ohne Wirbelströme (rot $\boldsymbol{E} = \boldsymbol{0}$) in einem Medium mit der elektrischen Leitfähigkeit κ ausgegangen. Sowohl in (3.19) als auch in (3.20) ist zur Berechnung die Koppelgröße als eine Linearkombination aus Formfunktionen und Koeffizienten für die gekoppelten Größen entsprechend (3.4) zu wählen. Die entstandenen zwei Teilsimulationen lassen sich bei jeweiliger Berechnung mittels des Newton-Verfahrens entsprechend (3.9) allgemein als

$$\begin{aligned} &\text{Simulation 1:} \quad \bar{\boldsymbol{J}}_1\, \Delta \boldsymbol{u}_1 = -\boldsymbol{f}_1 \\ &\text{Simulation 2:} \quad \bar{\boldsymbol{J}}_2\, \Delta \boldsymbol{u}_2 = -\boldsymbol{f}_2 \end{aligned} \tag{3.21}$$

mit den abhängigen Variablen $\boldsymbol{u}_1 = \boldsymbol{u}_1^{(0)} + \sum_{n=1}^{k} \Delta \boldsymbol{u}_1^{(n)}$ und analog $\boldsymbol{u}_2$ beschreiben. Da zu berücksichtigende Quellen, Randbedingungen oder Materialeinflüsse Abhängigkeiten voneinander aufweisen, ist eine Berechnung beider Teilsimulationen unter gegenseitiger Berücksichtigung der Ergebnisse sinnvoll. Eine Unterscheidung der zur Kopplung verwendeten Materialparameter und Quellen in linear und nicht linear ist dabei analog zu ungekoppelten Simulationen möglich. Da die Kopplungen zudem modellspezifisch verschieden stark ausgeprägt sind, werden in der Praxis häufig Kopplungen nur unidirektional berücksichtigt.

Die Diskretisierung beider Simulationen ist unter Berücksichtigung der Anforderungen der jeweiligen Teilsimulationen empfehlenswert [94]. Dabei gilt es, Anforderungen der Materialien, der Randbedingungen und der physikalischen Zusammenhänge zu berücksichtigen, sodass verschieden aufwendige Teilsimulationen entstehen. Neben den in Kapitel 3.3 beschriebenen Vorteilen verschiedener Diskretisierungen wird auch eine Nutzung bestehender ergebnisbezogener Fehlerschätzer zur Verfeinerung der Diskretisierung möglich. Allerdings sind aufgrund der Kopplung auch Werte der anderen Teilsimulation zu berücksichtigen. Zu deren Bestimmung werden unterschiedliche Interpolations- und Projektionsverfahren eingesetzt, die abhängig vom Verhältnis der Diskretisierungsdichte unterschiedlich stark fehlerbehaftet sind [95–97]. Eine mögliche Alternative stellt die einheitliche Vernetzung unter Berücksichtigung aller Anforderungen dar. Dieser in der Praxis häufig anzutreffende Fall erfordert aufgrund der widersprüchlichen Anforderungen jedoch Kompromisse zur Optimierung der Diskretisierung.

Eine Beschreibung möglicher Lösungsarten von (3.21) erfolgt in Kapitel 3.3. Weitere Beispiele für feldgekoppelte Anwendungen finden sich in [83, 98] und [71].

3.2.4 Gebietskopplungen

Werden in einer Simulation zwei Gebiete $\Omega_{1,2}$ gemeinsam betrachtet, so ist von einer Simulation gekoppelter Gebiete die Rede. Historisch gesehen, werden gekoppelte Teilgebiete erstmals 1869 in einem Algorithmus von Schwarz betrachtet [99]. Der heute als alternierendes Schwarz-Verfahren bezeichnete Algorithmus besteht aus einem abwechselnden Lösen eines Dirichlet-Problems innerhalb zweier überlappender Teilgebiete. In dem vorangegangenen Berechnungsschritt ermittelte Werte werden dabei als Randbedingungen und in Form von Startwerten in den im Anschluss folgenden Rechenschritten berücksichtigt [100]. Diese Vorgehensweise ist heute Bestandteil der Klasse der Gebietszerlegungsverfahren, die inzwischen in einer Vielzahl an numerischen Verfahren eingesetzt werden [99, 101]. Dabei wird eine Simulation zu Beginn in Teilsimulationen zerlegt. Diese Teilgebiete werden dann abhängig vom jeweiligen Verfahren bearbeitet. Die Lösung der gekoppelten Fragestellung ist die, bei der eine gemeinsame Lösung auf den Rändern aneinander grenzender Teilgebiete existiert und die eine gültige Lösung innerhalb der Teilgebiete enthält. Dabei sind Gebietskopplungen aufgrund der Stetigkeit der abhängigen Variablen lineare Kopplungen.

Eine Unterscheidung der existierenden Algorithmen ist anhand mehrere Charakteristika möglich, wobei diese in verschiedenen Kombinationen auftreten und so jede Unterscheidung eine Berechtigung hat [102]. Ein Beispiel ist die Betrachtung der Zerlegungen in Teilgebiete anhand von Knoten, Kanten oder Elementen der Diskretisierung. Alternativ ist eine Unterscheidung anhand der Größe des Überlappens der Teilgebiete möglich. Auch eine Unterscheidung anhand der Verarbeitung der Werte an den Schnittstellen zwischen den Gebieten, beispielsweise mittels des Schurkomplements $\hat{\boldsymbol{S}}$ oder auf Grundlage von Lagrange-Multiplikatoren Λ, ist möglich [102, 103]. Abschließend sei auch die Charakterisierung anhand der Art der Berechnung der Teilsimulationen (direkt oder iterativ) genannt [100].

Anwendungen der Gebietskopplung sind vielfältig, sodass die hier angeführten Beispiele dazu dienen, die Möglichkeiten eines dynamischen Berechnungssystems unter Berücksichtigung der Gebietskopplungen zu verdeutlichen. Eine Anwendung ist die Abstraktion von Modellen, bei der ein Simulationsmodell stückweise um weitere Simulationsteile erweitert wird. Diese Erweiterung kann angelehnt an den Entwicklungsprozess eines Bauteils mit mehreren Baugruppen betrachtet werden. Elektrotechnische Beispiele stellen Schaltungssimulationen [104–106] dar. Dabei setzt sich die Funktion einer Schaltung aus im Vorfeld bestimmten Übertragungsfunktionen verschiedener Bauteile zusammen, die sich beispielsweise mittels der FEM bestimmen lassen. Die Vernachlässigung des Einflusses anderer Bauteile auf einzelne Übertragungsfunktionen (unidirektionale Kopplung) macht modellabhängig eine zusätzliche Untersuchung der elektromagnetischen Verträglichkeit (EMV) der Schaltung notwendig (bidirektionale Kopplung). Diese stellt eine weitere Anwendung der gebietsgekoppelten FEM dar. Auch ist die Kopplung einer Schaltungssimulation und der FEM an definierten Knoten, beispielsweise über die Grö-

ßen des elektrischen Stromes I und der elektrischen Spannung U, möglich. Weitere Beispiele stellen Mehrfeldsimulationen dar, falls in Teilgebieten verschiedene zugrunde liegende physikalische Zusammenhänge betrachtet und über die Ränder gekoppelt werden. Dies hat sich in Kombination mit verschiedenen numerischen Berechnungsmethoden und deren im Folgenden beschriebenen Kopplung als numerisch hilfreich erwiesen [72].

3.2.5 Methodenkopplungen

Bisher wurde zur Simulation der physikalischen Zusammenhänge lediglich die FEM betrachtet. Werden verschiedene Methoden genutzt, um berechnete Gebiete oder Felder miteinander zu koppeln, so lässt sich von Methodenkopplungen sprechen. Einen Überblick über bestehende Methoden zur Berechnung elektromagnetischer Simulationen gibt [107]. Gründe zur Nutzung mehrerer Methoden ergeben sich aus den methodenspezifischen Eigenschaften, wie beispielsweise Schwierigkeiten der FEM bei der Betrachtung offener Gebiete oder dem hohen Diskretisierungsaufwand [71].

Dass eine Kopplung verschiedener Methoden vorteilhaft bei der Berechnung ist, zeigt beispielhaft die gekoppelte Berechnung einer Fluid-Struktur-Wechselwirkung [108] und die Betrachtung der Funktionsweise eines elektromechanischen Relais [109]. Die Methodenkopplung wird hier erwähnt, da sie einen konzeptionellen Einfluss auf das in Kapitel 4 dargestellte Agentensystem hat. Zudem konnte die Kopplung der FEM mit der Randelementmethode (BEM) im Zusammenhang mit dem Agentensystem zielführend zur Berechnung einer elektromagnetischen Welle eingesetzt werden [110, 111].

3.3 Gleichungssystemlöser

Bei der Lösung einer Fragestellung, die uni- oder bidirektionale Kopplungen beinhaltet, wird zwischen zwei Lösungsstrategien unterschieden [112, 113]. Die erste Strategie berechnet die Fragestellung an einem Stück (monolithisch). Entsprechend wird ein einzelnes Gleichungssystem aufgestellt und gelöst. Die zweite Lösungsstrategie betrachtet Teilsimulationen getrennt voneinander und berücksichtigt Kopplungen iterativ und je Rechenschritt linearisiert. Da beide Strategien in dieser Arbeit zum Einsatz kommen, werden diese im Folgenden eingeführt.

3.3.1 Monolithische Löser

Für zwei gemeinsam berechnete Simulationen entsprechend (3.21), wird bei der monolithischen Lösung ein Gleichungssystem

$$\underbrace{\begin{bmatrix} \bar{\boldsymbol{J}}_{11} & \bar{\boldsymbol{J}}_{12} \\ \bar{\boldsymbol{J}}_{21} & \bar{\boldsymbol{J}}_{22} \end{bmatrix}}_{\tilde{\boldsymbol{J}}} \underbrace{\begin{bmatrix} \Delta \boldsymbol{u}_1 \\ \Delta \boldsymbol{u}_2 \end{bmatrix}}_{\Delta \tilde{\boldsymbol{u}}} = \underbrace{\begin{bmatrix} -\boldsymbol{f}_1 \\ -\boldsymbol{f}_2 \end{bmatrix}}_{\tilde{\boldsymbol{f}}} \tag{3.22}$$

aufgestellt. Die Kopplungsmatrizen $\bar{\boldsymbol{J}}_{12}$ und $\bar{\boldsymbol{J}}_{21}$ beinhalten die partiellen Ableitungen der jeweiligen Simulationen nach den gekoppelten abhängigen Variablen $\boldsymbol{u}_2$ und $\boldsymbol{u}_1$. Beispielsweise entsprechen die Einträge der Kopplungsmatrix für die elektro-thermischen Simulation aus Kapitel 3.2.3 unter Berücksichtigung der Wärmequelle $q(\varphi)$ den partiellen Ableitungen der schwachen Formulierung nach den Freiheitsgraden des elektrischen Potenzials

$$\int_{\Omega_e} 2\kappa N_{je} \mathrm{grad}\,(N_{ie})\,\mathrm{grad}\,(\varphi(\boldsymbol{x}))\,\mathrm{d}\Omega. \tag{3.23}$$

Eine zu $\tilde{\boldsymbol{J}}\Delta\tilde{\boldsymbol{u}} = \tilde{\boldsymbol{f}}$ äquivalente Darstellung wird erreicht, wenn der Lösungsvektor $\tilde{\boldsymbol{u}}$ knotenbezogen formuliert wird. Abbildung 3.5 zeigt die sich unterscheidende Struktur der Jacobimatrix $\tilde{\boldsymbol{J}}$ für zwei bidirektional gekoppelte Teilsimulationen mit je 20 Freiheitsgraden.

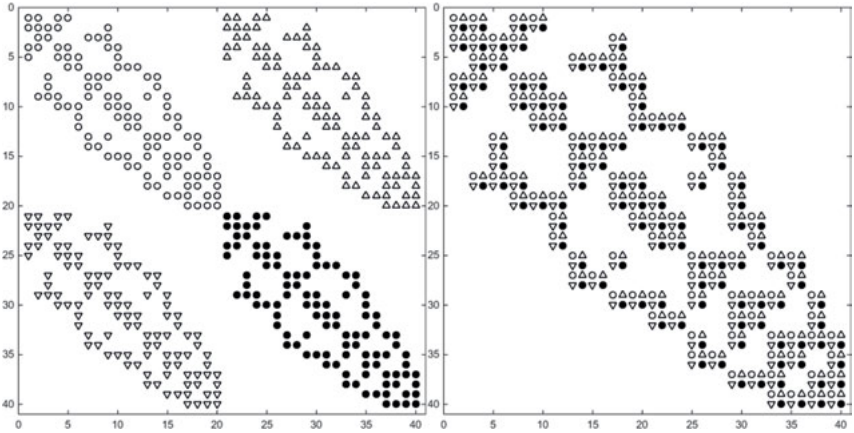

Abbildung 3.5: Monolithische Jacobimatrix $\tilde{\boldsymbol{J}}$ sortiert nach Teilsimulationen (links) und nach Knotennummern (rechts) mit ∘ als $\bar{\boldsymbol{J}}_{11}$, △ als $\bar{\boldsymbol{J}}_{12}$, ▽ als $\bar{\boldsymbol{J}}_{21}$ und • als $\bar{\boldsymbol{J}}_{22}$

Dabei bleibt im Fall der knotenbezogenen Formulierung von $\tilde{\boldsymbol{u}}$ die Bandstruktur des gekoppelten Problems weitestgehend erhalten. Bedingt durch verschiedene Wertebereiche der verwendeten Materialwerte innerhalb der Teilmatrizen sowie der Kopplung verschlechtert sich jedoch die Kondition der Jacobimatrix $\tilde{\boldsymbol{J}}$. Eine weitere Auswirkung ist, dass durch die kopplungsbedingt verschiedenen Koeffizienten der Teilmatrizen $\bar{\boldsymbol{J}}_{12}$ und $\bar{\boldsymbol{J}}_{21}$ meist die Symmetrie der Jacobimatrix $\tilde{\boldsymbol{J}}$ verloren geht. Besonders deutlich wird dies im Fall einer unidirektionalen Kopplung (z.B. $\bar{\boldsymbol{J}}_{12} = \boldsymbol{0}$). In diesem Fall ist $\tilde{\boldsymbol{J}}$ zudem reduzibel, was bei der Auswahl des

zur Lösung eingesetzten Verfahrens berücksichtigt werden muss [114]. Bei monolithischer Betrachtung verschiedener gekoppelter Berechnungsmethoden, wie der BEM und der FEM, geht die Bandstruktur der Jacobimatrix $\tilde{\boldsymbol{J}}$ gänzlich verloren.

Die Berechnung von (3.22) erfolgt dabei entweder direkt oder iterativ. Direkte Lösungsverfahren sind Verfahren, die in endlich vielen Schritten die exakte Lösung mittels sequenzieller Elimination von Abhängigkeiten bestimmen. Die Berechnungsverfahren enden, sobald deren Berechnungsvorschrift erfüllt ist. Direkte Lösungsverfahren werden heute meist in unvollständiger Form als Vorkonditionierer eines iterativen Lösungsverfahrens oder zur Lösung von kleineren Fragestellungen eingesetzt [114]. Zu ihnen zählen die Gauß-Elimination ($\boldsymbol{LR}$), die Cholesky-Zerlegung ($\boldsymbol{LL}^\top$ bzw. $\boldsymbol{LDL}^\top$) und die $\boldsymbol{QR}$-Zerlegung nach Gram-Schmidt, Givens oder Householder. Anwendungsbeispiele stellen die in Kapitel 5 berechneten Simulationen dar. Orientierung bei der Auswahl eines geeigneten direkten Lösungsverfahrens bietet beispielsweise Abbildung 3.6. Dabei berücksichtigte Matrixeigenschaften gilt es im Vorfeld zu bestimmen, was bei deren Auswertung meist zur Anwendung eines der genannten Verfahren führt.

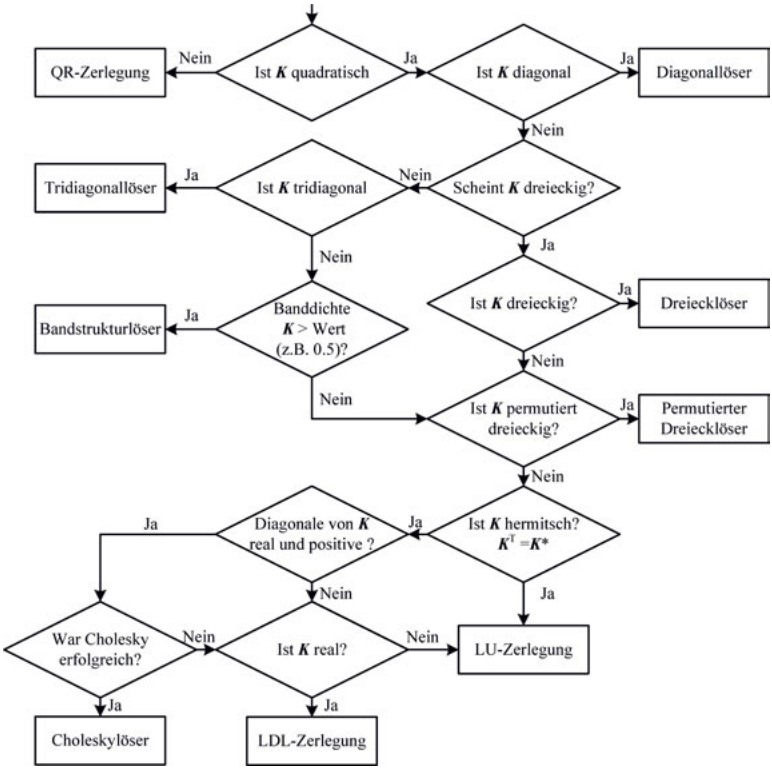

Abbildung 3.6: Entscheidungsbaum zur Auswahl von direkten Lösungsverfahren [115]

Iterative Lösungsverfahren, wie beispielsweise das Newton-Verfahren aus (3.8), erzeugen für (3.22) eine Sequenz von Vektoren $\tilde{\boldsymbol{u}}^{(k)}$, für die im Fall einer Konvergenz

$$\tilde{\boldsymbol{u}} = \lim_{k \to \infty} \tilde{\boldsymbol{u}}^{(k)} \tag{3.24}$$

gilt. Dabei wird in der Praxis der Startvektor $\tilde{\boldsymbol{u}}^{(0)}$ häufig zu null gewählt und die Berechnung nach k Iterationen mit einem Restfehler

$$\tilde{\epsilon} > \|\tilde{\boldsymbol{u}}^{(k)} - \tilde{\boldsymbol{u}}^{(k-1)}\| \tag{3.25}$$

beendet [82]. Die iterativen Lösungsverfahren lassen sich dabei in stationäre und nicht stationäre Verfahren unterteilen [116]. Stationäre Verfahren sind Verfahren, bei denen sich der nächste Schritt entsprechend

$$\tilde{\boldsymbol{u}}^{(k+1)} = \boldsymbol{G}\tilde{\boldsymbol{u}}^{(k)} + \boldsymbol{g} \tag{3.26}$$

berechnen lässt. Dabei werden die Matrix $\boldsymbol{G}$ und der Vektor $\boldsymbol{g}$ so gewählt, dass diese unabhängig von der Anzahl an Iterationen k sind [116]. Zu den stationären Verfahren zählen im Besonderen die Splittingverfahren, wie das Jacobi-Gesamtschrittverfahren, das Gauß-Seidel-Einzelschrittverfahren, Relaxationsverfahren, Gebietszerlegungsverfahren und Mehrgitterverfahren [117]. Aufbauend auf diesen Verfahren wird im Weiteren die Berechnung im Agentensystem umgesetzt. Dabei lässt sich zeigen, dass beispielsweise das Jacobiverfahren für einen beliebigen Startvektor $\tilde{\boldsymbol{u}}^{(i)}$ und eine beliebige rechte Seite $\tilde{\boldsymbol{f}}$ im Fall einer regulären, irreduziblen und diagonaldominanten Matrix $\tilde{\boldsymbol{J}}$ konvergiert [114].

Die Lösungssequenz nicht stationärer Verfahren lässt sich mittels

$$\tilde{\boldsymbol{u}}^{(k+1)} = \tilde{\boldsymbol{u}}^{(k)} + \alpha^{(k)}\boldsymbol{X}^{(k)} \tag{3.27}$$

beschreiben. Zu unterscheiden ist dabei, dass die Berechnung Informationen beinhaltet, die sich während jeder Iteration ändern [116, 118]. So stellt α den Skalierungsfaktor für den Suchrichtungsvektor $\boldsymbol{X}$ dar. Beispiele für nicht stationäre Verfahren sind die Projektionsverfahren, wie das für symmetrische Matrizen geeignete CG-Verfahren oder die Lösungsverfahren (F)GMRES und BiCGStab. Auch diese Verfahren kommen in dieser Arbeit zum Einsatz. Eine ausführliche Darstellung iterativer Lösungsverfahren findet sich in [119]. Hinweise zur Auswahl eines geeigneten Lösungsverfahrens lassen sich dem Entscheidungsbaum in Abbildung 3.7 entnehmen. Dabei werden analog zu direkten Lösungsverfahren die Eigenschaften der Systemmatrix $\boldsymbol{K}$ berücksichtigt. Vor- und Nachteile von iterativen und direkten Verfahren sowie eine Gegenüberstellung, ab welcher Matrixgröße es effizienter ist, das Gleichungssystems direkt zu berechnen, findet sich in [120, 121].

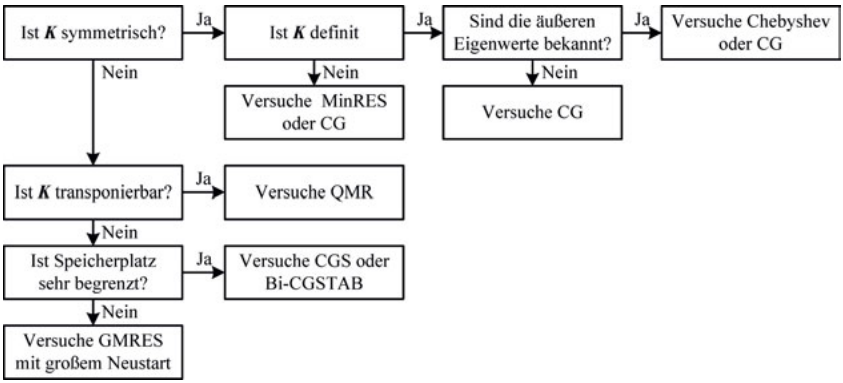

Abbildung 3.7: Entscheidungsbaum zur Auswahl von iterativen Lösungsverfahren [116]

Da eine kleine Konditionszahl der Jacobimatrix $\tilde{\boldsymbol{J}}$ von Vorteil für die iterative Lösung von (3.22) ist, wird versucht, diese mittels äquivalenter Umformung von (3.22) zu verringern [116]. Dazu werden beispielsweise unvollständig ausgeführte direkte Lösungsverfahren oder Gebietszerlegungsverfahren entsprechend Kapitel 3.2.4 genutzt. Erreicht wird mit diesen Methoden eine Beschleunigung und Stabilisierung der beispielsweise nicht stationären Krylov-Unterraumverfahren [114]. Auch gelingt ein zusätzliches Parallelisieren der Berechnung aufgrund erzeugter hierarchischer Strukturen, wie beispielsweise durch eine Gebietszerlegung.

Abschließend sei darauf hingewiesen, dass selbst bei bekannter Lösung die Auswahl des am besten geeigneten Lösungsverfahrens selten offensichtlich ist und häufig heuristisch bestimmt wird. Existierende Empfehlungssysteme, wie die in [122] und [123] beschriebenen, benötigen zu deren Training eine große Anzahl ähnlicher Simulationen. Gerade deren Bereitstellung ist für selten auftretende Kopplungen schwierig.

3.3.2 Gestaffeltes Lösen

Eine Alternative zum monolithischen Lösen zweier gemeinsam betrachteter Simulationen stellt das gestaffelte Lösen dar [124]. Dabei werden die zwei Gleichungssysteme entsprechend (3.21) unabhängig voneinander aufgestellt und nacheinander berechnet. Abbildung 3.8 stellt den sequenziellen Lösungsverlauf einer gestaffelten Berechnung dar.

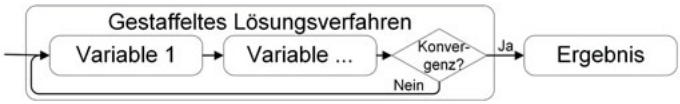

Abbildung 3.8: Aufbau eines gestaffelten Lösungsverfahrens

Das Lösungsverfahren endet, wenn die Änderungen ϵ_i in allen Teilsimulationen i kleiner als eine Toleranz ϵ mit beispielsweise

$$\epsilon > \max \epsilon_i = \max |\boldsymbol{u}_i^{(k)} - \boldsymbol{u}_i^{(k-1)}| \tag{3.28}$$

sind. Andere Kriterien, wie beispielsweise $\epsilon > \max \|\epsilon_i\|$ finden ebenfalls Anwendung [125].

Besteht zwischen den Teilsimulationen keine Abhängigkeit, so entspricht der Lösungsverlauf bei sequenzieller Berechnung dem der berücksichtigten Teilsimulationen. Ist eine Kopplung zwischen den Teilsimulationen vorhanden, so ist durch die getrennte Betrachtung der Teilsimulationen keine Berechnung der partiellen Ableitungen nach den gekoppelten Variablen aus (3.22) mehr möglich. Stattdessen werden die Kopplungen in $\bar{\boldsymbol{J}}$ und $\boldsymbol{f}$ linearisiert und iterativ berücksichtigt. Zum Aufstellen von (3.21) gilt es dabei, Initialwerte der gekoppelten Simulation bereitzustellen. Im Fall einer unidirektionalen Kopplung der Simulationen gelingt dies durch eine auf die Kopplung angepasste Reihenfolge der Berechnung. Hängen beispielsweise weitere Simulationen unidirektional von einer bereits berechneten Simulation ab, so werden diese im Anschluss berechnet. Analog werden Simulationen, die unidirektional Werte für gekoppelte Simulationen bereitstellen, im Vorfeld berechnet. Für bidirektionale Kopplungen und Simulationen mit Ringabhängigkeiten innerhalb des Simulationsprozesses (Kreise im Kopplungsgraph) ist die Wahl hinreichend genauer Startwerte notwendig. In der Praxis werden diese häufig als konstant oder als null angenommen. Entsprechend ist die Reihenfolge, in der gekoppelte Teilsimulationen gelöst werden, vom betrachteten Beispiel und dessen Kopplungen abhängig. Sie gilt es bisher vor Berechnungsbeginn simulationsbezogen statisch zu definieren. Änderungen in den Kopplungsgrößen werden indirekt über Veränderungen der abhängigen Variablen im Konvergenzkriterium ausgewertet. Auch eine parallele Berechnung von Teilsimulationen ist simulationsabhängig individuell zu prüfen [126].

Bei einem Vergleich des iterativen Ansatzes mit dem monolithischen Ansatz haben die Teilsimulationen beim iterativen Ansatz weniger Freiheitsgrade und sind besser konditioniert [127]. Auch bleiben die Eigenschaften der Jacobimatrizen $\bar{\boldsymbol{J}}$, wie beispielsweise die Symmetrie oder die Struktur der Matrix erhalten, und werden nicht durch die Kopplungsterme beeinflusst. Jedoch wirken sich Änderungen in der abhängigen Variablen um wenigstens eine Iteration verzögert auf die gekoppelte Simulation aus. Dies macht auch für lineare Aufgabenstellungen mehrere Iterationen notwendig [112]. Deutlich wird dies im Fall eines kopplungsabhängigen Materialparameters. Bei jeder Berechnungsiteration ändert sich die Matrix $\boldsymbol{K}$ und somit auch $\bar{\boldsymbol{J}}$. Dies macht das erneute Aufstellen von $\bar{\boldsymbol{J}}$ und $\boldsymbol{f}$ je gestaffelter Lösungsiteration notwendig. Die Anzahl zur Lösung notwendiger Iterationen lässt sich beispielsweise mit dem in [125] beschriebenen dezentralen Lösungsansatz mit zentraler Jacobimatrix verringern. Untersuchungen zu gestaffelt gelösten und nicht linear gekoppelten Teilsimulationen finden sich häufig im Zusammenhang mit zeitabhängigen Teilsimulationen [128]. Dabei treten Instabilitäten auf-

grund sich gegenseitig aufschwingender Teilsimulationen und verschiedener Zeitschrittweiten auf [129]. Modifizierte gestaffelte Lösungsansätze ermöglichen jedoch deren Berechnung [113]. Analysen zur Stabilität der zeitabhängigen Verfahren finden sich in [130] und [131]. Da bisher eine allgemeingültige Theorie zur Konvergenz von gestaffelt lösbaren Simulationen fehlt [112], scheitert diese Lösungsmethode manchmal unerwartet [113]. Andererseits finden sich in der Literatur eine Vielzahl an erfolgreich erprobten Lösungssequenzen sowie die dazugehörigen numerischen Modelle [71, 132].

Aus Sicht eines Softwareentwicklers bietet der gestaffelte Ansatz die Möglichkeit einer flexiblen und aufgabenspezifischen Softwareentwicklung. Zudem ermöglicht er eine einfachere Wartung, entsprechend dem Stand der Technik in den jeweiligen Disziplinen [71, 126, 133]. Entsprechend werden verschiedene Werkzeuge, mathematische Modelle und Lösungsverfahren für die Teilsimulation genutzt. Verglichen mit dem monolithischen Ansatz sind zu deren Kopplung nur geringfügige Anpassungen der Simulationswerkzeuge nötig [125], was eine kombinierte Nutzung eigener Simulationsbibliotheken als auch der von kommerziellen Anbietern erlaubt. Bei nicht konformer Diskretisierung auftretende Schwierigkeiten [134] lassen sich durch die Interpolation von Werten aus gekoppelten Teilsimulationen [95] oder einer diskretisierungsfreien Wertebestimmung [135] handhaben. Ein weiterer Vorteil des gestaffelten Ansatzes ist, dass er eine natürliche Möglichkeit zur Lastverteilung auf verschiedene Rechner bietet [136]. Allerdings existiert im Vergleich zu monolithischen Lösungsverfahren keine Garantie zur Leistungssteigerung. Insbesondere dann nicht, wenn große Datenmengen zwischen den Simulationen ausgetauscht werden, wie beispielsweise bei volumenbezogenen Größen [126].

3.4 Berechnungsumgebungen

Bei der softwaretechnischen Umsetzung einer disziplinübergreifenden Berechnungssoftware lassen sich zwei prinzipielle Ansätze unterscheiden. Im einen Fall wird genutzt, dass sich verschiedene physikalische Zusammenhänge durch ähnliche Gleichungen, wie beispielsweise partielle Differenzialgleichungen, beschreiben lassen. Dementsprechend werden bei der Programmierung disziplinspezifische Eingabemöglichkeiten bereitgestellt, die intern auf möglichst allgemeine Gleichungen zurückgeführt und anschließend gelöst werden. Alternativ lassen sich unterschiedlich spezialisierte und voneinander gekapselte Programme über softwarespezifische Schnittstellen miteinander in eine zentral organisierte Struktur überführen [137]. Um die am Ende von Kapitel 3.3.2 genannten Vorteile zu erreichen, ergeben sich für eine erfolgreiche Kopplung verschiedener Simulationswerkzeuge die Anforderungen [19]:

- einer kooperativen Interoperabilität, die es ermöglicht, dass verschiedene eigenständige Programmteile nebeneinander und miteinander arbeiten können,

- einer geringen Quelltextanpassung, die zur Bereitstellung der Schnittstellen zwischen den Programmteilen erforderlich ist,
- eines korrekten Datenaustausches zwischen den Programmteilen unter Berücksichtigung der Diskretisierungen, physikalischer Randbedingungen und numerischer Eigenschaften,
- einer dynamischen Anpassungsfähigkeit des Systems auf Zustände während der Simulation,
- einer Balance zwischen Flexibilität und Leistungsfähigkeit der Implementierung [138].

Bedingt durch die Komplexität der Implementierung kommen hierfür häufig abstrakte Programmieransätze zum Einsatz [139]. Als Beispiele für die objektorientierte Programmierung seien *MpCCI* und das *Functional Mockup Interface* genannt, die jeweils eine Schnittstellensoftware zur interdisziplinären Quelltextkopplung bereitstellen [140, 141]. Eine dienstorientierte Implementierung für den Einsatz im Netzwerk ist *PDE.MART* [18]. Ein mittels Java umgesetztes Agentensystem zur Verfeinerung der Diskretisierung, das auch eine Gebietszerlegung beherrscht, wird in [142, 143] vorgestellt. Dabei repräsentieren Agenten analog zu dieser Arbeit Rechenressourcen. Diese werden dort einer Zentralinstanz zur Verfügung gestellt. Eine Kommunikation der Agenten findet dabei ausschließlich mit einer Zentralinstanz statt. Der agentenorientierte Ansatz *SciAgents* findet Anwendung zur Berechnung mehrerer interdisziplinärer Wärmequellen, die gebietsabhängig zweidimensional miteinander gekoppelt werden, um den Wärmefluss in einem chemischen Reaktor zu berechnen [21]. Der im Folgenden dargestellte Ansatz wurde unabhängig von *SciAgents* entwickelt. Er ermöglicht zudem eine Berechnung verschiedener dreidimensionaler Simulationsaufgaben mit einer bidirektionalen Kopplung der Disziplinen aufeinander. Auch die dort statisch und zentral durchgeführte Aufgabenteilung erfolgt hier dezentral, dynamisch und angepasst auf die dem jeweiligen System bereitstehenden Berechnungsressourcen. Zudem lassen sich weitere Rechenressourcen in eine Berechnung einbinden und von deren Fähigkeiten profitieren. Dies gilt auch dann, wenn es sich um kleine Simulationsaufgaben handelt. Auch die Bereitstellung individueller Fähigkeiten der Agenten an das Berechnungssystem, deren funktionelle Gruppierung und dynamische Berücksichtigung bei gekoppelten Simulationen grenzen das im Folgenden entwickelte Agentensystem von bestehenden Systemen ab.

4 Softwareagenten-basierte Simulationsumgebung

Nachfolgend wird die Umsetzung eines Softwareagentensystems zur Berechnung von gekoppelten interdisziplinären Simulationen beschrieben. Dabei stehen die prinzipielle Realisierbarkeit der dezentralen Aufgabenteilung, die verteilte Bearbeitung der Teilaufgaben und das Austauschen der Teilergebnisse zum Ermöglichen einer Gesamtlösung im Vordergrund. Die selbstständige Planung der Zusammenarbeit von Agenten stellt eine eigenständige Fragestellung dar [45], die hier entsprechend entwurfsspezifischer Vorgaben erfolgt. Die Darstellung der agentenbasierten Simulationsumgebung gliedert sich in drei Abschnitte. Zu Beginn werden die Architektur des Agentensystems, die Funktionalität der eingesetzten Softwareagenten und deren Schnittstellen vorgestellt. Eine Darstellung des Verhaltens der Agenten und deren Vererbung untermauern den Aufbau und veranschaulichen die Abläufe innerhalb des Agentensystems.

Der zweite Abschnitt beschreibt den numerischen Berechnungsablauf einer Simulation durch das Agentensystem. Dazu wird der agentenbasierte Lösungsprozess angelehnt an einen monolithischen Lösungsansatz dargestellt und damit verglichen. Notwendige Teilsequenzen werden, ausgehend von der Initialisierung der Berechnung über die Kopplung bis hin zur iterativen Lösungssequenz, diskutiert. Zudem wird auf die Wahrnehmung des Lösungsprozesses aus Agentensicht und dessen Umgang mit den zur Kopplung verwendeten Daten eingegangen. Abgerundet wird der zweite Abschnitt durch eine Betrachtung von Gebiets- und Methodenkopplungsverfahren sowie einer auf verschiedene Disziplinen angepassten Diskretisierung.

Der dritte Abschnitt betrachtet das Agentensystem im Hinblick auf die Ausnutzung verfügbarer Ressourcen. Dabei wird der Umgang eines Agenten mit der zur Verfügung stehenden Rechenzeit und das Ende einer Berechnung dargestellt. Die Ressourcennutzung im Hinblick auf das Agentensystem wird anhand der Unterscheidung von zentralen und dezentralen Algorithmen und deren Implementierung am Beispiel eines agentenbasierten Rechenclusters verdeutlicht. Auch die verzögerte Bereitstellung und der Ausfall von Agenten werden betrachtet.

4.1 Systemarchitektur

Im Folgenden wird die Systemarchitektur des Agentensystems ausgehend von zwei Kategorien an Anforderungen beschrieben, die es zur Realisierung eines Agentensystems zur Simulation gekoppelter Fragestellungen zu berücksichtigen gilt [144].

Die erste Kategorie beinhaltet Anforderungen an das gesamte Simulationssystem. Um damit die Berechnung einer Simulation zu ermöglichen, sind die zur Simulation eingesetzten numerischen Feldberechnungsbibliotheken zu steuern. Zudem ist deren Ausführung während der einzelnen Phasen einer Simulation zu überwachen. Zur Nutzung mehrerer Berechnungseinheiten in dem Simulationssystem sind deren Zustände zu kennen, zu interpretieren und auf diese angepasst zu reagieren. Dabei wird auf eine Implementierung mittels des monolithischen Programmieransatzes verzichtet, der der ältesten Form an Programmen entspräche und aus einer Vielzahl an Funktionen besteht, die sich gegenseitig aufrufen [145]. Stattdessen wird ein verteiltes System entwickelt, bei dessen Umsetzung es die folgenden acht Aspekte zu berücksichtigen gilt [36]:

Nebenläufigkeit, lokale Datenhaltung, Heterogenität, Fehlerbehandlung, Skalierbarkeit, Offenheit, Sicherheit und Transparenz.

Diese sind gleichzeitig Anforderungen an das gesamte Simulationssystem, die am Beispiel der Datenverwaltung transparent werden. So wird bisher ein Simulationsmodell mit der zugehörigen Modellgeometrie und den Materialbeschreibungen zentral anstatt parallel bereitgestellt. Für eine parallele Berechnung werden zentral meist homogene statt heterogene Arbeitspakete erstellt und von dort zur verteilten Berechnung beauftragt. Die Bearbeitung aller Teilaufgaben ist eine Voraussetzung, um ein Ergebnis bereitzustellen. Diese Notwendigkeit der erfolgreichen Bearbeitung aller Teilaufgaben gilt es während der Berechnung, beispielsweise bei einer Skalierung des Simulationssystems oder bei der Behandlung von Fehlern, zu beachten.

Anforderungen der zweiten Kategorie beziehen sich auf die zur Umsetzung verwendete Programmiersprache. Dabei hat die Programmiersprache einen Umgang mit der Komplexität eines verteilten Systems sowie der dazu notwendigen Kommunikation zu ermöglichen. Bedingt durch eine zu erwartende Verarbeitung numerischer Ergebnisse, ist schnell und leistungsstark mit großen Datenmengen umzugehen. Da bestehende Simulationswerkzeuge in das System eingebunden werden, sind Schnittstellen zu etablierter Software zu realisieren. Dabei gilt es, die in Kapitel 3.4 angegebenen Eigenschaften bei der Kopplung von Simulationswerkzeugen zu berücksichtigen.

Bestehende Entwicklungsumgebungen für Agentensysteme, wie beispielsweise das in Kapitel 2.5 beschriebene JADE, berücksichtigen weitestgehend die durch die Simulation als Anwendung entstandenen Anforderungen. Sie ermöglichen die Kommunikation zwischen verteilt ausgeführten Softwareagenten, die hier zur Berechnung einer Simulation eingesetzt werden. Der systembedingt hinzukommende Speicheraufwand je Agent entspricht bei minimalistischer Auslegung wenigen Kilobyte und für die Agentenplattform wenigen hundert Kilobyte [146]. Weitere Hinweise zur Umsetzung von Agentensystemen finden sich in [147] und [148] oder ergeben sich aus bereits umgesetzten Projekten, wie beispielsweise der Integration betrieblicher Aufgaben in Chemiewerken [64].

Innerhalb der Berechnungseinheiten wird der Agent als Abstraktionsebene oberhalb der Ebene bestehender Simulationswerkzeuge betrachtet. Er ermöglicht im Bedarfsfall die Zusammenarbeit mit anderen Agenten innerhalb des Agentensystems sowie den Datenaustausch. Die Fähigkeiten eines Agenten werden durch die ihm lokal zur Verfügung stehenden numerischen Simulationswerkzeuge und die Leistungsfähigkeit seiner Hardware bestimmt. Da die zur Höchstleistungsrechnung eingesetzten Rechenressourcen meist mit Linux-basierten Betriebssystemen betrieben werden [5], Anwender Simulationen jedoch häufig auf Windows entwickeln, erhöht eine einheitliche, betriebssystemunabhängige Laufzeitumgebung für die Agenten die Flexibilität des Systems. Die in JADE eingesetzte Programmiersprache Java wird zudem als Bindeglied zwischen verschiedenen wissenschaftlichen Quelltexten beschrieben [149]. Sie ist ein Beispiel für Programmiersprachen, mit denen sich diese und weitere der genannten Anforderungen erfüllen lassen. Die Objektorientierung lässt sich nutzen, um komplexe Zusammenhänge abstrahiert darzustellen (siehe Abbildung 2.1). Eine Parallelisierung von Java Quelltext ist analog zur Parallelisierung von C-Quelltext mittels MPI in [150] dargestellt und untermauert die Nutzbarkeit von Java auch für rechenintensive Operationen. Zusammenfassend stellt Abbildung 4.1 eine mögliche Verteilung der Softwareagenten in einem Netzwerk dar. Dargestellt sind auch die gemeinsame Nutzung von Ressourcen sowie verschiedene Plattformen zur Ausführung der Agenten.

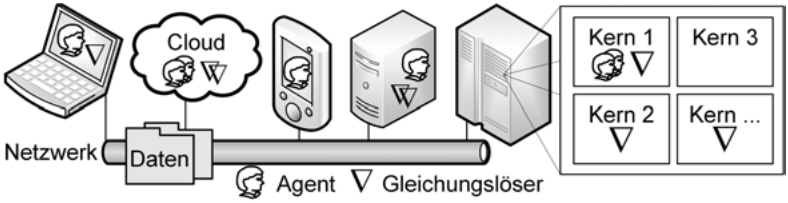

Abbildung 4.1: Verteilte Agenten auf verschiedenen Systemen [151]

Die Agentenplattform und dessen Verwaltung (der AMS-Agent) hat hier lediglich Kenntnis von der Existenz verfügbarer Agenten. Eine Zuteilung von Aufgaben oder Weisungen in Bezug auf den Ablauf der numerischen Simulation erfolgen nicht. Dies bewahrt die Autonomie der Agenten, sodass alle Entscheidungen entsprechend dezentral getroffen werden. Zudem entspricht dies dem agilen Projektmanagement während der Bearbeitung einer gekoppelten Simulation innerhalb eines interdisziplinären Agententeams. Ein Vergleich der Effizienz, der Planungssicherheit und der Ergebnisqualität zwischen einem agilen und einem klassischen Projektmanagement zeigt, dass die Umsetzung eines agilen Managements vorteilhaft ist [152]. Allerdings bedeutet die dezentrale Durchführung einer Aufgabe im Vergleich mit einer möglichen zentralen Durchführung meist einem Anstieg der Bearbeitungsdauer, gerade wenn die Aufgaben starke Abhängigkeiten aufweisen. Numerisch bestätigt der Vergleich der Konvergenzgeschwindigkeiten von iterativen asynchronen mit iterativen synchronen Rechenverfahren dies [153]. Eine

auf die Aufgabe und die Fähigkeiten der Agenten angepasste Aufgabenaufteilung ermöglicht jedoch, flexibel und unter Berücksichtigung äußerer Einflüsse, wie hinzukommende Ressourcen, zu agieren.

4.1.1 Agentenbeschreibungen

Im Folgenden wird ein Überblick über die Orchestrierung der zur Simulation eingesetzten Softwareagenten gegeben. Anhand dessen wird die in Abbildung 4.2 dargestellte Choreographie der Agenten im Gesamtkontext der Berechnung interdisziplinärer gekoppelter Simulationen veranschaulicht. Im Anschluss folgt eine detaillierte Beschreibung der dargestellten Agenten.

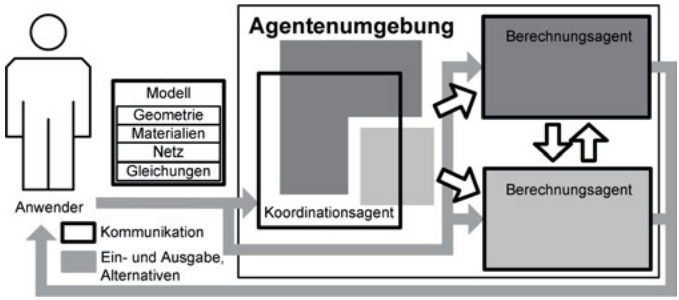

Abbildung 4.2: Aufbau des Agentensystems

Bei der klassischen Simulation wird zu Beginn eine Beschreibung des Modells, bestehend aus der zu betrachtenden Geometrie, deren Materialien, den auszuwertenden Zusammenhängen und der (hier optional) festgelegten Diskretisierung, von einem Anwender vorbereitet und dem Berechnungssystem übergeben. Als Empfänger dieser Modellbeschreibung fungiert ein Koordinationsagent (KA). Kapitel 2.1 entsprechend lässt dieser das Agentensystem für Anwender lokal erscheinen. Der KA analysiert das Modell im Hinblick auf im Agentensystem vorhandene Fähigkeiten und prinzipiell mögliche Kopplungen. Anschließend stellt er das Modell allen Berechnungsagenten (BA) zur Verfügung, die im System vorhanden sind, an der Berechnung der Lösung mitarbeiten wollen und dazu in der Lage sind. Alternativ besteht die Möglichkeit, dass ein Anwender direkt an die BA herantritt und diesen die jeweiligen Teilmodelle zur Bearbeitung übergibt. In diesem Fall gilt es dem KA mitzuteilen, welche BA gemeinsam an der Lösung eines Modells arbeiten. Die Berechnung der Teilmodelle sowie deren Kopplung erfolgt kooperativ und automatisiert durch die BA. Die dezentral erzeugten Ergebnisse werden dem Anwender über Visualisierungsschnittstellen der Agenten oder dateibasiert zur Verfügung gestellt. Zusätzlich existiert ein Dokumentationsagent (DA), der dem Entwickler als Ansprechpartner für die Verwaltung der dezentral anfallenden Statusmeldungen und der Verfolgung der Ressourcenauslastung dient [154].

4.1.1.1 Koordinationsagent (KA)

Zusätzlich zu den bereits beschriebenen Tätigkeiten eines KA ist dessen Kernaufgabe die Koordination im Agentensystem. Die Koordination behandelt Abhängigkeiten zwischen Aktivitäten, um eine Erhöhung der Effizienz von Arbeitsabläufen durch Handhabung bzw. Auflösung von Konflikten zu erreichen [14]. Ein typisches Merkmal eines kooperativen Koordinationsszenarios ist es, dass die Fähigkeiten eines Einzelnen für eine Gesamtlösung nicht ausreichen und eine Aufgabe stattdessen innerhalb einer Gruppe, beispielsweise in Form von Teamarbeit, bearbeitet wird [36]. Währenddessen entstehen Konflikte aufgrund der in Kapitel 3.2 beschriebenen Kopplungsarten. So macht beispielsweise eine fehlende Disziplinkenntnis einzelner BA deren Koordination notwendig. Fehlende Methodenkenntnisse machen die Koordination der Aufgaben innerhalb der Disziplinen nötig. Eine Zerlegung der betrachteten Gebiete innerhalb einer Methode ist aufgrund unterschiedlicher Gebietseigenschaften, wie beispielsweise der Materialwerte oder fehlender Rechenleistung, hilfreich. Sind verschiedene Arten von Parametern während der Berechnung wählbar, so ist eine Zerlegung auf Parameterebene möglich. Zur Handhabe der Koordination wird das in Abbildung 4.3 abgebildete und bisher statische hierarchische Modell von Koordinationsebenen innerhalb des Simulationsmodells eingeführt.

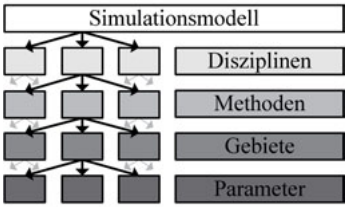

Abbildung 4.3: Überblick über die Ebenen der Koordination

Die Hierarchie berücksichtigt die zur Kopplung notwendige auszutauschende Datenmenge zwischen den BA für eine Iteration. So wird bei der Kopplung verschiedener Disziplinen der gesamte Lösungsvektor, bei der Kopplung von Methoden entweder der gesamte oder analog zur Gebietskopplung der Lösungsvektor am Gebietsrand und für eine Parameterauswahl deren Werte untereinander ausgetauscht. Eine modellbezogene Anpassung dieser Hierarchie ist abhängig vom Einfluss möglicher Nichtlinearitäten vorstellbar. Die Berechnung und die dazu notwendige Modifikation am jeweiligen Teilmodell finden ausschließlich durch die BA statt. Tritt ein Konflikt nicht auf, so wird auch die entsprechende Koordination unnötig.

Beispielhaft wird der Konflikt des Fehlens eines Simulationswerkzeugs zur vollständigen Berücksichtigung aller Teildisziplinen eines Simulationsmodells betrachtet. Dieser wird vom KA gelöst, indem bei der Analyse des Modells eine Modellbeschreibung erzeugt wird, die es den BA ermöglicht, Teilsimulationen zu erzeugen und diese miteinander zu koppeln. Ohne diese

Konfliktlösung wäre keine Berechnung des Simulationsmodells möglich. Alternativ wird vom Anwender erwartet, dass er mehrere Teilmodelle bereitstellt, um den Konflikt zu vermeiden.

Ein anderer Konflikt ergibt sich aus der Frage, welcher BA welches Teilmodell berechnet. Im Fall einer klassischen Parallelisierung, einiger Gebietszerlegungsmethoden [155] oder bei Lösungsverfahren, wie dem parallelen Quasi-Newton-Verfahren [86], erfolgt eine zentrale Planung und Zuweisung von Aufgabenteilen. Nimmt der Anwender die Zuweisung der Teilmodelle manuell vor, tritt der Konflikt nicht auf. Um die Eigenständigkeit der Berechnungseinheiten zu betonen, wird hier ein dezentraler Ansatz verfolgt (siehe Kapitel 2.3). Dieser ermöglicht es jedem Agenten, jede dafür befähigte Teilsimulationen zu bearbeiten, wobei lediglich die Lösung des schnellsten noch verfügbaren BA im Lösungsprozess berücksichtigt wird. Jeder verfügbare BA hat so die Möglichkeit, sich entsprechend seiner Fähigkeiten an der Simulation zu beteiligen oder die Mitarbeit zu beenden, wenn beispielsweise ein Anwender einem BA andere Aufgaben überträgt. Diese Individualität der BA stellt eine Besonderheit im Vergleich zur klassischen Feldberechnung dar. Der KA beteiligt sich nicht an der Berechnung des Modells und hat lediglich Verwaltungsfunktionen. Er teilt die im System vorhandenen Agenten anhand Ihrer Fähigkeiten in funktionale Gruppen ein. Dabei greift er auf eine von den Agenten bereitgestellte Funktionsbeschreibung zurück. In der Funktionsbeschreibung berücksichtigte Eigenschaften sind:

- der Agententyp (KA, BA oder DA),
- berechenbare Disziplinen,
- die zur Berechnung anwendbare Methode,
- ermöglichte Arten der Gebietskopplungen (Dirichlet-, Neumann-, Robin-Randbedingungen),
- anwendbare Lösungsverfahren,
- der Auftraggeber der Simulation, sodass mehrere KA im Agentensystem möglich sind,
- der aktuelle Status des Agenten.

Für die erstellten Gruppen fungiert der KA als Ansprechpartner. Entsprechend hat er eine Übersicht über alle an der Berechnung beteiligten Agenten. Weiter könnte der KA die Mitarbeit von BA an Teilmodellen zulassen oder ablehnen, indem er Agenten den Zugang zu funktionalen Gruppen verweigert. Bedingt durch das Bereitstellen des Modells und die Verwaltung der funktionellen Gruppen hätte der KA eine Möglichkeit zur Beeinflussung des Lösungsprozesses. Die konsequente Zulassung aller befähigten Agenten sowie die Bereitstellung der Teilmodelle an alle BA unterbinden jedoch die zentrale Planung und Steuerung.

Ist eine vollständige Aufteilung der Teilaufgaben einer Berechnung innerhalb einer Gruppe erforderlich, wie beispielsweise bei der Gebietszerlegung, so ist dafür ein Bieterverfahren umgesetzt. Dabei steigern BA um den Zuschlag zur Berechnung von Teilgebieten anhand einer einheitlich durchgeführten Leistungsbewertung der BA. Vom KA wird lediglich die Eignung zur Berechnung der Teilsimulation anhand der Funktionsbeschreibung des Agenten überprüft.

4.1.1.2 Berechnungsagent (BA)

Um die Zusammenarbeit von Berechnungseinheiten zu einer gemeinschaftlichen Lösung von Simulationen zu ermöglichen, ist eine Kopplung sowohl auf Modell- als auch auf Programmebene möglich [156]. Eine Modellkopplung liegt vor, falls während der Berechnung ein Modell existiert das alle Disziplinen, Methoden und Teilgebiete beinhaltet. Der hier realisierte Ansatz entspricht einer Kopplung auf Programmebene. Dabei kommunizieren die Simulationswerkzeuge während der Berechnung miteinander, tauschen berechnete Größen aus, integrieren diese in die eigene Berechnung und setzen anschließend die Berechnung fort. Bei der Umsetzung von BA als Abstraktion von Simulationswerkzeugen stellt die Interaktion mit dem Simulationswerkzeug eine der Eigenschaften von Softwareagenten dar (siehe Kapitel 2.2). Die Fähigkeit des Agenten zur selbstständigen Wahrnehmung von Veränderungen innerhalb der Umgebung lässt sich als eine Berücksichtigung aktueller numerischer Randbedingungen innerhalb der eigenen Simulation und die Kenntnis über den dafür zuständigen Ansprechpartner verstehen. Ausgehend davon zu agieren, entspricht der Agenteneigenschaft der Reaktionsfähigkeit, die bei der verteilten Berechnung gekoppelter Simulationen eine Voraussetzung zur erfolgreichen Lösung ist. Das Ziel jedes BA, seine Simulation erfolgreich und möglichst schnell zu berechnen und die Möglichkeit dorthin verschiedene Wege zu beschreiten (siehe Kapitel 4.3.1), verdeutlicht die Eigenschaft des zielgerichteten Verhaltens der BA. Die Eigenständigkeit und Verschiedenheit jedes Simulationswerkzeugs sowie der unterschiedlichen Hardware der BA stärken deren Individualität.

Die in dieser Arbeit als Blackbox betrachteten Simulationswerkzeuge werden von einem BA jeweils als externes Werkzeug (siehe Abbildung 2.2) eingebunden. Da Simulationswerkzeuge meist alle bereitgestellten Rechenressourcen nutzen, ist die Verwendung eines BA je Berechnungssystem empfehlenswert. Auch größere Ressourcen, wie beispielsweise ein Rechencluster, lassen sich exklusiv BA zur Verfügung stellen. Dessen Verwaltung erfolgt dann durch den BA selbst. Die Nutzung mehrerer Simulationswerkzeuge durch einen BA macht eine zusätzlich agenteninterne Rechenzeit- und Datenverwaltung notwendig, sodass hier jedem BA jeweils ein Simulationswerkzeug zugeordnet ist.

Zum agentenübergreifenden Datenaustausch sind die Eingabe- und Ausgabeparameter sowie Parameter zur Steuerung der Simulationswerkzeuge innerhalb des Agenten zu verwalten. Auch werden Funktionen, die zur Datenaufbereitung für verschiedene Simulationswerkzeuge eingesetzt werden, im Zug der Wiederverwendbarkeit im Agenten umgesetzt. Abbildung 4.4 stellt eine Auswahl der zum Betrieb des BA notwendigen Parameter der Schnittstelle zu einem Simulationswerkzeug dar. Eine detaillierte Funktionsbeschreibung der Schnittstelle zwischen BA und Simulationswerkzeug findet in Kapitel 4.1.2.3 statt. Funktionen des BA zum Datenaustausch mit anderen Agenten sind beispielsweise die Datenverwaltung in Bezug auf Ergebnisdaten oder zu berücksichtigende Zwischenwerte. Diese werden detailliert in Kapitel 4.2.4 betrachtet.

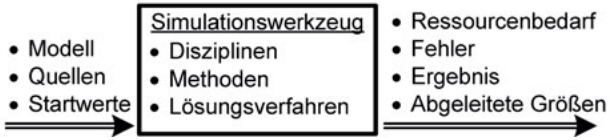

Abbildung 4.4: Parameter der Simulationswerkzeuge

Die in Kapitel 4.1.1.1 beschriebenen Fähigkeiten eines BA werden während dessen Starts im Agentensystem veröffentlicht und während der Laufzeit des Agenten aktualisiert. Die Beschreibung der Fähigkeiten erfolgt abhängig von den zur Verfügung stehenden Simulationswerkzeugen und auf Grundlage einheitlicher Bezeichnungen. Darauf Bezug nehmend analysiert der KA zu Beginn jedes Simulationsmodell und erstellt die zugehörige Modellbeschreibung. Die Umsetzung der Modellbeschreibung erfolgt durch eine hierarchische XML-Datei entsprechend Abbildung 4.5.

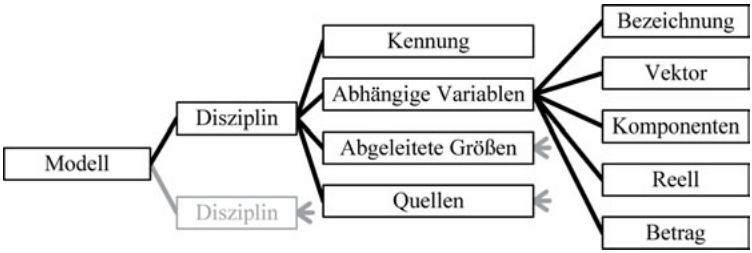

Abbildung 4.5: Aufbau der Datei zur Beschreibung der Fähigkeiten eines Agenten

Als Attribute beinhaltet die Modellbeschreibung eindeutige Bezeichnungen der Disziplinen. Darunter sind deren abhängigen Variablen $\boldsymbol{u}$, daraus berechenbare abgeleitete Größen und zur Feldberechnung nutzbare Quellen detailliert spezifiziert. Beispiele für hier implementierte abhängige Variablen sind das elektrische Potenzial φ, die elektrische Feldstärke $\boldsymbol{E}$ und die Temperatur T. Eindeutige Bezeichnungen werden auch zur Beschreibung der Methoden- und Gebietskopplung angewendet. Ein Vergleich zwischen der Modellbeschreibung und der Beschreibung der Fähigkeiten eines BA zeigt dessen anwendbare Fähigkeiten innerhalb der jeweiligen Simulation. Mögliche Kopplungen ergeben sich durch die Bestimmung von Paaren mit gleicher Bezeichnung von Quellen und abgeleiteten Größen bzw. abhängigen Variablen oder durch den Vergleich der Gebietsnummerierung mit angrenzenden Gebietsnummern.

Anwendbare Funktionen eines Simulationswerkzeugs werden beim Entwurf der Schnittstelle zum Agenten festgelegt. Ob das Simulationswerkzeug und entsprechende Fähigkeiten des BA genutzt werden, entscheidet der BA autonom und dynamisch während der Berechnung (siehe

Kapitel 4.3.1.1). Die Mitarbeit eines BA an der Simulation erfolgt auf Grundlage seiner Vermutung einer Befähigung. Dies ist analog zu einem Simulationswerkzeug zu sehen, bei dem zu Berechnungsbeginn keine Verpflichtung zur Bereitstellung einer Lösung eingegangen wird. Es wird jedoch versucht, dieses Ziel zu erreichen (Rationalität des Agenten). Die Rolle, die ein BA innerhalb des Lösungsprozesses einer Simulation einnimmt, ist so erst nach erfolgreicher Teilberechnung und im Vergleich mit anderen BA bestimmbar. Entsprechend ist die Rolle abhängig von der Konfiguration und Zusammensetzung des Agentensystems. Der statische Entwurf möglicher Rollen der BA innerhalb des Lösungsprozesses und die zur Laufzeit getroffene Auswahl entsprechen dem adaptiven Rollenverständnis der BA aus Tabelle 2.1. Da der BA als Abstraktionsebene oberhalb eines Simulationswerkzeugs betrachtet wird, ermöglicht dessen graphische Oberfläche auch eine direkte Interaktion zwischen Anwender und Simulationswerkzeug. Diese Interaktion verdeutlicht die Notwendigkeit des adaptiven Rollenverständnisses der BA. Dabei führt eine Simulation, die einem einzelnen BA zu Berechnung übergeben wird, zu deren unabhängigen Berechnung und dem außer Acht lassen möglicher Ergebnisse anderer BA. Die Belegung der Rechenressource wird dem Agentensystem über die Aktualisierung des Agentenstatus mitgeteilt (Güte der Agenten). Werden hingegen weitere BA mit Teilen der Simulation betraut, kommt es zur simulationsabhängigen Kopplung und einer kooperierenden Rolle des Agenten im System.

Beim Austausch der Ergebnisse wird von der Wahrhaftigkeit der BA ausgegangen. Erwartet wird, dass Ergebnisse ausgehend von der aktuellen Datenlage berechnet, unverfälscht weitergegeben und vom Empfänger verstanden werden (Ontologie). Voraussetzungen zu deren Weiterverarbeitung sind ein gemeinsames Referenzkoordinatensystem für das Geometriemodell und die Nutzung der im KA umgesetzten automatisch generierten einheitlichen Modellbeschreibung. Die Beschreibung und die Bereitstellung von Werten und Parametern, wie beispielsweise der Diskretisierung oder die Auswahl des Lösungsverfahrens, erfolgt autonom je BA. Auch Empfehlungen für Kopplungen der jeweiligen Simulation zu anderen Disziplinen, für die bisher kein Modell zur Verfügung steht, lassen sich agentenspezifisch aus dem Erfahrungswissen bereitstellen [157, 158].

Nach erfolgreicher Berechnung ist die Visualisierung der numerischen Ergebnisse zur physikalischen Interpretation erforderlich. Bei der Visualisierung verteilter Simulationen sind dezentral berechnete Ergebnisse von unterschiedlich großer Relevanz für den auswertenden Anwender. So sind beispielsweise aus abhängigen Variablen abgeleitete Werte häufig nur innerhalb bestimmter Teilgebiete von Interesse. Eine Möglichkeit ist es, die verteilten Teilergebnisse zu sammeln und aufzubereiten, wie es beispielsweise durch das Visualisierungsprogramm COVISE ermöglicht wird [159]. Alternativ ist die Auswertung der Teilergebnisse dezentral möglich. Dazu verfügt jeder BA über die in Abbildung 4.6 dargestellte webbasierte Visualisierungsschnittstelle [160].

Abbildung 4.6: Webbasierte Visualisierung eines Zwischenergebnisses der ortsabhängig gekoppelten Berechnung aus Kapitel 5.2.1

Diese ermöglicht dem Anwender die Visualisierung selbst auf mobilen Endgeräten. Vorausgesetzt wird die Verfügbarkeit eines Internetbrowsers. Für die Auswertung der Teilergebnisse durch die berechnende Instanz spricht zudem, dass dort Verfahren zur Nachbereitung des Ergebnisses einfach umsetzbar sind [161]. Für das Erzeugen zentraler Ergebnisse spricht eine einfache gebietsübergreifende Restfehleranalyse der Ergebnisse. Da diese gerade für Gebietszerlegungsverfahren wesentlich ist, sammelt im Fall einer Gebietszerlegung ein BA die Ergebnisse der anderen Teilgebiete und fügt diese zu einer gebietsübergreifenden Lösung zusammen. Aufgrund der in diesem Fall gleichen Berechnungseigenschaften der BA ist so auch eine mögliche Weiterverarbeitung sichergestellt. Verschiedene numerische Fähigkeiten der BA bei einer Disziplin- und Methodenkopplung verhindern die Weiterverarbeitung, weshalb hierbei auf eine zentrale Lösung verzichtet wird.

4.1.2 Kommunikationsarten

Innerhalb des Agentensystems treten drei grundverschiedene Arten der Kommunikation auf, die im Folgenden näher betrachtet werden. Dabei sind alle drei Arten der Kommunikation wesentlich, um den Betrieb das Agentensystem zu ermöglichen. Zu Beginn wird die Kommunikation zwischen dem Menschen als Anwender des Systems und den Agenten betrachtet. Danach wird die für Agentensysteme wesentliche Kommunikation der Agenten untereinander dargestellt, die zum Betrieb des verteilten Softwareagenten-basierten Simulationssystems miteinander kommunizieren. Abschließend wird auf die Kommunikation eines Agenten mit seinem über eine umgesetzte Schnittstelle angeschlossenen Simulationswerkzeug eingegangen.

4.1.2.1 Mensch - Agent

Um als Anwender des Simulationssystems Anwendungsschritte auszuführen, ist ein Dialog mit dem Rechner zu führen [162]. Innerhalb des Dialogs werden die benötigten Parameter bereitgestellt, um anschließend Methoden zu deren Verarbeitung auszuführen. Dabei beinhalten die für den Dialog notwendigen Benutzungsschnittstellen alle Bestandteile, die für den Anwender notwendig sind, um bestimmte Anwendungsschritte zu erledigen [163]. Für das verteilte Simulationssystem als reines Softwaresystem gilt es, entsprechende Benutzungsschnittstellen zu schaffen, die eine Ein- und Übergabe aller Einstellungen und Daten ermöglichen. Dabei werden die Interaktionselemente einer graphischen Benutzeroberfläche eingesetzt [162, 164]. Da entsprechend Abbildung 4.2 bei jedem Agenten eine Interaktion mit dem Anwender möglich ist, sind hier an die Gebrauchstauglichkeit (usability) angepasste graphische Benutzeroberflächen umgesetzt, die den Lösungsprozess ermöglichen. Während bei einem BA Dialoge zur numerischen Berechnung der Teilsimulationen, zur Auswertung und zur Visualisierung der Ergebnisse im Vordergrund stehen, sind es bei einem KA Dialoge zur Koordination.

4.1.2.2 Agent - Agent

Bei der Kommunikation der Agenten untereinander handelt es sich um eine M2M Kommunikation. Dabei kommunizieren verteilte Einheiten desselben Typs über ein Netzwerk miteinander und ermöglichen so Anwendungen zum Überwachen und Steuern verteilter Prozesse [165]. Der Aufbau eines typischen M2M Systems ist in Abbildung 4.7 abgebildet.

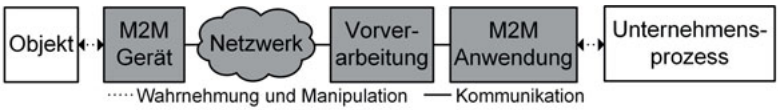

Abbildung 4.7: Aufbau eines auf der M2M Kommunikation basierenden Systems [165]

Dabei stellt hier die M2M Anwendung die interdisziplinäre Simulationsumgebung dar [166]. Diese ist einem Unternehmensprozess zugeordnet, beispielsweise der Produktentwicklung. Die einbezogenen Objekte entsprechen den verschiedenen Simulationswerkzeugen, die über einen BA (M2M Gerät) und dessen Kommunikation über das Netzwerk zusammenarbeiten, um die Simulationsergebnisse bereitzustellen. Eine Vorverarbeitung findet beispielsweise in Bezug auf die Verwaltung der BA in den KA oder im Hinblick auf ausgetauschte Daten im BA statt. Bei einem Vergleich zwischen einer verteilten und einer zentralistischen Software kommt es verteilt zu kommunikationsbedingten Verzögerungen. Die Erweiterung des Agentensystems um die Möglichkeit einer Berücksichtigung weiterer Kopplungen und Ressourcen ist hier jedoch gewichtiger, sodass diese Verzögerungen in Kauf genommen werden. Beim Entwurf der

Kommunikation ist unter Berücksichtigung der entsprechenden Entwurfsrichtlinien [165] zu achten auf:

- die Ausführung als System von gleichberechtigten Diensten,
- die Wiederverwendbarkeit in einem anderen Kontext, die dessen Entwicklung bzw. Ausführung unterstützt,
- eine Umsetzung, die sicher und vertrauenswürdig ist,
- die Skalierbarkeit, Effizienz und Leistungsfähigkeit,
- die Möglichkeit verschiedener Abstraktionsebenen zum Verbergen von Komplexität,
- die einfache Integration und Verwaltung in bzw. durch bestehende heterogene Systeme mit vollständigem Lebenszyklus,
- die kontextabhängige Nutzung verschiedener Rollen und entsprechender unterschiedlicher Dienstleistungen.

Angewendet auf das Agentensystem folgt daraus die zum Betrieb notwendige und in Abbildung 4.8 dargestellte Kommunikation der Agenten untereinander.

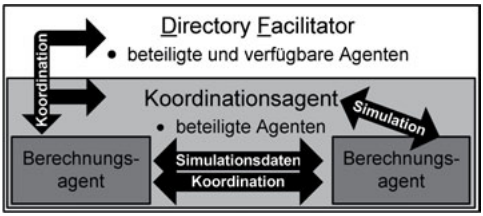

Abbildung 4.8: Agent-Agent Kommunikation innerhalb der Lösungsumgebung

Während der Austausch großer Datenmengen direkt zwischen den beteiligten Agenten geschieht, dienen Koordinationsnachrichten zur Verwaltung der funktionalen Gruppen. Beispiele für die Kommunikation zwischen den BA sind das Bekanntgeben von berechneten Lösungen oder die aktive Suche nach bereits existierenden Lösungen beim Start eines BA. Die Koordinationsnachrichten sorgen dafür, dass die Informationen über beteiligte Agenten aktuell sind. Ein Beispiel stellt der Austausch über die Fähigkeiten der Agenten dar. Die Umsetzung der Kommunikation erfolgt ereignisorientiert mittels eines eindeutigen Nachrichtenformats. Zur Übertragung großer Datenmengen, wie beispielsweise der Diskretisierung oder einer Teillösung werden entsprechende Objekte serialisiert versendet. Auf eine systemunabhängige Spezifikation der Objekte, wie beispielsweise mittels CORBA, RMI, XML-RPC oder SOAP, wird hier bewusst verzichtet, um individuelle Kommunikationsarten zwischen den Agenten zu ermöglichen.

Ein Agentensystem zur Berechnung numerischer Simulationen lässt erwarten, dass die Menge zwischen BA ausgetauschter Daten groß ist. Beispieldaten sind die Diskretisierung, Zwischenergebnisse zur Kopplung und Ergebnisse zur Visualisierung. Eine Reduktion der Datenmenge

ist mittels Kompressionsverfahren möglich. Während [167] einen Überblick über Verfahren zur Kompression der Diskretisierung gibt, gilt es bei der Kompression numerischer Ergebnisse zwischen zwei Anwendungsfällen zu unterscheiden. Wird ausgehend von den ausgetauschten Werten weitergerechnet, so ist zur Vermeidung von Rechenfehlern eine verlustfreie Kompression erforderlich. Wird ein numerisch berechnetes Ergebnis zur Visualisierung verwendet, so ist bei der Auswertung zu berücksichtigen, dass die Anzahl signifikanter Nachkommastellen des Ergebnisses stets kleiner als die Anzahl berechneter Stellen ist. Entsprechend ist eine verlustbehaftete Kompression dann möglich, wenn die Genauigkeit zur Darstellung ausreicht [168].

4.1.2.3 Agent - Simulationswerkzeug

Um als BA Simulationen berechnen zu können, werden bei dessen Start die zur Verfügung stehenden Simulationswerkzeuge ermittelt. Diese macht sich der BA entsprechend Abbildung 2.2 zunutze. Ermöglicht wird so der Zugriff auf die in Abbildung 4.9 dargestellten ausgewählten Modellparameter. Dies ist erforderlich, um eigene Berechnungen durchzuführen und aufgrund der Güte der BA auch Simulationen im Sinne des Agentensystems zu bearbeiten.

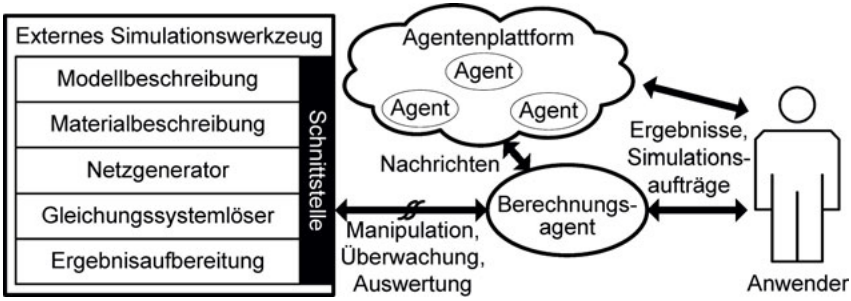

Abbildung 4.9: Darstellung eines BA mit seinen Schnittstellen

Die Schnittstelle wird als Dienstschicht (Middleware) umgesetzt. Dabei gilt es bestehende Rahmenbedingungen, wie beispielsweise das Betriebssystem, die Programmiersprachen der verbundenen Systeme oder bestehende Schnittstellen zur Anwendungsprogrammierung (API), zu berücksichtigen. Ein Überblick über verschiedene Umsetzungen von Dienstschichten unter der Berücksichtigung einer Kopplung verschiedener Programmiersprachen findet sich in [64]. Die Java-basierte Umsetzung von JADE sowie die bestehende Java-basierte Simulations-API zu einem eingesetzten Simulationswerkzeug legen die Java-basierte Implementierung der Schnittstelle im BA nahe. Dabei ist der Programmablauf desto schneller, je mehr Anwendungsschritte innerhalb der Simulationswerkzeuge erfolgen. Gründe finden sich beispielsweise in der compilergestützten softwareinternen Optimierung der Datenverarbeitung. Werden von

einem Agenten mehrere Simulationswerkzeuge genutzt, so ist zur Leistungssteigerung die Implementierung eines direkten Datenaustauschs zwischen diesen Werkzeugen zu prüfen.

4.1.3 Ausführung von Verhalten

Softwareagenten agieren durch ihr zielgerichtetes Verhalten. Eine Unterbrechung der Ausführung des Verhaltens ist hilfreich, um Rechenzeit freizugeben. Ein Beispiel dafür ist die Verarbeitung der Kommunikation, die nur dann sinnvoll ist, wenn zu verarbeitende Nachrichten existieren. Verschiedene Verhalten werden bei JADE in einer Liste verwaltet und sequenziell ausgeführt. Um ein gegenseitiges Blockieren von Verhalten (Deadlock) zu vermeiden, wird hier vor Ausführungsbeginn anhand innerer Zustände geprüft, ob eine Durchführung des Verhaltens möglich ist. Ist dies nicht der Fall, wird das Verhalten übergangen und zum erneuten Aufruf in die Warteschlange eingereiht. Ein Verhalten wird dabei so lange ausgeführt bis sein Ausführungsziel erreicht ist oder es von der Liste entfernt wird. Der Verhaltensablauf eines Agenten ist in Abbildung 4.10 zusammenfassend dargestellt.

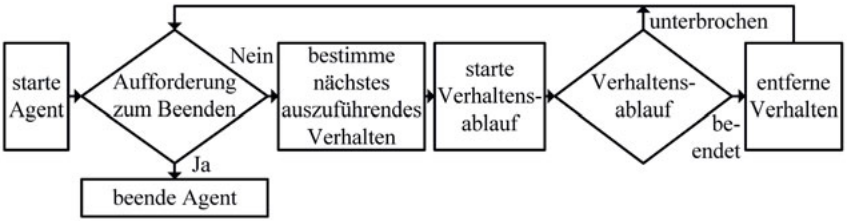

Abbildung 4.10: Verhaltensablauf eines Softwareagenten [61]

Nutzt ein Agent die ihm exklusiv zur Verfügung stehenden Werkzeuge während der Durchführung eines Verhaltens, so ist die Ausführungsdauer des Verhaltens auch bestimmt durch die Geschwindigkeit mit der diese Werkzeuge arbeiten. Beispiele für implementierungsabhängige Bearbeitungszeiten der als Werkzeug betrachteten Simulationsprogramme sind das Aufstellen und Lösen eines Gleichungssystems oder die Diskretisierung der Geometrie. Um den Agenten auch während dieser Zeit für das Agentensystem erreichbar zu machen, wird die Schnittstelle zum Simulationswerkzeug in einem eigenständigen Thread betrieben. Das entsprechende Verhalten wird nach dem Aufruf so lange unterbrochen, bis die Bearbeitung durch das Werkzeug beendet ist, und anschließend fortgesetzt. Diese Nebenläufigkeit stellt einen Unterschied zu derzeitigen Simulationswerkzeugen dar, bei denen meist die Beauftragung weiterer Abläufe (Scheduling) erst nach dem Ende bzw. einem Abbruch laufender Prozesse möglich ist. Da die Werkzeuge als Blackbox betrachtet werden, ist ein Zugriff auf das Simulationswerkzeug nur über die Schnittstelle möglich. Andere Aufrufe und Verhalten werden dann zwar initialisiert,

warten jedoch bis die Ressource freigegeben wird. Abbildung 4.11 zeigt den Umgang mit Verhalten, die eine lange Bearbeitungszeit haben.

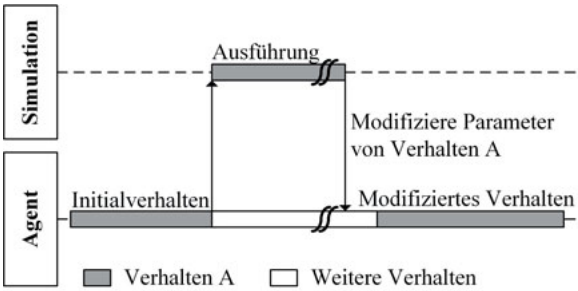

Abbildung 4.11: Organisation von Verhalten mit langer Simulationszeit

4.1.4 Vererbung

Zur Implementierung der beschriebenen Agenten wird auf die Vererbung zurückgegriffen [169]. Die von der Entwicklungsumgebung JADE bereitgestellte Klasse an Agenten dient als Vorlage. Diese wird in der davon abgeleiteten abstrakten Klasse der Basisagenten um Methoden zur Anmeldung der Agenten innerhalb des Agentensystems und zur einheitlichen Dokumentation des Verhaltens aller Agenten erweitert. Diese Methoden stehen aufgrund der Vererbung allen im System instanziierten Agenten zur Verfügung. Beispielsweise nutzt der Dokumentationsagent (DA) diese zur Behandlung aller Statusmeldungen. Die Klasse der Numerik-Agenten beinhaltet Methoden zur Realisierung der in Kapitel 4.1.2.3 beschriebenen Schnittstelle. Diese ist sowohl für BA als auch für KA notwendig. Der KA nutzt diese, um das vom Anwender an die Lösungsumgebung übergebene Modell zu analysieren. Der BA nutzt die Schnittstelle zur Simulation oder zur Durchführung vom Anwender ausgeführter lokaler Anweisungen. Die umgesetzte Vererbung spiegelt sich auch in der Funktionsbeschreibung zur Gruppenbildung wider (siehe Kapitel 4.1.1.1). Abbildung 4.12 stellt die beschriebene Klassenhierarchie der Agenten innerhalb der Lösungsumgebung dar.

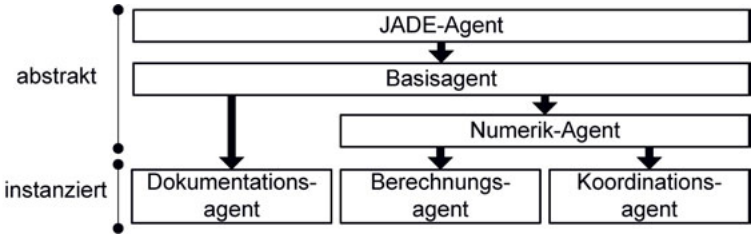

Abbildung 4.12: Klassenhierarchie der Agentenplattform

4.2 Numerischer Lösungsprozess

Im Folgenden wird der numerische Lösungsprozess, der sich aus der dynamischen Kopplung der Fähigkeiten gleichberechtigter BA bei der Lösung einer gekoppelten interdisziplinären Simulation ergibt, dargestellt. Dabei wird anstelle einer statisch gestaffelten Lösungssequenz (siehe Abbildung 3.8) ein dynamisches Lösungsverhalten erreicht und die Teilberechnungen, falls möglich, parallel berechnet. Erfahrungswissen des Anwenders zum Festlegen der Lösungssequenz wird so unnötig. Zum automatisierten Vergleich verschiedener Lösungsansätze untereinander wird das Simulationsmodell dabei so spät wie möglich assembliert (Aufstellen der Jacobimatrizen $\boldsymbol{J}_{ii}$). Dies ermöglicht gleichzeitig ein Anknüpfen an Themenfelder der Optimierung und wird dort als „First optimize then discretize“ beschrieben [170]. Abbildung 4.13 stellt die Kooperation der BA innerhalb der Lösungsumgebung schematisch dar. Dabei berechnet jeder BA ein Modell mit eigenständigen Lösungsvariablen. Die Kopplung der Teilmodelle erfolgt ereignisorientiert nach der Bereitstellung von Teillösungen.

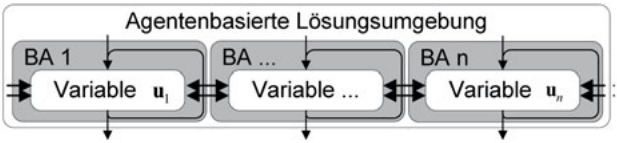

Abbildung 4.13: Umsetzung eines parallelen Lösungsalgorithmus

Der erste Abschnitt dieses Kapitels beschreibt, wie BA zu Lösungsbeginn initiale Simulationsteile erzeugen und diese anschließend parallel gelöst werden. Der zweite Abschnitt behandelt die Kopplung der Teilsimulationen, die nach der Bereitstellung einer Teillösung ereignisbasiert erfolgt. Der sich anschließende iterative Lösungsprozess führt in Bezug auf diese Kopplung zur Lösung der gekoppelten Teilsimulationen und wird in Kapitel 4.2.3 beschrieben. Vorausgesetzt wird dabei, dass sich die Fragestellung iterativ lösen lässt. Details zur Umsetzung der Kopplung innerhalb eines Agenten sind im darauf folgenden Abschnitt beschrieben.

4.2.1 Initialisierung und Segmentierung

In der Praxis ist bei der Modellierung zu beobachten, dass Modelle zu Beginn unvollständig modelliert und sequenziell um vom Entwickler für relevant erachtete Zusammenhänge, Teilgebiete und Kopplungen erweitert werden. So werden beispielsweise interdisziplinäre Fragestellungen zu Beginn entsprechend der einzelnen Disziplinen entwickelt und falls möglich durch Berechnungen validiert, bevor Kopplungen berücksichtigt werden. Übertragen auf ein Team mit agilem Projektmanagement entspricht dies einem zu Beginn unabhängigen Wirken aller Projektbeteiligten und einer anschließenden bedarfsorientierten Zusammenarbeit. Ausnahmen

stellen numerisch stark gekoppelte Fragestellungen dar, bei denen eine getrennte Betrachtung entsprechend Kapitel 3.2.1 nicht möglich ist. Diese werden im Folgenden als eine Teilsimulation mit einem Lösungsvektor $\boldsymbol{u}$ betrachtet. Ein entsprechendes Beispiel stellt die gemeinsame Betrachtung von drei abhängigen Variablen zur Berechnung eines Transistors in Kapitel 5.2.2 dar. BA gehen daher initial von unabhängig berechenbaren und vollständig beschriebenen Teilmodellen aus. Dabei berechnet BA i die Teilsimulation entsprechend der i-te Zeile von

$$\begin{bmatrix} \bar{\boldsymbol{J}}_{11} & & & \\ \vdots & & & \\ & \bar{\boldsymbol{J}}_{ii} & & \\ & \vdots & & \\ & & & \bar{\boldsymbol{J}}_{nn} \end{bmatrix} \begin{bmatrix} \Delta \boldsymbol{u}_1^{(0)} \\ \vdots \\ \Delta \boldsymbol{u}_i^{(0)} \\ \vdots \\ \Delta \boldsymbol{u}_n^{(0)} \end{bmatrix} = \begin{bmatrix} -\boldsymbol{f}_1 \\ \vdots \\ -\boldsymbol{f}_i \\ \vdots \\ -\boldsymbol{f}_n \end{bmatrix}, \tag{4.1}$$

mit $i = 1 \ldots n$ und n entsprechend der Anzahl gemeinsam betrachteter Teilsimulationen. Der Index (0) weist auf mögliche kopplungsbedingten Iterationen hin. Gleichung (4.1) entspricht dabei dem initialen Schritt des gestaffelten Lösungsprozesses, der hier jedoch parallel und ungekoppelt für alle Teilsimulationen ausgeführt wird. Das Vorgehen ist dabei analog zur Initialisierung einer monolithischen Berechnung mit dem Block-Jacobi-Verfahrens zu betrachten [125].

Aus der gewählten monolithischen Betrachtung von (4.1) lässt sich zudem die Adjazenzmatrix $\tilde{\boldsymbol{C}}^{(0)}$ des Kopplungsgraphen extrahieren (siehe Kapitel 3.2). Der Index (0) weist auch hier auf die Veränderlichkeit während des Lösungsprozesses hin. Der Kopplungsgraph verdeutlicht den zur Berechnung erfolgten Datenaustausch zwischen den Teilsimulationen. Zur Bestimmung der Adjazenzmatrix $\tilde{\boldsymbol{C}}^{(0)}$ sind die abhängigen Variablen $\boldsymbol{u}$ bezogen auf die n Teilsimulationen zu sortieren. Existiert die partielle Ableitung in der analog sortierten Jacobimatrix $\tilde{\boldsymbol{J}}$, so wird dies binär ausgedrückt. Die anschließende Oder-Verknüpfung von Einträgen mit einer Zugehörigkeit zur gleichen Teilsimulation ergibt die Kanteneinträge von $\tilde{\boldsymbol{C}}^{(0)}$. Die Größe der Adjazenzmatrix $\tilde{\boldsymbol{C}}^{(0)}$ entspricht so der Anzahl an Teilsimulationen. Für die Initialisierung gilt

$$\tilde{\boldsymbol{C}}^{(0)} = \begin{bmatrix} 1 & 0 & \ldots & 0 \\ 0 & 1 & \ddots & \vdots \\ \vdots & \ddots & 1 & 0 \\ 0 & \ldots & 0 & 1 \end{bmatrix}. \tag{4.2}$$

Die Schleifen in $\tilde{\boldsymbol{C}}^{(0)}$ weisen dabei auf die inneren Abhängigkeiten der Teilsimulationen hin. Zusätzlich ist auch eine Codierung von linearen bzw. nicht linearen Abhängigkeiten vorstellbar.

Da zur verteilten Berechnung von (4.1) die globale Systemmatrix $\tilde{\boldsymbol{J}}$ nicht existieren muss, ist das Aufstellen des Gleichungssystems und die Diskretisierung disziplinspezifisch lokal beim jeweiligen BA möglich („First optimize then discretize“). Die nun folgende Berechnung

der Teilsimulationen bestimmt den Einfluss von Quellen und Randbedingungen, die aus der Teilsimulation selbst in die Lösung eingehen. Sobald ein Agent eine erste Lösung gefunden hat, kommuniziert dieser die Lösung an alle interessierten und an der Simulation beteiligen Agenten. Dies setzt jedoch voraus, dass sich die beteiligten Agenten initiativ handelnd in Listen der an der Lösung interessierten Agenten eingetragen haben. Jeder Agent entscheidet dazu individuell, ob eine Berücksichtigung der Lösung innerhalb seiner Simulationsumgebung möglich ist.

4.2.2 Iterative Kopplung

Während sich die Notwendigkeit der Kopplung an Gebietsgrenzen direkt aus der Modellbeschreibung ergibt, lassen sich Hinweise zur Berücksichtigung von Disziplinkopplungen aus der analytischen Auswertung der Zusammenhänge der Disziplinen extrahieren [157, 158]. Dazu wird jede Größe eines physikalischen Zusammenhangs als Knoten eines Graphen interpretiert. Existiert eine Möglichkeit verschiedene Graphen zu verbinden, so entsprechen gemeinsame Knoten den bei einer Kopplung zu berücksichtigenden Größen. Dabei sind physikalische Zusammenhänge nur bedingt reversibel. Die innerhalb der eingebundenen Simulationswerkzeuge gewählte Formulierung, deren abhängige Variablen und Randbedingungen schränken die Kopplungen zusätzlich ein, sodass deren Umsetzung über eine Kopplungsgröße meist unidirektional erfolgt. Abhängigkeiten innerhalb von Materialparametern, beispielsweise die temperaturabhängige Leitfähigkeit $\kappa(T)$, setzen voraus, dass entsprechende Materialmodelle hinterlegt sind. Abbildung 4.14 stellt einen so bestimmten, reduzierten Kopplungsgraphen verschiedener physikalischer Größen dar. Dieser dient auch weniger erfahrenen Anwendern als Hinweis auf ergänzende Kopplungen einer bereits bestehenden Simulation.

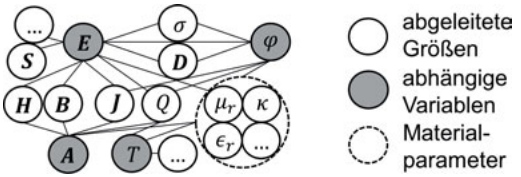

Abbildung 4.14: Kopplungsgraph verschiedener physikalischer Größen

Welche der bereitgestellten Lösungen in Form einer Kopplung berücksichtigt werden, entscheiden BA auf Grundlage ihrer Fähigkeiten (siehe Abbildung 4.4). Zur Kopplung werden mögliche eigene Quellen und Materialabhängigkeiten mit den Lösungsgrößen des BA verglichen, der die Lösung bereitstellt. Voraussetzung dazu ist eine einheitliche Bezeichnung aller Größen im Agentensystem, wobei lokale Umbenennungen möglich sind. Lösungsgrößen des BA bestehen aus dessen abhängigen Variablen und daraus abgeleiteten Größen. Beispiele abgeleiteter Größen stellen für eine Berechnung des elektrischen Potenzials φ die elektrische Flussdichte $\boldsymbol{D}$ oder

die Flächenladungsdichte σ dar. Diese lassen sich von dem BA, der die Lösung bereitstellt, auf Anfrage ebenfalls bestimmen und zur Kopplung kommunizieren.

Die Entscheidung, ob eine Berücksichtigung der Lösung sinnvoll ist oder der BA stattdessen weiter rechnet, erfolgt autonom nach Auswertung des bereits erfolgten Berechnungsfortschritts und der Änderung der Koppelgröße im Vergleich zu deren Startwerten (näheres siehe Kapitel 4.2.4). Eine unmittelbare Unterbrechung der Berechnung und Berücksichtigung der Kopplung findet statt, wenn der relative Berechnungsfortschritt kleiner als ein Schwellwert δ ist. Anderenfalls wird die Berechnung fortgesetzt und eine Berücksichtigung der Kopplung erfolgt im Anschluss. Verglichen mit der statischen Reihenfolge des gestaffelten Berechnens wird so eine dynamische Berücksichtigung der Kopplungen im Lösungsverlauf möglich. Zudem werden ausschließlich Teilsimulationen mit aktualisierten Kopplungswerten mehrfach berechnet und diese Werte frühst möglich berücksichtigt.

Erfolgt eine Kopplung, so lässt sich die Adjazenzmatrix der Kopplungen $\tilde{\boldsymbol{C}}$ als Summe der Kopplungsmatrizen $\tilde{\boldsymbol{C}}^{(k)}$ verstehen. Für ein Beispiel mit zwei BA, wobei der erste BA seine Lösung veröffentlicht und diese vom Zweiten berücksichtigt wird, ist

$$\tilde{\boldsymbol{C}} = \tilde{\boldsymbol{C}}^{(0)} + \tilde{\boldsymbol{C}}^{(1)} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}. \tag{4.3}$$

Dabei wird die Adjazenzmatrix $\tilde{\boldsymbol{C}}^{(1)}$ aus Kapitel 3.2 zu der bestehenden Adjazenzmatrix $\tilde{\boldsymbol{C}}^{(0)}$ aus Kapitel 4.2.1 hinzugefügt. Das Entfernen einer Kopplung während des Rechenvorgangs erscheint nicht sinnvoll und wäre lediglich im Fall keiner Konvergenz der Teilsimulation zur Stabilisierung denkbar. Eine Überprüfung der Adjazenzmatrix $\tilde{\boldsymbol{C}}$ ist leicht anhand einer monolithischen Berechnung für unidirektional gekoppelte Simulationen möglich.

BA berechnen dabei weiterhin die Teilsimulationen (4.1). Die Berücksichtigung einer Kopplung ist dabei entweder als Quelle oder als Materialabhängigkeit möglich. Für Kopplungsquellen erweitert sich die Energiebetrachtung aus (3.6) um die Energie der Koppelgröße, sodass die zusätzliche Quelle lediglich Einfluss auf $\boldsymbol{f}_i^{(k)}$ hat. Eine Kopplung über Materialabhängigkeiten verändert hingegen die Systemmatrix $\boldsymbol{K}_{ii}^{(k)}$ und somit auch die Jacobimatrix $\bar{\boldsymbol{J}}_{ii}^{(k)}$ und die rechte Seite $\boldsymbol{f}_i^{(k)}$. Nicht lineare Abhängigkeiten wie $\boldsymbol{f}_i^{(k)}\left(\boldsymbol{u}_i^{(k)}\right)$ oder $\bar{\boldsymbol{J}}_{ii}^{(k)}\left(\boldsymbol{u}_i^{(k)}\right)$ werden je Rechenschritt im BA berücksichtigt.

Ausgehend vom Beispiel (4.1), in dem BA 1 als erster seine Lösung veröffentlicht, folgt

$$\begin{bmatrix} \boldsymbol{I} \\ \bar{\boldsymbol{J}}_{22}^{(0)} \\ \vdots \\ \bar{\boldsymbol{J}}_{nn}^{(1)}\left(\boldsymbol{u}_1^{(0)}\right) \end{bmatrix} \begin{bmatrix} \Delta\boldsymbol{u}_1^{(1)} \\ \Delta\boldsymbol{u}_2^{(0)} \\ \vdots \\ \Delta\boldsymbol{u}_n^{(1)} \end{bmatrix} = \begin{bmatrix} \boldsymbol{0} \\ -\boldsymbol{f}_2^{(0)} \\ \vdots \\ -\boldsymbol{f}_n^{(1)}\left(\boldsymbol{u}_1^{(0)}\right) \end{bmatrix}. \tag{4.4}$$

Dabei wird exemplarisch für BA 2 angenommen, dass dieser im Gegensatz zu BA n ungekoppelt ist. Gleichung 4.4 bedeutet für eine Simulation mit n beteiligten BA, dass $n-1$ Agenten ein bereitgestelltes Ergebnis parallel auf eine Berücksichtigung hin prüfen und im Anschluss berechnen. Beim gestaffelten Lösen entsprechend Kapitel 3.3.2 erfolgt dies sequenziell. Jede Teilsimulation wird dabei mittels individuell angepasster Verfahren berechnet. Dies ist bei monolithischer Betrachtung als eine blockweise Aktualisierung des Lösungsvektors und der Jacobimatrix $\bar{\boldsymbol{J}}$ zu verstehen. Zur Unterscheidung des Vorgehens zur absoluten Asynchronität [171] ist hier zu beachten, dass Blöcke bedingt durch die als Blackbox realisierten Löser stets gleichbleibend groß sind und vollständig aktualisiert werden.

Wird im BA ein direktes Verfahren zur Berechnung verwendet, so ist dieses nach jeder Kopplung neu zu initialisieren. Eine Besonderheit bei der direkten Berechnung stellt die Kopplung über eine Quelle dar. Direkte Lösungsverfahren ermöglichen, dass eine bereits berechnete invertierte Jacobimatrix $\bar{\boldsymbol{J}}$ weiterverwendet werden kann. Allerdings ist dazu deren Speicherung notwendig. Für iterative Verfahren lässt sich der bis zur Unterbrechung berechnete Lösungsvektor $\boldsymbol{u}_i^{(k)}$, als Startvektor für den Berechnungsneustart nutzen. Weist das iterative Verfahren eine kurze Rekursion auf, wie beispielsweise das CG-Verfahren, so wird der Konvergenzverlauf des Verfahrens durch die Unterbrechung nur geringfügig verändert. Bei einer Unterbrechung von Verfahren mit langer Rekursion m, wie beispielsweise dem GMRES(m)-Verfahren, sind die m-Basisvektoren des Lösungsraums entweder zu speichern, oder sie gehen verloren. Dies wirkt sich entsprechend eines Neustarts auf die lokale Konvergenz aus. Abbildung 4.15 zeigt den Verlauf ausgewählter iterativer Lösungsverfahren nach einer Unterbrechung bei einem Fehler kleiner 10^{-1} bis zur Konvergenz kleiner 10^{-3}. Für die dargestellten Verfahren mit Multigrid-Vorkonditionierung ist trotz der Neuinitialisierung nur ein geringer Einfluss auf den Lösungsverlauf zu beobachten.

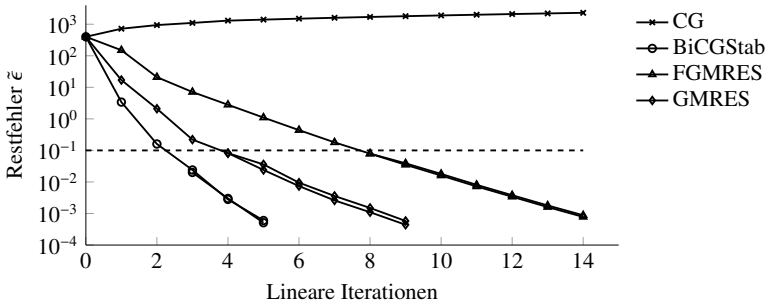

Abbildung 4.15: Lösungsverlauf verschiedener Lösungsverfahren nach dem Unterbrechen für ein Gleichungssystem entsprechend Kapitel 5.1.1

Weitere Iterationen des Agentensystems erfolgen im Anschluss analog. Der Zustand der Gesamtsimulation entspricht zum Zeitpunkt der Lösungsbereitstellung dabei stets (4.4).

4.2.3 Lösungsverlauf und Konvergenz

Das Systemverhalten des Agentensystems zur Entwurfszeit vorherzusagen, ist bedingt durch das individuelle Verhalten einzelner Agenten sehr schwierig bis unmöglich [11]. Um allgemeine Aussagen über die Zuverlässigkeit verteilter Systeme bemüht sich der 2015 innerhalb der VDI/VDE-Gesellschaft Mess- und Automatisierungstechnik gegründete Fachausschuss 7.25 „Testen vernetzter Systeme für Industrie 4.0“. Im Rahmen der Plattform Industrie 4.0, einem gemeinsamen Projekt der Verbände Bitkom e.V., VDMA e.V. und ZVEI e.V., wird mit robusten und zuverlässigen Algorithmen für zentrale und dezentrale Intelligenz gegen 2030 gerechnet [172]. Die Betrachtung der Zuverlässigkeit des Agentensystems bei der Berechnung gekoppelten Simulation, entsprechend der Konvergenz des eingesetzten Lösungsverfahrens, erfolgt daher anhand verschiedenen Einsatzszenarien. Dabei wird davon ausgegangen, dass alle Ressourcen über den gesamten Berechnungszeitraum verfügbar sind oder gleichwertig ersetzt werden. Die Berechnung gilt als beendet, wenn alle beteiligen Agenten ihre Berechnung beendet haben und keiner der Agenten einen Bedarf sieht, weitere verfügbare Ergebnisse zur Kopplung zu berücksichtigen.

Szenario 1: Bestehen zwischen mehreren Teilsimulationen keine Möglichkeiten zur Kopplung, so entspricht die Fragestellung der des Initialisierungsschritts aus (4.1). Die Koppelmatrix $\tilde{\boldsymbol{C}}$ ist dauerhaft die Identitätsmatrix $\boldsymbol{I}$. Der zugehörige Kopplungsgraph der Teilsimulationen ist unverbunden und die Berechnung der Teilsimulation erfolgt unabhängig voneinander. Konvergenz für die Gesamtsimulation ist erreicht, sobald alle Teilsimulationen konvergiert haben. Die offensichtliche Möglichkeit einer Parallelisierung der Teilsimulationsberechnung erfolgt im Agentensystem ohne Zutun eines Anwenders. Der kritische Pfad des Kopplungsgraphen entspricht dem längsten Wege innerhalb des Graphen und besteht aus der am längsten andauernden Teilsimulation, die gleichzeitig die Berechnungsdauer der Gesamtsimulation bestimmt.

Szenario 2: Bei Betrachtung zweier Teilsimulationen, die sich unidirektional von Simulation 1 zu Simulation 2 koppeln lassen, entspricht die Kopplungsmatrix $\tilde{\boldsymbol{C}}$ der Dreiecksmatrix (4.3). Zum Erreichen der Gesamtkonvergenz ist die lokale Konvergenz der Initialsimulation von BA 1 und die von BA 2 nach Berücksichtigung der Kopplung nötig. Die Rechenzeit ergibt sich aus dem kritischen Pfad durch den Kopplungsgraphen, der von Simulation 1 zu Simulation 2 verläuft. Die Berechnungsdauer entspricht so der Summe der beiden Teilrechnungen.

Monolithisch betrachtet lässt sich die Simulation beschreiben mit

$$\begin{bmatrix} \bar{\boldsymbol{J}}_{11} & \boldsymbol{0} \\ \bar{\boldsymbol{J}}_{21} & \bar{\boldsymbol{J}}_{22} \end{bmatrix} \begin{bmatrix} \Delta \boldsymbol{u}_1 \\ \Delta \boldsymbol{u}_2 \end{bmatrix} = \begin{bmatrix} -\boldsymbol{f}_1 \\ -\boldsymbol{f}_2 \end{bmatrix}. \tag{4.5}$$

Dabei stellt $\tilde{\boldsymbol{J}}$ eine reduzible Matrix dar. Während die Berechnung von (4.5) mit Splittingverfahren die Irreduzibilität voraussetzt [173], erfolgt die Reduzierung auf irreduzible Matrizen automatisiert durch das Agentensystem. Eine Darstellung der Vorgehensweise ist in [174] gegeben. Alternativ ist eine monolithische Lösung mittels Projektionsverfahren möglich.

Die parallel initialisierte Berechnung durch das Agentensystem ermöglicht zudem eine Reduktion der Rechenzeit. Deutlich wird dies am Beispiel einer unidirektionalen Quellenkopplung und einem direkten Lösungsverfahren von BA 2. Erfolgt die Berechnung durch das Agentensystem, so invertiert BA 2 während der Initialsimulation die Matrix $\bar{\boldsymbol{J}}_{22}$, die nach der Lösungsbereitstellung durch BA 1 lediglich mit der neuen rechte Seite $-\boldsymbol{f}_2$ zu multiplizieren ist. Die Bereitstellung geeigneter Zwischenlösungen für die gekoppelten Teilsimulationen wird so im Agentensystem automatisiert ausgeführt, wobei keine Konvergenz der Initialsimulation von BA 2 vorausgesetzt wird. Stattdessen entscheidet sich zum Zeitpunkt der Veröffentlichung der Lösung von BA 1, ob bisher erfolgte Berechnungen von BA 2 zur weiteren Berechnung genutzt oder verworfen werden. Der hierbei statisch festgelegte Zeitpunkt zum Austausch der Lösungen lässt sich ergänzend als Optimierungsaufgabe verstehen. Werden beide Teilsimulationen iterativ berechnet, so lassen sich bereits Zwischenergebnisse der Berechnung von BA 1 nutzen, um für eine lange andauernde Berechnung von BA 2 gute Startwerte zu erreichen. Die Bestimmung des Zeitpunkts zur Veröffentlichung der Zwischenergebnisse stellen genau wie der Restfehler des iterativen Lösungsverfahrens Parameter zur Optimierung der Rechenzeit dar. Alternative Optimierungsziele sind die Konvergenzgeschwindigkeit oder die Anzahl der linearen iterativen Rechenschritte der beteiligten BA.

Szenario 3: Werden zwei Teilsimulationen bidirektional miteinander gekoppelt, so entspricht die Adjazenzmatrix des Kopplungsgraphen

$$\tilde{\boldsymbol{C}} = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}. \tag{4.6}$$

Diese Adjazenzmatrix stellt den kleinsten Kopplungsgraphen mit einem Kreis dar. Der Kreis drückt die ringförmige Abhängigkeit der Teilsimulationen untereinander aus. Eine Ausdehnung des Kreises aus (4.6) um weitere gekoppelte Teilsimulationen führt zur Adjazenzmatrix

$$\tilde{\boldsymbol{C}} = \begin{bmatrix} 1 & 0 & \dots & 0 & 1 \\ 1 & 1 & \ddots & & 0 \\ 0 & 1 & 1 & \ddots & \vdots \\ \vdots & \ddots & 1 & 1 & 0 \\ 0 & \dots & 0 & 1 & 1 \end{bmatrix}. \tag{4.7}$$

Die in (4.7) dargestellte Struktur ergibt sich bei der monolithischen Betrachtung von beispielhaft drei Teilsimulationen entsprechend

$$\begin{bmatrix} \bar{\boldsymbol{J}}_{11} & \mathbf{0} & \bar{\boldsymbol{J}}_{13} \\ \bar{\boldsymbol{J}}_{21} & \bar{\boldsymbol{J}}_{22} & \mathbf{0} \\ \mathbf{0} & \bar{\boldsymbol{J}}_{32} & \bar{\boldsymbol{J}}_{33} \end{bmatrix} \begin{bmatrix} \Delta \boldsymbol{u}_1 \\ \Delta \boldsymbol{u}_2 \\ \Delta \boldsymbol{u}_3 \end{bmatrix} = \begin{bmatrix} -\boldsymbol{f}_1 \\ -\boldsymbol{f}_2 \\ -\boldsymbol{f}_3 \end{bmatrix}. \tag{4.8}$$

Diese Darstellung beinhaltet sowohl Material- als auch Quellenkopplungen und deren Mischung. Ein Beispiel für Letzteres ist die in Kapitel 3.2.3 dargestellte elektro-thermische Fragestellung.

Beim gestaffelten Lösen erfolgen die Berechnungsschritte innerhalb einer Lösungssequenz sequenziell. Zudem ist ein Startpunkt der Berechnung durch den Anwender festzulegen. Die Kreise im Kopplungsgraphen verhindern jedoch ein Überführen in eine Baumstruktur (kreisfrei und zusammenhängend). Entsprechend ist für die gestaffelte Berechnung kein eindeutiger Startpunkt mehr bestimmbar. Den sich durch das Agentensystem rechenzeitabhängig ergebenden Berechnungsablauf einer einzelnen Lösungssequenz zeigt Abbildung 4.16. Für die Darstellung ist angenommen, dass BA 1 die Berechnungen beginnt und nachfolgende BA jeweils die Kopplungsergebnisse abwarten, bevor diese ihre Lösung veröffentlichen.

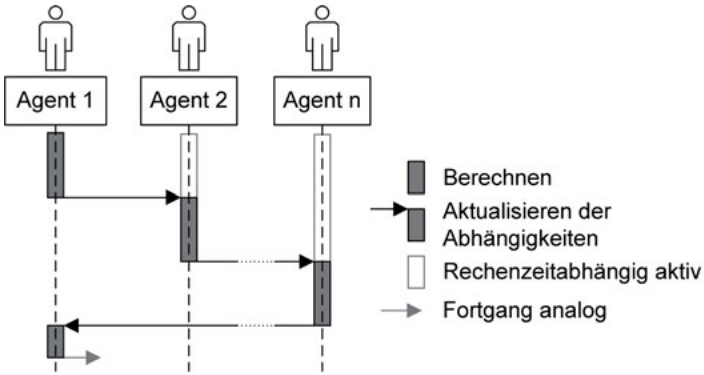

Abbildung 4.16: Aktivität der Agenten für sequenzielle Kopplungen

Dabei existiert innerhalb der dargestellten Lösungssequenz mit exklusiv befähigten BA nur ein BA, der aktiv zur Lösung beiträgt. Die Implementierung im Agentensystem ermöglicht durch den gleichzeitigen Berechnungsbeginn aller BA mehrere parallele Lösungssequenzen und somit, dass parallel Startwerte für alle Teilsimulationen berechnet werden. Dieses parallele Vorgehen bei der Entwicklung weitestgehend unabhängiger Teile stellt den Stand der Technik beispielsweise bei der Entwicklung mechatronischer Systeme in der Automatisierungstechnik dar [3]. Es dient hier zu Einsparung von Rechenzeit und der Anzahl von Rechenschritten.

Die sich aufgrund des parallelen Berechnungsbeginns und der parallel existierenden Lösungssequenzen ergebende Eigenschaften werden am Beispiel dreier sequenziell gekoppelter Simulationen mit jeweils identischen Rechenzeiten dargestellt. Tabelle 4.1 zeigt den Ablauf der parallel berechneten Lösungssequenzen für die Teilsimulationen A, B und C.

Tabelle 4.1: Parallele berechnete Lösungssequenzen zur Startvektorverbesserung

Sequenz 1:	A	neu $\longrightarrow$	B	aktualisiert $\longrightarrow$	C	aktualisiert $\longrightarrow$	A	abhängig $\longrightarrow$	B	…
Sequenz 2:	B	neu $\longrightarrow$	C	aktualisiert $\longrightarrow$	A	aktualisiert $\longrightarrow$	B	abhängig $\longrightarrow$	C	…
Sequenz 3:	C	neu $\longrightarrow$	A	aktualisiert $\longrightarrow$	B	aktualisiert $\longrightarrow$	C	abhängig $\longrightarrow$	A	…

Die drei hier parallel ausgeführten Lösungssequenzen berechnen zu Beginn die initialen Teilsimulationen A, B, C des Simulationsmodells. Bei Betrachtung von Lösungssequenz 1 werden im 2. Rechenschritt die Ergebnisse von B analog zu Szenario 2 berücksichtigt. Allerdings liegen durch die parallel ausgeführte Lösungssequenz 2 bereits Startwerte für B vor, die sich nutzen lassen. Im 3. Schritt folgt die Berechnung von C, für die bedingt durch Lösungssequenz 3 ebenfalls Startwerte vorliegen und die aufgrund Lösungssequenz 2 bereits die Kopplung von B beinhaltet. Abhängig vom kopplungsbedingten Einfluss von A auf C ist nach der Berechnung zu prüfen, ob weitere Iterationen notwendig sind. Analoges gilt für die Betrachtung der Lösungssequenzen 2 und 3. Dementsprechend gestatten parallele Lösungssequenzen ein vorzeitiges Bereitstellen bereits parallel berechneter Startwerte verglichen mit dem gestaffelten Berechnen.

Für ungleiche Rechenzeiten der Teilsimulationen findet ein Einholen der langsameren Lösungssequenz durch die schnelleren statt. Unterbricht der BA seine Berechnung und berücksichtigt die bereitgestellten Werte sofort, so entspricht dies dem Beenden der langsameren Lösungssequenz und dem Fortsetzen der schnelleren. Wird die Berechnung zuerst beendet und die bereitgestellten Werte dann berücksichtigt, so bleiben beide Lösungssequenzen erhalten. Dabei sind allgemeine Aussagen zur Konvergenz bzw. zum optimalen Zeitpunkt der Berücksichtigung der Lösungen noch Gegenstand der Forschung (siehe Szenario 2). In [125] wird die Konvergenz unter Verwendung eines nicht näher beschriebenen Relaxationsfaktors bestätigt, der hier zu Eins gewählt wird und so bisher keine Relaxation gestattet. Die Berücksichtigung der Ergebnisse erfolgt hier sofort nach deren Veröffentlichung, wenn eine Berücksichtigung möglich und der bisherige Berechnungsfortschritt kleiner als 50 % ist. Dieser frei gewählte Schwellwert stellt für iterative Berechnungsverfahren eine relative Beziehung zwischen dem akzeptierten Restfehler $\tilde{\epsilon}$ und dem bisher erzielten Restfehler entsprechend (3.25) dar. Für direkte Verfahren ist der Fortschritt eine auf die verbleibende Anzahl zu eliminierender Freiheitsgrade bezogene Größe. Ein erneutes Beleben von Lösungssequenzen, bei denen ein Einholen stattgefunden hat, bringt keinen Mehrwert. Stattdessen erlaubt es die Individualität der BA, freie Rechenzeiten entsprechend der eigenen Ziele zu nutzen. Ein Unterbinden paralleler Lösungssequenzen ist durch ein temporäres Zulassen benötigter BA möglich. Diese Form der Koordination des Lösungsverlaufs

wird jedoch nicht näher betrachtet, da sie den Handlungsspielraum der BA einschränkt.

Gesamtkonvergenz wird erzielt, nachdem wenigstens eine Lösungssequenz alle Teilsimulationen durchlaufen hat und zudem ein weiterer Lösungsschritt ausgeführt wurde. Vorausgesetzt wird, dass der entsprechend der Lösungssequenz dann aktive BA keine nennenswerte Veränderung seiner Lösung feststellt. Dabei ist es ausreichend, wenn eine der bis zum Ende aktiven Lösungssequenzen aus konvergierenden Teilsimulationen besteht. Eine nachträgliche Bewertung der Qualität eines Lösungsschritts ist durch die in [175] gegebenen Fehlerschranken möglich. Da auch einfache Fehlerschranken wie (3.28) ausreichen, um eine Lösung zu bewerten, finden diese hier Anwendung. Zur Bestimmung der Berechnungsdauer ist eine Erweiterung der Methode des kritischen Pfads notwendig, da diese nicht für Graphen mit Kreisen geeignet ist [176]. Dafür wird ein Knoten im Kopplungsgraphen als unberücksichtigt betrachtet, wenn sich die Lösung des mit einer Kante verbundenen Vorgängers geändert hat. Die Berechnungsdauer entspricht so der Summe von Rechenzeiten aller Lösungsschritte entlang dieses erweiterten kritischen Pfades und beinhaltet alle Lösungsschritte der ausgewerteten Lösungssequenz.

Szenario 4: Die hier betrachtete Erweiterung von Szenario 3 bestehend aus drei ringförmig gekoppelten Teilsimulationen mit einer weiteren unidirektionalen Kopplung. Der Kopplungsgraph entspricht dabei

$$\tilde{\boldsymbol{C}} = \begin{bmatrix} 1 & 0 & 1 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix}. \tag{4.9}$$

Eine Permutation der Adjazenzmatrix ist zudem möglich, sodass die Teilsimulation mit bidirektionaler Kante des Graphen hier Simulation 1 zugeordnet ist. Abbildung 4.17 stellt mögliche Lösungsverläufe dar. Analog zu Szenario 3 sind auch hier mehrere parallele Lösungssequenzen möglich, die ein sich gegenseitiges Einholen ermöglichen. Dabei und im Folgenden wird zur Darstellung erneut davon ausgegangen, dass BA 1 seine Lösung zuerst bereitstellt.

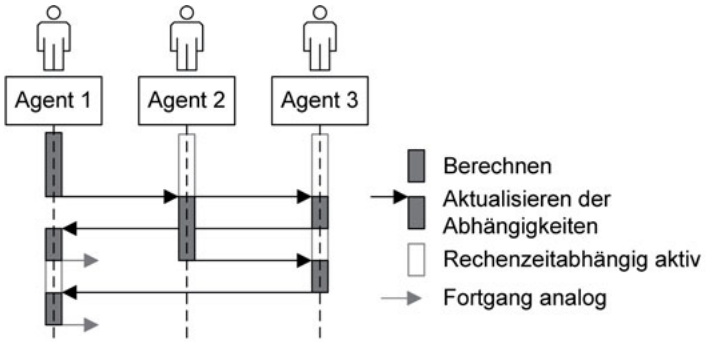

Abbildung 4.17: Aktivität der Agenten für erweitert sequenzielle Kopplungen

Nachdem BA 1 das erste Mal seine Lösungen veröffentlicht hat, werden diese von BA 2 und 3 in deren aktuelle Berechnung entsprechend

$$\begin{bmatrix} \boldsymbol{I} \\ \bar{\boldsymbol{J}}_{22}^{(1)}\left(\boldsymbol{u}_1^{(0)}\right) \\ \bar{\boldsymbol{J}}_{33}^{(1)}\left(\boldsymbol{u}_1^{(0)}\right) \end{bmatrix} \begin{bmatrix} \Delta\boldsymbol{u}_1^{(1)} \\ \Delta\boldsymbol{u}_2^{(1)} \\ \Delta\boldsymbol{u}_3^{(1)} \end{bmatrix} = \begin{bmatrix} \boldsymbol{0} \\ -\boldsymbol{f}_2^{(1)}\left(\boldsymbol{u}_1^{(0)}\right) \\ -\boldsymbol{f}_3^{(1)}\left(\boldsymbol{u}_1^{(0)}\right) \end{bmatrix} \tag{4.10}$$

berücksichtigt. Die nach der Berechnung von (4.10) folgende Fragestellung lässt sich nach einer Permutation des Kopplungsgraphen entsprechend des BA, der als nächstes seine Lösung bereitstellt, eindeutig darstellen als

$$\begin{bmatrix} \bar{\boldsymbol{J}}_{11}^{(2)}\left(\boldsymbol{u}_2^{(1)}\right) \\ \boldsymbol{I} \\ \bar{\boldsymbol{J}}_{33}^{(2)}\left(\boldsymbol{u}_1^{(0)}, \boldsymbol{u}_2^{(1)}\right) \end{bmatrix} \begin{bmatrix} \Delta\boldsymbol{u}_1^{(2)} \\ \Delta\boldsymbol{u}_2^{(2)} \\ \Delta\boldsymbol{u}_3^{(2)} \end{bmatrix} = \begin{bmatrix} -\boldsymbol{f}_1^{(2)}\left(\boldsymbol{u}_2^{(1)}\right) \\ \boldsymbol{0} \\ -\boldsymbol{f}_1^{(2)}\left(\boldsymbol{u}_1^{(0)}, \boldsymbol{u}_2^{(1)}\right) \end{bmatrix}. \tag{4.11}$$

Gilt für die Rechenzeiten t_{1-3} der Teilsimulation 1-3 stets $t_1 < t_2 < t_3$, so lässt sich bei sofortigem Berücksichtigen der Ergebnisse die Lösungssequenz eindeutig vorhersagen. Dabei werden die parallel ausgeführten Berechnungen durch die Rechenzeiten bedingt innerhalb einer Lösungssequenz vereinigt. Eine Untersuchung des entstandenen sequenziellen Lösungsverlaufs und der Konvergenz ist dann analog zu Szenario 3 möglich. Gleiches gilt bei einer Koordination des Lösungsverlaufs entsprechend der Wünsche des Anwenders, beispielsweise durch den KA. Auch dann ist die Lösungssequenz vorhersagbar und ermöglicht durch das Zusammenfassen von Teilsimulationen eine Konvergenzanalyse und Rechenzeitauswertung entsprechend Szenario 2 und 3. Dies wäre hier beispielsweise durch das Zusammenfassen der Teilsimulation 1 und 3 zu einer bidirektionalen Simulation, die dann wiederum bidirektional mit Teilsimulation 2 gekoppelt ist, möglich. Die Koordination des Lösungsverlaufs macht diesen zwar vorhersehbar, schränkt allerdings Eigenschaften des Agentensystems entsprechend Kapitel 2.3 ein. Beispiele sind ein Verlust an Autonomie der BA und eine Einschränkung im zielgerichteten Verhalten, schnellstmöglich eine Lösung für die eigene Teilsimulation zu finden. Um BA im Kontrast zu [142] und [143] als autonome Einheiten zu realisieren, wird hier auf den Einfluss einer zentralen Koordinationsinstanz verzichtet und sich mit einer posteriori Konvergenzanalyse begnügt. Diese setzt jedoch die Konvergenz der gekoppelten Berechnung voraus.

Zu einem Aufspalten der Lösungssequenz in zwei Teilsequenzen kommt es beispielsweise für $t_3 < t_2$. Analoges gilt für Simulationen mit einem Kopplungsgraphen entsprechend

$$\tilde{\boldsymbol{C}} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 0 & 1 \end{bmatrix} \quad \text{oder} \quad \tilde{\boldsymbol{C}} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}. \tag{4.12}$$

Dabei ermöglicht jede Aktualisierung der Werte ein erneutes Durchlaufen des Kopplungsgra-

phen, wobei wiederum ein Aufteilen möglich wird. Aufgrund des Einholens der Lösungssequenzen untereinander und der drei verfügbaren BA sind hier jedoch maximal drei Teilsequenzen möglich. Für die Kopplungsmatrix (4.9) ergeben sich aus (4.11) im dritten Berechnungsschritt nach der Veröffentlichung von BA 3 die zu berechnenden Simulationen

$$\begin{bmatrix} \bar{\boldsymbol{J}}_{11}^{(3)}\left(\boldsymbol{u}_2^{(1)},\boldsymbol{u}_3^{(2)}\right) \\ \bar{\boldsymbol{J}}_{22}^{(3)}\left(\boldsymbol{u}_1^{(0)},\boldsymbol{u}_3^{(2)}\right) \\ \boldsymbol{I} \end{bmatrix} \begin{bmatrix} \Delta\boldsymbol{u}_1^{(3)} \\ \Delta\boldsymbol{u}_2^{(3)} \\ \Delta\boldsymbol{u}_3^{(3)} \end{bmatrix} = \begin{bmatrix} -\boldsymbol{f}_1^{(3)}\left(\boldsymbol{u}_2^{(1)},\boldsymbol{u}_3^{(2)}\right) \\ -\boldsymbol{f}_2^{(3)}\left(\boldsymbol{u}_1^{(0)},\boldsymbol{u}_3^{(2)}\right) \\ \boldsymbol{0} \end{bmatrix}. \tag{4.13}$$

Bedingt durch die Aufteilung der Lösungssequenz ist auch ein Veröffentlichen von BA 1 möglich. Dies führt im dritten Schritt zur Berechnung von

$$\begin{bmatrix} \boldsymbol{I} \\ \bar{\boldsymbol{J}}_{22}^{(3)}\left(\boldsymbol{u}_1^{(2)}\right) \\ \bar{\boldsymbol{J}}_{33}^{(3)}\left(\boldsymbol{u}_1^{(2)},\boldsymbol{u}_2^{(1)}\right) \end{bmatrix} \begin{bmatrix} \Delta\boldsymbol{u}_1^{(3)} \\ \Delta\boldsymbol{u}_2^{(3)} \\ \Delta\boldsymbol{u}_3^{(3)} \end{bmatrix} = \begin{bmatrix} \boldsymbol{0} \\ -\boldsymbol{f}_2^{(3)}\left(\boldsymbol{u}_1^{(2)}\right) \\ -\boldsymbol{f}_3^{(3)}\left(\boldsymbol{u}_1^{(2)},\boldsymbol{u}_2^{(1)}\right) \end{bmatrix}. \tag{4.14}$$

Zum möglichen Erreichen der Gesamtkonvergenz ist es notwendig, dass lediglich eine der parallel ausgeführten Lösungssequenzen zu einem Ergebnis kommt. Im Vergleich des hier realisierten Lösungsablaufs mit dem gestaffelten Lösen bedeutet dies selbst dann eine Lösungsbereitstellung, wenn eine der parallelen Teilsimulationen in beispielsweise (4.13) oder (4.14) nicht konvergiert. Das Agentensystem stellt so ein System dar, das sich adaptiv an verschiedene Rechenzeiten der Simulationen anpasst und verfügbare Zwischenlösungen dynamisch in der Lösungssequenz berücksichtigt.

Eine allgemeine Beschreibung des Konvergenzverhaltens des Agentensystems ist bei monolithischer Betrachtung angelehnt an das zweistufige äußere Block-asynchrone Jacobi-Verfahren [171] möglich, wenn ausschließlich lineare Teilsimulationen betrachtet werden. Das zweistufige äußere Block-asynchrone Jacobi-Verfahren unterteilt dabei die zu berechnende Matrix (hier entsprechend der Jacobimatrix $\tilde{\boldsymbol{J}}$) blockweise und berechnet diese Blöcke in beliebiger Reihenfolge iterativ unter Berücksichtigung bereits verfügbarer Teillösungen $\tilde{\boldsymbol{u}}^{(s)}$. Dabei können Blöcke auch mehrfach hintereinander, parallel oder mittels verschiedener Berechnungsverfahren berechnet werden. Trotz dieser Dynamik während der Berechnung und der blockabhängigen Berechnungszeiten existiert für das zweistufige äußere Block-asynchrone Jacobi-Verfahren eine allgemeine Konvergenzaussage [171, 177, 178]. Vorausgesetzt wird dabei die strikte Diagonaldominanz der Jacobimatrix $\tilde{\boldsymbol{J}}$. Für gekoppelte nicht lineare Teilsimulationen wird erwartet, dass die Diagonaldominanz ebenfalls für alle linearisierten Teilsimulationen $\bar{\boldsymbol{J}}_{i,o}$ erforderlich ist. Die Diagonaldominanz entspricht dabei einem geringen Einfluss der Kopplungsblöcke $\bar{\boldsymbol{J}}_{i,o\neq i}$ im Vergleich zu dem disziplinspezifischen Einfluss von $\bar{\boldsymbol{J}}_{i,i}$. Entsprechend ist es auch hier möglich, von einer schwachen Kopplung zu sprechen (siehe Kapitel 3.2.1).

Der Verzicht auf eine Koordinierung der globalen Berechnung bedeutet dabei, dass den BA

ausschließlich lokale oder mittels Kommunikation bestimmbare Informationen zur Verfügung stehen. Auf eine Beschleunigung der globalen Konvergenz, beispielsweise mit einer Anderson-Beschleunigung oder dem Quasi-Newton-Verfahren [113], muss daher verzichtet werden. Eine Beschleunigung der lokalen Konvergenz ist hingegen beispielsweise mit dem Jacobi-Überrelaxationsverfahren [179] vorstellbar. Allerdings setzt dies die Speicherung der Geschichte der gekoppelten Werte und sich daraus berechenbarer abhängiger Variablen voraus.

4.2.4 Agentenspezifischer Berechnungsschritt

Die Umsetzung der dargestellten Lösungsschritte in den BA erfolgt anhand einer Betrachtung des BA n, der eine Nachricht über verfügbare Ergebnisse von BA m empfängt. Das darauf folgende Verhalten von BA n gliedert sich in die drei Phasen der Vorauswertung, der Berechnung und der abschließenden Auswertung.

Zu Beginn steht die Vorauswertung, die parallel zu einer möglichen stattfindenden Berechnung von BA n erfolgt. Dabei wird überprüft, ob bereits eine Kopplung mit dem Herausgeber der Lösungsnachricht erfolgt ist. Ist dies nicht der Fall, werden die veröffentlichten Ergebnistypen mit den lokalen Fähigkeiten des Agenten daraufhin verglichen, ob eine Verwendung in der lokalen Berechnung möglich ist. Die Verwendung ist als Quelle entsprechend dem Produkt aus Systemmatrix $\boldsymbol{K}_{nm}$ und Lösungsvektor $\boldsymbol{u}_m$, oder als Materialabhängigkeit innerhalb von $\boldsymbol{K}_{nn}$ möglich. Können keine Ergebnisse berücksichtigt werden, endet das Verhalten in Bezug auf die eingegangene Lösung. Das weitere Verhalten von BA n findet dann wie in Kapitel 4.1.3 beschrieben statt.

Ist hingegen einer der Ergebnistypen zur Kopplung nutzbar, so bestimmt BA n die Koordinaten der Punkte im Modell, die eine Berücksichtigung ermöglichen. Diese Koordinaten werden dem Herausgeber der Lösung einschließlich des benötigten Ergebnistyps mitgeteilt (hier BA m). Im Fall, dass bereits eine Kopplung mit dem Herausgeber stattgefunden hat und es zu keiner Veränderung an der Diskretisierung gekommen ist, sind die Koordinaten dem Herausgeber bereits bekannt. Eine Aufforderung zur erneuten Übermittlung der Ergebnisse ist in diesem Fall ausreichend. BA m bestimmt daraufhin die angeforderten Ergebnisse und teilt sie BA n mit. Dabei wird von der Wahrhaftigkeit als Eigenschaft der Agenten ausgegangen, sodass jeder Agent die angefragten Ergebnisse möglichst präzise bestimmt und weitergibt.

Ist der Ergebnistyp bereits in der Simulation berücksichtigt, so sind bei gleichem Herausgeber die vorhandenen Kopplungswerte zu aktualisieren. Stammen bereitgestellte Ergebnisse von einem bisher unberücksichtigten Ergebnistyp, so sind die Kopplungswerte entsprechend (4.4) in der Teilsimulation zu berücksichtigen, da deren Auswirkungen auf die Simulation unbekannt sind. Mittels (3.28) bestimmte Änderungen bei den berücksichtigten Kopplungswerten und bisher unbekannte Ergebnistypen legen so die Neuberechnung der Simulation nahe. Dabei ist

eine Änderung nur in dem summierten Gesamtwert des jeweiligen Ergebnistyps ausschlaggebend und zur Minimierung der Auswertezeit auch nur dort zu berechnen. Der dazu notwendige lokale Ergebnisspeicher verwaltet alle in der Teilsimulation berücksichtigten Kopplungswerte und stellt im Fall von mehreren Datenquellen eines Ergebnistyps einen Gesamtwert zur Berücksichtigung in der Simulation bereit. Abbildung 4.18 stellt einen Ergebnisspeicher und dessen Befüllen mit Ergebnissen verschiedener Ergebnistypen dar.

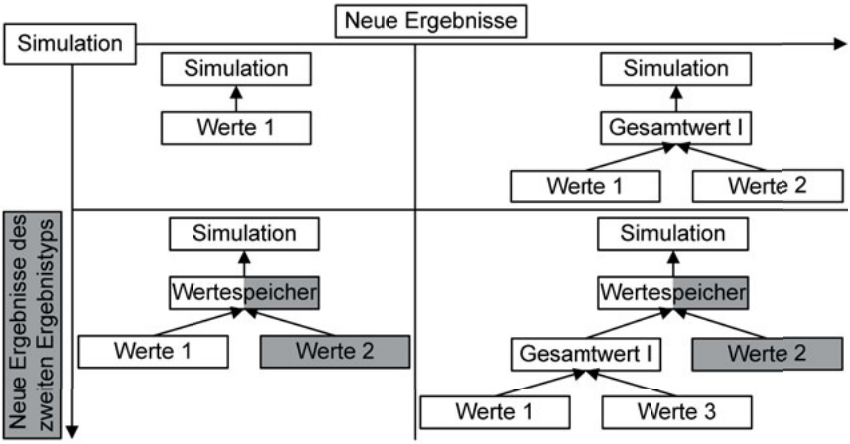

Abbildung 4.18: Verwaltung der zu integrierenden Ergebnisse [180]

Bedingt durch die Vorauswertung wird das gestaffelte Lösungsverfahren so um eine kopplungsabhängige und automatisch auf die Aufgabenstellung angepasste Bestimmung der Änderungen der Kopplungsgrößen vor jedem Berechnungsschritt erweitert. Damit lässt sich simulationsabhängig die Neuberechnung des Gleichungssystemes bei nahezu unveränderten Kopplungsgrößen, jedoch veränderten abhängigen Variablen der gekoppelten Simulationen vermeiden. Beispielhaft zeigt sich dies an einer durch Materialwerte bedingten Skalierung einer zur Kopplung verwendeten abgeleiteten Größe. Dabei reduziert die Skalierung die Aussagekraft des Schwellwerts im gestaffelten Lösungsverfahren zur Beurteilung der Änderung des Lösungsvektors entsprechend (3.28) für die Kopplung.

Die Berechnung einer Teilsimulation entspricht der zweiten Phase eines agentenspezifischen Berechnungsschritts. Ist das Simulationswerkzeug des BA bereits belegt, ist eine Unterbrechung und das Fortsetzen mit aktualisierten Werten zu prüfen.

Die Auswertung der Lösungsvariablen in Hinblick auf eine Änderung erfolgt in Phase 3 des Berechnungsschritts. Haben nur sehr kleine oder numerisch bedingte Veränderungen stattgefunden [181], wird das Ergebnis nicht veröffentlicht und unnötige Berechnungsiterationen werden

verhindert. Ausnahmen stellen die Berechnungen der Initialwerte dar, deren Ergebnisse immer veröffentlicht werden. Mit der Veröffentlichung einer Lösung beginnt ein weiterer agentenspezifischer Berechnungsschritt. Abbildung 4.19 fasst den beschriebenen Ablauf zur Kopplung zweier BA einschließlich eines initialen eigenständigen Suchens nach bereits existierenden Lösungen durch BA 2 zusammen.

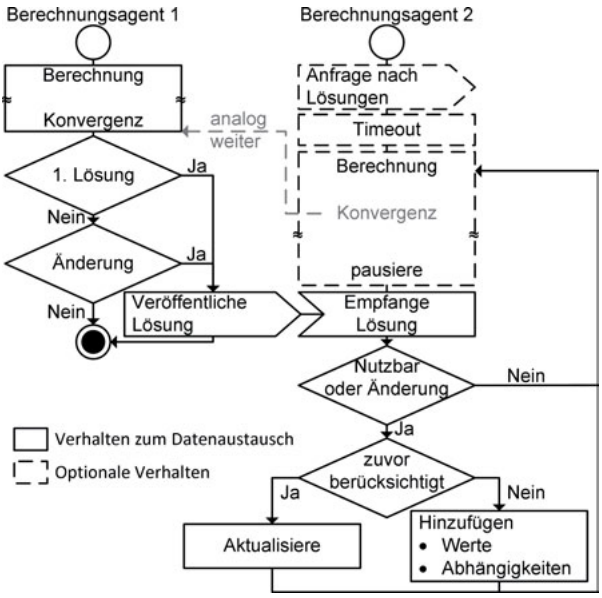

Abbildung 4.19: Datenaustausch bei einer Kopplung zweier Agenten [39]

4.2.5 Gebiets- und Methodenkopplung

Die für eine Kopplung verschiedener Disziplinen beschriebene Vorgehensweise lässt sich auf die Kopplung verschiedener Gebiete und unterschiedlicher Berechnungsmethoden übertragen (siehe Kapitel 3.2.4 und 3.2.5). Aus Gründen der Analogie wird im Folgenden jeweils nur eine kurze Einführung in die Themenbereiche gegeben. Die Darstellung erfolgt dabei anhand von im Vorfeld linearisierten Teilsimulationen.

4.2.5.1 Methodenkopplung

Die Kopplung verschiedener Methoden innerhalb einer Simulation, wie beispielsweise der FEM und der BEM, ist in Abbildung 4.3 hierarchisch unterhalb der Disziplinkopplung dargestellt.

Dies ermöglicht, dass innerhalb einer Disziplin verschiedene Methoden um die Lösungsbereitstellung konkurrieren können. Entsprechend sind Methoden, deren Nutzbarkeit ein Alleinstellungsmerkmal der BA darstellt, Disziplinen zugeordnet. Eine interdisziplinäre Kopplung über Methodengrenzen hinweg stellt eine durch Koordination mögliche Erweiterung des Systems dar. Da jede Methode sehr unterschiedliche Eigenschaften hat, sind zur Kopplung Schnittstellen zwischen den Methoden zu definieren. Auf Anfrage stellen die umgesetzten Schnittstellen dabei koordinatenabhängig Werte bereit. Zu beachten ist, dass benötigte Werte methodenspezifisch unterschiedlich vorliegen. Für eine unidirektionale Kopplung von der BEM hin zur FEM gilt es so, benötigte Werte durch Nachbereitung (Postprocessing) aus der Teillösung erst zu generieren [135]. Der zur Wertebestimmung benötigte Rechenaufwand ist dabei nennenswert [95] und wird aufgrund dessen innerhalb des konkurrierenden Systems berücksichtigt.

Die mittels der BA realisierte Abstraktionsebene verwaltet diese Kopplungsschnittstellen und übersetzt einen BA erreichende Werteanfragen auf das verfügbare Werkzeug und somit auf die entsprechende Methode. Wird beispielsweise das Modellgebiet Ω betrachtet, so ist zur Umsetzung der Methodenkopplung ein Aufteilen in zwei Teilgebiete Ω_1 und Ω_2 notwendig, die jeweils mit unterschiedlichen Methoden berechnet werden. Die Kopplung der beiden Methoden findet an den Grenzen der Teilgebiete statt (siehe Abbildung 4.20). Dabei ist die Methodenkopplung mit überlappendem Volumen an den Gebietsgrenzen von $\Omega^*_{1,2}$ analog zur bereits beschriebenen Disziplinkopplung möglich. Das Überlappen der Gebiete führt dabei zwar zu einem Mehraufwand an Kommunikation und zur doppelten Berechnung der zugehörigen Freiheitsgrade, jedoch ermöglicht es die Kopplung und dient der Konvergenz. Eine Methodenkopplung ohne Überlapp erfolgt an den gemeinsamen Gebietsrändern $\Gamma_{1,2}$. Beide Arten der Kopplung weisen dabei eine Analogie zur Gebietskopplung auf, die im Folgenden beschrieben wird.

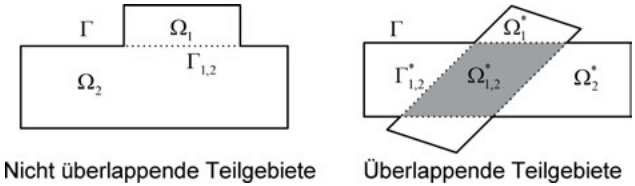

Abbildung 4.20: Kopplung zweier Teilgebiete

4.2.5.2 Gebietskopplung

Die hierarchische Untergliederung der Kopplungen in Abbildung 4.3 ermöglicht es, disziplin- und methodengleiche Teilmodelle im Agentensystem zu einem Gesamtmodell zu verbinden. Die Auswahl der zur Lösung eingesetzten Simulationswerkzeuge beschränkt sich unter den Randbedingungen der Disziplin- und Methodengleichheit auf einige wenige Werkzeuge. Dies

bedeutet, dass zur Berechnung eingesetzte BA ähnliche Fähigkeiten aufweisen und in gleichen funktionellen Gruppen verwaltet werden. Entsprechend ist auch ein Wiederverwenden der bereits realisierten Koppelschnittstellen zu bestehenden Werkzeugen möglich. Auch das zur Disziplinkopplung eingeführte einheitliche globale Koordinatensystem bleibt notwendig. Es dient als Basis zur Spezifikation der Gebiete und deren Kopplungen und ermöglicht die Kommunikation benachbarter BA.

Für die Gebietskopplung als Teil der Klasse der Gebietszerlegungsverfahren gilt im Idealfall, dass die Diskretisierungen auch an den Gebietsgrenzen konform sind. Entsprechend kann ein Gleichungssystem analog zu (3.7) formuliert werden. Ein Vergleich des Gauß-Seidel-Lösungsverfahrens mit dem multiplikativen Schwarz-Gebietszerlegungsverfahren zeigt, dass beide Verfahren die berechneten Teillösungen sofort berücksichtigen. Analog zum Block-Jacobi-Verfahren ist das additive Schwarz-Verfahren zu sehen. Bei diesen beiden Verfahren wird der Lösungsvektor erst nach der Berechnung aller Komponenten aktualisiert [100]. Unter Berücksichtigung der in [103] und [182] beschriebenen Bedingungen, wie beispielsweise disjunkter Teilgebiete, stimmen die genannten vier Verfahren paarweise sogar überein. Dabei führt ein Überlappen der Teilgebiete bei dem zweistufigen äußeren Block-asynchronen Gauß-Seidel-Verfahren zur schnelleren Fortpflanzung der Informationen auf den Nebendiagonalen und lässt so eine verbesserte Konvergenz erwarten [171]. In Anlehnung daran ist eine ebenfalls asynchrone Reihenfolge, in der die Teilgebiete berechnet werden, bei dem additiven Schwarz-verfahren möglich [183]. Dies legt die Nutzung des Agentensystems auch zur Gebiets- und Methodenkopplung nahe.

Die bisher betrachteten Schwarz-Verfahren zählen zu den direkten Gebietskopplungsverfahren [103]. Dabei werden ausgehend von (3.7) die durch Randbedingungen vorgegebenen Lösungsvariablen $\boldsymbol{u}_x$ von den zu berechnenden Lösungsvariablen $\boldsymbol{u}_f$ entsprechend

$$\begin{bmatrix} \boldsymbol{K}_{ff} & \boldsymbol{K}_{fx} \\ \boldsymbol{K}_{xf} & \boldsymbol{K}_{xx} \end{bmatrix} \begin{bmatrix} \boldsymbol{u}_f \\ \boldsymbol{u}_x \end{bmatrix} = \begin{bmatrix} \boldsymbol{r}_f \\ \boldsymbol{r}_x \end{bmatrix}. \tag{4.15}$$

getrennt. Direkte Verfahren lösen

$$\boldsymbol{K}_{ff}\boldsymbol{u}_f = \boldsymbol{r}_f - \boldsymbol{K}_{fx}\boldsymbol{u}_x. \tag{4.16}$$

Alternativ zu den direkten Verfahren sind Lagrange-Multiplikator-Algorithmen zu betrachten. Dabei werden Randbedingungen an Gebietsgrenzen mittels Lagrange-Polynomen beschrieben [184]. Für diese Verfahren lässt sich (4.15) darstellen als

$$\begin{bmatrix} \boldsymbol{K}_{ff} & \boldsymbol{K}_{fx} & \boldsymbol{0} \\ \boldsymbol{K}_{xf} & \boldsymbol{K}_{xx} & \boldsymbol{I} \\ \boldsymbol{0} & \boldsymbol{I} & \boldsymbol{0} \end{bmatrix} \begin{bmatrix} \boldsymbol{u}_f \\ \boldsymbol{u}_x \\ \boldsymbol{\lambda} \end{bmatrix} = \begin{bmatrix} \boldsymbol{r}_f \\ \boldsymbol{r}_x \\ \boldsymbol{u}_x \end{bmatrix}. \tag{4.17}$$

Die Beziehung zwischen den Lagrange-Multiplikatoren $\boldsymbol{\lambda}$ und einer Formulierung mittels eines zweifachen Steklov-Poincaré-Operators wird in [185] und [184] gezeigt. Dabei überführt der Steklov-Poincaré-Operator Randwerte von einem Gebiet in das des anderen. Bestehende Methoden lassen sich dabei auf der Art der Kopplung der Ränder und deren Berücksichtigung in Dirichlet-Dirichlet, Dirichlet-Neumann und Neumann-Neumann Algorithmen unterteilen. Die unter dem Begriff FETI (Finite Element Tearing and Interconnecting) zusammengefassten Gebietszerlegungsverfahren können aufgrund der Lösung eines Neumann-Neumann Problems analog zu diesen betrachtet werden. Sie stellen zusammen mit den FETI-Algorithmen die am besten untersuchten Verfahren dar [101].

Im Folgenden wird auf die Verfahren eingegangen, die (3.7) mittels Schurkomplement berechnen, das als Approximation des Steklov-Poincaré Operators betrachtet wird [101, 184]. Dazu werden die Lösungsvariablen aus (3.7) so sortiert, dass Lösungsvariablen der Kontaktstellen der Gebiete als $\hat{\boldsymbol{u}}$ und Lösungsvariablen innerhalb eines Gebiets als $\boldsymbol{u}_o$ bezeichnet werden. Es folgt

$$\begin{bmatrix} \boldsymbol{K}_{oo} & \boldsymbol{K}_{or} \\ \boldsymbol{K}_{ro} & \boldsymbol{K}_{rr} \end{bmatrix} \begin{bmatrix} \boldsymbol{u}_o \\ \hat{\boldsymbol{u}} \end{bmatrix} = \begin{bmatrix} \boldsymbol{f}_o \\ \hat{\boldsymbol{f}} \end{bmatrix}. \tag{4.18}$$

Der Darstellung folgend beschreibt die Teilmatrix $\boldsymbol{K}_{oo}$ die Zusammenhänge innerhalb eines Gebiets, während $\boldsymbol{K}_{or}$ und $\boldsymbol{K}_{ro}$ die Einflüsse vom Rand auf das Gebiet bzw. umgekehrt abbilden. $\boldsymbol{K}_{rr}$ stellt den Einfluss der Gebietsränder aufeinander dar. Bei einer Berechnung des Schurkomplements und unter Berücksichtigung von (4.15) gilt

$$\begin{bmatrix} \boldsymbol{K}_{oo} & \boldsymbol{K}_{or} & \boldsymbol{0} \\ \boldsymbol{K}_{ro} & \boldsymbol{K}_{rr} & \boldsymbol{I} \\ \boldsymbol{0} & \boldsymbol{I} & \boldsymbol{0} \end{bmatrix} \begin{bmatrix} \boldsymbol{v}_{or} \\ \boldsymbol{v} \\ -\hat{\boldsymbol{S}}\boldsymbol{v} \end{bmatrix} = \begin{bmatrix} \boldsymbol{0} \\ \boldsymbol{0} \\ \boldsymbol{v} \end{bmatrix}. \tag{4.19}$$

Der Vektor $\boldsymbol{v}$ wird beim iterativen Lösen verwendet, sodass $\hat{\boldsymbol{S}}\boldsymbol{v} = \left(\boldsymbol{K}_{rr} - \boldsymbol{K}_{ro}\boldsymbol{K}_{oo}^{-1}\boldsymbol{K}_{or}\right)\boldsymbol{v} = \boldsymbol{K}_{rr}\boldsymbol{v} - \boldsymbol{K}_{ro}\boldsymbol{v}_{or}$ mit $\boldsymbol{K}_{oo}\boldsymbol{v}_{or} = -\boldsymbol{K}_{ro}\boldsymbol{v}$ nicht explizit berechnet werden muss [103]. Zu beachten ist, dass das Gleichungssystem mit $\boldsymbol{K}_{oo}$ sehr genau gelöst werden muss [100]. Das entstehende globale Schurkomplement $\hat{\boldsymbol{S}}$ lässt sich mittels geeigneter Zerlegung gebietsweise für Gebiet i als $\hat{\boldsymbol{S}}^i$ formulieren. Allerdings entstehen in Teilgebieten ohne Dirichlet-Randbedingungen nicht invertierbare Schurkomplemente $\hat{\boldsymbol{S}}^i$ (floating domains). Auch ursprüngliche (primal) Neumann-Neumann und duale FETI-Methoden ermöglichen keine Berechnung von Teilgebieten ohne Dirichlet-Randbedingungen.

Bei einer Interpretation der Teilsimulationen als eigenständige Simulationen und deren anschließender Kopplung sind Teilsimulationen ohne Dirichlet-Randbedingungen ebenfalls nicht lösbar. Bei korrekter Modellierung erscheinen die Kopplungsverfahren im Kontext der Agentenbasierten Lösung zur numerischen Simulationen vielversprechend. Allerdings bedeutet eine wachsende Anzahl an Teilgebieten eine steigende Zahl an Iterationsschritten zum Austausch

der benachbarten Teilgebiete und somit eine längere Rechenzeit. Eine Reduktion benötigter Iterationen ist durch Koordination des Lösungsverlaufs beispielsweise mittels Färbeverfahren möglich. Färbeverfahren sind ein bewährtes Mittel der Graphentheorie. Dabei werden alle Knoten eines Graphen mit möglichst wenig Farben eingefärbt, wobei sichergestellt wird, dass alle mit Kanten verbundene Knoten stets verschiedene Farben aufweisen. Bezogen auf die Gebietszerlegung wird so erreicht, dass ausschließlich Teilgebiete gleichzeitig berechnet werden, die keine gemeinsamen Ränder aufweisen. Die Umsetzung eines zentral organisierten Färbeansatzes ist jedoch leistungsfähiger als die eines dezentralen agentenbasierten Ansatzes. Lediglich für kleine Änderungen innerhalb eines bereits gefärbten Graphen ist die Leistung vergleichbar mit der eines zentralen Ansatzes [186]. Zudem stellt der zentral koordinierte Lösungsablauf einen Gegensatz zu asynchronen Verfahren dar, die einen Prozessorleerlauf durch eine Reduktion der Synchronisationspunkte innerhalb eines Algorithmus zu erreichen versuchen [187].

Gebietszerlegungsverfahren wie das FETI-DP und das BDDC ermöglichen eine Verbesserung der Konvergenzgeschwindigkeit und die Betrachtung von Teilsimulationen auch ohne Dirichlet-Randbedingungen [103, 188]. Dabei werden die inneren Lösungsvariablen parallelisiert eliminiert und die Teilsimulation in Abhängigkeit der gemeinsamen Ränder formuliert. Zur Berechnung der Fragestellung gemeinsamer Ränder wird ein zusätzlicher zentraler BA benötigt. Dieser kommuniziert seine Lösung an alle beteiligten BA und ermöglicht so erst eine Ergebnisbereitstellung der Teilsimulationen. Dabei wird jedoch die Agenteneigenschaft der Autonomie der BA stark eingeschränkt. BA erledigen dann lediglich von einer Zentralinstanz zugewiesene Berechnungen. Zudem ist eine erfolgreiche Berechnung aller Teilsimulationen eine Voraussetzung für einen Rechenfortschritt. Eine entsprechende mit Agenten parallelisierte Umsetzung einer Gebietszerlegung findet sich in [142]. Da in dieser Arbeit das Verbinden dezentraler, autonomer Experteneinheiten im Vordergrund steht, ist ein automatisiertes Verbinden von Teilmodellen zu einem Gesamtmodell ein Anwendungsfall der Gebietskopplung. Ein nachträgliches Aufteilen einer Simulation auf disziplin- und methodengleich befähigte BA scheint nur für sehr große Probleme sinnvoll, zu deren Berechnung die Rechenleistung nicht ausreicht. Unter Berücksichtigung stetig leistungsfähigerer Rechner und moderner Verwaltungswerkzeuge für Rechencluster mit einer auf die Leistungsfähigkeit der Rechenknoten bezogenen Lastverteilung ist dann eine Betrachtung der Gebietszerlegung im Rahmen der Höchstleistungsrechnung sinnvoll.

Neben der beschriebenen Gebietskopplung finden sich in der Literatur auch Beispiele von an Gebietsgrenzen gekoppelten Methoden und Disziplinen [72, 189]. Die konkurrierende Kopplung verschiedener Methoden wird für das beschriebene Agentensystem in [110, 111] betrachtet. Ergänzend wird erwartet, dass eine simulationsspezifische Anpassung der zur Kopplung eingeführten Hierarchie entsprechend Abbildung 4.3 in Form von Koordination vorteilhaft in Bezug auf den Lösungsverlauf ist.

4.2.6 Umgang mit unterschiedlicher Diskretisierung

Die Autonomie ermöglicht es BA, eine für die jeweilige Teilsimulation angepasste Diskretisierung zu verwenden. Diese wird entweder automatisch durch ein dafür umgesetztes Verhalten im BA oder manuell durch den Anwender unter Zuhilfenahme des dem BA zur Verfügung stehenden Simulationswerkzeugs erstellt. Dass verschiedene Diskretisierungen vorteilhaft sind, lässt sich anhand von Fehlerschranken, die diskretisierungsabhängige Aussagen über die Qualität der berechneten Lösung ermöglichen, verdeutlichen [94]. Als Beispiel sei die Kopplung zweier Disziplinen betrachtet, bei der das Material in der einen Disziplin lineare und in der zweiten nicht lineare Eigenschaften aufweist. Bei gleicher Diskretisierung wird der Fehler im linearen Fall geringer ausfallen als im nicht linearen Fall. Entsprechend sind zur Diskretisierung disziplin-, methoden-, und gebietsabhängige Parameter zu berücksichtigen. Deren Bestimmung ist direkt durch den BA oder durch Kommunikation möglich.

Der BA wird dabei als Experte seiner Teilsimulation betrachtet. Zur Kopplung mit anderen BA werden Werte explizit und auf Grundlage des globalen Koordinatensystems angefragt und ausgetauscht. Ein Austausch von Wissen zur Weiterverarbeitung von Werten, beispielsweise über die Materialeigenschaften und Materialgrenzen, wird somit unnötig. Für eine Simulation mit mehreren BA bleibt so die Expertise jedes einzelnen gewahrt. Allerdings liegen die zur Kopplung angefragten Werte nur in Ausnahmefällen als Lösung auf der Diskretisierung des anderen BA vor. Augenscheinlich wird dies am Beispiel einer Anfrage nach Werten auf dem Rand $\Gamma_{1,2}$ zur Gebietskopplung bei nicht übereinstimmender Diskretisierung. Dabei kommt es regional zu einem Überlappen bzw. zu Lücken in der Diskretisierung. Abbildung 4.21 veranschaulicht dies für einen stark vergrößerten Ausschnitt einer Diskretisierung.

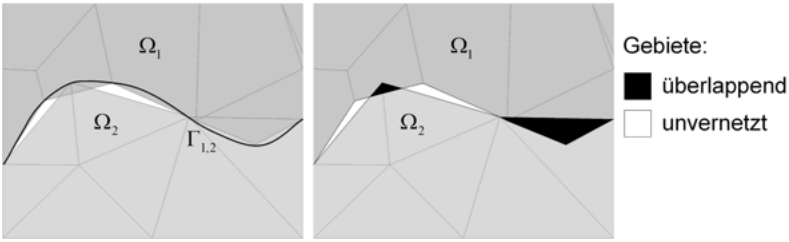

Abbildung 4.21: Darstellung der zu koppelnden Diskretisierung (links) mit einer regionalen Auswertung der Geometrie (rechts) angelehnt an [190]

Entsprechend gilt es, angefragte Werte mittels Nächster-Nachbar-, Interpolations- oder Projektionsverfahren gebietsabhängig zu bestimmen [95]. Die Umsetzung mindestens eines dieser Verfahren ist so in jedem BA notwendig. Hierfür nutzbar ist beispielsweise die in Kapitel 4.2.5.1 beschriebene Schnittstelle. Diese ist dabei um eine Behandlung von Werteanfragen außerhalb

der Diskretisierung gebietskonform zu erweitern. Der aufgrund der Interpolation bzw. Projektion entstehende Fehler ist dabei für alle außer den Nächster-Nachbar-Verfahren kleiner als der Diskretisierungsfehler [190]. Eine ähnliche Größe der Elemente der gekoppelten Teilsimulationen ist dabei jedoch zu bevorzugen [95].

4.3 Ressourcennutzung

Im Folgenden werden die Eigenschaften des vorgestellten dezentralen Agentensystems zum verteilten Lösen gekoppelter Feldprobleme im Hinblick auf die Nutzung der Ressourcen dargestellt. Dabei wird zu Beginn auf die Nutzung der Rechenzeit eines einzelnen BA eingegangen. Die Betrachtung eines KA entfällt, da dieser nicht an der Berechnung der Simulationen beteiligt ist. Im Anschluss folgt die Darstellung ausgewählter Aspekte der Ressourcennutzung des Agentensystems. Dabei steht die Flexibilität innerhalb des Agentensystems und die Unterstützung des Berechnungsablaufs durch hinzugefügte redundante Agenten im Vordergrund.

4.3.1 Agenteninterne Rechenzeitnutzung

Aus wirtschaftlichen Gründen ist ein Simulationsingenieur daran interessiert, verfügbare Rechenzeit möglichst effizient zu nutzen. Enden Berechnungen eines BA ohne dass weitere Aufgaben vorliegen, so ermöglicht das zielgerichtete Verhalten eines jeden BA weitere und über die Aufgaben des Simulationsingenieurs hinausgehende zielgerichtete Berechnungen. Beispiele zusätzlicher zielgerichteter Berechnungen sind die Verbesserung der Genauigkeit mittels Verfeinerung der Diskretisierung, die Verbesserung des Lösungsablaufs zur schnelleren Berechnung von Parametervariationen oder Optimierungsaufgaben und eine umfassendere Modellierung zur zusätzlichen Auswertung von interdisziplinären Zusammenhängen. Dabei bedingen sich die genannten Aspekte gegenseitig, sodass eine Aussage über die Notwendigkeit der Ausführung der im BA realisierten Verhalten schwierig ist. So ist das Erreichen der Konvergenz eines Lösungsverfahrens durch eine verfeinerte Diskretisierung simulationsabhängig möglich. Das dabei im Idealfall verwendete schnellste konvergierende Lösungsverfahren setzt zu dessen Bestimmung jedoch eine geeignete Diskretisierung sowie eine vollständige Modellierung voraus. Anhand dieses Konfliktes wird im Folgenden exemplarisch die Einführung einer Priorisierung zum Umgang mit dem lokalen Ressourcenkonflikt dargestellt. Eine Unterstützung durch zusätzliche redundante BA wird in Kapitel 4.3.2 beschrieben.

4.3.1.1 Auswahl des Lösungsverfahrens

Die Auswahl eines geeigneten Lösungsverfahrens zur Berechnung der Simulation stellt ein vielfach analysiertes Themenfeld dar [191, 192]. Eine Herangehensweise ist die Analyse ausgewählter Parameter bei erfolgreich berechneten Simulationen und das Erstellen einer davon abgeleiteten Wissensbasis in Bezug auf die Leistungsfähigkeit verschiedener Lösungsverfahren. Das in der Wissensbasis hinterlegte Wissen dient anschließend zur Auswahl der Numerikbibliothek und dem ausgeführten Verhalten des BA. Beispiele von Simulationsparameter die sich Berücksichtigen lassen sind die Eigenschaften der Jacobimatrix $\tilde{\boldsymbol{J}}$, der Diskretisierung $\mathcal{D}$ und der betrachteten PDE. Auch rechnerbezogene Größen, wie die benötigte Rechendauer, die CPU-Generation oder der verfügbare Arbeitsspeicher lassen sich bei entsprechender Normierung verwenden. Diese Informationen werden in generalisierter Form in der Wissensbasis abgelegt und beispielsweise zum Training eines neuronalen Netzes genutzt [123, 193]. Dabei wird eine Zielfunktion, wie beispielsweise die erwartete Rechenzeit, mittels eines Feed-Forward-Netzes approximiert. Die approximierte Funktion ermöglicht es, ausgehend von einer Analyse der Parameter einer bisher ungelösten Simulation, Empfehlungen für das Lösungsverfahren abzugeben. Der Aufbau eines solchen Empfehlungssystems ist in Abbildung 4.22 abgebildet.

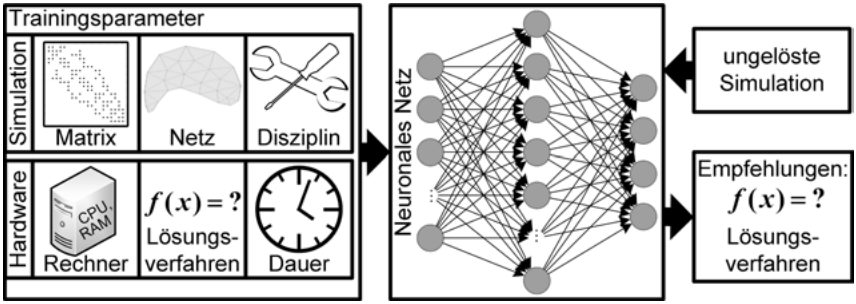

Abbildung 4.22: Neuronales Netz zur Empfehlung der Lösungsverfahren

Abbildung 4.23 zeigt beispielhaft die Auftrittshäufigkeit der Abweichungen zwischen den Empfehlungen eines neuronalen Netzes und der tatsächlichen, aus gemessener Rechenzeit bestimmten relativen Position von 114 Konfigurationen an Lösungsverfahren. Eine positive Position der Wertung entspricht dabei einer vom Empfehlungssystem um eine entsprechende Anzahl an Positionen bessere Empfehlung eines Lösungsverfahrens, als diese durch Messungen bestätigt wurde. Berücksichtigt sind dabei vier direkte und fünf iterative Lösungsverfahren mit je 14 Vorkonditionierern. Besteht innerhalb der Lösungsverfahren kein Unterschied, ob der Vorkonditionierer links oder rechts angewendet wird, so werden beide Optionen als eine Konfiguration betrachtet. Analog dargestellt sind in Abbildung 4.23 die Auftrittshäufigkeit der Abweichungen für die 10 schnellsten Verfahren. Dabei ersetzt die auf Statistik beruhende

Empfehlung des neuronalen Netzes im Einzelfall der jeweiligen Simulation nicht die Entscheidung für ein Lösungsverfahren. Die Nutzung des Expertenwissens gestattet lediglich die Berechnung mit vielversprechenden Lösungsverfahren zu beginnen. Entsprechend dient die Vorgehensweise der Unterstützung des Anwenders zur schnellen Berechnung der Simulationen durch die automatisierte Empfehlung geeigneter Lösungsverfahren.

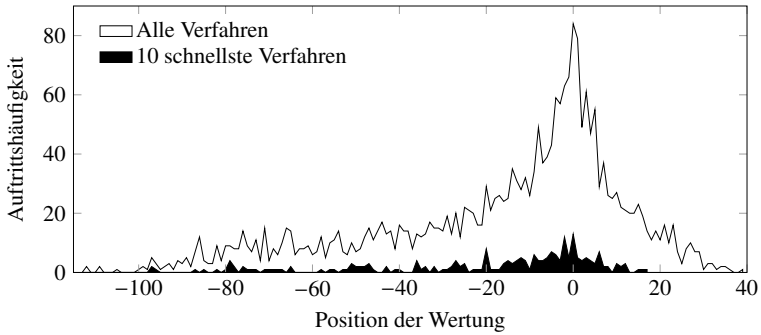

Abbildung 4.23: Fehlerhistogramm eines neuronalen Netzes [193]

Die Qualität der Empfehlungen hängt dabei von den für den Aufbau einer Wissensbasis verwendeten Simulationen und den extrahierten Parametern ab. Zusätzliche von einem BA durchgeführte Berechnungen lassen sich zur Verbesserung seiner Wissensbasis nutzen. Jedoch gilt es dazu möglichst vollständige Parametersätze in Bezug auf Lösungsverfahren und Rechenzeit zu erstellen [194]. Das Erzeugen dieser Parametersätze repräsentiert eines der zielgerichteten und über die Anforderungen an eine einmalige Simulation hinausgehenden Verhalten eines BA.

4.3.1.2 Verfeinerung der Diskretisierung

Die Genauigkeit der berechneten Lösung ist abhängig von der charakteristischen Elementgröße h und dem Grad der Formfunktionen N_i eines finiten Elements [142]. Die Elementgröße h wird dabei als Durchmesser einer Kugel, die das finite Element beinhaltet oder entsprechend der größten Kantenlänge des finiten Elements gewählt [195]. In Bezug auf interdisziplinär gekoppelte Simulationen sind die durch Fehlerschranken bestimmten Gebiete zur Verfeinerung zudem disziplinspezifisch [94]. Abbildung 4.24 zeigt beispielhaft zu verfeinernde Gebiete der Diskretisierung $\mathcal{D}$ eines Transistors für die drei gekoppelt berechneten Größen der Konzentration der Elektronen n, der Löcher p und des elektrischen Potenzials φ.

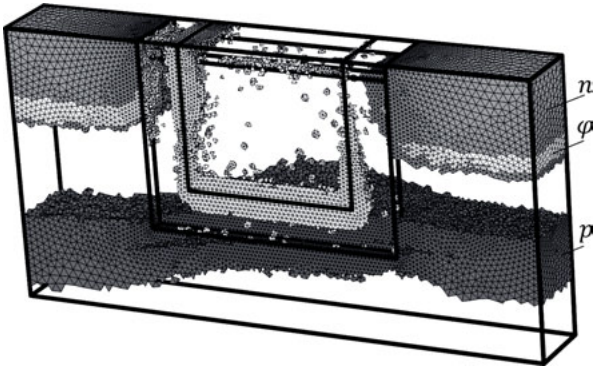

Abbildung 4.24: Finite Elemente mit geringer Genauigkeit je Lösungsvariablen für einen Transistor entsprechend Kapitel 5.2.2

Die gestaffelte Berechnung innerhalb des Agentensystems und die darin umgesetzte disziplinspezifische Diskretisierung ermöglichen den BA so auch derartige disziplinspezifische Verfeinerungen. Ein Vergleich der agentenspezifischen mit einer agentenübergreifenden Verfeinerung der Diskretisierung zeigt dabei einen reduzierten Rechenaufwand der agentenspezifischen Netzverfeinerung. Allerdings macht diese ein erneutes Austauschen der verfeinerten Diskretisierung zwischen den BA zur Kopplung notwendig. Eine zusätzliche Reduktion der Rechenzeit ist im Fall der Berechnung mittels iterativer Lösungsverfahren automatisiert möglich, indem Interpolationsverfahren genutzt werden, um eine Approximation des Startvektors aus einer unverfeinerten Berechnung zu bestimmen. Die Anpassung der Diskretisierung, die dazu verwendete Methode sowie das eingesetzte Lösungsverfahren stellen disziplin-, methoden- und gebietsspezifisches Expertenwissen dar. Dessen automatisierte Berücksichtigung erfolgt durch die BA anhand der in [142] dargestellen und innerhalb der BA verfügbaren Netzverfeinerungsverfahren. Ein Verfeinerungsschritt lässt sich während einer Berechnung auch mehrfach ausführen, um die Diskretisierung $\mathcal{D}$ dort zu verfeinern, wo große Fehler auftreten. Dabei ist eine Verfeinerung mit einer Steigerung an Freiheitsgraden verbunden. Dies wiederum führt zu einer Steigerung der Rechenzeit und so zu einem zweiten, über die einmalige Berechnung der Simulation hinausgehenden Verhalten eines BA. Entsprechend ist eine weitere Fähigkeit der Softwareagenten-basierten Simulation, dass die Anzahl an Verfeinerungsschritten abhängig von der verfügbaren Rechenzeit je BA variiert werden kann.

4.3.1.3 Rechenzeitverteilung

Ungenutzte BA stellen ihre Fähigkeiten verfügbaren KA zur Verfügung und bemühen sich um Aufgaben aus ihrer Umgebung. Neben einem KA als Auftraggeber ermöglicht die graphische

Benutzeroberfläche einem Anwender, Simulationen mittels des Simulationswerkzeugs zu bearbeiten. Eine Priorisierung der beiden Auftraggeber findet hier zugunsten des Anwenders statt, bei dem von einer bewussten Ressourcennutzung ausgegangen wird. Die Ressourcen des BA werden nun innerhalb der funktionellen Gruppen des Agentensystems als belegt betrachtet. Eine weitere Priorisierung ist zwischen der Berechnung einer Simulation mit verschiedenen Lösungsverfahren und der Berechnung verschiedener Diskretisierungen notwendig. Grund ist, dass beide Verhalten alle einem BA zur Verfügung stehenden Ressourcen ausnutzen. Da die Ergebnisse beider Verhalten von Interesse sind und keine Aussage über die Verfügbarkeit der Rechenressourcen des BA erfolgen, ist eine Planung der Arbeitsabläufe (Scheduling) nicht möglich. Die Beauftragung des entsprechenden Verhaltens hat so jeweils nach dem Ende des vorhergehenden Verhaltens zu erfolgen. Dies wird durch einen Vergleich der Ausführungshäufigkeiten der jeweiligen Verhalten mit einer vom Anwender vorgegebenen Häufigkeitsverteilung γ möglich [196]. Da bei der ersten Berechnung sowohl eine Diskretisierung als auch das dazu gehörige Lösungsverfahren zum ersten Mal berechnet werden, sind Verteilungen von 100 % nicht zu realisieren. Die Priorisierung der Verhalten ermöglicht es einem Anwender so, bei ausreichender Rechenzeit und ohne entsprechende Vorkenntnisse, genau und schnell Feldprobleme zu berechnen. Es vermeidet zudem ungenutzte Rechenressourcen innerhalb des Agentensystems.

4.3.1.4 Ende der Agententätigkeit

Auch bei einer Betrachtung bisheriger verteilter Systeme zur Berechnung von Simulationen endet mit der Konvergenz der Simulation auch die Aktivität daran beteiligter Einheiten. Zur Feststellung der Konvergenz existieren sowohl zentrale als auch dezentrale Ansätze. Letztere finden beispielsweise in Rechenclustern Anwendung und werden mittels Umlaufverfahren realisiert [197]. Dabei unterstützt ein dezentraler Ansatz die Autonomie der BA. Auch eine zentrale Feststellung der Konvergenz ist, bedingt durch die hier realisierte Struktur des Agentensystems und die Möglichkeit zu dessen Koordination, beispielsweise mittels einer zentral geführten Liste, umsetzbar. In diesem Agentensystem liegt Konvergenz vor, falls wenigstens einer der konkurrierenden und alle sich ergänzenden BA ihre Berechnung beendet haben und kein Bedarf an einer Veröffentlichung der Lösung mehr besteht. Dessen Überwachung obliegt hier dem Anwender. Von einem Berechnungsende ist die Rede, wenn für alle Teilsimulationen wenigstens eine Lösung gefunden wurde.

Bis zum Erreichen der Konvergenz hat wenigstens ein BA mit den für die Simulation notwendigen Fähigkeiten verfügbar zu sein. Anderenfalls ist kein Auswerten der Lösungen oder Bereitstellen von Koppelgröße mehr möglich. Verfügbar meint, dass im Bedarfsfall ein BA seine Tätigkeit an der entsprechenden Simulation wieder aufnehmen kann. Fehlen dem Agentensystem Fähigkeiten, so bleiben entsprechende Teilsimulationen unberücksichtigt. Unter Berücksichtigung der Nutzbarkeit der Rechenzeit auch über die Konvergenz hinaus, ergibt sich

eine sehr lange Auslastung der BA. Die Priorisierung der Verhalten und die Interaktion mit dem Anwender ermöglichen eine sinnvolle Rechenzeitverteilung innerhalb eines BA. Zusätzlich lassen sich, angelehnt an die prinzipiell sehr lange rechnenden iterativen Lösungsverfahren, Grenzen definieren, bei deren Erreichen die Bearbeitung einer Simulation beendet wird. Die im Folgenden berücksichtigte Häufigkeitsverteilung γ gestattet dabei lediglich die Berechnung einer Diskretisierung je BA.

4.3.2 Rechenzeit des Agentensystems

Bei der verteilten Rechnung gilt es zwischen zentral ausgeführten Algorithmen und dezentralen Algorithmen zu unterscheiden. Abhängig von der jeweiligen Implementierung sind beide in der Lage, verteilte Ressourcen zur Ausführung zu nutzen. Beispiele für zentral ausgeführte Algorithmen mit verteilter Ressourcennutzung sind direkte Lösungsverfahren oder Gebietszerlegungsalgorithmen. Diese berechnen unter der Verwendung von Bibliotheken wie OpenMP, MPI oder GPI [198] parallelisiert Ergebnisse und stellen die Ergebnisse im Anschluss zusammengeführt bereit. Je geringer die Synchronisation zwischen den parallel ablaufenden Prozessen ist, desto besser die Skalierbarkeit der Verfahren. Die dazu benötigte Funktionalität, wie beispielsweise das Verwalten und Überwachen beteiligter Ressourcen, die dynamische Lastverteilung oder die Rekonfigurierbarkeit, sind leistungsstark zentral umgesetzt. Der Leistungsvergleich zwischen einem zentralen und einem verteilten Algorithmus fällt entwurfsbedingt zugunsten des zentralen Algorithmus aus. Beispielhaft wird dies an der Datenhaltung sichtbar, die im einen Fall zentral und im anderen Fall verteilt und gekapselt stattfindet. Werden die in Kapitel 4.1 beschriebenen Eigenschaften einer verteilten Architektur als wesentlich eingeschätzt, so gilt es den Algorithmus verteilt zu realisieren.

Die durch das beschriebene Agentensystem verteilt berechnete Simulation stellt ein Beispiel eines dezentralen Algorithmus dar. Je ausgeprägter die Individualität der Agenten und deren Verhalten ist, desto geringer wird die Ähnlichkeit mit einem zentralen Algorithmus. Entsprechend ist dieses Agentensystem grundsätzlich vom parallelen verteilten Rechnen zu unterscheiden. Eigenschaften des Agentensystems lassen sich dabei angelehnt an agentenbasierte Rechencluster (ACG) betrachten. Durch angebotene Dienstleistungen und deren Durchführung ermöglichen BA darin eine Abstraktion sowie eine dynamische und robuste Anpassung des Systems an Gegebenheiten innerhalb eines verteilten Systems [199]. Exemplarisch sei die Agenteneigenschaft der Mobilität betrachtet, die durch Funktionen wie das Klonen von Agenten und dessen Migration die Fehlertoleranz in einem Rechencluster erhöhen [200]. Auf deren Umsetzung wurde hier aufgrund der lokal bereitstehenden Simulationswerkzeuge und den großen bei einer Simulation anfallenden Datenmengen bei einer erneuten Initialisierung der Berechnung bewusst verzichtet. Alle anderen Eigenschaften der starken sowie der schwachen Beschreibung eines Agenten aus Kapitel 2.2 sind hingegen in diesem Agentensystem realisiert. Global betrachtet bietet das

Agentensystem so Zugriff auf alle Dienstleistungen der am Agentensystem beteiligten Agenten. Entsprechend ist das Agentensystem als Kollektiv von Agenten zu betrachten, innerhalb dessen Dienstleistungen und Ressourcen unter seinen Mitgliedern bereitgestellt werden und das eine einfache Erweiterung und Verwaltung bestehender Infrastruktur ermöglicht [201]. Ein Beispiel stellt die in [202] beschriebene Vermittlung von Diensten zur Verwaltung und Planung verteilter Ressourcen dar. Eine auf numerische Simulationen bezogene Realisierung ist der so erzielte Lastausgleich bei der Gebietszerlegung [143].

Für einfache Aufgaben mit kurzer Ausführungsdauer ist deren Verteilung auf mehrere autonom agierende Agenten wenig sinnvoll. Zusätzliche redundante Agenten, die automatisiert konkurrieren und unterstützende Informationen zur Lösung beitragen, nützen hingegen der Lösung. Dies ist in der dargestellten Agentenplattform zur Berechnung einzelner Teilsimulationen umgesetzt. Dazu werden BA mit gleichen Fähigkeiten innerhalb des Agentensystems als funktionelle Gruppe betrachtet. Innerhalb dieser Gruppen konkurrieren Agenten darum, möglichst günstig eine Lösung bereitzustellen. Die zur Beurteilung notwendige Kostenfunktion lässt sich angelehnt an die Rechenzeit, die benötigten Ressourcen und die Betriebskosten aufstellen. Agenten, deren Lösungsbeitrag aufgrund von längerer Rechenzeit oder größerem Restfehler nicht mehr relevant ist, beenden ihre Berechnung. Dieses Vorgehen findet angelehnt an evolutionäre Algorithmen [203] statt, wobei ein erneuter Lösungsversuch eines Agenten mit modifizierten Parametern möglich ist. Dies findet beispielsweise nach dem Scheitern eines Lösungsverfahrens statt, wenn eine weitere Berechnung einen Mehrwert bietet. Ermöglicht wird so ein verteiltes Bearbeiten der in Kapitel 4.3.1 beschriebenen Aufgaben unter Berücksichtigung des im Agentensystem vorhandenen Wissens und bereitgestellter Fähigkeiten. Dabei wird durch Kommunikation sichergestellt, dass Konfigurationen mit identischen Parametern (siehe Abbildung 4.3) nicht mehrfach berechnet werden. Für verteilt berechnete interdisziplinäre Simulationen ermöglicht dies beispielsweise, dass Agenten, die zur Berechnung aller Disziplinen in der Lage sind, mit verteilten Systemen konkurrieren. Zudem werden im System verfügbare, bereits berechnete Informationen zur weiteren Berechnung genutzt.

Die dynamische Erweiterung des Agentensystems während der Laufzeit gestattet es, dass Teilsimulationen bereits berechnet werden, ohne das alle notwendigen Fähigkeiten im Agentensystem vorhanden sind. Auch Teilmodelle, die während der Entwicklung einer gekoppelten Simulation entstehen, lassen sich durch das Agentensystem auswerten. Vorausgesetzt wird, dass eine Lösung für die Teilsimulation existiert. Kommen während der Berechnung neue Rechenressourcen in Form von BA hinzu, so nutzen diese das beim KA hinterlegte Simulationsmodell, um anhand ihrer Kompetenzen einen Beitrag zur Simulation zu leisten. Dabei informieren sich BA vor Berechnungsbeginn automatisiert über für sie relevante und bereits existierende Ergebnisse und berücksichtigen diese bei der im Anschluss durchgeführten Rechnung (siehe Abbildung 4.19). Dieses Vorgehen stellt ein Beispiel für die Agenteneigenschaften des initiativen und zielgerichteten Handelns eines BA dar. Entstehen durch hinzukommende BA weitere

Möglichkeiten zur Kopplung, deren Berechnung aufgrund fehlerhafter Modellierung jedoch fehlschlägt, so wird eine manuelle Modellierung notwendig. Entsprechende Hinweise lassen sich aus dem Kopplungsgraphen aus Abbildung 4.14 entnehmen.

Entfällt ein BA, beispielsweise aufgrund anderweitig ihm übertragener Aufgaben oder durch einen Hardwaredefekt, so bleiben die zuletzt verteilten Ergebnisse jedoch im Agentensystem präsent. Dessen Funktion innerhalb des Agentensystems wird dabei durch die konkurrierenden BA innerhalb der funktionellen Gruppe übernommen. Entfällt bei einer Simulation der letzte befähigte BA, so ist eine Lösung nur noch in reduzierter Form möglich. Dabei werden ausgetauschte, vorhandene Kopplungswerte weiterverwendet, jedoch nicht weiter präzisiert und aktualisiert. Auch hinzukommende BA werden nun nicht mehr über bis dahin berechnete Ergebnisse informiert. Abhilfe schafft die Erweiterung des Agentensystems um Agenten, die berechnete Lösungen der BA sammeln und daraus zur Kopplung benötigte Werte bestimmen und bereitstellen. Da die Bereitstellung von zur Kopplung benötigter Werte meist jedoch eine Auswertung disziplinspezifischer Zusammenhängen erfordert, würden so Agenten mit großer funktioneller Ähnlichkeit zu den jeweiligen BA notwendig. Entsprechend gilt es hier nach dem Ausfall des letzten befähigten BA, wenn das Erreichen der Konvergenz nicht mehr möglich ist, den Beitrag des ausgefallenen Agenten innerhalb der Teilsimulationen zu verwerfen.

5 Numerische Beispiele

Anhand zweier Problemstellungen wird im Folgenden demonstriert, dass der Einsatz von Softwareagenten zur numerischen Feldberechnung interdisziplinärer gekoppelter Fragestellungen vorteilhaft ist. In die numerische Feldberechnung übertragene Eigenschaften der agentenorientierten Programmierung und deren Einfluss auf die Simulation werden dabei betont.

5.1 Nahbereichsradar

Eine erste Demonstration erfolgt anhand von Untersuchungen zum Berechnungsverlauf eines 24,15 GHz Nahbereichsradars, das beispielsweise zur Objekterkennung im Auto eingesetzt wird. Dazu wird ein Antennenarray, bestehend aus mehreren Patch-Antennen und deren Zuleitungen auf einer Leiterplatine, betrachtet. Das Antennenarray befindet sich in einem Kunststoffgehäuse mit eingelassener Linse. Auf dem Gehäuse und der Linse befindet sich eine Schicht Wasser. Das Antennenarray sowie das seitlich durch den Modellrand begrenzte Gehäuse sind von Luft umgeben. Abbildung 5.1 stellt das Antennenarray und einen Schnitt durch das Gehäuse dar.

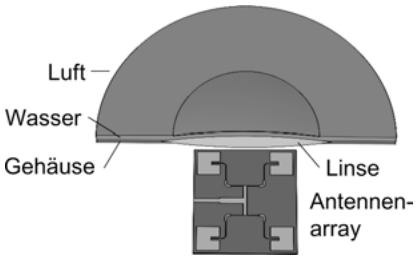

Abbildung 5.1: Modell des Nahbereichsradars

5.1.1 Konkurrierende Wellensimulationen

Die Auslegungen der im Antennenarray eingesetzten Patch-Antennen, deren Zuleitungen sowie die Positionierung der Linse erfolgen unter Auswertung der Helmholtz-Gleichung

$$\Delta \boldsymbol{E} - \mu_r k_0^2 \left(\varepsilon_r - \frac{\mathrm{j}\kappa}{\omega \varepsilon_0} \right) \boldsymbol{E} = 0 \tag{5.1}$$

für die elektrische Feldstärke $\boldsymbol{E}$ mittels der FEM. An den Rändern des über das Modell hinaus reichenden Luftgebiets wird dieses als frei von Reflexionen angenommen. Die Materialparameter sind gebietsweise konstant. Diskretisiert ist das Modell mit $2 \cdot 10^6$ Tetraedern. Dem Agentensystem wird die Modellbeschreibung einschließlich der Diskretisierung zur Berechnung übergeben. Dabei ermöglicht die Kommunikationsfähigkeit der Agenten die Interaktion mit dem Anwender sowie ein zentrales Bereitstellen der Simulation für verteilt betriebene BA. Die Eigenschaft des rationalen Verhaltens der Agenten bewirkt, dass BA das bereitgestellte Simulationsmodell auch berechnen.

Das sich bei festgelegter Diskretisierung ergebende Gleichungssystem hat $14 \cdot 10^6$ Freiheitsgrade, ist komplex, symmetrisch, hat eine relative Bandbreite von 7 %, eine Besetzungsdichte von $3 \cdot 10^{-6}$ und benötigt zu dessen Speicherung in dünn besetzter Form 14 GB. Diese Eigenschaften sowie Hardwareeigenschaften der BA dienen zur Empfehlung möglicher Lösungsverfahren durch ein neuronales Netz. Die so erhaltenen Empfehlungen werden entsprechend der Reihenfolge der Platzierung von den verfügbaren BA angewendet. Anwenderspezifische Vorgaben werden dabei bevorzugt berücksichtigt und Empfehlungen für skalare Vorkonditionierer verworfen.

Für dieses Beispiel nutzbare BA befinden sich innerhalb der funktionellen Gruppe zur Berechnung elektromagnetischer Wellen. Das eingesetzte Rechnersystem besteht, angelehnt an ein Büronetz aus 6 ähnlichen Rechnern, wobei diese aufgrund der Größe der Simulation leistungsstark ausgeführt sind (siehe Tabelle 5.1). Individualität der BA entsteht durch unterschiedliche Hardware, die bei der Auswahl der Lösungsverfahren berücksichtigt wird. Entsprechend Abbildung 4.3 konkurrieren BA hier auf Parameterebene darum einen Beitrag zur Lösung zu leisten. Dies entspricht der Agenteneigenschaft des zielgerichteten Verhaltens. Tabelle 5.1 fasst den Aufbau des Agentensystems, gewählte Lösungsverfahren, deren Platzierung durch das neuronale Netz und die benötigten Rechenressourcen zusammen.

Tabelle 5.1: Agentensystem zur Antennensimulation

	BA 1	BA 2	BA 3	BA 4	BA 5	BA 6
Platz	1	2	3	4	5	Anwender
Lösungsverfahren:	PARDISO	BiCGStab, MG(SORV, MUMPS)	MUMPS Out-of-Core	FGMRES, DD(·, 2× MUMPS)	FGMRES, MG(SORV, MUMPS)	FGMRES, MG(SORV, FGMRES)
Rechenkerne:	4	8	8	8	8	4
CPU:	Xeon-E5-2643	Xeon-E5-2630v3	Xeon-E5-2630v3	Xeon-E5-2630v3	Xeon-E5-2630v3	Xeon-E3-1275v5
Rechnenzeit:	7 h	21 min	5 h	2 h	19 min	-
Speicherbelegung:	364 GB	62 GB	108 GB	221 GB	62 GB	-

Die zudem dargestellten Rechenzeiten und der jeweilige Speicherbedarf verdeutlichen, dass Softwareagenten in der Lage sind eigenständig Simulation zu berechnen. PARDISO und MUMPS sind dabei direkte und BiCGStab und FGMRES iterative Lösungsverfahren. Die multiplikative Schwarz-Gebietszerlegung (DD,·), das vektorielle Überrelaxationsverfahren (SORV) und das Mehrgitterverfahren (MG) sind Vorkonditionierer. Der dabei erreichte Konvergenzverlauf ausgewählter sowie zusätzlicher Lösungsverfahren ist in Abbildung 4.15 abgebildet.

Unter der Annahme, dass erfolgreich berechnete Ergebnisse gleichwertig sind, ist ein Abbrechen von langsameren Lösungsverfahren möglich. Dies trifft auch dann zu, wenn Agenten verspätet hinzukommen, verspätet mit der Berechnung beginnen oder aufgrund von zusätzlichen Aufgaben eine längere Rechenzeit aufweisen. Dabei kompensiert das Agentensystem auch den Ausfall von Agenten. So wird hier für einen parallelen Start aller BA das Ergebnis von BA 5 bereitgestellt, anstelle auf das Ergebnis fehlerhafter oder langsamerer BA zu warten. Im Vergleich mit einer sequenziellen Implementierung bedarf das Agentensystem so keiner Neuinitialisierung der Berechnung. Dieses Beispiel verdeutlicht zudem die Autonomie der BA, da das von BA 6 gewählte nicht konvergierende Lösungsverfahren für diesen zwar keinen Lösungsbeitrag bedeutet, dem Agentensystem jedoch die erfolgreiche Berechnung gelingt.

Findet kein Abbrechen statt, so vergleicht das Agentensystem automatisiert die Leistungsfähigkeit von sowohl parallel als auch sequenziell berechneten Lösungsverfahren. Dieser Vergleich setzt die Agenteneigenschaft der Wahrhaftigkeit in Bezug auf die Angabe der Rechenzeit voraus. Während ein parallelisiertes Berechnen verschiedener Lösungsverfahren für den geübten Anwender wenig sinnvoll ist, profitiert dieser von der Möglichkeit getroffene Entscheidungen bei der Konfiguration des Lösers zu verifizieren. Zudem ermöglichen die Agenten für nicht lineare Lösungsverfahren oder im Fall eines bidirektional gekoppelten Modells verschiedene automatisiert konkurrierende Lösungsverfahren für sequenzielle Berechnungsschritte. Letzteres demonstriert Kapitel 5.1.2.3. Ein ungeübter Anwender profitiert bereits vom parallelen Berechnen, da ihm das Agentensystem eine schnelle Lösungsbereitstellung ermöglicht.

5.1.2 Interdisziplinär gekoppelte Simulationen

Erfahrungswissen eines Anwenders oder das in Kapitel 4.2.2 dargestellte Empfehlungssystem legen neben der Betrachtung der elektrischen Feldstärke $\boldsymbol{E}$ eine Auswertung unter Berücksichtigung der Temperatur T nahe (siehe Abbildung 4.14). Entsprechend wird zusätzlich zu der in Kapitel 5.1.1 beschriebenen Berechnung der elektrischen Feldstärke $\boldsymbol{E}$ nun die stationäre, nichthomogene Wärmeleitungsgleichung (3.15) betrachtet. Diese ist für das hier betrachtete Beispiel innerhalb des Gehäuses von Interesse. Das Gehäuse ermöglicht auf der Oberfläche einen Wärmefluss vom Wasser in das Luftgebiet, hat auf der Unterseite konstant $50°C$ und ist ansonsten isoliert. Dementsprechend entsteht im Agentensystem der Bedarf an zusätzlicher Dis-

ziplinkenntnis. Im Folgenden werden mögliche Szenarien von sich ergebenden Fragestellungen vorgestellt, die einem Konflikt verschieden befähigter, selbstständiger und gleichberechtigter BA auf Disziplinebene entsprechen (siehe Abbildung 4.3).

5.1.2.1 Szenario 1

Das Szenario 1 entspricht analog zu Kapitel 4.2.3 einer ungekoppelten Berechnung von (5.1) und (3.15). Dies ist dann von Interesse, wenn zusätzlich zur elektrischen Feldstärke $\boldsymbol{E}$ der durch Randbedingungen vorgegebene Temperaturverlauf innerhalb des Gehäuses ausgewertet wird. Tabelle 5.2 stellt die Konfiguration des Agentensystems dar, die zur Berechnung dieses Szenarios eingesetzt wird. Die dargestellten Befähigungen der BA beschreiben deren Eigenschaften zur Berechnung der angegebenen Disziplinen durch verfügbare Simulationswerkzeuge entsprechend Abbildung 4.4.

Tabelle 5.2: Agentensystem zur Demonstration von Szenario 1

	Befähigung	Lösungsverfahren	Freiheitsgrade	Rechnenzeit
BA 1	$\boldsymbol{E}$, T	FGMRES, MG(SORV, MUMPS)	$16 \cdot 10^6$	24 min
BA 2	T	FGMRES, MG(SOR, MUMPS)	10^6	1 min
BA 3	$\boldsymbol{E}$	FGMRES, MG(SORV, MUMPS)	$14 \cdot 10^6$	19 min

Die angegebenen Rechenzeiten sind auf drei gleichen Rechnern mit 8 Kernen einer Xeon-E5-2630v3 CPU berechnet. Berücksichtigt ist, dass eine monolithische Berechnung meist mit einheitlicher Diskretisierung aller Teildisziplinen stattfindet. Die reellen Freiheitsgrade der thermischen Simulation kommen so knotengleich zu den Komplexen der Wellenberechnung hinzu. Dies führt bei monolithischer Berechnung zu einem Gleichungssystem mit $16 \cdot 10^6$ Freiheitsgraden. Das gestaffelte Berechnen der disziplinspezifischen Teilsimulationen erfolgt hier unter Verwendung von auf die Disziplin angepassten unterschiedlichen Diskretisierungen. Verglichen mit der einheitlichen Diskretisierung wird so eine thermische Simulation mit lediglich 10^6 Freiheitsgraden möglich. Tabelle 5.2 zeigt zudem, dass die Realisierung von BA mit Kenntnis mehrerer Disziplinen gelingt. Da die Rechenzeit von BA 1 mit 24 min hier länger dauert als das Erreichen der Konvergenz der BA 2 und 3 nach 19 min, ist ein Abbrechen der Berechnung von BA 1 möglich. Entsprechend zeigt die Berechnung der Simulation durch das Agentensystem auch, dass die parallelisierte gestaffelte Berechnung mit auf die Disziplin angepasster Diskretisierung hier schneller ist als die monolithische Berechnung mit einheitlicher Diskretisierung.

5.1.2.2 Szenario 2

Im Folgenden wird Szenario 2 analog zu Kapitel 4.2.3 betrachtet, das der unidirektionalen interdisziplinären Kopplung zweier Disziplinen entspricht. Dies entsteht beispielsweise dann, wenn bei der Berechnung von (3.15) elektrische Verluste in Form einer stationären Wärmequelle $q(\boldsymbol{E})$ berücksichtigt werden. Dabei ist die entsprechend (3.17) aus der Stromwärme bestimmte Wärmequelle $q(\boldsymbol{E})$ und somit auch die unidirektionale Kopplung nicht linear von der elektrischen Feldstärke $\boldsymbol{E}$ abhängig. Deren Berücksichtigung ist in allen Gebieten möglich, jedoch nur in den Gebieten mit einer elektrischen Leitfähigkeit $\kappa \neq 0$ notwendig. Abbildung 5.2 stellt einen Vergleich des Temperaturverlaufs der Modelle mit und ohne Berücksichtigung der Kopplung sowie die Konturen der Geometrie anhand eines Schnitts durch das Modell dar.

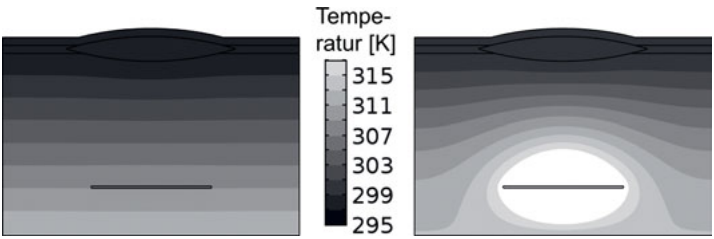

Abbildung 5.2: Temperatur ohne (links) und mit (rechts) Berücksichtigung der Kopplung

Der zur Kopplung notwendige Datenaustausch erfolgt anhand der für die Integration in (3.16) benötigten Werte an den Gaußpunkten. Diese Koppelwerte stellt ein BA aufgrund der Eigenschaften seiner Güte und Rationalität auf Anfrage bereit. Ausgehend von den Teilmodellen aus Szenario 1 erfolgt so der Austausch von $6 \cdot 10^5$ Werten, was bei einer Fließkommadarstellung mit je 8 $\frac{\text{Byte}}{\text{Wert}}$ in Summe 5 MB entspricht. Aufgrund der verschiedenen Diskretisierungen der gekoppelten Teilsimulationen sind einmalig die Koordinaten der zu koppelnden Diskretisierung auszutauschen, was hier 15 MB entspricht. Der serialisierte Austausch dieser Datenmenge stellt bei aktueller Netzwerktechnik einen vernachlässigbaren Zeitaufwand dar. Für größere Datenmengen ist anzumerken, dass eine verlustlose Kompression der Koppelwerte hier keine nennenswerte Speicherreduktion ermöglicht [168]. Da der Ablauf dieser und der im Folgenden dargestellten unidirektionalen Kopplungen aus Agentensicht analog verläuft, wird hier auf eine Rechenzeitauswertung verzichtet.

Eine ebenfalls unidirektionale Kopplung entsteht bei einer Berücksichtigung des Einflusses der Temperatur T auf die Materialwerte in (5.1). Ein Vergleich der elektrischen Leitfähigkeiten κ der verwendeten Materialien verdeutlicht, dass bei der Modellierung Kupfer als perfekter elektrischer Leiter angenommen werden kann. Auch der Temperatureinfluss auf dessen linearisierte elektrische Leitfähigkeit kann im Temperaturbereich von 0 bis 100°C vernachlässigt werden, da

sich diese lediglich von $5{,}8 \cdot 10^7$ auf $4{,}2 \cdot 10^7$ ändert. Die Permittivität ε_r wird in Luft als reelle Größe mit $\varepsilon_r = 1$ und in den anderen Materialien als komplexe Größe betrachtet. Komplexe Permittiviäten werden hier entweder über den von der Temperatur unabhängigen Verlustfaktoren $\tan(\delta)$ oder im Fall von Wasser über das temperatur- und frequenzabhängige Debye Modell berücksichtigt. Abbildung 5.3 stellt die berücksichtigte komplexe Permittivität $\varepsilon_r = \varepsilon' + \mathrm{j}\varepsilon''$ von Wasser nach Real- und Imaginärteil getrennt bei konstanter Temperatur (links) und konstanter Frequenz (rechts) dar [204]. Die Kopplung von (3.15) auf (5.1) ist aufgrund der nicht linearen Zusammenhänge $\varepsilon'(T)$ und $\varepsilon''(T)$ somit ebenfalls nicht linear.

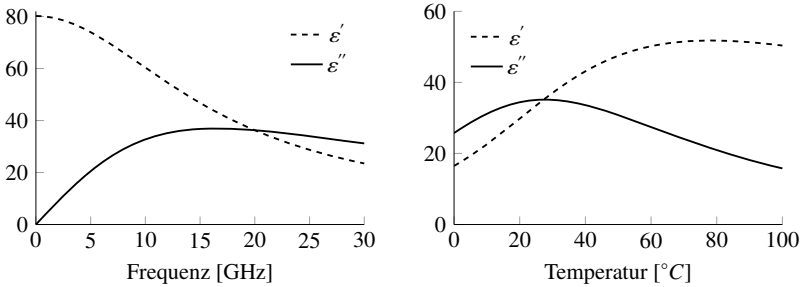

Abbildung 5.3: Komplexe Permittivität von Wasser bei 20°C (links) und 24,15 GHz (rechts)

Abbildung 5.4 stellt den Berechnungsverlauf und die Aktivitäten der BA für diese unidirektional gekoppelte Simulation dar. Dabei entsprechen die eingesetzten BA und deren Konfiguration jenen der vorhergehenden Kapitel, wobei auf den dort langsameren BA 1 verzichtet wird. Dessen Ressourcen werden stattdessen genutzt, um die zwei zum Vergleich dargestellten gestaffelten Lösungsverläufe 3 und 4 zu berechnen. Diese ergeben sich bei manueller Konfiguration möglicher gestaffelter Berechnungsreihenfolgen mittels des vom BA eingesetzten Simulationswerkzeugs und führen zu gleichwertigen Ergebnissen.

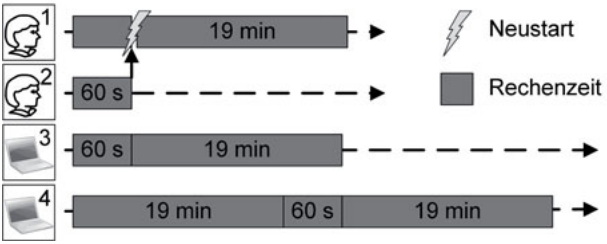

Abbildung 5.4: Rechenzeiten einer unidirektionalen Kopplung

Angelehnt an einen erfahrenen Anwender, der sich aus Rechenzeitgründen zur Konfiguration des Simulationswerkzeugs entsprechend Lösungsweg 3 entscheidet, berechnet das Agentensystem

mit den BA 1 und BA 2 automatisiert die Lösungssequenz mit der kürzeren Berechnungszeit. Lediglich die Zeit zum Abbrechen der aktuellen Berechnung und zu dessen Neustart kommt hinzu. Diese entspricht hier 3 s, ermöglicht jedoch eine Rechenzeitersparnis von 19 min für eine nicht optimale Löserkonfiguration. Die vorteilhafte Nutzung zweier parallel begonnener Berechnungssequenzen entsprechend Tabelle 4.1 wird so offensichtlich. Die von BA 1 berechneten Werte bis zum Zeitpunkt der Ergebnisbereitstellung durch BA 2 haben hier keinen nennenswerten Einfluss auf die Rechenzeiten. Dementsprechend wird hier auf eine Untersuchung einer frühzeitigeren Bereitstellung der Lösungen, mit dann größerem Restfehler als Startwert für BA 1 verzichtet (siehe Szenario 2 in Kapitel 4.2.3). Zusätzlich ist auch hier ein Vergleich von verschiedenen Lösungsverfahren für die Teilsimulationen, angelehnt an Kapitel 5.1.1, möglich.

Die dargestellten Ergebnisse verdeutlichen die Kooperation der Agenten zu einem gemeinschaftlichen Berechnen der Simulation. Dies gelingt trotz des Mangels eines BA mit Fähigkeiten zur Berechnung beider Disziplinen. Während der Berechnung der gekoppelten Fragestellung wird von der notwendigen Wahrhaftigkeit und Güte der Agenten ausgegangen. Das Beispiel der unidirektionalen Kopplung demonstriert auch die umgesetzte interaktive Schnittstelle zur Kopplung verschiedener gekapselter Simulationswerkzeuge, die mittels BA gelingt. Zudem bewahrt die realisierte Kapselung die Expertise einzelner BA. Da eine derartige Schnittstelle in der numerischen Feldberechnung selten ist, wird auf deren Umsetzung hier explizit hingewiesen.

5.1.2.3 Szenario 3

In diesem Szenario werden die beiden in Kapitel 5.1.2.2 beschriebenen unidirektionalen Kopplungen gemeinsam betrachtet. Dabei entsteht eine bidirektionale interdisziplinäre Kopplung entsprechend Szenario 3 aus Kapitel 4.2.3, die sowohl die elektrischen Verluste $q(\boldsymbol{E})$ als auch die temperaturabhängige Permittivität $\varepsilon(T)$ berücksichtigt. Wird das Agentensystem mit der bidirektional gekoppelten Simulation initialisiert, so werden zur vollständigen Berechnung BA aus den funktionellen Gruppen zur Berechnung der elektrischen Feldstärke $\boldsymbol{E}$ und der Temperatur T benötigt. Für ein Agentensystem entsprechend Tabelle 5.2 ist analog zu Kapitel 5.1.2.1 sowohl eine monolithische als auch die gestaffelte Berechnung möglich. Zusätzlich kommt hier BA 4 während der Berechnung hinzu, der die Fähigkeiten von BA 2 aus Tabelle 5.1 besitzt. Die Zusammensetzung des Agentensystems ermöglicht so ein Konkurrieren der monolithischen und der gestaffelten Berechnungsmethoden. Eine analoge Fragestellung entsteht durch die Möglichkeit, verschiedene Berechnungsmethoden innerhalb der Berechnung einer Disziplin einzusetzen (siehe Kapitel 3.2.5). Der konkurrierende Umgang dieses Agentensystems mit den verschiedenen Berechnungsmethoden ist in [111, 166] beschrieben.

Das sich bei einheitlicher Diskretisierung ergebende monolithische Gleichungssystem (3.22) wird entsprechend Abbildung 3.5 nach Knotennummern sortiert, um die zum Lösen vorteilhafte

Bandstruktur zu erhalten. Bei einer Initialisierung der Lösungsvariablen mit *Null* ist (3.22) anfangs symmetrisch, wobei die Symmetrie bedingt durch die unsymmetrische Kopplung zwischen den Disziplinen anschließend verloren geht. Entsprechend gilt es, benötigte Parameter zur Bestimmung des geeigneten Lösungsverfahrens durch ein Empfehlungssystem bei jedem Assemblieren der Matrix zu extrahieren und auszuwerten. Hat das hier als Empfehlungssystem eingesetzte neuronale Netz zu wenig Trainingsdaten für eine Empfehlung, so scheitert die Generalisierung der Eingangsparameter. Dies trifft für die hier gekoppelten Disziplinen in Kombination mit dem daraus resultierenden unsymmetrischen Gleichungssystem zu. Aufgrund dessen entfällt für die monolithische Berechnung eine Berücksichtigung der Empfehlung des Lösungsverfahrens. Bei der gestaffelten Berechnung bleibt hingegen die Symmetrie der Teilsimulationen erhalten, sodass beispielsweise zur Berechnung der elektrischen Feldstärke $\boldsymbol{E}$ eine Empfehlung entsprechend Kapitel 5.1.1 gelingt. Das hinzugekommene und bei der gestaffelten Berechnung zu lösende Gleichungssystem der thermischen Teilsimulation ist symmetrisch, reell und hat bei einer dünn besetzten Speicherung eine Speichergröße von 210 MB. Es berücksichtigt die Kopplung in (3.7) als zusätzlichen Quellterm $\boldsymbol{r}$. Die Kopplung der Wellensimulation erfolgt über die Materialparameter in der Systemmatrix $\boldsymbol{K}$.

Abbildung 5.5 zeigt den Lösungsverlauf zur Berechnung der Simulation im Agentensystem, wobei das abwechselnde Berechnen von BA 2 und BA 3 aufgrund des gestaffelten Lösens der zwei gekoppelten Simulationen notwendig ist. Für eine gröbere Diskretisierung mit beispielsweise $7 \cdot 10^6$ Freiheitsgraden werden weitere und hier entsprechend in Summe fünf Iterationen notwendig [166]. Das Hinzukommen von BA 4 während des Lösungsprozesses ermöglicht eine zu BA 3 analoge Berechnung mit einem weiteren konkurrierenden Lösungsverfahren.

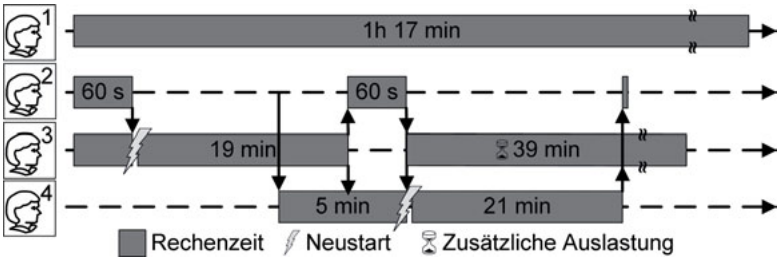

Abbildung 5.5: Lösungssequenz des bidirektional gekoppelten Modells

Dabei informiert sich BA 4 proaktiv über bereits verfügbare Kopplungswerte und Lösungsverfahren und berücksichtigt diese. Die anschließende Information über die Existenz einer Lösung der Fragestellung nimmt BA 4 zwar wahr, ein Abbrechen der Berechnung erfolgt jedoch erst, wenn neue Aufgaben zur Berechnung verfügbar werden. Ermöglicht wird so ein Vergleich von Rechenzeiten verschiedener Berechnungsverfahren. Die Information von BA 2 über dessen Ergebnis des zweiten Iterationsschritts veranlassen BA 4 zum Abbruch der Berechnung.

Die anschließende Neuinitialisierung der Berechnung geschieht ausgehend von den durch die BA 2 und 3 bereitgestellten Lösungsvektoren. Zu dessen Bereitstellung werden aufgrund der gleichen Diskretisierung der BA 3 und 4, mit jeweils $14 \cdot 10^6$ Freiheitsgraden, 254 MB je Iteration ausgetauscht. Die von BA 2 kommunizierten Daten entsprechen je Iteration denen aus Kapitel 5.1.2.2. Die durch externe Einflüsse herbeigeführte zusätzliche Auslastung von BA 3 kompensiert BA 4 im Folgenden und stellt nach Abschluss der Berechnung die Lösung BA 2 bereit. Diese ist in Abbildung 5.6 für eine Komponente der elektrischen Feldstärke und anhand eines Schnitts durch das Modell dargestellt. Der Einfluss der Linse sowie die dämpfende Wirkung der verlustbehafteten Materialien zeigt sich dabei anhand des Amplitudenverlaufs.

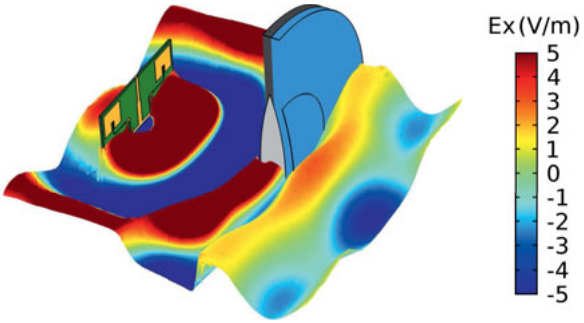

Abbildung 5.6: Berechnete elektrische Feldstärke $\boldsymbol{E}$ von BA 4

BA 2 stellt daraufhin keine nennenswerte Veränderung der gekoppelten Wärmequellen fest und beendet die Berechnung mit dem zuvor berechneten Ergebnis. Informationen über den Lösungsverlauf der von BA 3 verzögert bereitgestellte Lösung, lassen sich zur Erweiterung der Wissensbasis des Empfehlungssystems nutzen. Dazu ist eine Gewichtung der Rechenzeit mit den tatsächlich genutzten Ressourcen erforderlich. Die von BA 1 konkurrierend durchgeführte monolithische Berechnung der bidirektionalen Kopplung dauert hier länger als das gestaffelte kooperative Lösen der Simulation. Dabei werden bedingt durch die einheitliche Diskretisierung der beiden Disziplinen vom direkten Groblöser des Multigrids 155 GB Arbeitsspeicher benötigt (FGMRES, MG(SORV, MUMPS)). Die Anzahl nicht linearer Rechenschritte des Newtonverfahrens stimmt dabei mit der Anzahl an Iterationen des Agentensystems überein.

Alternativ zu dem in Abbildung 5.5 dargestellten Lösungsverlauf ist prinzipiell auch eine Berechnung mehrerer parallel verlaufender Berechnungssequenzen möglich (siehe Abbildung 4.16). Auf die Darstellung der Lösungsverläufe wurde verzichtet, da diese nur dann zustande kommen, wenn kein Unterbrechen der gleichzeitig gestarteten Berechnungen erfolgt. Eine weitere alternative Lösungssequenz entsteht, wenn das in Kapitel 5.1.1 erfolgreich berechnete Modell der Wellensimulation um die thermischen Zusammenhänge einschließlich deren Kopplungen erweitert und einem KA zur Berechnung übergeben wird. Im Modell erfolgte Änderungen

werden dabei durch den KA festgestellt. Dieser ermittelt so den Bedarf von BA mit den Fähigkeiten zur Berechnung von thermischen Simulationen und informiert diese über mögliche neue Aufgaben. Gleiches gilt, wenn zu Berechnungsbeginn ausschließlich BA mit Fähigkeiten zur Berechnung der elektrischen Feldstärke $\boldsymbol{E}$ im Agentensystem vorhanden sind und BA mit der Fähigkeit zur thermischen Berechnung erst nach Abschluss der ersten Iteration hinzukommen. Dies ist in der Praxis beispielsweise aufgrund einer endlichen Anzahl an Softwarelizenzen, der Verfügbarkeit von Rechenressourcen oder deren Auslastung anzutreffen. Dabei informieren sich neu hinzugekommene BA mit der Fähigkeit zur Berechnung thermischer Teilsimulationen proaktiv über bereits verfügbare Ergebnisse in ihrer Umgebung. Dieses Vorgehen ist analog zur interdisziplinären Kopplung zu betrachten und ermöglicht die automatisierte Berücksichtigung bestehender Werte als Startwerte für die iterative Berechnung oder deren Kopplung. Entsprechend werden hier die bereits berechneten Werte der elektrische Feldstärke $\boldsymbol{E}$ und der davon abhängigen Wärmequellen $q(\boldsymbol{E})$ bereitgestellt und bei der Berechnung der thermischen Simulation berücksichtigt. Die Auswertung des Beispiels unter Berücksichtigung der Startwerte zeigt jedoch, dass hier keine Verkürzung der Berechnungssequenz erreicht wird. Gesondert ist der Umgang mit BA zu betrachten, die unter Ausnutzung eines Teils ihrer Fähigkeiten Lösungen bereitstellen. Da eine Modifikation des Modells den bereitgestellten Lösungsvektor als unvollständig erscheinen lässt, wird auf deren bedarfsorientierte Funktionserweiterung zugunsten der Lösungsbereitstellung verzichtet.

Zusammenfassend zeigt das dargestellte Beispiel, dass ein Lösungsverlauf mit geringer Komplexität durch einen Agenten realisiert wird, der in der Lage ist die Simulation eigenständig zu bearbeiten. Fehlen diesem Kompetenzen, wie Möglichkeiten zur Simulationen oder zur Kooperation, so ermöglichen zusätzliche Agenten eine konkurrierende Berechnung. Das demonstrierte konkurrierende Verhalten auf Disziplinebene unterscheidet sich dabei von der Konkurrenz der Lösungsverfahren dadurch, dass anstelle von identischen Fragestellungen sich ergänzende Teilsimulationen berechnet und gekoppelt werden. Dabei wird die Individualität und Autonomie der Agenten automatisiert und zielführend berücksichtigt. Beispiele stellen das individuelle Anpassen der gemeinschaftlichen Fragestellung je BA und das demonstrierte kooperative Lösen der Fragestellung dar. Den beiden konkurrierenden Abläufen gemein ist die dynamische Rechenzeitberücksichtigung der Ressourcen zum schnellen Erreichen einer Lösung. Dies zeigt sich analog bei dem in [166] beschriebenen Konkurrieren verschiedener Berechnungsmethoden für verschiedene Teilgebiete der Geometrie. Eine Berücksichtigung von redundanten Agenten beweist sich ebenfalls als vorteilhaft. Beispiele stellen die während einer laufenden Berechnung einfach hinzuzufügenden BA mit aktualisierter Feldberechnungssoftware oder modernerer Hardware dar. Zudem gelingt es, auf unvorhergesehene Ereignisse wie den Ausfall von BA oder deren reduzierte Leistungsfähigkeit zu reagieren, ohne dass eine Neuberechnung notwendig wird. Dies gilt insbesondere auch für kleinere Simulationen, bei denen ein Aufteilen der Berechnung, beispielsweise auf ein Rechencluster, unnötig ist. Im Rahmen der

alternativen Lösungssequenzen zeigt sich, dass Agentensysteme einen hinzukommenden Bedarf an benötigter Funktionalität auch während der Laufzeit der Berechnung decken. Dabei bereits berechnete Zwischenergebnisse werden modellorientiert und automatisiert berücksichtigt. Dies stellt ein weiteres Unterscheidungsmerkmal zu bestehenden Feldberechnungsumgebungen dar. Im System befindliche Agenten passen sich dabei dynamisch an die geänderte Systemkonfiguration an. Dieses Beispiel zeigt, dass in einem Agentensystem ein gemeinsames Bearbeiten von Aufgaben gelingt, zu deren Bearbeitung ein Einzelner nicht in der Lage ist.

5.2 Transistor

Anhand der im Folgenden dargestellten Berechnungen eines NPN-Tri-Gate-Transistors werden weitere Vorteile einer Softwareagenten-basierten numerischen Feldberechnung dargestellt. Da die Funktion eines NPN-Tri-Gate-Transistors von dem sich bildenden dreidimensionalen Leitungskanal abhängt, ist die dreidimensionale Simulation des Modells notwendig [205]. Die zur Demonstration der prinzipiellen Funktionsweise gewählten typischen geometrischen Abmessungen des symmetrischen Transistormodells zeigt Abbildung 5.7. Ebenso dargestellt ist die durch Dotierung erreichte fiktive Konzentration an Fremdatomen N.

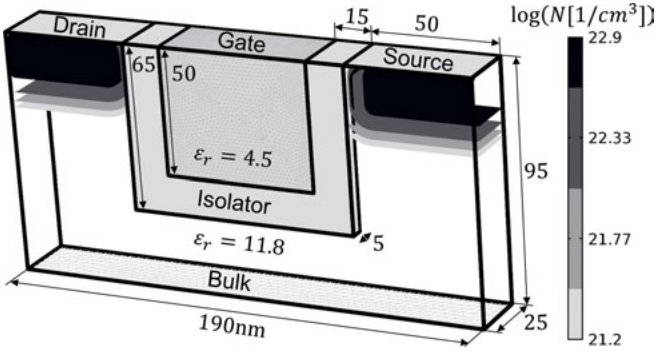

Abbildung 5.7: Modell eines NPN-Tri-Gate-Transistors

Die Berechnung erfolgt ausgehend von der Poisson-Gleichung (3.1). Die wird die Raumladungsdichte ρ als Produkt der Elementarladung $q = 1{,}6 \cdot 10^{-19}$ As mit einer Elektronenkonzentration n, einer Löcherkonzentration p und einer Konzentration an Fremdatomen N ausgedrückt. Bei der Konzentration der Fremdatome N ist zudem eine Unterscheidung in die Konzentration der Donatoren N_D^+ und Akzeptoren N_A^- üblich. Die so zur Simulation des elektrischen Potenzials φ auszuwertende Poisson-Gleichung ist

$$-\operatorname{div}(\varepsilon \operatorname{grad}(\varphi)) = q(p - n + N_D^+ - N_A^-). \tag{5.2}$$

Diese Darstellung ermöglicht im Folgenden eine getrennte Betrachtung der Ladungsträgerkonzentrationen n und p. Die Auswertung der Ladungserhaltung $\mathrm{div}\,(\boldsymbol{J}) + \frac{\partial \rho}{\partial t} = 0$ ergibt für die zeitabhängige Änderung der Elektronenkonzentration $\frac{\partial n}{\partial t} = \frac{1}{q}\mathrm{div}(\boldsymbol{J}_n)$. Analog ist auch eine zeitabhängige Simulation der Änderung der Löcherkonzentration $\frac{\partial p}{\partial t} = \frac{1}{q}\mathrm{div}(\boldsymbol{J}_p)$ notwendig. Nicht lineare Effekte wie die Generation und Rekombination von Ladungsträgern werden hier nicht berücksichtigt. Die Teilstromdichten sind $\boldsymbol{J}_n = qD_n\,\mathrm{grad}\,(n) - q\mu_n n\mathrm{grad}\,(\varphi)$ und $\boldsymbol{J}_p = qD_p\,\mathrm{grad}\,(p) - q\mu_p p\mathrm{grad}\,(\varphi)$ und berücksichtigen das Gefälle der betrachteten Ladungsträgerkonzentration und die an das Bauteil angelegte elektrische Potenzialdifferenz. Dabei entsprechen D_n und D_p den Diffusionskoeffizienten und μ_n und μ_p der Ladungsträgerbeweglichkeit. So sind die Transportgleichungen

$$\frac{\partial n}{\partial t} = \frac{1}{q}\mathrm{div}\left(\underbrace{qD_n\,\mathrm{grad}\,(n)}_{\boldsymbol{J}_{n,\mathrm{Dif.}}} - \underbrace{q\mu_n n\mathrm{grad}\,(\varphi)}_{\boldsymbol{J}_{n,\mathrm{Drift}}}\right) \tag{5.3}$$

und

$$\frac{\partial p}{\partial t} = \frac{1}{q}\mathrm{div}\left(\underbrace{qD_p\,\mathrm{grad}\,(p)}_{\boldsymbol{J}_{p,\mathrm{Dif.}}} - \underbrace{q\mu_p p\mathrm{grad}\,(\varphi)}_{\boldsymbol{J}_{p,\mathrm{Drift}}}\right) \tag{5.4}$$

zusätzlich zu (5.2) bei einer Simulation auszuwerten [206]. Entsprechend sind hier die drei abhängigen Variablen n, p und φ zu berechnen. Die Vernetzung der Geometrie erfolgt unter Berücksichtigung der Péclet-Zahl $P_e \leq 2$, die durch das Verhältnis des Driftstroms $\boldsymbol{J}_{\mathrm{Drift}}$ zum Diffusionsstrom $\boldsymbol{J}_{\mathrm{Dif.}}$ angegeben wird. Die Auswertung der Péclet-Zahl P_e ist dabei unter Berücksichtigung der beiden Ladungsträgerarten p und n notwendig.

5.2.1 Ortsabhängige Kopplungen

Da zur zeitabhängigen Berechnung des Transistors geeignete stationäre Startwerte für das elektrische Potenzial φ erforderlich sind, wird auf deren Berechnung im Folgenden eingegangen. Dies erfolgt beispielhaft für eine agentenbasierte Kopplung mehrerer Teilmodelle an Gebietsrändern (siehe Abbildung 3.3). Zur randgekoppelten Berechnung des Transistors wird dieser aus drei jeweils benachbarten und überlappenden Teilgebieten zusammengefügt. Dies entspricht einer Koordination des Agentensystems auf Gebietsebene (siehe Abbildung 4.3). Die äußeren Teilgebiete sind jeweils 40 nm lang, wobei 5 nm mit den Gebietsnachbarn überlappen. Die Kopplungsmatrix $\tilde{\boldsymbol{C}}$ entspricht

$$\tilde{\boldsymbol{C}} = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 1 \end{bmatrix} \tag{5.5}$$

und verdeutlicht die bidirektionale Kopplung an den Rändern benachbarter Simulationsteile. Die Berechnung erfolgt analog zur alternierenden Schwarzmethode mit überlappenden Teilgebieten, wobei die ausgetauschten Dirichlet-Randbedingungen hier dem elektrischen Potenzial φ entsprechen. Die zu berechnende Näherung des elektrischen Potenzials φ erfolgt ausgehend von (5.2) für das thermische Gleichgewicht ($n_i^2 = p \cdot n$). Dabei werden die Ladungsträgerkonzentrationen p und n in Abhängigkeit von der intrinsischen Ladungsträgerdichte n_i als

$$n = n_i \cdot e^{\frac{q(\varphi_n - \varphi)}{\kappa_B T}} \tag{5.6}$$

und

$$p = n_i \cdot e^{\frac{q(\varphi - \varphi_p)}{\kappa_B T}} \tag{5.7}$$

mit den Quasi-Fermi-Potenzialen φ_n und φ_n beschrieben. Aufgrund von (5.6) und (5.7) ist die Berechnung des elektrischen Potenzials φ in (5.2) im Folgenden nicht linear. Nachdem eine verteilte Rechnung für Simulationen mit geringerem Aufwand nicht erforderlich ist, werden hier drei BA auf einem Rechner ausgeführt. Dieser Rechner hat einen Xeon-E3-1275v5 Prozessor der bei 3,6 GHz betrieben wird und dem 64 GB Arbeitsspeicher zur Verfügung stehen. Dabei wird jeder BA exklusiv auf einem Rechenkern ausgeführt, um die Nebenläufigkeit der Prozesse sicherzustellen. Alternativ ist auch eine ressourcenangepasste Aufteilung der Teilsimulationen analog zu Kapitel 5.1.1 möglich, die unter Berücksichtigung von Leistungskennzahlen der eingesetzten Hardware erfolgt [5]. Die Vernetzung erfolgt selbstständig durch die beteiligten drei BA. Entsprechend entstehen drei Teilsimulationen mit $5{,}6 \cdot 10^4$ Freiheitsgraden für die Teilgebiete 1 und 3 und $8{,}4 \cdot 10^4$ Freiheitsgraden für das Teilgebiet 2. Für die Auswertung wird angenommen, dass sich BA entsprechend ihrer Nummerierung für die Bearbeitung des entsprechenden Teilgebiets entschließen. Die gesamte Fragestellung hat so $2 \cdot 10^5$ Freiheitsgrade, wobei die Teilgebiete im Bereich des Überlapps keine gemeinsamen Knoten aufweisen. Zur Kopplung benötigte Werte werden so durch Interpolation ermittelt. Abbildung 5.8 stellt anhand eines Schnitts durch den Transistor die Teilsimulationen, den jeweils zuständigen BA, die durch ihn erstellte Diskretisierung und das im Folgenden berechnete elektrische Potenzial φ dar.

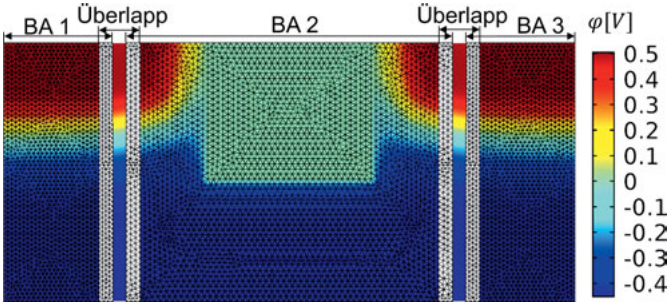

Abbildung 5.8: Mit Gebietszerlegung berechnetes Potenzial φ im thermischen Gleichgewicht

Die Rechenzeiten der zum Lösen der Gleichungssysteme verwendeten und durch die BA verwalteten Simulationswerkzeuge sind in Abbildung 5.9 in Form eines Sequenzdiagramms dargestellt. Dabei ermöglicht die parallele Initialisierung der Berechnung einen Start von drei parallelen Lösungssequenzen (I, II, III). Der unterschiedliche Berechnungsbeginn ergibt sich durch den teilmodellspezifischen Aufwand zur Vorbereitung der Berechnung, wie beispielsweise deren Vernetzung. Die Berechnung der jeweils nicht linearen Teilsimulationen erfolgt mittels eines auf das Teilgebiet angepassten Newton-Verfahrens unter Anwendung eines direkten Berechnungsverfahrens für die linearisierten Rechenschritte.

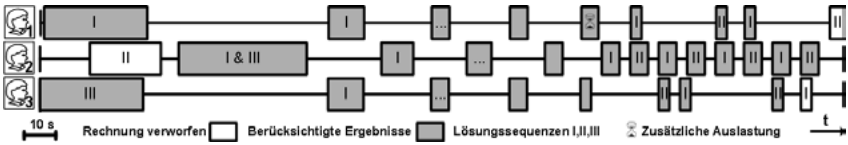

Abbildung 5.9: Lösungssequenz und Rechenzeiten der Gebietskopplung

Der Abbruch des ersten Berechnungsschritts von BA 2 dient dem sofortigen Berücksichtigen der bereits verfügbaren Zwischenergebnisse der Lösungssequenzen I und III. Die bis dahin von BA 2 berechneten 5 nicht linearen und dazu in Summe notwendigen 42 linearen Iterationen werden als Startwerte für die folgende Berechnung genutzt. Unter Berücksichtigung der Kopplung werden so im nächsten Rechenschritt 13 nicht lineare und 114 lineare Iterationen notwendig. Eine Berücksichtigung der Kopplung erst nach Abschluss des ersten Rechenschritts von Lösungssequenz II bedeutet hier ein Abwarten von 15 nicht linearen und 132 linearen Rechenschritten. Das im Anschluss berechnete identische Ergebnis wird ansonsten erst nach weiteren 5 nicht linearen und 44 linearen Iterationen erreicht. Dieses vorteilhafte Vorgehen entspricht so einem Abbruch von Lösungssequenz II und dem geometrie- und rechenzeitbedingten Zusammenfassen der Lösungssequenzen I und III zu der im Anschluss fortgesetzten Lösungssequenz I. Da sowohl BA 1 als auch BA 3 das berechnete Ergebnis berücksichtigen, ist die Abfolge der nächsten Rechenschritte als ein wiederkehrendes Aufspalten der Lösungssequenz und anschließendes Zusammenführen zu verstehen. Das Ergebnis von BA 2 nach Berechnung des zweiten Iterationsschritts ist beispielhaft in Abbildung 4.6 dargestellt.

Das Zusammenführen der Ergebnisse gelingt bis zum Zeitpunkt einer verzögerten Lösungsbereitstellung durch BA 1. Da der Berechnungsverlauf der BA ohne Synchronisationspunkte auskommt, spaltet sich die Lösungssequenz hier in zwei Teilsequenzen auf. Diese werden im Folgenden parallel berechnet, wobei ein erneutes Zusammenfassen der Lösungssequenz hier aufgrund der sehr ähnlichen Rechenzeiten der Teilsimulationen bis zum Ende der Berechnung nicht mehr gelingt. Die Berechnung endet mit den Entscheidungen der BA 1 und 3. Eine Auswertung der Fehlerentwicklung während der iterativen Lösungssequenz der drei BA ist in Abbildung 5.10 dargestellt. Deren nahezu gleicher Verlauf für die BA 1 und 3 weist auf die

Ähnlichkeit und Symmetrie des in Abbildung 5.8 dargestellten elektrischen Potenzials φ der äußeren Teilgebiete hin. Zu nur sehr geringen Änderungen des Fehlers $\tilde{\epsilon}$ kommt es nach dem Aufspalten der Lösungssequenz bei jeder zweiten Iteration. Dies ist begründet durch die zwei nun parallel verlaufenden Lösungssequenzen und deren ähnliche Berechnungsverläufe. Die Berücksichtigung des durch BA 2 berechneten elektrischen Potenzials φ führt dabei nur noch in einer der von BA 1 oder 3 berechneten Lösungen zu wesentlichen Änderungen. In der Anderen sind die Änderungen äußerst gering, da die Lösung bereits Randbedingung der Berechnung von BA 2 ist und Änderungen nur aufgrund des Überlapps des Koppelrands auftreten.

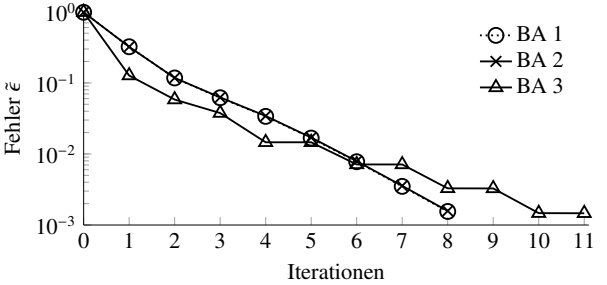

Abbildung 5.10: Fehlerentwicklung während der Rechenschritte

Der Lösungsverlauf aus Abbildung 5.9 gibt zudem Auskunft über die Verarbeitungszeiten des Agentensystems, die für Vor- und Nachbereiten einer Rechnung, den Umgang mit Schnittstellen und den Datenaustausch untereinander benötigt werden. Dazu werden in diesem Beispiel zwischen 3 s und 5 s, bei einer Berechnungsdauer je Iterationsschritt zwischen 36 s und 3 s, notwendig. Für größere Fragestellungen, wie in den Beispielen der Kapitel 5.1 und 5.2.2, steigt die Verarbeitungszeit bei verteilter Berechnung im Mittel auf 20 s an. Auch dies ist im Vergleich zu den dort je Iterationsschritt notwendigen Rechenzeiten vernachlässigbar.

Dieses Beispiel zeigt die Anpassungsfähigkeit verwendeter BA und des damit berechneten Lösungsverlaufs an die in der Umgebung verfügbaren Zwischenlösungen. Die Autonomie der BA ermöglicht situationsabhängig eine oder mehrere Teillösungen bei der Berechnung zu berücksichtigen, um situationsangepasst zu reagieren. Die parallele Existenz mehrerer Lösungssequenzen wird dabei zielführend eingesetzt, wobei ein Zusammenfassen ebenso wie ein Hinzukommen möglicher Lösungssequenzen rechenzeitspezifisch berücksichtigt wird. Zudem werden die Fähigkeiten des Agentensystems zur automatisierten Konfiguration und Berechnung einer gebietsgekoppelten Simulation bei gleich befähigten BA demonstriert. Dabei stellen benachbarte Gebietsgrenzen die Grundlage zur Kooperation und Kopplung dar, wobei die Kopplung von Gebieten mit verschiedener Diskretisierung automatisiert erfolgt. Ermöglicht wird die randgekoppelte Berechnung der Simulation dabei erst durch die Kommunikation zwischen den Agenten und durch deren Wahrhaftigkeit im Hinblick auf ausgetauschte Lösungen.

5.2.2 Zeitabhängige Kopplungen

Die in Kapitel 5.2.1 berechneten Startwerte werden nun verwendet, um den Drain-Strom des Transistors bei einer Source-Drain-Spannung von 1 V und einer harmonischen Gate-Source-Spannung von 0,6 V + 0,3 · cos(ωt) V mit $\omega = 2\pi \cdot 50$ GHz zu berechnen. Die Berechnung der unidirektionalen Kopplung zwischen verschiedenen Zeitpunkten gelingt mittels impliziter Lösungsverfahren. Gleich befähigte BA berechnen dabei von einem Startzeitpunkt ausgehend parallel verschiedene Zeitschritte, wobei die Entscheidung zur Auswahl eines geeigneten Zeitschritts anhand von rekursiven Fehlerschranken erfolgt. Zur Auswertung eines periodischen Systems mit kleinem Spektrum der Anregungen und der auszuwertenden nicht linearen Zusammenhänge ist eine frequenztransformierte multiharmonische Berechnung sinnvoll [89, 207]. Diese Berechnung wird im Folgenden durch das Agentensystem ausgeführt. Eine Beschreibung der Berechnung zeitabhängiger Simulationen dieses Agentensystem ist in [208] gegeben.

Die zur multiharmonischen Berechnung notwendige Fourierreihenapproximation der abhängigen Variablen erfolgt entsprechend (3.13). Für die drei abhängigen Variablen n, p und φ ergeben sich so die drei komplexen Größen $\mathfrak{n}_k$, $\mathfrak{p}_k$ und ϕ_k mit $k = 0 \ldots N$ entsprechend der berücksichtigten harmonischen Frequenzen. Diese in die Poisson-Gleichung (5.2) eingesetzt und entsprechend der frequenzspezifischen Anteile separiert, ermöglicht die Berechnung von

$$-\operatorname{div}\left(\varepsilon \operatorname{grad}\left(\phi_k\right)\right) = q(\mathfrak{p}_k - \mathfrak{n}_k + N_D^+ - N_A^-) \tag{5.8}$$

für die betrachtete Frequenz k. Die Fouriertransformation der beiden Transportgleichungen (5.3) und (5.4) erfolgt analog. Dabei führt die Multiplikation der abhängigen Variablen innerhalb des Diffusionsstroms $\boldsymbol{J}_{\text{Dif.}}$ zu einer Faltung dann komplexen Größen. Die analytische Berechnung der Faltung unter Berücksichtigung der Eigenschaften einer reellen Funktion $\mathfrak{u}(\omega) = \mathfrak{u}^*(-\omega)$ ermöglicht die anschließende Separation der Frequenzanteile. Hier berechnete Frequenzanteile sind der Gleichanteil, die Grundfrequenz und deren Vielfache. Aufgrund einer als stark angenommenen Kopplungen der sich je Frequenz ergebenden drei transformierten Gleichungen untereinander, erfolgen die Berechnungen der drei abhängigen Variablen gemeinsam.

Die separierten Frequenzanteile teilen sich die verfügbaren, verteilt ausgeführten und gleich befähigten BA beginnend bei der Grundfrequenz und sich anschließend frequenzbezogen steigernd untereinander auf. Die Anzahl der Frequenzanteile entspricht hierbei der Anzahl verfügbarer BA. Da die zu erwartende Anzahl nicht linearer Berechungsiterationen bei höheren Frequenzen abnimmt, werden diese Teilsimulationen von BA mit niedriger Leistungsfähigkeit berechnet (siehe Kapitel 4.1.1.1). Dabei berechnet ein BA die abhängigen Größen $\mathfrak{n}_k$, $\mathfrak{p}_k$ und ϕ_k für eine Frequenz k. Die Berechnung eines BA erfolgt hier beispielhaft für eine vom Anwender vorgegebene Diskretisierung mit $6 \cdot 10^5$ komplexen Freiheitsgraden. Die Teilsimulationen werden mittels eines durch die BA auf die Teilsimulation angepassten Newton-Verfahrens und

des direkten Lösers MUMPS unter Verwendung aller verfügbaren Rechenkerne berechnet. Tabelle 5.3 stellt das zur Berechnung des Transistors eingesetzte Agentensystem dar. Zudem ist die hardwareabhängig erfolgte Zuordnung der BA zu den Teilsimulationen dargestellt. Eine Berücksichtigung der in Kapitel 5.2.1 bestimmten Startwerte erfolgt bei der Berechnung des Gleichanteils und ist in der Rechenzeit von BA 1 berücksichtigt.

Tabelle 5.3: Agentensystem der multiharmonischen Berechnung

Agent	Teilsimulation	Rechnerkonfiguration
BA 1	Gleichanteil	Xeon E5-2630v3 8-Core 2,4 GHz, 256 GB
BA 2	Grundfrequenz	Xeon E5649 6-Core 2,5 GHz, 144 GB
BA 3	Oberwelle 1	Xeon E5-2643 4-Core 3,3 GHz, 385 GB
BA 4	Oberwelle 2	i7 4790 4-Core 3,6 GHz, 32 GB
BA 5	Oberwelle 3	i5 4690 Quad-Core 3,5 GHz, 16 GB

Bedingt durch die Faltungen in den transformierten Transportgleichungen sind alle Teilsimulationen bidirektional miteinander gekoppelt. Die Kopplungsmatrix $\tilde{\boldsymbol{C}}$ ist entsprechend voll besetzt. Lediglich während der Initialisierung ergibt sich eine dünnere Besetzung. Diese ist begründet durch die zu Berechnungsbeginn als null angenommenen Werte der gekoppelten BA. Dabei sind die zur Kopplung ausgetauschten Werte, analog zur Kopplung verschiedener physikalischer Zusammenhänge, volumenbezogene Größen. Die während der Berechnung dieses Beispiels dynamisch entstandene asynchrone Lösungssequenz zeigt Abbildung 5.11.

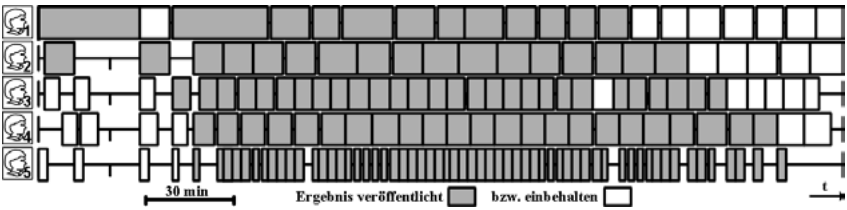

Abbildung 5.11: Lösungssequenz der agentenbasierten multiharmonischen Berechnung

Dabei werden Teilergebnisse, deren relative Abweichung aller abhängigen Variablen kleiner 10^{-5} ist, nicht veröffentlicht. Dies trifft zu Beginn der Berechnung auf die Ergebnisse der BA 3 bis 5 zu, da die Anregung lediglich einen Gleichanteil und die Grundfrequenz beinhaltet. Ebenso führen die Änderungen in den Teilergebnissen der höheren Frequenzen gegen Ende der Berechnung zu keiner wesentlichen Änderung der Lösungen niedrigerer Frequenzen mehr. Dies bestätigt den wesentlichen Beitrag der niedrigen Frequenzen für das Ergebnis und die Notwendigkeit der Berechnung endlich vieler Oberwellen [91]. Auch der aus den jeweiligen Lösungen der BA berechnete und in Tabelle 5.4 dargestellte Anteil an der Gesamtleistung bekräftigt dies.

Tabelle 5.4: Anteil der BA an der berechneten Gesamtleistung

Agent	BA 1	BA 2	BA 3	BA 4	BA 5
Leistung	$2{,}2 \cdot 10^{-3}$ W	$5{,}6 \cdot 10^{-5}$ W	$7 \cdot 10^{-7}$ W	$4{,}8 \cdot 10^{-8}$ W	$1{,}3 \cdot 10^{-8}$ W

Die mittels des Agentensystems benötigte Rechenzeit zur Bereitstellung einer Lösung beträgt 4 h 15 min. Zum Erreichen der Lösungssequenz werden bei jedem der dargestellten Recheniterationen jeweils alle bis dahin verfügbaren Ergebnisse durch die BA berücksichtigt. Hingewiesen sei auch auf den angenommenen Ausfall der Hardware eines BA während der Berechnung. Das Agentensystem ermöglicht dabei eine Fortsetzung der Berechnung nach erneutem Lösungsaustausch der verbleibenden BA mit dem ausgefallenen BA. Im Vergleich mit einer auf dem Rechner von BA 1 durchgeführten gestaffelten Vergleichsrechnung, die 9 h in Anspruch nimmt, wäre dort ein Neustart der Rechnung erforderlich. Eine sequenzielle Berechnung aller Simulationsschritte würde über 18 h dauern.

Die zur Auswertung der abhängigen Variablen notwendige Rücktransformation der berechneten Ergebnisse in den Zeitbereich ist nach Abschluss der Berechnungen durch jeden BA möglich. Dies ist aufgrund der bidirektionalen Kopplung aller BA untereinander möglich, da jeder BA kopplungsbedingt gültige Lösungen aller anderen besitzt. Eine Darstellung der logarithmischen Ladungsträgerkonzentration ist für die Zeitpunkte $t = 0$ s und $t = 10$ ps als Schnittbild durch den Transistor in Abbildung 5.12 dargestellt. Deutlich wird darin auch die dreidimensionale Struktur des sich ausbildenden n-Kanals.

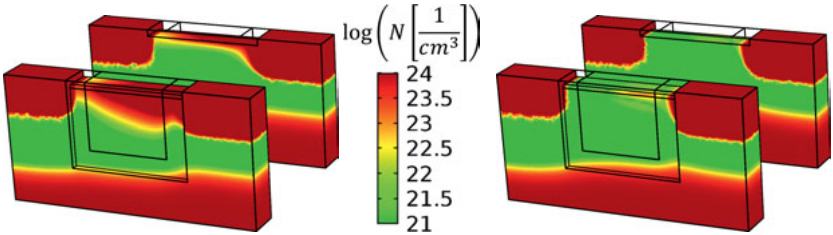

Abbildung 5.12: Logarithmische Ladungsträgerkonzentration für $t = 0$ s und $t = 10$ ps

Eine Auswertung des so berechneten zeitlichen Verlaufs des Drain-Stroms sowie der dazu angelegten Gate-Source- und der Source-Drain-Spannung ist in Abbildung 5.13 dargestellt. Deutlich wird dabei der nicht linear von der Gate-Source-Spannung abhängige Drain-Strom sowie die Sperrspannung des Transistors.

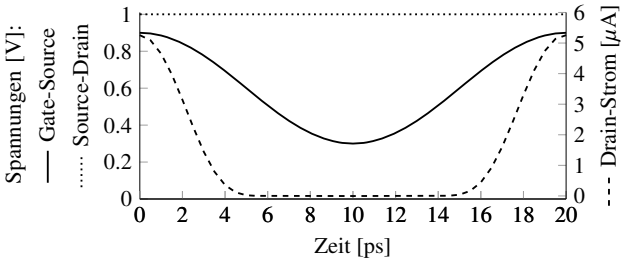

Abbildung 5.13: Strom- und Spannungsverlauf des Transistors

Zusammenfassend zeigt dieses Beispiel, dass auch eine Berechnung mehrfach bidirektional untereinander gekoppelter Simulationen mittels des Agentensystems gelingt. Dabei erfolgt eine selbstständige und sinnvolle Aufgabenverteilung durch das Agentensystem innerhalb eines heterogenen Rechnernetzes anhand festgelegter Leistungsmerkmale. Die demonstrierte dynamische Kopplung der Teilsimulationen gelingt mittels Agenten sowohl an gemeinsamen Rändern als auch innerhalb von Gebieten. Zusätzlich zu den in Kapitel 5.1 betrachteten stationären Fragestellungen zeigt dieses Beispiel, dass auch zeitabhängige Fragestellungen vom Einsatz von Agenten profitieren. Die erfolgreiche und verteilte Berechnung der Teilsimulationen gelingt dabei aufgrund der Eigenschaften von Agenten (siehe Kapitel 2.2).

6 Resümee und Ausblick

Zu Beginn dieser Arbeit wurde die Notwendigkeit der Berechnung von interdisziplinären und gekoppelten Simulationen auf sich dynamisch verändernden Rechnernetzen dargelegt. Um sich daraus ergebenden Anforderungen an eine Simulationsumgebung Rechnung zu tragen, ist hier zweckdienlich das Programmierparadigma der agentenorientierten Softwareentwicklung angewendet worden. Dessen Eignung wurde anhand einer Darstellung der so erreichbaren Eigenschaften eines so entwickelten Systems und dessen Komponenten dargestellt. Zudem erfolgte eine Abgrenzung zu bestehenden Simulationsumgebungen. Die Darstellung der Grundlagen der zu berechnenden Simulationen ermöglichte eine Strukturierung dieser zur anschließenden Abbildung in ein Softwaresystem. Das dafür entwickelte und vorgestellte Softwareagentensystem eignet sich zur gemeinschaftlichen und verteilten numerischen Feldberechnung interdisziplinärer gekoppelter Simulationen. Befähigt durch die Eigenschaften und umgesetzten Verhalten der Softwareagenten behandelt es automatisiert Fragestellungen, die sich aus den verschiedenen Arten der Zusammenarbeit von autonomen Softwareeinheiten während der Feldberechnung ergeben. Die detaillierten Darstellungen der umgesetzten Kommunikationsschnittstellen und des erreichten numerischen Lösungsprozesses verdeutlichten die Funktionsweise des Agentensystems. Dabei stand die Autonomie der Berechnungseinheiten im Vordergrund, die beispielsweise den Umgang mit unterschiedlichen Diskretisierungen und die zielführende Nutzung von redundanten Rechenressourcen ermöglichte. Eine Betrachtung der Ressourcennutzung eines einzelnen Softwareagenten und des Agentensystems rundeten die Darstellung des entwickelten Softwareagentensystems ab. Anhand der Berechnung ausgewählter Simulationsbeispiele wurde die Funktionalität des Agentensystems belegt und der durch die agentenbasierte Herangehensweise gewonnene Mehrwert nachgewiesen. Ein Beispiel stellte die Unterstützung des Anwenders bei der dynamischen Konfiguration des sich an Randbedingungen anpassenden Agentensystems dar, wobei lernfähige Systemkomponenten eingesetzt wurden. Die Leistungsfähigkeit des realisierten Systems demonstrierte die Berechnung verschieden aufwendiger Modelle mit unterschiedlichen Kopplungen sowie die dafür durchgeführte Auswertung.

Die vorgestellte Arbeit steht durch ihre Betrachtung verteilter autonomer Systeme und deren Interaktion sowie der Analogie zum Themenkomplex der Industrie 4.0 im Interessenfokus der Anwender. Das dargestellte komplexe und verteilte System wurde zielführend und zuverlässig zur Bearbeitung von herausfordernden numerischen Beispielen genutzt. Das entwickelte System zeichnet sich dabei durch seine Flexibilität und Zuverlässigkeit aus. Zusätzlich stellt die Arbeit eine zukunftsorientierte Vorgehensweise vor, um das wachsende Interesse an der Berechnung

gekoppelter interdisziplinärer Fragestellungen zu handhaben. Auch zur Verwaltung der zur Simulationsberechnung notwendigen Rechnerinfrastruktur bietet das umgesetzte Agentensystem zukunftsorientierte Funktionalität an. Die Unterstützung des wachsenden Anwenderkreises, der Simulationen zur Produktgestaltung verwendet, stößt im Besonderen auf Industrieseite auf ein verstärktes Interesse. Der geleistete Forschungsbeitrag bietet so für alle genannten Zielgruppen Anknüpfungspunkte sowie Antworten auf aktuelle Fragestellungen.

Die folgenden Aspekte stellen zukünftige Erweiterungen des vorgestellten Agentensystems und des darin realisierten Konzeptes dar. Dabei ist eine Erweiterung innerhalb der drei Themenbereiche des Lernens, der besseren Konvergenz und eines weiteren Parallelisierens erstrebenswert. So ist eine Erweiterung des bisher eingesetzten lernenden Systems zur Auswahl numerischer Verfahren um alle Ebenen der Hierarchie in Abbildung 4.3 denkbar. Möglich wird dann, dass Agenten bei wiederkehrenden Aufgaben Kooperationspartner bevorzugen, mit denen eine Zusammenarbeit zielführend und schnell gelungen ist. So lässt sich auch der Einsatz der beispielsweise in der Cloud betriebenen Berechnungsagenten statistisch analysieren. Zudem gelingt so eine Erweiterung der Empfehlungen von zusätzlichen gekoppelten Disziplinen oder der Einsatz von Methoden, die sich bisher als besonders geeignet bei der Berechnung bewiesen haben. Auch lässt sich der während der Exploration verschiedener Lösungswege zu erkundete Graph (siehe Kapitel 2.3) weiter einschränken. Angelehnt an die Simulation des Transistors in Kapitel 5 wird beispielsweise eine erfahrungsbasierte Entscheidung zwischen den ebenfalls für derartige Fragestellungen eingesetzten Berechnungsmethoden der Finiten Volumen und der Methode der Finiten Differenzen möglich. Auch die automatisierte Entscheidung für geeignete gebietsweise Formulierungen in Abhängigkeit der Modelleigenschaften ist so vorstellbar.

Das Erreichen der Konvergenz stellt gerade dann eine Herausforderung dar, wenn gekoppelte Simulationen kooperativ berechnet werden. Diese zeigt sich beispielhaft an keiner bzw. nur sehr schlechter lokaler Konvergenz oder einer Vielzahl erforderlicher globaler Recheniterationen während einer Berechnung. Letzteres trifft beispielsweise auf den in Kapitel 5 berechneten Transistor bei einer Berücksichtigung der Rekombination zu. Dabei führt die Rekombination zu einer stärkeren Kopplung der Oberwellen untereinander. Eine Stabilisierung und Beschleunigung der Berechnung gelingt durch eine zentrale Sammlung und Bereitstellung von nunmehr globalen Informationen [89]. Die so mögliche Koordination des Agentensystems eignet sich zudem zur bedarfsorientierten Umsetzung holoner Agenten. Diese geben Teile der Autonomie ab, um nach außen als Einheit aufzutreten. Entsprechend wäre der Erhalt der Individualität und Flexibilität solange nutzbar, bis diese aus Gründen der Zielerreichung in den Hintergrund treten. Auch der Umgang mit verschiedenen gekoppelten Diskretisierungen bei der Berechnung stellt seinerseits ein eigenes Forschungsfeld dar.

Weiterentwicklungen im Hinblick auf eine Parallelisierung der Berechnung des Agentensystems lassen sich beispielsweise für Parameterstudien umsetzen. Dabei durch Agenten automatisiert

zu beantwortende Fragen ergeben sich aus der bisher zu Berechnungsbeginn festzulegenden Abfolge der Rechenschritte, wie beispielsweise mit welchen Parametern die Berechnung begonnen wird, oder inwieweit sich die bereits berechneten Lösungen positiv in Bezug auf Konvergenz oder Rechenzeit weiterverwenden lassen. Auch ob eine Zerlegung der Simulation sinnvoll ist oder ein anschließendes Wiederverwenden von Teilergebnisse gelingt, stellen mögliche Fragestellungen dar. Diese knüpfen an Themen der Optimierung, wie beispielsweise der Modellreduktion oder der Rechenzeitoptimierung, an. Die Handhabe der Dynamik im Rechnersystem ergänzt diese Fragen. Beispiele dafür sind die dynamische Anpassung von Modellgrenzen oder die des Überlapps beim Ausfall von Agenten während der Berechnung einer gebietsgekoppelten Simulation. Auch eine wiederkehrende Verifikation der bisher statischen Zerlegungshierarchie der Modelle dient einer schnelleren Berechnung. Dabei wird durch eine flexible Anpassung eine Reduktion der Berechnungsschritte und eine geringe ausgetauschte Datenmenge erwartet.

Zusätzliche Möglichkeiten bietet das Agentensystem bei einer detaillierten Untersuchung der hier lediglich referenzierten Betrachtung von Fragestellungen im Zeitbereich. Dabei stellt die adaptive und individuelle Anpassung der Zeitschrittgrößen für gekoppelte Simulationen ein weiteres Forschungsfeld dar [2, 106]. Auch der Entwurf und die Umsetzung eines Standards für eine Schnittstelle zwischen verschiedenen Simulationswerkzeugen, die sowohl von der Programmiersprache als auch vom eingebundenen Quelltext unabhängig ist, würde eine wesentliche Vereinfachung bei der Kopplung von Simulationswerkzeugen mit sich bringen.

Literaturverzeichnis

[1] Wissenschaftsrat, „Bedeutung und Weiterentwicklung von Simulation in der Wissenschaft“, Brohler Straße 11, 50968 Köln, 2014.

[2] D. E. Keyes, L. C. McInnes, C. Woodward, W. D. Gropp, E. Myra u. a., „Multiphysics simulations: Challenges and opportunities“, *International Journal of High Performance Computing Applications*, Bd. 27, Nr. 1, S. 4–83, 2013.

[3] M. Rauscher, „Agentenbasierte Konsistenzprüfung heterogener Modelle in der Automatisierungstechnik“, Dissertation, Universität Stuttgart, 2015.

[4] C. A. MacK, „Fifty years of Moore's law“, *IEEE Transactions on Semiconductor Manufacturing*, Bd. 24, Nr. 2, S. 202–207, 2011.

[5] E. Strohmaier, H. Meuer, J. Dongarra und H. Simon, „The TOP500 List and Progress in High-Performance Computing“, *Computer*, Bd. 48, Nr. 11, S. 42–49, 2015.

[6] T. O'Reilly, „What Is Web 2.0 - Design Patterns and Business Models for the Next Generation of Software“, in *Online Communication and Collaboration: A Reader*, Routledge, 2010, S. 225–235.

[7] I. Gouy, *The Computer Language Benchmarks Game*. Adresse: http://benchmarksgame.alioth.debian.org/.

[8] C. Jones, „Function Points as a Universal Software Metric“, *ACM SIGSOFT Software Engineering Notes*, Bd. 38, Nr. 4, S. 1–27, 2013.

[9] International Standard ISO/IEC, „Software and systems engineering – Software measurement – IFPUG functional size measurement method“, Nr. 20926, 2009.

[10] Y. Shoham, „An Overview of Agent-Oriented Programming“, in *Software Agents*, MIT Press, 1997, S. 271–286.

[11] N. R. Jennings, „On agent-based software engineering“, *Artificial Intelligence*, Bd. 117, Nr. 2, S. 277–296, 2000.

[12] G. Weiß, „Agent Orientation in Software Engineering“, *The Knowledge Engineering Review*, Bd. 16, Nr. 4, S. 349–373, 2001.

[13] Z. Zhang, „An Agent-Based Hybrid Framework for Decision Making on Complex Problems“, Dissertation, Deakin University, 2001.

[14] T. W. Malone und K. Crowston, „The Interdisciplinary Study of Coordination“, *ACM Computing Surveys*, Bd. 26, Nr. 1, S. 87–119, 1994.

[15] R. Rico, M. Sánchez-Manzanares, F. Gil und C. Gibson, „Team Implicit Coordination Processes: A Team Knowledge-Based Approach“, *The Academy of Management Review*, Bd. 33, Nr. 1, S. 163–184, 2008.

[16] H. Kargermann, W. Wahlster und J. Helbig, *Umsetzungsempfehlungen für das Zukunftsprojekt Industrie 4.0*. Lyoner Straße 9, 60528 Frankfurt/Main, 2013.

[17] International Standard ISO/IEC, „Software engineering – Software product Quality Requirements and Evaluation (SQuaRE)“, Nr. 25000, 2007.

[18] X. Zhang, M. Mu, G. Mao und W. Zhang, „Performance of Grid-Based PDE.Mart“, in *IEEE International Conference on e-Science and Grid Computing*, 2006, S. 108–115.

[19] E. D. Sturler, J. Hoeflinger, L. Kale und M. Bhandarkar, „A New Approach to Software Integration Frameworks for Multi-physics Simulation Codes“, in *The Architecture of Scientific Software*, Springer, 2001, S. 87–104.
[20] S. Weerawarana, E. N. Houstis, J. R. Rice, A. Joshi und C. E. Houstis, „PYTHIA: A Knowledge-Based System to Select Scientific Algorithms“, *ACM Transactions on Mathematical Software*, Bd. 22, Nr. 4, S. 447–468, 1996.
[21] T. T. Drashansky, A. Joshi, J. R. Rice, E. N. Houstis und S. Weerawarana, „A Multi-Agent Environment for MPSEs“, Technical Report, Purdue University, 1996.
[22] L. Davis und G. Williams, „Evaluating and Selecting Simulation Software Using the Analytic Hierarchy Process“, *Integrated Manufacturing Systems*, Bd. 5, Nr. 1, S. 23–32, 1994.
[23] R. Rabenseifner, G. Hager und G. Jost, „Hybrid MPI/OpenMP Parallel Programming on Clusters of Multi-Core SMP Nodes“, in *Conference on Parallel, Distributed and Network-Based Processing*, 2009, S. 427–436.
[24] J. Geiser, *Decomposition Methods for Differential Equations: Theory and Applications*. CRC Press, 2009.
[25] M. Santasusana Isach, „Continuum modelling using the Discrete Element Method. Theory and implementation in an object-oriented software platform.“, Dissertation, Escola de Camins UPC BarcelonaTECH, 2012.
[26] M. Armbrust, I. Stoica, M. Zaharia, A. Fox, R. Griffith u. a., „A View of Cloud Computing“, *Communications of the ACM*, Bd. 53, Nr. 4, S. 50–58, 2010.
[27] R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg und I. Brandic, „Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility“, *Future Generation Computer Systems*, Bd. 25, Nr. 6, S. 599–616, 2009.
[28] A. Iosup, S. Ostermann, N. Yigitbasi, R. Prodan, T. Fahringer u. a., „Performance Analysis of Cloud Computing Services for Many-Tasks Scientific Computing“, *IEEE Transactions on Parallel and Distributed Systems*, Bd. 22, Nr. 6, S. 931–945, 2011.
[29] T. Carroll und J. Russell, „Why HPC is Absolutely Ready for the Cloud“, in *HP-Cwire Soundbite*, 120234456, 8445 Camino Santa Fe, Suite 101 San Diego: Tabor Communications Inc., 2014.
[30] C. B. Ries, *BOINC - Hochleistungsrechnen mit Berkeley Open Infrastructure for Network Computing*. Springer, 2012.
[31] University of California, *BOINC, Open-source software for volunteer computing*, 2017. Adresse: http://boinc.berkeley.edu/.
[32] P. Göhner, *Agentensysteme in der Automatisierungstechnik*. Springer, 2013.
[33] K. M. Muscholl, „Interaktion und Koordination in Multiagentensystemen“, Dissertation, Universität Stuttgart, 2001.
[34] G. Weiss, *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*. MIT Press, 1999.
[35] J. Odell, „Objects and Agents Compared“, *Journal of Object Technology*, Bd. 1, Nr. 1, S. 41–53, 2002.
[36] L. Braubach, „Architekturen und Methoden zur Entwicklung verteilter agentenorientierter Softwaresysteme“, Dissertation, Universität Hamburg, 2008.
[37] M. Wooldridge und N. R. Jennings, „Intelligent Agents: Theory and Practice“, *The Knowledge Engineering Review*, Bd. 10, Nr. 2, S. 115–152, 1994.
[38] A. S. Rao und M. P. Georgeff, „BDI Agents: From Theory to Practice“, in *International Conference on Multi-Agent Systems*, 1995, S. 312–319.

[39] M. Jüttner, A. Buchau, D. Vögeli, W. M. Rucker und P. Göhner, „Iterative Software Agent Based Solution of Multiphysics Problems“, in *Scientific Computing in Electrical Engineering*, Springer, 2016, S. 123–131.
[40] F. Bellifemine, A. Poggi und G. Rimassa, „JADE - A FIPA - compliant agent framework“, in *International Conference on the Practical Application of Intelligent Agents and Multi-Agent Technology*, 1999, S. 97–108.
[41] C. Wang, F. Mueller, C. Engelmann und S. L. Scott, „Proactive Process-Level Live Migration in HPC Environments“, in *ACM/IEEE Conference on Supercomputing*, 2008.
[42] I. Wachsmuth, „Mensch-Maschine-Interaktion“, in *Handbuch Kognitionswissenschaft*, J. B. Metzler, 2013, S. 361–364.
[43] Foundation For Intelligent Physical Agents, „Communicative Act Library Specification“, Nr. SC00037J, 2002.
[44] Foundation For Intelligent Physical Agents, „ACL Message Structure Specification“, Nr. SC00061G, 2002.
[45] E. H. Durfee, „Distributed Problem Solving and Planning“, in *Multi-Agent Systems and Applications*, Springer, 2001, S. 118–149.
[46] J. M. Swaminathan, S. F. Smith und N. M. Sadeh, „Modeling Supply Chain Dynamics: A Multiagent Approach“, *Decision Sciences*, Bd. 29, Nr. 3, S. 607–631, 1998.
[47] I. Badr, F. Schmitt und P. Göhner, „Integrating Transportation Scheduling with Production Scheduling for FMS: An Agent-Based Approach“, in *IEEE International Symposium on Industrial Electronics*, 2010, S. 3539–3544.
[48] P. Vrba, „JAVA-Based Agent Platform Evaluation“, in *Holonic and Multi-Agent Systems for Manufacturing*, Springer, 2003, S. 47–58.
[49] M. T. Huda, H. W. Schmidt und I. D. Peake, „An Agent Oriented Proactive Fault-tolerant Framework for Grid Computing“, in *IEEE International Conference on e-Science and Grid Computing*, 2005, S. 304–311.
[50] R. M. Jones und R. E. Wray, „Intelligent Agents“, in *Artificial Intelligence: A Modern Approach*, Prentice-Hall, 1995, S. 31–52.
[51] H. S. Nwana, L. Lee und N. R. Jennings, „Co-ordination in software agent systems“, *British Telecom Technical Journal*, Bd. 14, Nr. 4, S. 79–89, 1996.
[52] M. N. Huhns und L. M. Stephens, „Multiagent Systems and Societies of Agents“, in *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*, MIT Press, 1999, S. 79–120.
[53] V. Marik und D. McFarlane, „Industrial Adoption of Agent-Based Technologies“, *Computer - Intelligent Systems*, Bd. 20, Nr. 1, S. 27–35, 2005.
[54] C. Poulter, „Kooperation in Multiagentensystemen“, Diplomarbeit, Universität Hamburg, Fakultät für Mathematik, Informatik und Naturwissenschaften, 2007.
[55] J. F. Hübner, J. S. Sichman und O. Boissier, „A Model for the Structural, Functional, and Deontic Specification of Organizations in Multiagent Systems“, in *Symposium on Articicial Intelligence*, 2002, S. 118–128.
[56] E. Steegmans, K. Schelfthout, T. Holvoet, Y. Berbers, P. Valckenaers u. a., „A Basic Taxonomy for Role Composition“, in *Software Engineering for MultiAgent Systems II*, Springer, 2004, S. 92–110.
[57] F. Hohl, „Sicherheit in Mobile-Agenten-Systemen“, Dissertation, Universität Stuttgart, 2001.
[58] M. J. Wooldridge und N. R. Jennings, „Pitfalls of Agent-Oriented Development“, in *ACM International Conference on Autonomous Agents*, ACM, 1998, S. 385–391.

[59] C. Nikolai und G. Madey, „Tools of the Trade: A Survey of Various Agent Based Modeling Platforms“, in *Journal of Artificial Societies and Social Simulation*, 2, Bd. 12, 2009.
[60] R. Trillo, S. Ilarri und E. Mena, „Comparison and Performance Evaluation of Mobile Agent Platforms“, in *IEEE International Conference on Autonomic and Autonomous Systems*, 2007, S. 41–46.
[61] F. Bellifemine, G. Caire und D. Greenwood, *Developing Multi-Agent with JADE Systems*. John Wiley & Sons, 2007, Bd. 7.
[62] Telecom Italia S.p.A., *JADE: JAVA Agent DEvelopment Framework*, Via G. Reiss Romoli 274, 10148 Torino. Adresse: http://jade.tilab.com/.
[63] M. Berner, „Entwurf und Implementierung eines Managementsystems für Softwareagenten“, Bachelorarbeit, Universität Stuttgart, Institut für Theorie der Elektrotechnik, 2015.
[64] M. Nikraz, „Agent-Based Integration of Operational Tasks in Chemical Plants“, Dissertation, Murdoch University, 2007.
[65] M. Nikraz, G. Caire und P. a. Bahri, „A Methodology for the Development of Multi-Agent Systems using the JADE Platform“, *International Journal of Computer Systems Science and Engineering*, Bd. 21, Nr. 2, S. 99–116, 2006.
[66] D. Sanchez, D. Isern, A. Rodriguez-Rozas und A. Moreno, „Agent-based platform to support the execution of parallel tasks“, *Expert Systems with Applications*, Bd. 38, Nr. 6, S. 6644–6656, 2011.
[67] K. Chmiel, D. Tomiak, M. Gawinecki, P. Karczmarek, M. Szymczak u. a., „Testing the Efficiency of JADE Agent Platform“, in *IEEE International Symposium on Parallel and Distributed Computing*, 2004, S. 49–56.
[68] E. Cortese, F. Quarta und G. Vitaglione, „Scalability and Performance of JADE Message Transport System“, *Journal of Analysis of Suitability for Holonic Manufacturing Systems*, Bd. 3, Nr. 3, S. 52–65, 2002.
[69] J. Al-Jaroodi, N. Mohamed, H. Jiang und D. R. Swanson, „Agent-Based Parallel Computing in Java Proof of Concept“, Technical Report, University of Nebraska, 2001.
[70] N. Vyas, „Cloud Based Software Agents for High Performance Services“, Masterarbeit, Universität Stuttgart, Institut für Theorie der Elektrotechnik, 2016.
[71] M. Kaltenbacher, *Numerical Simulation of Mechatronic Sensors and Actuators*. Springer, 2015, Bd. 3.
[72] J. Fetzer, „Die Lösung statischer und quasistatischer elektromagnetischer Feldprobleme mit Hilfe der Kopplung der Methode der finiten Elemente und der Randelementmethode“, Dissertation, Universität Stuttgart, 1995.
[73] I. N. Bronštejn und G. Musiol, *Taschenbuch der Mathematik*. Harri Deutsch, 2005.
[74] W. Hackbusch, „Theorie und Numerik elliptischer Differentialgleichungen“, Vorlesungsskript, Max-Planck-Institut für Mathematik in den Naturwissenschaften, 2005.
[75] T. Damm, „Berechnung hochfrequenter 3-dimensionaler elektromagnetischer Felder mittels der Finite Elemente Methode auf Kantenbasis“, Dissertation, Technische Universität Berlin, 2000.
[76] A. Pflug, „Implementierung und Bewertung unterschiedlicher Multiphysikkopplungen für verschiedene Diskretisierungen“, Masterarbeit, Universität Stuttgart, Institut für Theorie der Elektrotechnik, 2016.
[77] A. Huerta, A. Huerta, A. Rodriguez-Ferran, A. Rodriguez-Ferran, P. Diez u. a., „Adaptive finite element strategies based on error assessment“, *International Journal for Numerical Methods in Engineering*, Bd. 46, Nr. 10, S. 1803–1818, 1999.

[78] J. R. Shewchuk, „What Is a Good Linear Finite Element? Interpolation, Conditioning, Anisotropy, and Quality Measures“, *International Meshing Roundtable*, S. 115–126, 2002.
[79] T. Erhart, „Strategien zur numerischen Modellierung transienter Impaktvorgänge bei nichtlinearem Materialverhalten“, Dissertation, Universität Stuttgart, 2004.
[80] J. R. Gilbert, C. Moler und R. Schreiber, „Sparse Matrices in MATLAB: Design and Implementation“, *Siam Journal on Matrix Analysis and Applications*, Bd. 13, Nr. 1, S. 333–356, 1992.
[81] H. Werkle, *Finite Elemente in der Baustatik*. Vieweg, 2008.
[82] F. Rapetti, „Analyse Numérique“, Vorlesungsskript, Université de Nice Sophia-Antipolis, Laboratoire de Mathématiques J.A. Dieudonné, 2011.
[83] B. Markert, „Weak or strong: On coupled problems in continuum mechanics“, Habilitation, Universität Stuttgart, 2010.
[84] K. Hameyer, J. Driesen, H. D. Gersem und R. Belmans, „The Classification of Coupled Field Problems“, *IEEE Transactions on Magnetics*, Bd. 35, Nr. 3, S. 1618–1621, 1999.
[85] O. Zienkiewicz und R. Taylor, *The Finite Element Method: Volume 1 The Basis*. Butterworth-Heinemann, 2000.
[86] B. Uekermann, H. Bungartz, B. Gatzhammer und M. Mehl, „A Parallel, Black-Box Coupling Algorithm for Fluid-Structure Interaction“, in *International Conference on Computational Methods for Coupled Problems in Science and Engineering*, 2013.
[87] J. R. Cash, „Efficient numerical methods for the solution of stiff initial-value problems and differential algebraic equations“, *Royal Society of London A: Mathematical, Physical and Engineering Sciences*, Bd. 459, Nr. 2032, S. 797–815, 2003.
[88] K. Strehml, R. Weiner und H. Podhaisky, *Numerik gewöhnlicher Differentialgleichungen - Nichtsteife, steife und differential-algebraische Gleichungen*. Springer, 2012.
[89] M. Wick, M. Jüttner und W. M. Rucker, „Fourier Transform based Analysis of Harmonic Nonlinear Magnetic Fields“, in *International Symposium on Electric and Magnetic Fields*, 2016.
[90] M. Kolmbauer, „The Multiharmonic Finite Element and Boundary Element Method for Simulation and Control of Eddy Current Problems“, Dissertation, Johannes Kepler Universität Linz, 2012.
[91] F. Bachinger, U. Langer und J. Schöberl, „Numerical analysis of nonlinear multiharmonic eddy current problems“, *Numerische Mathematik*, Bd. 100, Nr. 4, S. 593–616, 2005.
[92] S. Yamada, P. P. Biringer und K. Bessho, „Calculation of Nonlinear Eddy-Current Problems by the Harmonic Balance Finite Element Method“, *IEEE Transactions on Magnetics*, Bd. 27, Nr. 5, S. 4122–4125, 1991.
[93] G. Lehner, *Elektromagnetische Feldtheorie*. Springer, 2008.
[94] S. Grabmaier, „Adaptive Netzverfeinerung und Anfangswertbestimmung für ein Agentensystem“, Masterarbeit, Universität Stuttgart, Institut für Theorie der Elektrotechnik, 2014.
[95] M. Jüttner, A. Pflug, M. Wick und W. M. Rucker, „Experimental Evaluation of Numerical Errors for Multiphysics Coupling Methods using Disparate Meshes“, in *International IGTE Symposium on Numerical Field Calculation in Electrical Engineering*, 2016.
[96] A. D. Boer, A. H. V. Zuijlen und H. Bijl, „Comparison of Conservative and Consistent Approaches for the Coupling of Non-Matching Meshes“, *Computer Methods in Applied Mechanics and Engineering*, Bd. 197, Nr. 49, S. 4284–4297, 2008.

[97] R. K. Jaiman, X. Jiao, P. H. Geubelle und E. Loth, „Conservative load transfer along curved fluid-solid interface with non-matching meshes“, *Journal of Computational Physics*, Bd. 218, Nr. 1, S. 372–397, 2006.
[98] J. Geiser, *Coupled Systems: Theory, Models, and Applications in Engineering*. CRC Press, 2014.
[99] M. J. Gander, „Schwarz Methods in the Course of Time“, *Electronic Transactions on Numerical Analysis*, Bd. 31, Nr. 5, S. 228–255, 2008.
[100] Y. Saad, *Iterative Methods for Sparse Linear Systems*. Society for Industrial und Applied Mathematics, 2000, Bd. 2.
[101] A. Toselli und O. Widlund, *Domain Decomposition Methods — Algorithms and Theory*. Spring, 2005.
[102] T. Mathew, *Domain Decomposition Methods for the Numerical Solution of Partial Differential Equations*. Springer, 2008.
[103] E. M. Okrouhl, „Numerical methods in computational mechanics“, Dissertation, The Czech Academy of Sciences, 2012.
[104] J. Fijnvandraat, S. Houben, E. ter Maten und J. Peters, „Time domain analog circuit simulation“, *Journal of Computational and Applied Mathematics*, Bd. 185, Nr. 2, S. 441–459, 2006.
[105] P. Zhou, D. Lin, W. N. Fu, B. Ionescu und Z. J. Cendes, „A General Cosimulation Approach for Coupled Field-Circuit Problems“, *IEEE Transactions on Magnetics*, Bd. 42, Nr. 4, S. 1051–1054, 2006.
[106] S. Schöps, „Multiscale Modeling and Multirate Time-Integration of Field/Circuit Coupled Problems“, Dissertation, Bergische Universität Wuppertal & Katholieke Universiteit Leuven, 2011.
[107] Q. Fang, Z. Nie und J. Hu, „Computational Electromagnetics“, Vorlesungsskript, Dartmouth College, Hanover, 2003.
[108] J. Steindorf, „Partitionierte Verfahren für Probleme der Fluid-Struktur Wechselwirkung“, Dissertation, Universität Braunschweig, 2002.
[109] V. Rischmüller, „Eine Parallelisierung der Kopplung der Methode der finiten Elemente und der Randelementmethode“, Dissertation, Universität Stuttgart, 2004.
[110] S. Grabmaier, M. Jüttner, W. M. Rucker und P. Göhner, „Numerical Framework for the Simulation of Dielectric Heating using Finite and Boundary Element Method“, *International Symposium on Electric and Magnetic Fields*, 2016.
[111] S. Grabmaier, H. Li, M. Jüttner und W. M. Rucker, „Efficient Magnetic Field Calculation based on a novel Domain Decomposition Approach“, in *International IGTE Symposium on Numerical Field Calculation in Electrical Engineering*, 2016.
[112] M. Cervera, R. Codina und M. Galindo, „On the computational efficiency and implementation of block-iterative algorithms for nonlinear coupled problems“, *Engineering Computations*, Bd. 13, Nr. 6, S. 4–30, 1996.
[113] H. G. Matthies, R. Niekamp und J. Steindorf, „Algorithms for strong coupling procedures“, *Computer Methods in Applied Mechanics and Engineering*, Bd. 195, Nr. 17, S. 2028–2049, 2006.
[114] A. Meister, *Numerik linearer Gleichungssysteme*. Springer, 2008.
[115] The MathWorks Inc., *MATLAB Documentation*, Adalperostraße 45, 85737 Ismaning, 2014.
[116] R. Barrett, M. Berry, T. F. Chan, J. Demmel, J. Donato u. a., *Templates for the solution of linear systems: Building blocks for iterative methods*. SIAM, 1994, S. 118.

[117] D. Lahaye, „Algebraic Multigrid as Solvers and as Preconditioner“, Vorlesungsskript, Katholieke Universiteit Leuven, 2005.
[118] C.-K. Cheng, „Computer Aided Circuit Simulation and Verification“, Vorlesungsskript, University of California, 2013.
[119] C. T. Kelley, *Iterative Methods for Linear and Nonlinear Equations*. SIAM, 1995.
[120] A. H. van den Boogaard, A. D. Rietman und J. Huetink, „Iterative solvers in forming process simulations“, in *Simulation of Materials Processing: Theory, Methods and Applications*, 1998, S. 219–224.
[121] V. Rajaraman, *Computer Oriented Numerical Methods*. PHI Learning, 1993.
[122] J. Falk, „Entwicklung eines neuronalen Netzwerks zur Empfehlung von Lösungsverfahren für lineare Gleichungssysteme“, Forschungsarbeit, Universität Stuttgart, Institut für Theorie der Elektrotechnik, 2015.
[123] T. George, „Recommendation System for Preconditioning Iterative Solvers“, Dissertation, Texas A&M University, 2009.
[124] A. Yeckel, L. Lun und J. J. Derby, „An approximate block Newton method for coupled iterations of nonlinear solvers: Theory and conjugate heat transfer applications“, *Journal of Computational Physics*, Bd. 228, Nr. 23, S. 8566–8588, 2009.
[125] S. Sicklinger, V. Belsky, Engelmann B., H. Elmqvist, H. Olsson u. a., „Interface Jacobian-based Co-Simulation“, *International Journal for Numerical Methods in Engineering*, Bd. 98, Nr. 6, S. 418–444, 2014.
[126] C. a. Felippa, K. C. Park und C. Farhat, „Partitioned Analysis of Coupled Mechanical Systems“, *Computer Methods in Applied Mechanics and Engineering*, Bd. 190, Nr. 24-25, S. 3247–3270, 2001.
[127] C. Förster, W. A. Wall und E. Ramm, „Artificial added mass instabilities in sequential staggered coupling of nonlinear structures and incompressible viscous flows“, *Computer Methods in Applied Mechanics and Engineering*, Bd. 196, Nr. 7, S. 1278–1293, 2007.
[128] C. Farhat und M. Lesoinne, „Two efficient staggered algorithms for the serial and parallel solution of three-dimensional nonlinear transient aeroelastic problems“, *Computer Methods in Applied Mechanics and Engineering*, Bd. 182, S. 499–515, 2000.
[129] C. Förster, W. A. Wall und E. Ramm, „The Artificial Added Mass Effect in Sequential Staggered Fluid-Structure Interaction Algorithms“, in *European Conference on Computational Fluid Dynamics*, 2006.
[130] M. Busch, „Zur effizienten Kopplung von Simulationsprogrammen“, Disseration, Universität Kassel, 2012.
[131] M. Trčka, J. L. M. Hensen und M. Wetter, „Co-simulation for performance prediction of integrated building and HVAC systems - An analysis of solution characteristics using a two-body system“, *Simulation Modelling Practice and Theory*, Bd. 18, Nr. 7, S. 957–970, 2010.
[132] H. Liu und J. Rao, „Coupled modeling of electromagnetic-thermal problem in induction heating process considering material properties“, in *IEEE International Conference on Information Engineering and Computer Science*, 2009.
[133] M. A. Fernández, J. F. Gerbeau und C. Grandmont, „A projection semi-implicit scheme for the coupling of an elastic structure with an incompressible fluid“, *Journal for Numerical Methods in Engineering*, Bd. 69, Nr. 4, S. 794–821, 2007.
[134] L. Quiroz und P. Beckers, „Non Conforming Mesh Gluing in the Finite Elements Method“, *International Journal for Numerical Methods in Engineering*, Bd. 38, S. 2165–2184, 1995.

[135] A. Buchau, M. Jüttner und W. M. Rucker, „Automatic domain detection for a meshfree post- processing in boundary element methods“, in *International IGTE Symposium on Numerical Field Calculation in Electrical Engineering*, Graz, 2012, S. 386–391.
[136] S. Piperno und C. Farhat, „Design of Efficient Partitioned Procedures for the Transient Solution of Aeroelastic Problems“, *Revue Européenne Eléments Finis*, Bd. 9, Nr. 6-7, S. 655–680, 2000.
[137] B. Patzák, D. Rypl und J. Kruis, „MuPIF-A distributed multi-physics integration tool“, *Advances in Engineering Software*, Bd. 60-61, S. 89–97, 2013.
[138] M. Brenk, H. Bungartz, M. Mehl und T. Neckel, „Fluid-Structure Interaction on Cartesian Grids: Flow Simulation and Coupling Environment“, in *Fluid-Structure Interaction*, Springer, 2006, S. 233–269.
[139] D. Gaston, C. Newman, G. Hansen und D. Lebrun-Grandié, „MOOSE: A parallel computational framework for coupled systems of nonlinear equations“, *Nuclear Engineering and Design*, Bd. 239, Nr. 10, S. 1768–1778, 2009.
[140] M. G. Hackenberg, M. G. Hackenberg, P. Post, P. Post, R. Redler u. a., „MpCCI, Multidisciplinary Applications, and Multigrid“, in *European Congress on Computational Methods in Applied Sciences and Engineering*, 2000.
[141] T. Blochwitz, M. Otter, M. Arnold, C. Bausch, C. Clauß u. a., „The Functional Mockup Interface for Tool independent Exchange of Simulation Models“, in *International Modelica Conference*, 2011.
[142] J. Ruben, „Ein Multi-Agenten-System zur verteilten p-adaptiven Finite-Elemente-Simulation am Beispiel der Baugrund-Tragwerk-Interaktion“, Dissertation, Technischen Universität Darmstadt, 2005.
[143] M. Müller, „Ein Intra-Grid-System für die p-adaptive Finite-Elemente-Simulation auf Arbeitsplatzrechnern am Beispiel der Geotechnik“, Dissertation, Universität Darmstadt, 2006.
[144] M. Jüttner, A. Buchau und W. M. Rucker, „Software Agent Based Solution of Segregated Multiphysics Problems with Varying Study Types“, in *International Workshop on Finite Elements for Microwave Engineering*, 2014.
[145] J. Schlichte, „Grundlagen: Betriebssysteme und Systemsoftware (GBS)“, Vorlesungsskript, Technische Universität München, 2011.
[146] P. Vrba, „Java Based Agent Platform Evaluation“, in *Holonic and Multi-Agent Systems for Manufacturing*, Springer, 2003, S. 47–58.
[147] EURESCOM Participants in Project P907, „MESSAGE: Methodology for Engineering Systems of Software Agents“, Project Report, 2001.
[148] K.-H. Krempels, J. Nimis, L. Braubach, R. Herrler und A. Pokahr, „Entwicklung intelligenter Multi-Multiagentensysteme - Werkzeugunterstützung, Lösungen und offene Fragen“, in *Jahrestagung der Gesellschaft für Informatik*, 2003, S. 31–46.
[149] J. Villacis, „A Note on the Use of Java in Scientific Computing“, *Association for Computing Machinery - Special Interest Group on Applied Computing*, Bd. 7, Nr. 1, S. 14–17, 1999.
[150] J. Al-Jaroodi, N. Mohamed, H. Jiang und D. R. Swanson, „An Overview of Parallel and Distributed Java for Heterogeneous Systems: Approaches and Open Issues“, *Scalable Computing: Practice and Experience*, Bd. 5, Nr. 4, 2002.
[151] M. Jüttner, A. Buchau, M. Rauscher, W. M. Rucker und P. Göhner, „Software Agent Based Domain Decomposition Method“, in *International IGTE Symposium on Numerical Field Calculation in Electrical Engineering*, 2012, S. 89–94.

[152] A. Komus, „Status Quo Agile 2016 / 2017“, Abschlussbericht, Hochschule Koblenz, BPM-Labor für Business Process Management und Organizational Excellence, 2017.
[153] A. Uresin und M. Dubois, „Effects Of Asynchronism On The Convergence Rate Of Iterative Algorithms“, *Journal of Parallel and Distributed Computing*, Bd. 34, S. 66–81, 1996.
[154] F. Diener, „Entwicklung eines Softwareagenten zur Darstellung der Prozessparameter numerischer Simulationen“, Bachelorarbeit, Universität Stuttgart, Institut für Theorie der Elektrotechnik, 2014.
[155] C. Farhat, M. Lesoinne, P. Letallec, K. H. Pierson und D. Rixen, „FETI-DP: A dual-primal unified FETI method part I: A faster alternative to the two-level FETI method“, *International Journal for Numerical Methods in Engineering*, Bd. 50, Nr. 7, S. 1523–1544, 2001.
[156] M. Geimer, T. Krüger und P. Linsel, „Co-Simulation, gekoppelte Simulation oder Simulatorkopplung? Ein Versuch der Begriffsvereinheitlichung“, *O+P Zeitschrift für Fluidtechnik*, Nr. 50, S. 572–576, 2006.
[157] C. Schweikert, „Ein lernendes, rekursives Empfehlungssystem zur interdisziplinären Simulationskopplung“, Bachelorarbeit, Universität Stuttgart, Institut für Theorie der Elektrotechnik, 2017.
[158] E. Amet, „Entwurf und Implementierung eines Empfehlungssystems zur interdisziplinären Simulationskopplung“, Masterarbeit, Universität Stuttgart, Institut für Theorie der Elektrotechnik, 2016.
[159] A. Wierse, „Performance of the COVISE visualization system under different conditions“, in *Visual Data Exploration and Analysis II*, Bd. 218, International Society for Optics und Photonics, 1995, S. 218–229.
[160] M. Jüttner, S. Grabmaier und W. M. Rucker, „Web Based 3D Visualization for COMSOL Multiphysics“, in *COMSOL Conference*, 2014.
[161] A. Naga und Z. Zhang, „A Posteriori Error Estimates Based on the Polynomial Preserving Recovery“, *SIAM Journal on Numerical Analysis*, Bd. 42, Nr. 4, S. 1780–1800, 2004.
[162] A. M. Heinecke, *Mensch-Computer-Interaktion*. Springer, 2012, S. 386.
[163] DIN Deutsches Institut für Normung e.V., „Ergonomie der Mensch-System-Interaktion – Grundsätze der Dialoggestaltung“, Nr. 9241-110, 2008.
[164] Microsoft Corporation, „Windows User Experience Interaction Guidelines“, Redmond, WA 98052-6399, USA, 2010.
[165] J. Holler, V. Tsiatsis, C. Mulligan, S. Avesand, S. Karnouskos u. a., *From Machine-to-Machine to the Internet of Things: Introduction to a New Age of Intelligence*. Academic Press, 2014, S. 352.
[166] M. Jüttner, S. Grabmaier, D. Vögeli, W. M. Rucker und P. Göhner, „Coupled Multiphysics Problems as Market Place for Competing Autonomous Software Agents“, *IEEE Transactions on Magnetics*, Bd. 53, Nr. 6, 2017.
[167] A. Maglo, G. Lavoué, F. Dupont und C. Hudelot, „3D Mesh Compression: Survey, Comparisons, and Emerging Trends“, *Association for Computing Machinery - Computing Surveys*, Bd. 47, Nr. 3, 44:1–41, 2015.
[168] A. Pflug, „Kompressionsverfahren zur schnellen Datenübertragung numerischer Simulationsergebnisse“, Forschungsarbeit, Univeristät Stuttgart, Institut für Theorie der Elektrotechnik, 2014.

[169] A. Snyder, „Encapsulation and Inheritance in Object-Oriented Programming Languages“, *Association for Computing Machinery - Special Interest Group on Programming Languages*, Bd. 21, Nr. 11, S. 38–45, 1986.
[170] M. Hinze, R. Pinnau, M. Ulbrich und S. Ulbrich, *Optimization with PDE Constraints*. Springer, 2009.
[171] H. Anzt, „Asynchronous and Multiprecision Linear Solvers“, Dissertation, Karlsruher Instituts für Technologie, 2012.
[172] *Umsetzungsstrategie Industrie 4.0: Ergebnisbericht der Plattform Industrie 4.0*. BITKOM e.V.,VDMA e.V. und ZVEI e.V., 2015.
[173] O. Röhrle, „Höhere Mechanik: Numerische Methoden in der Mechanik“, Vorlesungsskript, Universität Stuttgart, 2010.
[174] T. Huckle, „Dünnbesetzte Matrizen: Algorithmen und Datenstrukturen“, Vorlesungsskript, Technische Universität München, 2005.
[175] J. P. Whiteley, K. Gillow, S. J. Tavener und A. C. Walter, „Error bounds on block Gauss Seidel solutions of coupled multiphysics problems“, *International Journal for Numerical Methods in Engineering*, Bd. 88, Nr. 11, S. 1219–1237, 2011.
[176] B. Malakooti, *Operations and Production Sytems with Multiple Objectives*. John Wiley & Sons, 2013.
[177] A. Frommer und D. B. Szyld, „Asynchronous two-stage iterative methods“, *Numerische Mathematik*, Bd. 96, Nr. 2, S. 141–154, 1994.
[178] D. P. Bertsekas und J. N. Tsitsiklis, „Convergence Rate and Termination of Asynchronous Iterative Algorithms“, in *ACM International Conference on Supercomputing*, 1989, S. 461–470.
[179] N. Köckler, *Mehrgittermethoden*. Springer, 2012.
[180] M. Jüttner, A. Buchau, A. Faul, W. M. Rucker und P. Göhner, „Segregated Parallel and Distributed Solution of Multiphysics Problems using Software Agents“, in *IEEE International Conference on Electromagnetic Field Computation*, Annecy, 2014.
[181] W. H. Press, S. A. Teukolsky, W. T. Vetterling und B. P. Flannery, *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press, 1996.
[182] W. Hackbusch, „Iterative Lösung großer Gleichungssysteme“, Vorlesungsskript, Max-Planck-Institut für Mathematik in den Naturwissenschaften, 2004.
[183] A. Frommer, H. Schwandt und D. B. Szyld, „Asynchronous Weighted Additive Schwarz Methods“, *Electronic Transactions on Numerical Analysis*, Bd. 5, S. 48–61, 1997.
[184] R. Tezaur, „Analysis of a Lagrange Multiplier Based Domain Decomposition“, Dissertation, University of Colorado at Denver, 1998.
[185] M. Dorr, „On the Discretization of Interdomain Coupling in Elliptic Boundary-value Problems“, in *Domain Decomposition Methods*, SIAM, 1989, S. 17–37.
[186] D. Preuveneers und Y. Berbers, „ACODYGRA: An Agent Algorithm for Coloring Dynamic Graphs“, in *International Symposium on Symbolic and Numeric Algorithms for Scientific Computing*, 2004, S. 381–390.
[187] A. Frommer und D. B. Szyld, „On asynchronous iterations“, *Journal of Computational and Applied Mathematics*, Bd. 123, Nr. 1-2, S. 201–216, 2000.
[188] M. Rauschnabel, „Implementierung der Gebietszerlegungsmethode FETI-DP für elektrische Potenzialprobleme“, Masterarbeit, Universität Stuttgart, Institut für Theorie der Elektrotechnik, 2015.
[189] B. Uekermann, B. Gatzhammer und M. Mehl, „Coupling Algorithms for Partitioned Multi-Physics Simulations“, *GI - Lecture Notes in Informatics*, Bd. 232, S. 113–124, 2014.

[190] A. de Boer, A. H. van Zuijlen und H. Bijl, „Review of coupling methods for non-matching meshes“, *Computer Methods in Applied Mechanics and Engineering*, Bd. 196, Nr. 8, S. 1515–1525, 2007.
[191] N. Ramakrishnan, E. N. Houstis und J. R. Rice, „Recommender Systems for Problem Solving Environments“, in *Association for the Advancement of Artificial Intelligence Workshop on Recommender Systems*, MIT Press, 1998, S. 91–95.
[192] R. Ewald, J. Himmelspach und A. Uhrmacher, „An Algorithm Selection Approach for Simulation Systems“, in *IEEE Workshop on Principles of Advanced and Distributed Simulation*, 2008, S. 91–98.
[193] M. Jüttner, J. Falk und W. M. Rucker, „A Neural Network based Recommendation System for Solvers and Preconditioners for Systems of Linear Equations“, *IEEE Transactions on Magnetics*, Bd. 53, Nr. 6, 2017.
[194] D. Kessler, „Inkrementelles Training eines Neuronalen Netzes für lineare Gleichungssysteme“, Bachelorarbeit, Universität Stuttgart, Institut für Theorie der Elektrotechnik, 2016.
[195] K.-J. Bathe, *Finite Element Procedures*. Klaus-Jurgen Bathe, 2006.
[196] H. Vietz, „Entwicklung eines Algorithmus zur Entscheidung zwischen der Evaluation von Lösungsverfahren und der Netzverfeinerung“, Bachelorarbeit, Universität Stuttgart, Institut für Theorie der Elektrotechnik, 2015.
[197] J.-C. Charr, R. Couturier und D. Laiymani, „A decentralized and fault tolerant convergence detection algorithm for asynchronous iterative algorithms“, *The Journal of Supercomputing*, Bd. 53, Nr. 2, S. 269–292, 2009.
[198] G. Calin, E. Derevenetc, R. Majumdar und R. Meyer, „A Theory of Partitioned Global Address Spaces“, in *Conference on Foundations of Software Technology and Theoretical Computer Science*, Bd. 24, 2013, S. 127–139.
[199] O. F. Rana und L. Moreau, „Issues in Building Agent-Based Computational Grids“, in *Workshop of the UK Special Interest Group on Multi-Agent Systems*, 2000.
[200] R. Aversa, B. Di Martino, N. Mazzocca und S. Venticinque, „MAGDA: A Mobile Agent based Grid Architecture“, *Journal of Grid Computing*, Bd. 4, Nr. 4, S. 395–412, 2006.
[201] L. Chunlin und L. Layuan, „Agent framework to support the computational grid“, *Journal of Systems and Software*, Bd. 70, Nr. 1–2, S. 177–187, 2004.
[202] J. Cao, S. A. Jarvis, S. Saini, D. J. Kerbyson und G. R. Nudd, „ARMS: An Agent-Based Resource Management System for Grid Computing“, *Scientific Programming*, Bd. 10, Nr. 2, S. 135–148, 2002.
[203] K. Weicker, *Evolutionäre Algorithmen*. Springer, 2015, Bd. 3.
[204] W. J. Ellison, „Permittivity of pure water, at standard atmospheric pressure, over the frequency range 0-25 THz and the temperature range 0-100C“, *Journal of Physical and Chemical Reference Data*, Bd. 36, Nr. 1, S. 1–18, 2007.
[205] A. Asenov, A. Brown, B. Cheng, X. Wang, N.Daval u. a., „FinFETs“, in *IEEE International Conference on Simulation of Semiconductor Processes and Devices (SISPAD)*, Glasgow, 2013.
[206] F. Thuselt, *Physik der Halbleiterbauelemente*. Springer, 2011, Bd. 2.
[207] B. Troyanovsky, „Frequency Domain Algorithms For Simulating Large Signal Distortion in Semiconductor Devices“, Dissertation, Stanford University, 1997.
[208] J. Rohloff, „Berechnung zeitabhängiger Simulationen mit Softwareagenten“, Masterarbeit, Universität Stuttgart, Institut für Theorie der Elektrotechnik, 2016.

Online-Buchshop für Ingenieure

VDI nachrichten

BUCHSHOP

Online-Shops

Fachliteratur und mehr - jetzt bequem online recherchieren & bestellen unter: www.vdi-nachrichten.com/Der-Shop-im-Ueberblick

Täglich aktualisiert: Neuerscheinungen VDI-Schriftenreihen

Im Buchshop von vdi-nachrichten.com finden Ingenieure und Techniker ein speziell auf sie zugeschnittenes, umfassendes Literaturangebot.

Mit der komfortablen Schnellsuche werden Sie in den VDI-Schriftenreihen und im Verzeichnis lieferbarer Bücher unter 1.000.000 Titeln garantiert fündig.

Im Buchshop stehen für Sie bereit:

VDI-Berichte und die Reihe **Kunststofftechnik:**

Berichte nationaler und internationaler technischer Fachtagungen der VDI-Fachgliederungen

Fortschritt-Berichte VDI:

Dissertationen, Habilitationen und Forschungsberichte aus sämtlichen ingenieurwissenschaftlichen Fachrichtungen

Newsletter „Neuerscheinungen":

Kostenfreie Infos zu aktuellen Titeln der VDI-Schriftenreihen bequem per E-Mail

Autoren-Service:

Umfassende Betreuung bei der Veröffentlichung Ihrer Arbeit in der Reihe Fortschritt-Berichte VDI

Buch- und Medien-Service:

Beschaffung aller am Markt verfügbaren Zeitschriften, Zeitungen, Fortsetzungsreihen, Handbücher, Technische Regelwerke, elektronische Medien und vieles mehr – einzeln oder im Abo und mit weltweitem Lieferservice

Die Reihen der Fortschritt-Berichte VDI:

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

ISBN 978-3-18-3**46920**-8