Systemorganisation und Subsysteme verteilter paralleler Simulationsumgebung. Parallele blockorientierte Simulationssprache: ein Entwicklungskonzept

Автор: Prof. Dr.-Ing. V.A. Svjatnyj.
Источник: Virtuelle parallele Simulationsmodelle und ein Devirtualisierungsvorgang der Entwicklung von parallelen Simulatoren für dynamische Netzobjekte mit verteilten Parametern. Vortrag am HLRS, Universität Stuttgart – 2010.

Inhaltsverzeichnis

1. Vorwort

Im Rahmen der wissenschaftlichen Zusammenarbeit von HLRS der Universität Stuttgart und FCWT (früher FRTI) der Nationalen technischen Universität Donezk werden sowohl allgemeine als auch die konkrete gegenstandsgebietbezogene Probleme paralleler

Simulationstechnik gelöst. Während der gemeinsamen Seminare am HLRS und an der FRTI (2006, 2007, 2008, 2009) wurden die Aufgabenstellungen, parallele Algorithmen und Parallelelisierungsansätze, vorhandene und perspektive Zielrechnerarchitekturen sowie HW/SW-Entwicklungen der problemorintierten parallelen Simulationsumgebungen diskutiert. Seit 2006 wird Thema „Virtuelle parallele Simulationsmodelle und ein Devirtualisierungsvorgang der Entwicklung von parallelen Simulatoren für dynamische Netzobjekte mit verteilten Parametern“ als Bestandteil der parallelen Simulationstechnik bearbeitet, die Implementierungsarbeiten werden fortgesetzt. Aktuelle Probleme der Aufbau von parallelen Simulatoren sind mit der Systemorganisation der verteilten parallelen Simulationsumgebung (VPSU) und Erhöhung ihrer Benutzerfreundlichkeit verbunden. Die erhaltene Ergebnisse der Zusammenarbeit wurden im ASIM-Vortrag „Forschungsgebiet: parallele Simulationstechnik“ (Feldmann L., Resch M., Svjatnyj V., Zeitz M.) zusammengefasst und haben deutliche Interesse bei den ASIM-Teilnehmern hervorgerufen. Neben den neuen erzielten Lösungen ist zu erwähnen, dass an der FCWT die Forschung und Lehre ins Gebiet „Parallele Simulationstechnik“ (ParSimTech) eine Gruppe von Nachwuchswissenschaftler geschafft wurde. Die junge Aspirantinnen und Aspiranten, Magistranden der deutschsprachigen Gruppe „Systemprogrammierung“ aktiv diese Problematik bearbeiten. Fakultät hat als humanitäre Hilfe vom HLRS den Cluster NEC 5800 im November 2009 erhalten und beschäftigt sich zusammen mit dem DonNTU-Rechenzentrum mit der Organisation der effizienten Nutzung des Clusters in Lehre und Forschung. Analyse der weltweiten HPC-Aktivitäten und aktueller Initiative an der Universität Stuttgart „SimTech-Cluster of Excellence“ hat gezeigt, dass für die Ukraine aktuell ist, schon erzielte Erfahrungen der obigen Zusammenarbeit für die Modernisierung der Lehre und Forschung in mehreren technischen Universitäten aktiver benutzen. Organisatorisch wäre sinnvoll diese Modernisierung im Rahmen des TEMPUS-Projektes zu verwirklichen.

Im vorliegenden Bericht werden die Ergebnisse des Forschungsaufenthaltes von Prof. Dr.-Ing. V. Svjatnyj am HLRS (01.02. – 31.03.2010) zur obigen Problematik sowie die Vorschläge zur möglichen Beantragung des TEMPUS-Projektes dargestellt.

2. Zur Nutzung der VPSU in der Systemorganisation von ParSimTech-Kompetenzzentren der ukrainischen technischen Universitäten

2.1 Motivation und Zielsetzung

Die Simulationstechnik steht für eine neue Art von Wissenschaft und hat zunehmende Bedeutung für die Industrie: die modellgestützte Projektentwicklung und simulationsbasierte frühzeitige Erkennung möglicher Projektfehlern sind die Entscheidungsfaktoren der Qualitätssicherung von Industrieprojekten und der Lebenszyklusdauer von Industrieprodukten. Die steigenden Industrieanforderungen an immer genauere und damit auch komplexere Simulationsmodelle haben andererseits intensive Entwicklung von neuen Simulationswerkzeugen hervorgerufen, die neue Ansätze zur Modellierung und Simulation industrieller Objekte anbieten und interdisziplinäres aktuelles Forschungsgebiet – parallele Simulationstechnik (ParSimTech) – bilden. Die Ukraine will den Übergang ihrer Wirtschaft vom sekundären Sektor in den tertiären vorantreiben, um sich der EU anzunähern. Es ist

notwendig, diesen Wandel zu unterstützen. Dafür ist die parallele Simulationstechnik in der ukrainischen Industrie (Flugzeugbau und Raumfahrt, Bergbautechnologien und Grubenbewetterung, IT-Dienste, Gastransportsysteme, Hüttenwesen und automatisierte Walzwerke, Chemische Verfahrenstechnik etc.) ein wesentliches Instrument. Derzeit besteht aber ein Mangel an der ParSimTech-Ausbildung. Dies liegt vor allem daran, dass zum einen das notwendige Know-how in der Simulation auf großen Rechenanlagen nicht vorhanden ist. Zum anderen ist die Infrastruktur nur eingeschränkt in der Lage, die Ausbildung zu unterstützen. Das für die parallele Simulation notwendige Know-how ist in der europäischen Union verbreitet. Andererseits ist die Ukraine ein Land, in dem traditionell eine sehr gute und sehr fundierte Ausbildung auf dem Gebiet der mathematischen Modellbildung stattfindet.

Auf dieser starken theoretischen Basis wollen wir ***das Problem – die ParSimTech in die Ausbildung zu bringen*** – allerseitig im Tempus-Projekt zu lösen. Das Ziel ist der Know-how Transfer im Bereich der parallelen Simulation, der aufbauend auf den theoretischen Grundlagen die Qualität der Lehre in der Ukraine auf diesem Gebiet nachhaltig verbessern kann. Die Nachhaltigkeit soll dadurch gesichert werden, dass der Know-how Transfer in der Lehre umgesetzt wird, und damit in die Studienpläne der beteiligten Universitäten Einzug hält.

Zur Zeit hat dieses Problem deutliche Prioritätsbedeutung für ukrainisches und EU-Hochschulwesen aus folgenden Gründen: ParSimTech wird mit den Studiengänge Computer Science und Computer Engineering (CS/CE) integriert, ist aber auch mit den theoretischen Grundlagen der Modellbildung in allen anderen Studiengängen (die Welt ist parallel!) eng verbunden; ParSimTech hat hervorragendes Kognitionspotential für die Lehregestaltung auf europäischer Ebene; ParSimTech bringt die industrienahen Methoden in der Lehre, aktiviert weltweit die Wissenstransfer zwischen den Universitäten, Forschungszentren und Firmen; ParSimTech-HW/SW-Realisierungen sind mit den IT-Dienste der Universitäten integriert.

Diese Projektproblemstellung ist eng mit den Tätigkeiten der Partnern verbunden und entspricht euren Entwicklungsstrategien: HLRS modernisiert planmäßig und generationsweise die HPC-Ressource, führt regelmäßig die Schulungen für die Benutzern verschiedener Institutionen sowie Fächer, bearbeitet die HPC-Projekte weltweiter Niveau, gehört zu den führenden HPC-Institutionen in Europa; beide weitere EU-Partnern haben solide Erfahrung in ParSimTech-Entwicklungen und Anwendungen, planen weitere HPC-Arbeiten gerade für betrachtete Projektperiode; für ukrainischen Partnern bedeutet dieser Projektvorschlag eine Zusammenfassung euren vorigen ParSimTech-Arbeiten und Realisierung neuen Ideen auf EU-Ebene im Rahmen des Bologna-Prozesses.

Die Problemstellung wird ***in drei Zielsektoren (ZS) und zehn Zielgruppen konkretisiert.*** **ZS-1** „Modernisierung der Lehre in den CS/CE-Studiengängen“ beinhaltet zwei Zielgruppen: 1) Modernisierung der Studienpläne, um ParSimTech-Spezialisierung der Bachelor/Master-CS/CE-Studiengänge proto-typisch zu gestalten; 2) Modernisierung der Lehrinhalte und Lehrunterlagen: es muss festgelegt werden, was und wie unterrichtet wird; es müssen die Module der Vorlesungen und notwendigen Unterlagen (Skripte, Folien, Bücher, Experimente etc.) für die Lehre erarbeitet werden.

ZS-2 „Modernisierung der Lehrinfrastruktur“ hat vier Zielgruppen: 1) Modernisierung von ParSimTech-Hardware mit der Installation entsprechender System- und Anwendersoftware; 2) Modernisierung der Laboratorien: als zentrale Einrichtung soll

verteilte parallele Simulationsumgebung (VPSU) installiert werden; die Clusters und PC-Pools der Partnern zusammen mit VPSU bilden den Universitätsnetz von ParSimTech-Kompetenzzentren; 3) Einsatz der parallelen CS/CE-Trainingssimulatoren; 4) Ausbildung der Lehrenden vor Ort und an den EU-Universitäten.

ZS-3 „Nachhaltige Ausdehnung der Projektergebnisse“ mit den Zielgruppen: 1) Entwicklung der problemorientierten Simulationssoftware und Installation der PC-Fakultätspools als VPSU-Anwendern; 2) ParSimTech- und fachorientierte Schulung der Nachwuchswissenschaftlern und Lehrenden als VPSU-Benutzern; 3) Entwicklung der Simulations- und Servicezentren für Fakultätsfachbereiche; 4) Anwendung der ParSimTech-Ansätze zur modellgestützten Projektierung und simulationsbasierten Qualitätssicherung der Industrieprojekte.

Profite vom Projekt: die ParSimTech-modernisierte Lehre und installierte VPSU mit ParSImTech- Kompetenzzentren als Lehrinfrastruktur in der Ukraine, die mit EU-Partnern integriert sind; die ersten Absolventen spezialisiertes Studienganges am Ende des dritten Projektjahres; nachhaltig erweiterte ParSimTech-Anwendung an den Technologiefakultäten; minimal möglichen Projektkosten wegen Nutzung vorhandener HW/SW-Ressourcen und Erfahrungen.

2.2 Aufbau der notwendigen Infrastruktur aus Hardware/Software, VPSU und Labors

Lehrinfrastruktur soll genau der Modernisierungsaufgaben bezüglich der für Studiengang zuständigen Lehrstühle entsprechen und Hierarchieaspekte der Installation von Ressourcen für Kompetenzzentren im Rahmen vorhandenen Universitätsrechenzentren und IT-Diensten berücksichtigen.

AP4.1 Partnerebene: Liste der vorhandenen und noch notwendigen Ressourcen

P4 – DonNTU (P4 ist Partnernummer im TEMPUS-Formular, P1 – P3 sind EU-Partnern)

P4.1 Cluster NEC 5800, 100 Knoten – Hardware wird im Serverraum 4.19S (DonNTU-Gebäude 4, am 1.Stock) installiert, zur Systemsoftware möglicherweise die Fragen entstehen, die von HLRS beantwortet werden können. Materiell und systemsadministrativ wird Cluster P1.1 dem DonNTU-Rechenzentrum zugeordnet, methodisch wird vom Lehrstuhl für Computer Engineering betreut.

P4.2 Studenten Pool, 2 Räume mit je 13 und 15 PC-Arbeitsplätze, territoriell mit Cluster P1.1 integriert. Raum-13PC (Lab.4.19) wird vorrangig in etwa April 2010 an Cluster eingeschaltet werden, um für deutschsprachige Gruppen Laborunterrichten, experimentelle Teile von Studien- und Masterarbeiten zu ermöglichen. Weitere 15PCs (Lab.4.20) werden Zugang zum Cluster etwa ab Juni 2010 bekommen. *WICHTIG: beide Räume haben Anschlüsse zum Lehrstuhlsrechnernetz und Internet, deshalb notwendige Arbeiten inhaltlich gehören zu den Systemadministrationsaufgaben.*

P4.3 Cluster von parallel funktionierten 12 PCs (Lab.4.18, territoriell neben P.1.2 – 4.19, 4.20) – zum RZ zugeornet, methodisch wird vom Lehrstuhl für angewandte Mathematik und Informatik betreut. Systemsoftware wurde von Microsoft geliefert, Firma hat auch die Schulung durchgeführt.

P4.4 Serverressource bei dem DonNTU-IT-Dienst für Zugriff zu den entfernten parallelen HLRS-Ressourcen (zuständige Systemadministratoren sind Absolventen des

Lehrstuhls für Computer Engineering D. Nadeev und O. Rytschka, alle Systemformalitäten sind geregelt)

P4.5 Notwendige Hardware wird aus folgenden Gesichtspunkten betrachtet:

- als Ersatz von obigen 28 PCs (Baujahr 2003) auf neue Modelle möglichst mit Multicore- Bauart (aktuelle Konfigurationen in 2012);
- Modernisierung des Clusters durch Hardware von CUDA-Ansätze (aktuelle Architekturen in 2012) in Richtung der hybriden Clusterknoten;
- Einrichtung des ParSimTechKompNetz-Servers am DonNTU-IT-Dienst mit dem Ziel – effizient der im Projekt geplanten Aktivitäten mit ZNTU und KhNURE zu realisieren, die nachhaltige Projektaufgaben für Hochschullehre in der Ukraine erfolgreich zu lösen;
- Einrichtung von Video-Konferenzsaal (Raum 5.427) für Präsentationen der ParSimTech-Aktivitäten und Ergebnissen, sowie als modernen FCWT-Hörsaal für Vorlesungen (100 Sitzplätze).

P4.6 Notwendige Software wird aus folgenden Gesichtspunkten betrachtet:

- Bereitstellung der Systemsoftware für neue 28PCs (Windows, Linux, Stand 2012);
- als Anwendersoftware für simulationstechnisch orientierten Lehre sehr wichtig ist die Programmpakete Mathematica, Matlab und konventionellen Simulationssprachen SIMULINK, DYMOLA,VHDL, UML, GPSS-Nachfolger (ARENA mit ISSOP-Optimisator u.a. aktuelle Produkte im 2012) an den obigen 28PCs zu haben;
- für SimTech-spezialisierten Bachelorstudiengang – Programmsysteme für Studium von Programmiersprachen C, C++, Java (u.a. aktuelle in 2012);
- für parallele Programmierung – MPI-, OpenMP-Bibliotheken, möglicherweise für hybrides Programmiermodell (MPI+OpenMP); Programmpakete für Performance Analyse u.a. nach Empfehlungen von HLRS; Software für CUDA-Programmiermodelle;
- für parallele Simulation – Software für die Visualisierung und Animation der Simulationsergebnissen; möglicherweise gibt es eine Bedienoberfläche für Cluster NEC 5800 (BOF, GUI – aktuelle Frage schon heute, 14.02.10, in Zusammenhang mit Inbetriebnahme des Clusters)
- Systemsoftwareunterstützung der VPSU-Struktur „HLRS-ParRessource – Zugriffsserver – INTERNET – DonNTU-ParSim-Server – ClusterAdmin/Arbeitsplätze – ClusterKnoten“ Systemsoftwareunterstützung der VPSU-Struktur im Abschnitte DonNTU – ZNTU, DonNTU – KhNURE , ZNTU – KhNURE (oder die direkten Strecken HLRS – ZNTU, HLRS – KhNURE ? Besonders wird für Forschungsaufgaben von Magistranden und Doktoranden aktuell sein). Es soll im Installationsplan beantwortet werden

P4.7 Die notwendigen Lehreunterlagen

- Lehrbücher und Monographieliteratur (Deutsch und Englisch);
- Einige Zeitschriften und Tagungsbände (im Druckschrift und in E-Form);
- Zugang zu den INTERNET-Quellen (zur ProjektZeit und nachhaltig);
- DEMO-Versionen von Simulationssprachen, Simulatoren, Gleichungslöser etc. (für Studierenden – wie etwas simulationstechnisches als Produkt aussehen und funktionieren soll)
- u. a. – Thema für Besprechungen: was wir selbst entwickeln werden, was von Dissertationsergebnissen angewendet werden kann usw.

P4.8 Strukturdarstellung des ParSimTechKompetenzNetzes

Projekt entwickelte Infrastruktur. Ziel ist schon heute mit der Dokumentation anfangen, um ganz deutlich vor Projektbeantragung alles darzustellen. Als Kern soll verteilte parallele Simulationsumgebung mit vorhandenen Hardware und Software sein. Dann kann man im Sinne TEMPUS-Projektes ergänzen und erweitern.

Die HLRS-Ressource und Zugänge von Benutzer. Die Aufbau der notwendigen Infrastruktur ist eine komplexe Systemlösung, die Ziele des ZS-2 „Modernisierung der Lehrinfrastruktur“ verfolgt und wird in folgenden TEMPUS-Teilprojekten (TTP) gegliedert, die unter Führung der erfahrener Wissenschaftler mit der Teilnahme als Projektentwicklern von Assistenten, Aspiranten und Studenten der Bachelor- und Masterstudiengänge erfüllen werden:

- TTP1 – Entwicklung der VPSU-Subsysteme und Inbetriebnahme der parallelen Simulationssoftware (Projekttermin 01.09.2010 – 31.12.2011, Verteidigung der Masterarbeiten) – zeitlich entspricht der Vorbereitungs- und Startphase des TEMPUS-Projektes, spezialisiert erste 10 Absolventen als ParSimTech-Magistern, endet sich mit der Inbetriebnahme der VPSU-Simulationssoftware als praktischer Beitrag für die Installation der Infrastruktur;
- TTP2 – Entwicklung der VPSU-Komponenten und ihren Einsatz in den ParSimTech-Kompetenzzentren (Projekttermin 01.09.2011 – 31.12.2012, Verteidigung der Masterarbeiten) – wird im 1.Projektjahr erfüllt, spezialisiert 10 ParSimTech-Magistern, bringt Beiträge zur Integration der VPSU mit den ParSimTech-Kompetenzzentren;
- TTP3 – Systemorganisation der ParSimTech-Kompetenzzentren an den Partneruniversitäten (Projekttermin 01.09.2011 – 31.12.2013, Verteidigung der Masterarbeiten) – soll in den 1. und 2.Projektjahren von den Wissenschaftlern und Studenten der Partneruniversitäten bearbeitet werden, es spezialisiert weitere ParSimTech-Magistern und endet sich mit der Inbetriebnahme der HW/SW-Ressourcen von integrierten ParSimTech-Kompetenzzentren;
- TTP4 – Entwicklung der problemorientierten verteilten parallelen Simulationsumgebungen (POVPSU) für Gegenstandsgebiete „Grubenbewetterung“ und „Verfahrenstechnik“ (Projekttermin 01.09.2012 – 31.12.2014, Verteidigung der Masterarbeiten) – wird in den 2. und 3.Projektjahren von den Wissenschaftlern und Studenten der DonNTU bearbeitet, spezialisiert weitere ParSimTech-Magistern und im Mai 2014 stellt den Fakultäten für Bergbauwesen und Chemische Verfahrenstechnik die benutzerfreundliche ParSimTech-Ressourcen zur Verfügung. Seit 01.09.2014 werden die Wissenschaftlern und Studenten-Magistranden dieser Fakultäten im TTP4 als POVPSU-Benutzern und Cliente des ParSimTech-Kompetenzzentrums teilnehmen;
- TTP5 – Erprobung der neuen Kursen und methodischen Unterlagen in der Infrastrukturumgebung (Projekttermin 01.10.2012 – 31.12.2014) – wird von den Wissenschaftlern und Studenten der Partneruniversitäten bearbeitet; die Erprobung wird nach individuelle Ausarbeitungspläne in den dem Studienplan entsprechenden Semestern durchgeführt;
- TTP6 – Modernisierung und Installation neuer Laboratorien von Partneruniversitäten – soll in den 1. und 2.Projektjahren von den Wissenschaftlern, Ingenieurkräfte und Studenten der Partneruniversitäten erfüllt werden, um möglichst früher die Erprobungsarbeiten beginnen zu können. Dabei sollen aus den Kostengründen die Arbeitsplätze einen Zugang zu den örtlich nahen und entfernten parallelen Ressourcen haben.

- TTP7 – Erarbeitung der fachbezogenen Trainingssimulatoren (TS) aufgrund der problemorientierten parallelen Simulationsumgebungen, vorhandenen Laboratorien der Fakultäten und Erfahrungen von HLRS Uni Stuttgart und FCWT NTU Donezk – soll schon im 3.Projektjahr von den Wissenschaftlern, Ingenieurkräfte und Studenten der CS/CE- und Technologiefakultäten der Partneruniversitäten erfüllt werden, um möglichst früher die Nachhaltungsarbeiten beginnen zu können. Dabei sollen aus den Kostengründen die TS-Arbeitsplätze der Technologiefakultäten einen Zugang zu den Ressourcen der Kompetenzzentren haben.
- TTP8 – Aufbau des ParSimTech-spezialisierten Videokonferenz-Hörsaals – soll im 2.Projektjahr erfüllt und mit dem TTP3-Ablauf integriert werden.
- TTP9 – Aufbau und weitere Administrierung des ParSimTech-spezialisierten Web-Portals – soll im 1.Projektjahr erfüllt und mit den TTP-Abläufen integriert werden.

3. Entwicklung der Subsysteme von verteilten parallelen Simulationsumgebung und Inbetriebnahme der parallelen Simulationssoftware (TTP1)

3.1 VPSU-Funktionalität

Die verteilte parallele Simulationsumgebung (VPSU) ist eine benutzerfreundliche simulationstechnische Systemorganisation der parallelen Hardware, der architekturrelevanten Systemsoftware, der speziell zielgerichtet entwickelten Simulationssoftware sowie der modellierungs-, simulations- und systemtechnisch bedingten Subsysteme, mit denen alle Etappen der parallelen Modellierung und Simulation von komplexen dynamischen Systemen mit konzentrierten (DSKP) und verteilten (DSVP) Parametern mit der möglichst vollen Berücksichtigung der aktuellen Anforderungen unterstützt werden. Die VPSU-Funktionalität, die dieser Definition entspricht, wird mit der Abbildungen 3.1, 3.2 verdeutlicht.

Die VPSU-Entwicklung wird als zentrales Problem der parallelen Simulationstechnik angesehen und von FCWT und HLRS nach der folgenden Konzeption durchgeführt:

- Berücksichtigung der aktuellen DSKP/VP-Anforderungen an die Simulationsmittel;
- Nutzung der vorhandenen und perspektiven parallelen Ressourcen von MIMD-Architekturen als verteilten Rechensysteme;
- Entwicklung der funktionsvollständigen Simulationssoftware für MIMD-Hardware der VPSU entsprechend den DSKP/VP-Anforderungen;
- Möglichst breite Verwendung von denselben Lösungsalgorithmen für DSKP- und ortsdiskretisierten DSVP;
- Parallele Simulationssoftware mit ähnlichen Eigenschaften wie bei den vorhandenen block-, objekt- und gleichungsorientierten Simulationssprachen;
- Problemorientierte VPS-Umgebungen werden als Teilprojekte betrachtet, die allgemeine Lösungen nutzen und Ergebnisse für mögliche Verallgemeinerungen liefern;
- VPSU wird als komplexes HW/SW-SYSTEM mit der umfangreichen Funktionalität (Abb. 3.1, 3.2) betrachtet, deren Entwicklung soll auf dem OO-Ansatz mit der UML-Technologie sowie mit der Dekomposition auf bestimmte Menge der Subsysteme erfolgen. Solche Organisation der VPSU-Entwicklung erlaubt nebenläufige Bearbeitung der

Teilprojekte (Entwicklungsaufgaben) und eine Erhaltung der Zwischenergebnisse, die zu den arbeitsfähigen Versionen der Simulationsumgebung führen.

Im vorgeschlagenen TEMPUS-Teilprojekt TTP1soll diese Konzeption aufgrund der vorhandenen und perspektiven Zielrechnerarchitekturen realisiert werden.

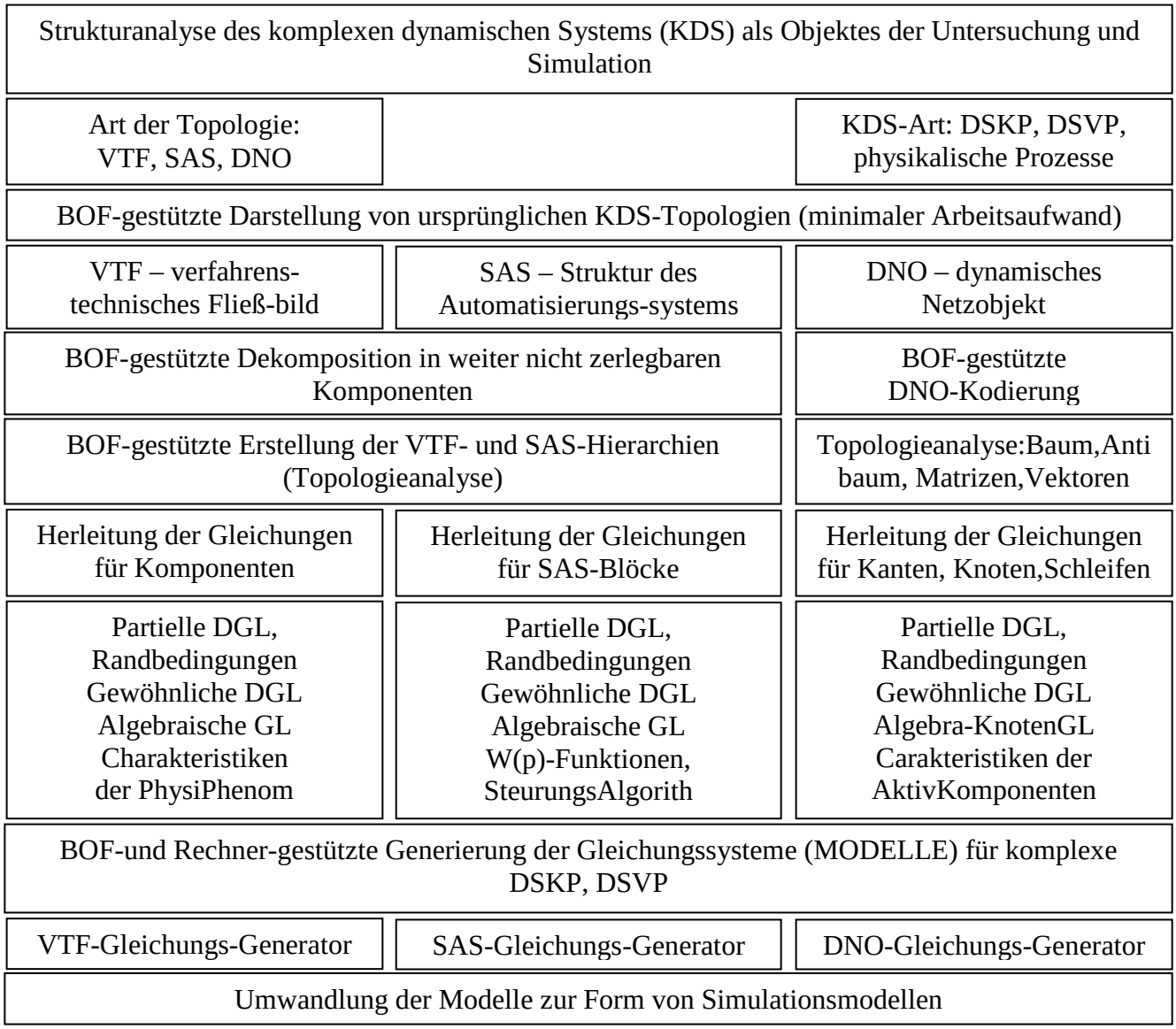

Abb. 3.1.VPSU-Funktionalität: Rechner- und BOF-gestützte Entwicklung der DSKP-, DSVP-Modelle und –Simulationsmodelle

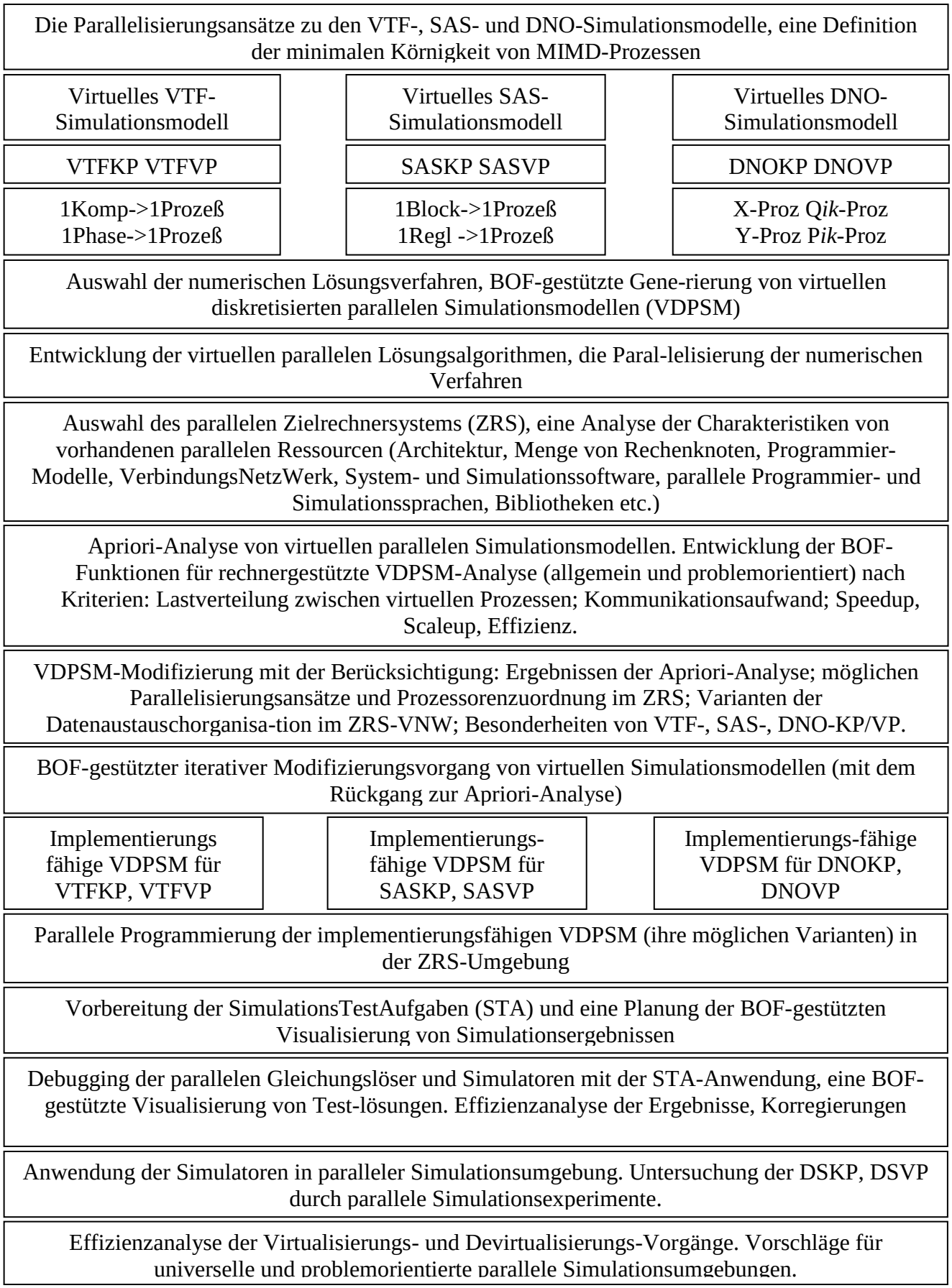

Abb. 3.2. VPSU-Funktionalität: Rechner- und BOF-gestützte Entwicklung von parallelen DSKP-, DSVP-SIMULATOREN, virtuelle parallele Simulationsmodelle und die Devirtualisierungsvorgänge

3.2 Vorläufige Liste und Funktionalität der VPSU- Subsysteme

Das Problem der vollfunktionellen Entwicklung der VPSU-Simulationssoftware wird mit Hilfe der Verteilung der gesamten Funktionalität zwischen den Subsystemen gelöst.

Das VPSU-Subsystem ist eine Teilkomponente der Hardware, Systemsoftware und Simulationssoftware, die benutzerfreundlich eine gewisse Gruppe von inhaltlich nahen Funktionen auf entsprechenden parallelen DSKP/DSVP-Modellierungs- und Simulationsetappen (Abb.3.1, 3.2) realisiert.

1. Dialogsubsystem (DiSuS, die Benutzeroberfläche – BOF) – eine Präsentation der VPSU; Schulung von Benutzern; aktives Dialog der Modellentwickler mit alle Ressourcen des verteilten Rechnersystems; Integration mit alle anderen Subsysteme; Anpassung an die Simulationsaufgaben der Gegenstandsgebiete; Planung und Führunung der Simulationsexperimente

2. **Topologieanalysesubsystem** (TASuS) – verbale und grafische Darstellung sowie geeignete Kodierung von ursprünglichen VTF-, SAS-, DNO-Topologien; Dekomposition und Aproximierung, Erstellung der sekundären Topologien;Formierung der topologierelevanten Vektoren und Matrizen; Umwandlung der ursprünglichen Kodierungstabellen in die Zwischenformen; Darstellung der topologischen Informationseinheiten in der für die Gleichungsgenerierung geeigneten Form; eine Ausgabe der Topologieanalyseergebnissen.

3. Subsysten der Gleichungsgenerierung (SuGG) – Kommunikation mit dem TASuS; Darstellung der ursprünglichen Gleichungen (Modelle) in Vektormatrixform; formale Vektor-Matrix-Operationen der Umwandlung „DSKP-Modell – Simulationsmodell“; Darstellung der approximierten DSVP-Gleichungen (1D-, 2D-, 3D-ortsdiskretisierten Modelle) in der Multivektor-Matrix-Form; die formale Umwandlung „DSVP-Modell – Simulationsmodell“; die Generierung von diskreten DSKP-, DSVP-Simulationsmodelle für gegebenes numerisches Lösungsverfahren; die Visualisierung der Modelle und Simulationsmodelle.

4. Subsystem der virtuellen parallelen Simulationsmodelle (VPSMSu) – interaktive Darstellung der Hierarchie von virtuellen parallelen Simulationsmodelle abhängig von möglichen Parallelisierungsansätze; TASuS-gestützte Erstellung der Topologie entsprechend den VPSM-Ebenen; SuGG-unterstützte Formierung der VPSM-Ebene Gleichungssysteme; Apriori-Analyse der diskreten VPSM aller Ebenen; interaktive Bereitstellung von implementierungsfähigen diskreten VPSM.

5. Subsystem von parallelen Gleichungslöser (SuPaGLö) – Kommunikation mit TASuS und VPSMSu, die Eingabe der implementierungsfähigen diskreten VPSM; die Lösung von algebraischen, gewöhnlichen und partiellen Differentialgleichungssysteme mit Hilfe der in den DSKP-, DSVP-Bibliotheken zusammengefassten parallelen numerischen Verfahren; die Analyse der Konvergenz, Stabilität und Genauigkeit der Lösungen, Optimierung der veränderlichen Parametern; Abschätzung und Optimierung der Effizienzmerkmalen von parallelen Lösungen in Vergleich mit den sequentiellen Verfahren; Formierung der Lösungsergebnisse für die anschauliche Präsentation.

6. **Datenaustauschsubsystem** (DASu) – die vollfunktionelle Liste der angemeldeten VPSU-Teilnehmern und vorhandenen VPSU-Ressourcen, die ein Datenaustausch nach gemeinsame Initiative durchführen sollen; eine Hierarchiedarstellung von Datenströme für

ausgewählten ZRS; Testsystem für die Ermittlung der realen Parametern von Austauschoperationen im ZRS-Verbindungsnetzwerk; Optimierung der parallelen Programme bezüglich der Datenaustauschoperationen; Zusammenstellung der in der Apriori-Analyse und im ZRS erhaltene Wirkung der DA-Operationen auf die Werte der Effizienz von Parallelisierungsansätzen.

7. **Lastbalancierungssubsystem** (LaBSu) **–** Abschätzung der Lasthöhe von virtuellen Prozessen in VPSM-Ebenen, statische Lastbalancierung der VPSM aller Ebenen; Spezifikation der Aufträge auf VPSU-Ressorcen von implementierten VPSM; eine Ermittlung der Lastverteilung zwischen Prozessen und zwischen ZRS-Prozessoren; Vergleichsanalyse der Parallelisierungsansätze nach Kriterium der gleichmäßigen Lastverteilung.

8. **Visualisierungssubsystem** (ViSuS) – in Integration mit BOF und Subsystem von Gleichungslösern eine Vorbereitung und Strukturierung der Simulationsergebnissen zu den Formen, die für grafischen 1D-, 2D- und 3D-Visualisierung geeignet sind; interaktive Erstellung von Grafiken während und nach der Durchführung von Simulationsexperimenten

9. Datenbanksubsystem (DaBSu) – in der Integration mit allen anderen Subsystemen: Daten über VPSU-Hardware und Systemsoftware; Benutzerdaten; ursprüngliche und umgewandelte Daten der modellierten DSKP/DSVP von VTF-, SAS- und DNO- Topologien; Archivierung der lauffähigen parallelen und sequentiellen Programme; Daten der Testaufgaben, Pläne der Simulationsexperimente, Archivierung der Simulationsergebnissen.

10. Subsystem der IT-Unterstützung (SuIT) – Betriebsorganisation von verteilten Rechen-, Kommunikations- und Simulationsressourcen durch die Technologien der modernen Netzwerke; entfernter WEB-basierter Ansatz zur parallelen Modellierung und Simulation; Funktionierung der Ketten „WEB-Client (Modellentwickler) – WEB-Server – PARSIMULATOR“ und „WEB-Client (Modellentwickler) – WEB-Server – DATABASE-Server“; Integration mit anderen Subsystemen.

3.3 Allgemeine Methodik der Entwicklung von VPSU-Subsysteme

Dekomposition des komplexen Systems auf bestimmte Menge der Subsysteme ist ein wertvoller Entwicklungsansatz, der den Entwicklern erlaubt, komplexes Problem als Teillösungen darzustellen, ohne gesamte Systemeigenschaften und Funktionen zu verpassen oder nicht zu berücksichtigen. Dabei spielen eine wichtige Rolle neben der Dekomposition selbst die Prozesse der Entwicklungskoordination, des Projektmanagements und der Komposition der Einheiten von höheren Ebenen der Parallelität aus erhaltenen Dekompositionslösungen.

Es wird von uns VPSU als komplexes HW/SW-System betrachtet und folgende Entwicklungsmethodik vorgeschlagen.

1. Problemstellung – wird auf der Ebene des Projektmanagements formuliert.

1.1. Es sind die oben genannten 10 VPSU-Subsysteme zu entwickeln, zu implementieren und auf den Testsimulationsaufgaben experimentell zu untersuchen.

1.2. VPSU-Hardwarestruktur ist in Abb.2.2 dargestellt und soll in Integration mit der Infrastruktur des ParSimTech-Kompetenzzentrums der FCWT DonNTU (Abb.2.1) betrachtet werden.

1.3. Die Projektlösungen sollen als unabhängige von Hardwareplattform konzipiert werden und erst bei der Implementierung die Charakteristiken von Ressource-Komponenten berücksichtigen.

2. Spezifikation der Funktionen des j-Subsystems (j=1,2,…10) – wird vom Entwickler des j-Subsystems erstellt und mit den Entwicklern von i-Subsysteme (i /= j) sowie mit dem Projektleiter koordiniert.

2.1 Verbale Spezifikation.

2.2 Grafische Darstellung von Funktionszusammenhängen.

2.3 Definition der Verbindungen mit den anderen Subsystemen.

2.4 Formale Beschreibung der Funktionen.

3. Anforderungen – werden von Subsystementwickler mit der Berücksichtigung der Anforderungen an Simulationsmitten [27], Subsystemfunktionen sowie VPSU-Entwicklungskonzept formuliert und mit den Entwicklern der anderen Subsysteme sowie mit dem Projektleiter vereinbart.

4. Beschreibung des OO-Ansatzes zur Entwicklung.

5. Beschreibung des Hardware-Teils.

6. Beschreibung des Systemsoftware-Umgebung, BOF/(GUI)-Funktionen.

7. Entwicklung des Simulationssoftware-Teils.

7.1 Entwicklung und Spezifikation der Funktionsalgorithmen.

7.2 UML-Diagrammen der Subsystem-Simulationssoftware.

7.3 Vereinbarungen mit den anderen Subsystemen.

7.4 Vorbereitung der Funktionstestaufgaben für jede Funktion.

7.5 Vorbereitung der Subsystem-Testaufgaben für Prüfung der Subsystems-Simulationssoftware.

7.6 Implementierung und Prüfung der Funktionsalgorithmen in der gewählten Systemsoftware-Umgebung (siehe P.6).

7.7 Implementierung und Prüfung der modellierungs-und simulationsspezifischen BOF/(GUI)-Funktionsalgorithmen.

7.8 Prüfung der Subsystem-Simulationssoftware aufgrund der Subsystem-Testaufgaben.

8. Entwicklung der subsystemübergreifenden TEST-Modellierung-Simulation-Aufgabe – wird auf der Ebene des Projektmanagements formuliert und zusammen mit TTP1-Vollzieher entwickelt.

8.1 Als Modellierungs- und Simulationstest für subsystemübergreifende Prüfung der entwickelten Subsysteme (VPSU-TEST) wird die Aufgabe „Parallele Modellierung und Simulation des dynamischen Netzobjektes mit verteilten Parametern (DNOVP)“ vorgeschlagen.

8.2 Es soll aufgrund der Berichte zum HLRS-FCWT-Thema „Virtuelle Parallele Simulationsmodelle und ein Devirtualisierungsvorgang der Entwicklung von parallelen Simulatoren für dynamische Netzobjekte mit verteilten Parametern“ sowie unserer gemeinsamer Veröffentlichungen [6, 8, 10, 13, 14, 27] eine methodische Unterlage zur Lösung der Testaufgabe erarbeitet werden.

9. Implementierung der VPSU-TEST-Aufgabe mit der Verwendung von Subsystem-Simulationssoftware

9.1 Verteilung der VPSU-TEST-Aufgabe zwischen den Subsysteme, eine Definition von Subsysteme-Abhängigkeiten und der subsystemübergreifenden Koordination.

9.2 Implementierung und Debugging der Subsystemteile in VPSU-TEST-Aufgabe.

9.3 Implementierung und Debugging der subsystemübergreifenden Koordinationsfunktionen im Dialogsubsystem.

9.4 Implementierung und Debugging der Subsystemabhängigkeiten in jedem Subsystem.

9.5 VPSU-TEST-Probe, experimentelle Untersuchungen.

10. Erstellung der TTP1-Präsentation aufgrund der VPSU-TEST-AUFGABE.

4. Die diskrete Simulationsmodelle für DNOVP aufgrund der blockartigen numerischen Verfahren

Betrachten wir die Problemstellung und die Lösungsansätze zu den Entwicklungsaufgaben von diskreten Simulationsmodellen als erste Etappe der Gleichungslöseraufbau für parallele DNOVP-Simulatoren aufgrund der blockartigen numerischen Verfahren.

4.1 Formale Beschreibung (Modell) des dynamischen Netzobjektes mit verteilten Parametern (DNOVP)

Topologisch wird DNOVP als Graph *G(m, n)* dargestellt und durch Tabelle 1 mit *m* Zeilen und *s+5* Spalten kodiert. Hier sind: QJ – ein Luftstrom in der J-Kante; AKI und EKK – Anfangs- und Endknoten der J-Kante; I $\in$ *(1,2, ... ,n)*, J $\in$ *(1,2, ... ,m)*; PAR(PJ_1, PJ_2, …, PJ_s) – Menge von *s* Parametern PJ der Kante; AEJ – aktives Element in der J-Kante; VECOMJ – Kommentar zur technologische Funktion der J-Kante.

Tabelle 1 – Kodierung des Graphens

AKI	EKK	QJ	PAR(PJ_1, PJ_2, …, PJ_s)	AEJ	*(i) VECOMJ*

Modell der dynamischen Prozesse in der *j*-Kante ohne Luftverlusten durch die Wände wird von der Gleichungen

$$\begin{cases} -\dfrac{\partial P_j}{\partial \xi} = r_j Q_j^2 + \dfrac{\rho}{F_j}\dfrac{\partial Q_j}{\partial t} + r_j(\xi_r, t) Q_j^2 \\ -\dfrac{\partial P_j}{\partial t} = \dfrac{\rho a^2}{F_j}\dfrac{\partial Q_j}{\partial \xi}, \end{cases} \qquad (4.1)$$

beschrieben. Hier sind: P_j, Q_j – Druck und Luftstrom der Koordinate ξ entlang, die von *AKI*- bis zum *EKK*-Knoten errechnet wird ; r_j – spezifischer aerodynamischer Widerstand; F_j – die Querschnittsfläche der Kante (Luftwegstrecke); ρ – Luftdichte; a – die Schahlegeschwindigkeit im Luft; $r_j(\xi_r, t)$ – regelbarer Widerstand; ξ_r – die Ortskoordinate des regelbaren Widerstands (z. B., ein Schieber).

Die Randbedingungen für (1) sind die Druckfunktionen P_{AKI}, P_{EKK} in den Knoten der *j*-Kante. Es sind drei Kanten- und Knotenarten nach Randbedingungen in DNOVP zu unterscheiden:

- die Kanten, die den inneren $\boldsymbol{n_1}$ DNO-Knoten inzident sind; hier werden die Druckwerten während des Lösens des DNOVP-Gleichungssystems entsprechend den dynamischen Knotenbedingungen

$$-\frac{\partial P_{wi}}{\partial t} = \frac{\rho a^2}{F_{wi}} \frac{\partial Q_{wi}}{\partial \xi} \quad (4.2)$$

berechnet; hier sind P_{wi} – Druck im *WI*-Knoten; Q_{wi} – Gesamtluftstrom durch *WI*-Knoten; F_{wi} –Querschnittsfläche des Knotenraums;

- die Kanten, die den $\boldsymbol{n_2}$ Knoten der Ventilatorenanschlüsse inzident sind; hier wird Druck als die Ventilatorcharakteristik vorgegeben

$$P_{wi} = P_{AEJ}(QJ); \quad (4.3)$$

- die Kanten, die den $\boldsymbol{n_3}$ Knoten der Athmosphereanschlüsse inzident sind:

$$P_{wi} = P_{ATM} = const. \quad (4.4)$$

DNO hat insgesamt

$$n = n_1 + n_2 + n_3 \quad (4.5)$$

Knoten und entsprechend die $\boldsymbol{n}$ Randbedingungen.

Die Anfangsbedingungen sind

$$P_j(\xi, 0),\ Q_j(\xi, 0)\ (j = 1,2,...,m) \quad (4.6)$$

4.2 Problemstellung

Für DNOVP, dessen Graph wird mit der topologischen Tabelle 1 kodiert und jede Kante wird mit den Gleichungssysteme (4.1) und Randbedingungen (4.2), (4.3), (4.4) bei den Anfangsbedingungen (4.6) beschrieben, sollen die parallelen algorithmischen, hardware- und softwaretechnischen Simulationsmitteln entwickelt und implementiert werden, die adequat die dynamischen Prozesse $P_j(\xi, t),\ Q_j(\xi, t)\ (j = 1,2,...,m)$ mit der Berücksichtigung von definierten Arbeitsbedingungen, Störungen und Regelungen von Luftströmen widerspiegeln.

4.3 Die Simulationsmodelle der *j*-Kante und des DNOVP

Durch Approximation der Gleichungen (1) nach Linienverfahren mit der Ortsschrittweite $\Delta\xi$ erhalten wir für *k*-Element der *j*-Kante das Gleichungssystem:

$$-\frac{P_{jk} - P_{j,k+1}}{\Delta\xi_{jk}} = \frac{\rho}{F_{jk}} \frac{dQ_{jk}}{dt} + r_{jk} Q_{jk} \left|Q_{jk}\right| + r_{jk}(\xi_p, t) Q_{jk} \left|Q_{jk}\right|,$$

$$-\frac{Q_{jk} - Q_{j,k+1}}{\Delta\xi_{jk}} = \frac{F_{jk}}{\rho a^2} \frac{dP_{j,k+1}}{dt} \quad (4.7)$$

Die inneren Randbedingungen von Type (2) werden von Gleichung

$$-\frac{dP_{wi}}{dt} = \frac{\rho a^2}{F_{wi}} \frac{Q_{jk} - \sum_{jwi} (Q_{jwi1} - Q_{jwiMj})}{\Delta\xi_{jk}} \quad (4.8)$$

approximiert. Hier sind: $P_{wi} = P_{jM\,j+1}$ – Druck im Endknoten des letzten Elementes $Q_{jk} = Q_{jMj}$ des *J*-Kantestroms, der in *wi*-Knote fließt; $jwi \in j$ – die Kantennumern aus Menge *j=1, 2, ..., m*, die dem Knoten *wi* inzident sind; dabei *jwi1* ist erstes Element der *j*-Kante mit den *wi* als Anfangsknoten (Ausfluß), während *jwiMj* letztes Element mit den *wi* als Endknoten (Zufluß) ist. Jede Kante wird bei der Aproximation nach M_j Elementen $Q_{j1},...,Q_{jMj}$ geteilt. Dabei wird

die Numerierung der Druckwerten als P_{j1}, P_{j2}, ... , P_{jMj+1} erfolgt. Es ist wichtig zu erwähnen, daß *j*-Kante eine Anfangsknote *wi* mit dem Druck $P_{wi}=P_{j1}$ und die Endknote *wi+a* (*a = const*) mit dem Druck $P_{wi+a} = P_{jMj+1}$ hat.

Für dynamisches Netzobjekt mit verteilten Parametern wird jede Kante nach obige Approximation durch zwei Vektoren Q_j, P_j *(j = 1…m)* präsentiert:

$$Q_j = (Q_{j1},Q_{j2},\ldots,Q_{jMj})^T \qquad (4.9)$$

ist Luftstrom in *j*-Kante,

$$P_j = (P_{j1},P_{j2},\ldots,P_{jMj+1})^T \qquad (4.10)$$

ist Druck in *j*-Kante. Dabei wird M_j – die Elementenmengen in den Kanten – abhängig von der Kantenlängen l_j bei der gleichen Ortschrittweite $\Delta\xi$ für ganzes DNOVP als

$$M_j = l_j / \Delta\xi \qquad (4.11)$$

berechnet. Ortsdiskretisiertes Modell der *j*-Kante beinhaltet M_j Gleichungssysteme (4.7) bei $k=1,2,\ldots,M_j$. Für die numerische Berechnung der Vektorenkomponenten in (4.9) und (4.10) werden die Gleichungen (4.7) zur Form des ortsdiskretisierten Simulationsmodells umgewandelt:

$$\begin{cases} \dot{Q}_{jk} = \alpha_j(P_{jk} - P_{j,k+1}) - \beta_j Q_{jk}\left|Q_{jk}\right| - \beta_{rj} Q_{jk}\left|Q_{jk}\right| \\ \dot{P}_{j,k+1} = g_j(Q_{jk} - Q_{j,k+1}) \end{cases} \qquad (4.12)$$

Hier sind α_j, β_j, β_{rj}, g_j – Koeffizienten, die von der aerodynamischen Parametern *j*-Kante abhängig sind.

Bei der Entwicklung des ortdiskretisierten Simulationsmodells des ganzen Netzobjektes sollen wir *m* Gleichungssysteme von Type (4.12) für alle Kanten, d. h. *j=1,2,...,m,* darstellen:

$$\begin{cases} \dot{Q}_{1k}=\alpha_1(P_{1k}-P_{1,k+1})-\beta_1 Q_{1k}\left|Q_{1k}\right|-\beta_{r1} Q_{1k}\left|Q_{1k}\right| \\ \dot{P}_{1,k+1}=g_1(Q_{1k}-Q_{1,k+1}) \end{cases}$$

$k=1,2,\ldots,M_1$

…… (4.13)

$$\begin{cases} \dot{Q}_{mk}=\alpha_m(P_{mk}-P_{m,k+1})-\beta_m Q_{mk}\left|Q_{mk}\right|-\beta_{rm} Q_{mk}\left|Q_{mk}\right| \\ \dot{P}_{m,k+1}=g_m(Q_{mk}-Q_{m,k+1}) \end{cases}$$

$k = 1,2,\ldots,M_m$

Das Netzobjekt hat $n = n_1+n_2+n_3$ Randbedingungen (4.2)/(4.8), (4.3), (4.4). Entsprechend der Gleichung (4.8) formulieren wir $\boldsymbol{n_1}$ Randbedingungen für inneren Knoten des Netzobjektes ($wi=1, 2,\ldots,n_1$):

$$\frac{dP_{w1}}{dt} = \frac{\rho a^2}{F_{w1}} \frac{Q_{jk} - \sum_{jw1} (Q_{jw11} - Q_{jw1Mj})}{\Delta\xi_{jk}}$$

…… (4.14)

$$\frac{dP_{wn1}}{dt} = \frac{\rho a^2}{F_{wn1}} \frac{Q_{jk} - \sum_{jwn1} (Q_{jwn11} - Q_{jwn1Mj})}{\Delta\xi_{jk}}$$

Die Ventilatorenchrakteristiken bilden nach (4.3) n_2 weiteren Randbedingungen

$$P_{wi} = P^1_{AEJ}(QJ) \ldots\ldots P_{wi} = P^{n2}_{AEJ}(QJ). \quad (4.15)$$

Dabei ist $wi=n_1+1,n_1+2,..., n_1+n_2$, wenn die Numerierung der Knoten fortlaufend von inneren Knoten durchgeführt wird. Aktive Elemente (Ventilatoren) sind in (4.15) von *1* bis n_2 zusätzlich numeriert.

Für n_3 Knoten der Athmosphereanschlüsse gelten die Randbedingungen nach (4.4):

$$P_{wi} = P_{(n1+n2)+1} = P^1_{ATM} = const, \ldots\ldots P_{wi} = P_n = P^{n3}_{ATM} = const. \quad (4.16)$$

Bei obiger Nummerierung ist in (4.16) $wi=(n_1+n_2)+1,...,(n_1+n_2)+(n_3-1), n.$

Die Gleichungen (4.13) bilden zusammen mit den Randbedingungen (4.14), (4.15), (4.16) das ortsdiskretisiertes DNOVP-Simulationsmodell als vollständiges Gleichungssystem:

$$\begin{cases} \dot{Q}_{11} = \alpha_1(P_{11} - P_{12}) - \beta_1 Q_{11}|Q_{11}| - \beta_{r1} Q_{11}|Q_{11}| \\ \dot{P}_{12} = g_1(Q_{11} - Q_{12}) \end{cases} \ldots \begin{cases} \dot{Q}_{1M1} = \alpha_1(P_{1M1} - P_{1M1+1}) - \beta_1 Q_{1M1}|Q_{1M1}| - \beta_{r1} Q_{1M1}|Q_{1M1}| \\ \dot{P}_{1M1+1} = g_1(Q_{1M1} - Q_{1M1+1}) \end{cases}$$

$$j=1, k=1,...,M_1$$

$$\begin{cases} \dot{Q}_{m1} = \alpha_m(P_{m1} - P_{m,2}) - \beta_m Q_{m1}|Q_{m1}| - \beta_{rm} Q_{m1}|Q_{m1}| \\ \dot{P}_{m,2} = g_m(Q_{m1} - Q_{m,2}) \end{cases} \ldots \begin{cases} \dot{Q}_{mk} = \alpha_m(P_{mk} - P_{mk+1}) - \beta_m Q_{mk}|Q_{mk}| - \beta_{rm} Q_{mk}|Q_{mk}| \\ \dot{P}_{mk+1} = g_m(Q_{mk} - Q_{mk+1}) \end{cases} \quad (4.17$$

$$j=m, k=1,...,M_m$$

Wir werden zwischen den Luftstrom(Q)- und Druck(P)-Gleichungen unterscheiden. Die gesamte Menge der zu lösenden Q-Gleichungen in (4.13) ist

$$N_{QGl} = \sum M_j, \ 1 \le J \le m.$$

Aus (4.13) und (4.14) folgt, dass Menge der P-Gleichungen pro Kante $M_j - 2$ ist, weil die zwei Randknotendrücke werden nach (4.14) berechnet. Deshalb

$$N_{PGl} = \sum (M_j - 2) + n_1, \ 1 \le J \le m.$$

Für die industrienahen DNOVP ($m \ge 1000$, $n \ge 300$, $M_j \ge 50$) geht es um $(N_{QGl}+N_{PGl}) \ge 10^5$, deshalb ist aktuell die rechnergestützte Erstellung der DNOVP-Simulationsmodelle zu realisieren. Diese Aufgabe wird mit Hilfe von Topologieanalysator und des Gleichungsgenerators gelöst [11].

Die gesuchten Variablen $Q(\xi,t)$ und $P(\xi,t)$ werden als Multivektoren betrachtet:

$$Q = \begin{bmatrix} Q_{11} & Q_{12} & \ldots & Q_{1M_1} \\ Q_{21} & Q_{22} & \ldots & Q_{2M_2} \\ \ldots & \ldots & \ldots & \ldots \\ Q_{m1} & Q_{m2} & \ldots & Q_{mM_m} \end{bmatrix} \quad (4.18)$$

$$P = \begin{bmatrix} P_{11} & P_{12} & \dots & P_{1M_1+1} \\ P_{21} & P_{22} & \dots & P_{2M_2+1} \\ \dots & \dots & \dots & \dots \\ P_{m1} & P_{m2} & \dots & P_{mM_m+1} \end{bmatrix} \quad (4.19) \quad .$$

Hier wird für die Illustration angenommen, dass die Vektoren Q_j, P_j gleiche M_j haben. Bei der praktischen Lösungen bekommen wir die Matrizen (4.18), (4.19), wo die Spaltenmenge wird von M_{jmax} definiert und werden in mehreren Zeilen die Null-Elemente vorhanden.

Die Knotendruckwerte werden mit der Gleichung beschrieben

$$\dot{P}_{U_{j,k+1}} = g_j(Q_{jk} - \sum Q_{Uj,k+1}) \quad (4.20)$$

Es werden die Differenzmatrizen mit den Elementen $\Delta Q_{jk} = Q_{jk} - Q_{j,k+1}$, $\Delta P_{jk} = P_{jk} - P_{j,k+1}$ und Matrix ZQ mit den Elementen $ZQ_{jk} = Q_{jk}\,|Q_{jk}|$ formiert und Gleichungssysteme (4.13), (4.17) werden in Matrix-Vektor-Form umgewandelt:

$$\begin{cases} \dot{Q} = \alpha \cdot \Delta P - \beta \cdot ZQ - \beta_r \cdot ZQ \\ \dot{P} = g \cdot \Delta Q \end{cases} \quad (4.21).$$

Führen wir Vektor P_U für die Knotendrücke $P_U = (P_{U1},P_{U2},\dots,P_{Un1})^T$ ein. Allgemeine Gleichung für die Berechnung von Elementen dieses Vektors sieht so aus:

$$\dot{P}_U = G(sumRows(Q_U)). \quad (4.22)$$

Hier ist G – Diagonalmatrix der Parametern g von Kanten, die den Knoten inzident sind; *sumRows* definiert eine Operation der zeilenweisen Summierung von Elementen Q_{Ujk} der Q_U – Matrix, welche von Inzidenzmatrixelementen A_{jk} abhängig sind. Aufbau der Inzidenzmatrix für die Kantenelementen Q_{j1} und Q_{jMj}, $j=1,\dots,m$, die als Asuflüße und Zuflüße für entsprechende inzidente Knoten sind, gehört zu den Aufgaben des Topologieanalysators. Das ortsdiskretisierte Simulationsmodell (4.21), (4.22) wird vom Gleichungsgenerator durch formale Matrix-Vektor-Operationen erstellt [23]. Es sollen die Algorithmen und Programme des Topologieanalysators und des Gleichungsgenerators zu den Anforderungen der blockartigen numerischen Lösungsverfahren angepasst werden.

4.4 Die numerische Lösungsverfahren und die diskrete Simulationsmodelle (DSM) der *j*-Kante und des DNOVP

Die Simulationsmodellgleichungen (4.12), (4.13), (4.17) entsprechen den minimalkörnigen MIMD-Prozessen und werden als Caushy-Aufgabe

$$x' = f(t, x),\ x(t_0) = x_0 \quad (4.23)$$

betrachtet. Durch Diskretisierung der Variable t mit gleichmäßigem Gitter h: $\{t_n = nh,\ n = 0, 1, 2, \dots\}$ und Approximation der Ableitungen bekommen wir die diskreten Simulationsmodelle (DSM), die nach beiden Koordinaten diskretisiert werden. Wenn die genaue Lösung der Caushy-Aufgabe $x(t)$ und angenäherte $u_n = u(t_n)$ sind, dann eine Differenz

$$\varepsilon_n = u_n - x_n \quad (4.24)$$

charakterisiert die Lösungsfehler am Schritt n.

4.4.1 DSM aufgrund der konventionellen numerischen Verfahren

Die konventionelle numerische Verfahren für die Lösung der Caushy-Aufgabe (4.23) werden in zwei Gruppen klassifiziert:

- Einschrittsverfahren, die für die Suche der Lösung in einem Punkt t_{n+1} nur die Information über zu lösenden Aufgabe auf Einschrittsintervall (t_n, t_{n+1}) verwenden. Zu dieser Gruppe gehören die Euler- und Runge-Kutta(RUKU)-Verfahren.
- Mehrschrittverfahren, die für die Suche der Lösung in einem Punkt t_{n+1} ein Teil der auf den vorigen Schritten erhaltenen Information verwenden. Diese Gruppe verfügt durch Adams-Verfahren.

Differenzverfahren convergiert im Punkt t, wenn $|x_n - u_n| \to 0$ bei $h \to 0$ und $t_n = t$. Das Verfahren convergiert am Intervall $[t_0, T]$, wenn es im jeden Punkt dieses Intervalls convergiert. Ein Verfahren hat p-Ordnung der Genauigkeit, wenn ein Wert p existiert, für welchen $|x_n - u_n| = O(h)^p$ beim $h \to 0$.

Hauptidee des Runge-Kutta-Verfahrens ist zu garantieren, dass auf einem Schritt Differenz der annähernden Lösung $\Delta u_n = u_{n+1} - u_n$ gleich wäre mit dem Differenz $\Delta x_n = x_{n+1} - x_n$ der genauen Lösung auf derselben Schritt.

Die Berechnungsformeln für explizites Runge-Kutta-Verfahren der 4.Ordnung bei der Lösung der Caushy-Aufgabe sehen so aus:

$$\begin{aligned} &k_1 = f(t_n, u_n);\\ &k_2 = f(t_n+h/4, u_n + hk_1/4);\\ &k_3 = f(t_n+h/2, u_n + hk_2/2);\\ &k_4 = f(t_n+h, u_n + hk_1 - 2hk_2 + 2hk_3);\\ &u_{n+1}=u_n + h(k_1 + 4k_3 + k_4)/6. \end{aligned} \qquad (4.25)$$

Approximationsfehler wird so definiert:

$$r_n = -\frac{x_{n+1} - x_n}{h} + (k_1 + 4k_3 + k_4)/6 = O(h)^4.$$

Die Formeln (4.25) werden für Aufbau der diskreten Simulationsmodelle der J-Kante und gesamten DNOVP von uns benutzt. Führen wir die folgenden Vektorbezeichnungen ein

$$X = \begin{pmatrix} Q_1 \\ P_2 \\ \vdots \\ Q_k \\ P_{k+1} \\ \vdots \\ Q_M \\ P_{M+1} \end{pmatrix}; X' = \begin{pmatrix} Q_1' \\ P_2' \\ \vdots \\ Q_k' \\ P_{k+1}' \\ \vdots \\ Q_M' \\ P_{M+1}' \end{pmatrix}; F(t,X) = \begin{pmatrix} f_1(t, Q_1, P_1, P_2) \\ f_2(t, Q_1) \\ \vdots \\ f_k(t, Q_k, P_k, P_{k+1}) \\ f_{k+1}(t, Q_k) \\ \vdots \\ f_M(t, Q_M, P_M, P_{M+1}) \\ f_{M+1}(t, Q_M) \end{pmatrix}; X_0 = \begin{pmatrix} Q_1^0 \\ P_2^0 \\ \vdots \\ Q_k^0 \\ P_{k+1}^0 \\ \vdots \\ Q_M^0 \\ P_{M+1}^0 \end{pmatrix}; \qquad (4.26)$$

hier sind $Q_k' = \frac{dQ_k}{dt}$; $P_{k+1}' = \frac{dP_{k+1}}{dt}$; die Funktionen $f_k(t,Q_k,P_k,P_{k+1})$, $f_{k+1}(t,Q_k)$ – die rechten Seiten entsprechend für Q- und P-Gleichungen in den Systeme (4.13), (4.17); X_0 – Vektor der Anfangsbedingungen. Unter diesen Bezeichnungen wird das Simulationsmodell als Caushy-Aufgabe so aussehen

$$X' = F(t,X),\ X(t_0) = X_0. \qquad (4.27)$$

Diskretes Simulationsmodell wird nach Ru-Ku-Berechnungsformeln erstellt:

$$K_1 = F(t_n, U_n);\ n = 0, 1, 2, \ldots$$

$$K_2 = F(t_n+0{,}5h,\ U_n + 0{,}5hK_1);$$
$$K_3 = F(t_n+h/2,\ U_n + hK_2/2); \qquad (4.28)$$
$$K_4 = F(t_n+h,\ U_n + hK_3);$$
$$U_{n+1}=U_n + h(K_1 + 2K_2 + 2K_3+ K_4)/6.$$

Die *m*-Schrittverfahren von Adams-Bashforth (AB-Verfahren) verwenden die Berechnungsformel

$$u_{n+1} = u_n + h\sum_{j=1}^{m} b_j f(t_{n+1-j}, u_{n+1-j}),\ b_0 = 0. \quad (4.29)$$

Bei $m = 2$ wird am meistens dieses Verfahren gebraucht:

$$u_{n+1} = u_n + \frac{h}{2}(3f(t_n, u_n) - f(t_{n-1}, u_{n-1})). \quad (4.30)$$

Diskretes Simulationsmodell der j-Kante nach (30) hat für Q- , P-Gleichungen bei den obigen Bezeichnungen folgende Form

$$U_{n+1} = U_n + hF(t_n, U_n),\ \textit{für } n = 0$$

$$U_{n+1} = U_n + 0.5h(3F(t_n, U_n) - F(t_{n-1}, U_{n-1})),\ \textit{für } n \geq 1 \quad (4.31)$$

Unabhängig vom ausgewählten Parallelisierungsansatz werden die DSM-Berechnungen so durchgeführt:

Schritt 0 – die Prozesse werden initialisiert;

Schritt 1 – es werden die rechten Seiten der Q- und P-Gleichungen berechnet;

Schritt 2 – Berechnung von Q_{jk} und $P_{j,k+1}$ mit dem ausgewählten numerischen Verfahren. Ausführungszeiten sind: $t_2=13t_{mult.} + 7t_{sum}$ *für RU-KU;*

$$t_2 = 2t_{mult} + 2t_{sum} \textit{ für A-B.}$$

Schritt 3 – Datenaustausch zwischen den Prozessen und Übergang zum Neutakt der Berechnungen.

Die diskrete RUKU- und AB-Simulationsmodelle (4.28), (4.31) aufgrund der konventionellen numerischen Verfahren wurden von uns in SIMD- und MIMD-Simulatoren realisiert. Sie brauchen noch der theoretischen Apriori-Analyse bezüglich der möglichen Parallelitätsebenen des DNOVP und der architekturabhängigen MIMD-Implementationen. Realisierungserfahrung zeigt aber, dass die Schlußfolgerungen aus Analyse der ortsdiskretisierten Simulationsmodelle [23] gut mit den Implementationsergebnissen übereinstimmen.

4.4.2. Hauptideen der blockartigen Verfahren (BAV)

In [26, 27] haben wir die Entwicklung der parallelen DNOVP-Gleichungslöser aufgrund der blockartigen numerischen Verfahren (BAV-Löser) als aktuelles Problem bezeichnet. Als erste Etappe dieser Entwicklung bertachten wir die Aufbau von diskreten DNOVP-Simulationsmodellen, die sich auf den Simulationsmodellen der j-Kante (4.12) und des DNOVP (4.13) – (4.17) und auf grundlegenden BAV-Ideen basieren.

Bei der Lösung des Caushy-Problems (4.23) mit Hilfe des blockartigen *k-Punktverfahrens*. wird die Zeitachse *t* für Dauer T_{sim} des simulierten Prozesses *x(t)* mit dem Schritt τ als homogenes Gitter $\{t_m\}$, $m = 1,2,\ldots, M$ dargestellt. Die Knotenmenge wird als

$$M=T_{sim}/\tau$$

berechnet, ohne Punkt *t=0* einzuschließen. In den oben erwähnten konventionellen numerischen Verfahren wird ein Wert des Prozesszustandes $x(t_m)$ mit gewisser Genauigkeit

im t_m-Punkt als $u(t_m)$ berechnet. Zerlegen wir dieser Gitter auf N Blöcke, die je k Punkte beinhalten, $kN \le M$ (Abb.4.1). Nummerieren wir die Blöcke als $n=1,2,...,N$ und in jedem Block die Punkte als $i = 0,1,\ldots, k;$ dann wird $t_{n,i}$ das Punkt i des n-er Blocks bezeichnen. Die Punkte $t_{n,0}$ und $t_{n,k}$ sind entsprechend der Anfang und das Ende des Blocks n . Letztes Punkt des Blocks n ist gleichzeitig das Anfangspunkt des Blocks $n+1$, d. h. $t_{n,k} = t_{n+1,0}$ und das Anfangspunkt wird dem Block nicht gehören.

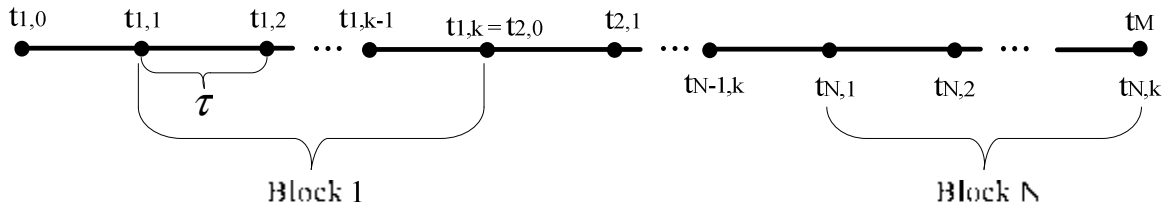

Abb. 4.1. Schema der Zerlegung auf die Blöcke

Für das Caushy-Problem, das mit den gewöhnlichen Differentialgleichungssysteme (4.12), (4.13), (4.17) beschrieben wird, berechnen die blockartigen Verfahren für einen Block n mit den k Punkten die neue k Funktionswerte $u_{n1}(\tau)$, $u_{n2}(2\tau)$, ..., $u_{nk}(k\tau)$ gleichzeitig, also parallel [18, 20, 21]. Diese Verfahrenseigenschaft passt gut zu den parallelen Rechnerarchitekturen und erlaubt die Berechnungen von Koeffizienten der Differenzformeln vor der Integrierung, schon während der Entwicklung des Verfahrens und erhöht die Effizienz des Rechnens.

Nach Art der Berechnungen der gesuchten Variablen (wie in obigen RUKU- und AB-Verfahren) unterscheidet man zwischen den blockartigen Einschritts- und Mehrschrittsverfahren . Im Einschritt-k-Punktverfahren wird nur einziger Wert im letztem Punkt k des Blocks n für die Berechnung alle k Werte des nächsten Blocks $n+1$ benutzt (Abb.4.2, letztes Zeitpunkt $t_{n,k}$ des Blocks n ist auch Anfangspunkt $t_{n+1,0}$ des Blocks $n+1$). Im Mehrschritts-k-Punktverfahren wird jeder Wert in alle k Punkte des nächsten Blocks $n+1$ mit der Benutzung der Werte aller k oder $m<k$ Punkte des vorigen Blocks n berechnet (Abb.4.3).

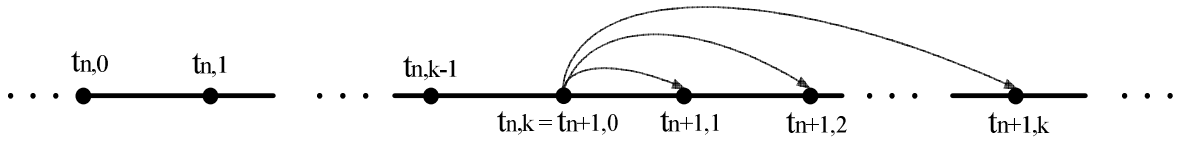

Abb. 4.2. Berechnungsschema beim blockartigen Einschrittsverfahren

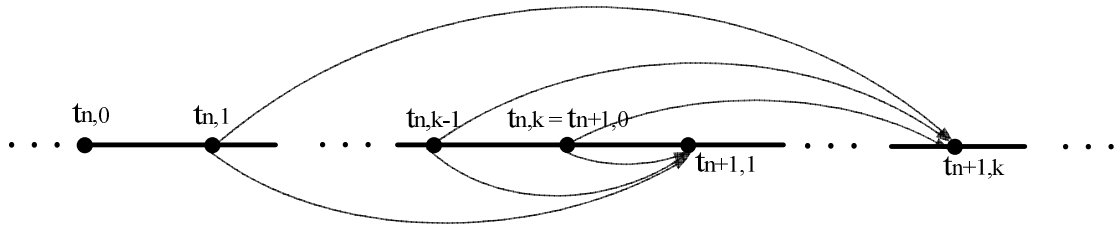

Abb. 4.3. Berechnungsschema beim blockartigen Mehrschrittsverfahren

Im allgemeinen Fall des blockartigen Verfahrens (m Schritte, k Punkte) haben die Gleichungen für die Berechnung der gesuchten Variablen folgende Form [20,21]:

$$u_{n,i} = u_{n,0} + i\tau\left[\sum_{j=1}^{m} b_{i,j} F_{n-1,j} + \sum_{j=1}^{k} a_{i,j} F_{n,j}\right] \quad (4.32).$$

Hier sind:

$u_{n,i}(i\tau)$ – die Nähernlösung im *i*-Punkt des Blocks *n*, die der gesuchten genauen Lösung $x(t = i\tau)$ entspricht; $n=1,2,...,N;\ i = 1,2,...,k;$

$u_{n,0}$ – Lösungswert im Anfangspunkt des n-Blocks; dieser Wert ist fürs Block n=1 die Anfangsbedingung des Caushy-Problems, für die Blöcke n >1 ist dieser Wert als eine Lösung im letzten Punk des vorigen Blocks zu entnehmen, d.h.

$$u_{2,0} = u_{1,k},...,u_{N,0} = u_{N-1,k};$$

$j = 1,2,...,m$ – die Nummern von Punkte (Schritte) des vorigen (*n-1*)-Blocks, dessen Werte werden für Berechnung der Werte $u_{n,i}$ im *n*-Block verwendet; im allgemeinen Fall wird vorausgesetzt, dass $m = k$ und Summierung nach Index $j = 1,2,...,k$ auch im Block *n* durchgeführt wird (die Summe in (32) von Produkte $a_{i,j}*F_{n,j}$);

$F_{n,j} = f(t_n+j\tau; u_{n,j})$ – Werte der rechten Seite im Caushy-Problem (4.23);

$F_{n-1,j} = f(t_{n-1}+j\tau; u_{n-1,j})$ – Werte der rechten Seite für Block *n-1* beim $j = 1,2,...,m;$

$a_{i,j}$, $b_{i,j}$ – die Koeffizienten, die nach dem Integrierung-Interpolationsverfahren gesucht werden:

1) es wird Interpolationsmehrglied $L_k(t)$ mit den Interpolationsknoten $t_{n,i}$ ($i=0,1,...,k$ – *die Knoten im Blockbereich*) und diesen Knoten entsprechenden Werten der rechten Seite $F_{n,i} = f(t_{n,i}, u_{n,i})$ gebaut;

2) $L_k(t)$ wird von t_n bis $t_n+i\tau$ für $i = 1,2,..,k$ integriert: $u_{n,i} = u_{n,0} + \int L_k(t)dt,\ t_n \le t \le t_n+i\tau.$

Methodik der Berechnung von Koeffizienten wurde in [20,21] mit Hilfe des Programmpackets MATHEMATICA realisiert, die Koeffizienten für Verfahren mit verschiedenen *m, k* sind ähnlich wie RUKU-Verfahren bekannt.

4.4.3. DSM aufgrund der BAV-Anwendung

Betrachten wir die Entwicklung der diskreten Simulationsmodelle für einzelne DNO-Kante und für gesamtes DNOVP aufgrund der blockartigen Verfahren, die die Parametern *m=1, k=2* und *m=1, k=4* haben. In Abb.4.4 ist den Schablone des Einschritts- und *k*-Punkt-Differenzschemas gezeigt.

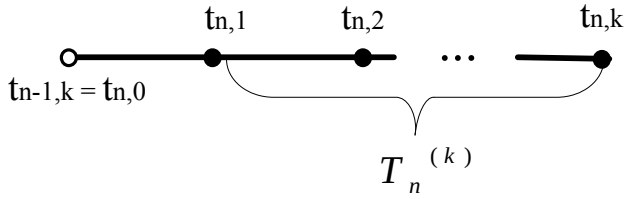

Abb. 4.4. Einschritts- und k-Punktdifferenzschema

Die allgemeine Berechnungsformel für die Definition der neuen *k* Werte folgt aus (4.32):

$$u_{n,i} = u_{n,0} + i\tau\left[b_i F_{n-1,j} + \sum_{j=1}^{k} a_{i,j} F_{n,j}\right]. \quad (4.33)$$

Für BAV mit m=1, k=4 gilt $F_{n-1,j} = F_{n,0}$ (siehe $t_{n-1,k} = t_{n,0}$, Abb.4.4) und aus (4.33) bekommen wir die folgenden Gleichungen

$$u_{n,1} = u_{n,0} + \tau\left[b_1 F_{n,0} + \sum_{j=1}^{4} a_{1,j} F_{n,j} \right],$$

$$u_{n,2} = u_{n,0} + 2\tau\left[b_2 F_{n,0} + \sum_{j=1}^{4} a_{2,j} F_{n,j} \right], \quad (4.34)$$

$$u_{n,3} = u_{n,0} + 3\tau\left[b_3 F_{n,0} + \sum_{j=1}^{4} a_{3,j} F_{n,j} \right],$$

$$u_{n,4} = u_{n,0} + 4\tau\left[b_4 F_{n,0} + \sum_{j=1}^{4} a_{4,j} F_{n,j} \right].$$

Die Berechnungsformeln fürs Verfahren sehen so aus [20, 21]:

$$U_{n,1} = \frac{\tau}{720}(251F_{n,0} + 646F_{n,1} - 264F_{n,2} + 106F_{n,3} - 19F_{n,4}) + U_{n,0}; \quad (4.35)$$

$$U_{n,2} = \frac{\tau}{90}(29F_{n,0} + 124F_{n,1} + 24F_{n,2} + 4F_{n,3} - F_{n,4}) + U_{n,0};$$

$$U_{n,3} = \frac{3\tau}{80}(9F_{n,0} + 34F_{n,1} + 24F_{n,2} + 14F_{n,3} - F_{n,4}) + U_{n,0};$$

$$U_{n,4} = \frac{2\tau}{45}(7F_{n,0} + 32F_{n,1} + 12F_{n,2} + 32F_{n,3} + 7F_{n,4}) + U_{n,0}.$$

Für die Gleichungen von Type (4.12) des Simulationsmodells der j-Kante, des *v*-Elements

$$\begin{cases} \dot{Q}_{jv} = \alpha_j (P_{jv} - P_{j,v+1}) - \beta_j Q_{jv} \left| Q_{jv} \right| - \beta_{rj} Q_{jv} \left| Q_{jv} \right| \\ \dot{P}_{j,v+1} = g_j (Q_{jv} - Q_{j,v+1}) \end{cases} \quad (4.36)$$

sollen wir mit der Berücksichtigung $v=1,2,...M_j$ die BAV-Berechnungsformeln (4.35) anwenden. Bedeutung der notwendigen Indexe werden wir örtlich definieren und erklären, weil es um recht komplexen Gleichungsstrukturen geht. Einige von Indexbezeichnungen haben wir schon in den Simulationsmodelle (j – Kantennummer, m – Kantenmenge im DNO, n – Knotenmenge im DNO, k – Elementennummer in der approximierten Kante u.a.) benutzt.

Methodik der DSM-Entwicklung für j-Kante:

1. Es werden die Parametern der j-Kante vorgegeben (siehe [28])

2. Es wird T_{sim} definiert – Dauer der untersuchten dynamischen Prozesse, Sec.

3. Für DNOVP und j-Kante wird die Ortsschrittweite $\Delta\xi$ vorgegeben; wir nehmen Bereich $5.0 \le \Delta\xi \le 50.0$ m

4. Es wird Zeitschritt $\tau=\Delta\xi/a$ berechnet: $\tau = (5.0 \le \Delta\xi \le 50.0)/330.0 - 0.015 \le \tau \le 0.15$

5. Die Mengen von Zeitschritten und Blöcken werden berechnet als

$$M = T_{sim} / \tau, \text{ also } 70T_{sim} \ge M \ge 7T_{sim};$$

$$N = M/k, \text{ für } k=4\ 17{,}5T_{sim} \ge N \ge 1.75T_{sim}$$

6.Es wird die Zeitachse im Bereich $0 \le t \le T_{sim}$ nach Schablone Abb.4.4 für $k=4$ blockweise für n = 1, 2, ... , r-1, r, r+1, ..., N-1, N dargestellt, um die Berechnungsabläufe und Zusammenhänge zwischen der Blöcke zu veranschaulichen. Dieser Schablone soll für

beide Gleichungen des Systems (4.35) erstellt werden, um die möglichen Zusammenhänge und den notwendigen Datenaustausch zwischen den Q- und P-Prozesse zu zeigen.

7. Es sollen die Bezeichnungen der Variablen, Koeffizienten sowie Indexe in (4.36) mit (4.35) übereinstimmen und auf Zeitachse darstellen (P.6). Als erste Version werden folgenden Bezeichnungen vorgeschlagen: $Q_{jvn,i} = U_{n,i}$, $P_{jv+1,\ n,i} = U_{n,i}$ und für die rechten Seiten $F_{n,0}$, $F_{n,1} - F_{n,4}$ werden allgemeine Bemerkung eingeführt, dass es über Q_j- oder P_j-Gleichungen geht.

Dann sind die Q_j- und P_j-Werte in jedem Punkt des n-Blocks als die Lösungen des folgenden Gleichungssystems zu berechnen:

$$U_{n,1} \Rightarrow Q_{jvn,1} = \frac{\tau}{720}(251F_{n,0} + 646F_{n,1} - 264F_{n,2} + 106F_{n,3} - 19F_{n,4})_{Qjv} + Q_{jvn,0};$$

$$U_{n,2} \Rightarrow Q_{jvn,2} = \frac{\tau}{90}(29F_{n,0} + 124F_{n,1} + 24F_{n,2} + 4F_{n,3} - F_{n,4})_{Qjv} + Q_{jvn,0};$$

$$U_{n,3} \Rightarrow Q_{jvn,3} = \frac{3\tau}{80}(9F_{n,0} + 34F_{n,1} + 24F_{n,2} + 14F_{n,3} - F_{n,4})_{Qjv} + Q_{jvn,0};$$

$$U_{n,4} \Rightarrow Q_{jvn,4} = \frac{2\tau}{45}(7F_{n,0} + 32F_{n,1} + 12F_{n,2} + 32F_{n,3} + 7F_{n,4})_{Qjv} + Q_{jvn,0}.$$

$$U_{n,1} \Rightarrow P_{jv+1,n,1} = \frac{\tau}{720}(251F_{n,0} + 646F_{n,1} - 264F_{n,2} + 106F_{n,3} - 19F_{n,4})_{Pjv+1} + P_{jv+1,n,0}; \quad (4.37)$$

$$U_{n,2} \Rightarrow P_{jv+1,n,2} = \frac{\tau}{90}(29F_{n,0} + 124F_{n,1} + 24F_{n,2} + 4F_{n,3} - F_{n,4})_{Pjv+1} + P_{jv+1,n,0};$$

$$U_{n,3} \Rightarrow P_{jv+1,n,3} = \frac{3\tau}{80}(9F_{n,0} + 34F_{n,1} + 24F_{n,2} + 14F_{n,3} - F_{n,4})_{Pjv+1} + P_{jv+1,n,0};$$

$$U_{n,4} \Rightarrow P_{jv+1,n,4} = \frac{2\tau}{45}(7F_{n,0} + 32F_{n,1} + 12F_{n,2} + 32F_{n,3} + 7F_{n,4})_{Pjv+1} + P_{jv+1,n,0}.$$

Die Variable $Q_{vn,0}$ und $P_{v+1,n,0}$ sind die Q- und P-Werte im Anfangspunkt des n-Blocks. Im ersten Block $n=1$ nehmen diese Variable die Anfangsbedingungen für Q und P über. In den nachfolgenden Blöcken werden $Q_{vn,0} = Q_{v,n-1,4}$ und $P_{v+1,n,0} = P_{v+1,n-1,4}$ sein (Abb.4.4, $t_{n-1,k} = t_{n,0}$).

Die Funktionen $F_{n,k}$ für Q_{jv} aus (4.36) sind:

$$(F_{v,n,0})_{Qjv} = \alpha_j(P_{jv,0} - P_{jv+1,n,0}) - \beta_j Q_{jv,n,0}|\ Q_{jv,n,0}| - \beta_{rj} Q_{jv,n,0}|\ Q_{jv,n,0}|$$
$$(F_{v,n,1})_{Qjv} = \alpha_j(P_{jv,0} - P_{jv+1,n,1}) - \beta_j Q_{jv,n,1}|\ Q_{jv,n,1}| - \beta_{rj} Q_{jv,n,1}|\ Q_{jv,n,1}|$$
$$(F_{v,n,2})_{Qjv} = \alpha_j(P_{jv,0} - P_{jv+1,n,2}) - \beta_j Q_{jv,n,2}|\ Q_{jv,n,2}| - \beta_{rj} Q_{jv,n,2}|\ Q_{jv,n,2}| \quad (4.38)$$
$$(F_{v,n,3})_{Qjv} = \alpha_j(P_{jv,0} - P_{jv+1,n,3}) - \beta_j Q_{jv,n,3}|\ Q_{jv,n,3}| - \beta_{rj} Q_{jv,n,3}|\ Q_{jv,n,3}|$$
$$(F_{v,n,4})_{Qjv} = \alpha_j(P_{jv,0} - P_{jv+1,n,4}) - \beta_j Q_{jv,n,4}|\ Q_{jv,n,4}| - \beta_{rj} Q_{jv,n,4}|\ Q_{jv,n,4}|$$

Die Funktionen $F_{n,k}$ für P_{jv+1} aus (4.36) sind:

$$(F_{v,n,0})_{Pjv+1} = g_j(Q_{jv,0} - Q_{jv+1,n,0})$$
$$(F_{v,n,1})_{Pjv+1} = g_j(Q_{jv,1} - Q_{jv+1,n,1})$$
$$(F_{v,n,2})_{Pjv+1} = g_j(Q_{jv,2} - Q_{jv+1,n,2}) \quad (4.39)$$
$$(F_{v,n,3})_{Pjv+1} = g_j(Q_{jv,3} - Q_{jv+1,n,3})$$
$$(F_{v,n,4})_{Pjv+1} = g_j(Q_{jv,4} - Q_{jv+1,n,4})$$

Also, dem Simulationsmodell (4.36) der *j*-Kante entspricht das diskrete Simulationsmodell (DSM) (4.37), (4.38) und (4.39), das aufgrund des blockartigen Einschritts-***k***-Punkt-Verfahrens bei $k=4$ gebaut wurde (BAV-DSM). Die obige Methodik wird zur Realisierung im Gleichungsgenerator der DNOVP-orientierten VPSU geplant.

Methodik der DSM-Entwicklung für ganzes dynamisches Netzobjekt mit verteilten Parametern (DNOVP):

1. Für jede DNOVP-Kante bei j=1,2,...,m (m – die Kantenmenge im Netzobjekt) wird BAV-DSM (4.37) – (4.39) entsprechend dem DNOVP-Gleichungssystem (4.17) erstellt.

2. Die inneren Randbedingungen (4.14) werden nach BAV-DSM-Darstellung umgewandelt.

3. Die äuseren Randbedingungen (4.15), (4.16) werden zu den inzidenten Kanten des DNOVP-BAV-DSM angepasst.

Diese DNOVP-BAV-DSM-Entwichklung ist als die Funktionen des Gleichungsgenerators zu realisieren.

5. Blockorientierte parallele Simulationssprache: ein Entwicklungskonzept

5.1. Motivation

Die Benutzerfreundlichkeit der parallelen Rechnersysteme ist in erster Reihe mit den vorhandenen Programmiermodelle und Mitteln für Realisierung der parallelen Lösungsalgorithmen verbunden. Die Systeme der 90-zigen Jahren verfügten über parallele SIMD- und MIMD-Programmiersprachen, die aufgrund von Fortran, C, C++, Modula-2 u. a. Sprachen gebaut wurden [1, 2]. Intensive Entwicklung der MIMD-Systeme und objektorientierten Ansätze hat die Standardisierung der Technologien der parallelen und verteilten Programmierung stimuliert. So haben die ANSI, ISO C++-Standard mit den MPI-, PVM- und Pthreads-Bibliotheken eingeführt. Trotz gewisser Fortschritt soll Gegenstandsexperte (Modellentwickler) heute wie früher mit der Programmiersprache die parallele Modelle realisieren und damit auf dem Niveau der ehemaligen konventionellen Simulationssysteme der zweiten und dritten Generation [3] bleiben. Diese Tatsache beschränkt wesentlich eine Nutzung der parallelen Ressourcen von Fachleuten, die mit den block- (BO), gleichungs- (GO) und objektorientierten (OO) Simulationssprachen [3, 9] dynamische Systeme modellieren und simulieren. Um VPSU zu den Simulationsmitteln etwa der fünften Generation [3] einzunäheren, sollen wir die parallele Simulationssoftware in Richtung parallelen Simulationssprachen weiter zu entwickeln. Diese Sprachen sollen eindeutige Verbindung zwischen den Gegenstandsgebiete der verfahrenstechnischen Ebene und den formalen Beschreibung von dynamischen Systemen mit Hilfe der Modellspezifikation benutzerfreundlich widerspiegeln (Abb.5.1) und in den lauffähigen parallelen Simulatoren umwandeln (Abb. 2.4, 2.5). Analyse zeigt, dass die VTF-, SAS- und DNO-Topologie allgemein in der Graphendarstellung umgewandelt werden können. Die entwickelten und experimentell untersuchten Paare „Topologieanalysatoren – Gleichungsgeneratoren“ geben die Simulationsmodelle, die direkte Anwendung der BO- und GO-Prinzipien der Gleichungslösung erlauben, ohne die diskrete Simulationsmodelle zu generieren. In [27] haben wir die Ideen der BO- und GO-Ansätze betrachtet. Hier wird das Entwicklungskonzept für parallele BO-Simulationssprache vorgeschlagen.

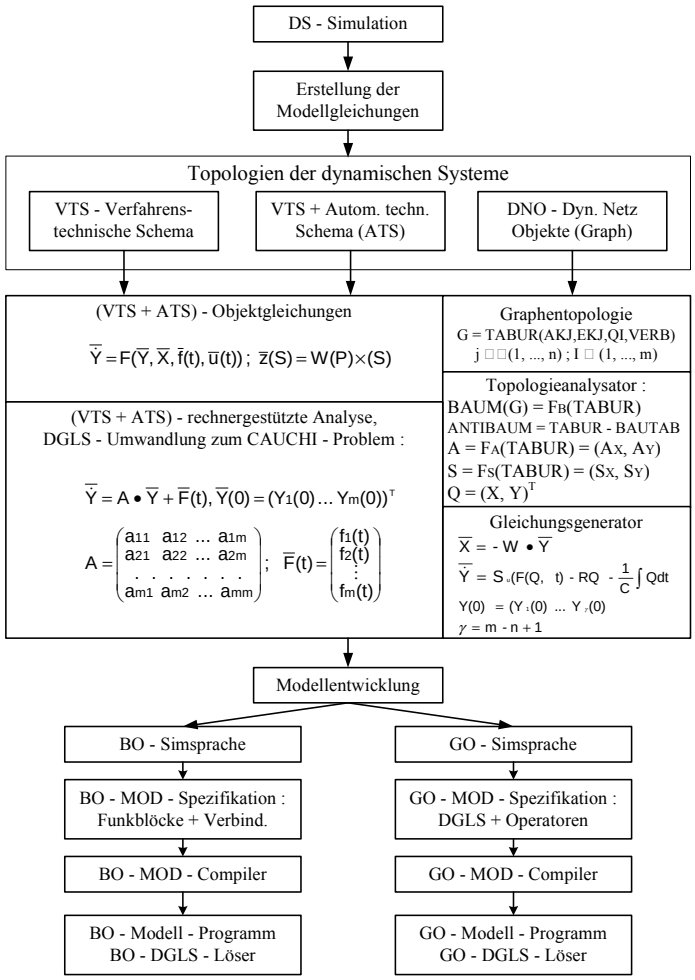

Abb. 5.1. Modellierung und Simulation von dynamischen Systemen mit den block- und gleichungsorientierten konventionellen Simulationssprachen

5.2. BO-Lösungsprinzip und die MIMD-Parallelität

Hauptkomponente der blockorientierten Simulationssprache ist ein Funktionsblock (Abb. 5.2), der programmtechnisch realisiert wird. Block beinhaltet n Eingänge, einziger Ausgang und **n** Stellen für Einführung der Koeffizienten. Ausgangsvariable wird als Ergebnis der Operation

$$Y = F(X_1, X_2, \ldots, X_n, a_1, a_2, \ldots, a_n, t) \qquad (5.1)$$

dargestellt. Für die Lösung der gewöhnlichen Differentialgleichungssysteme ist folgende Menge der Operationen notwendig

$$F \in \{a_i x_i;\ \sum_{i=1}^{n} a_i x_i;\ \int \sum_{i=1}^{n} a_i x_i dt;\ f(x_i), \varphi(x_i, x_k);\ f(t)\} \qquad (5.2)$$

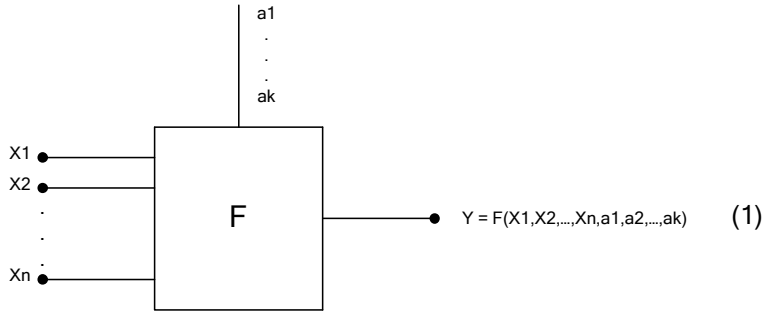

Abb.5.2. Allgemeine Darstellung des Funktionsblocks einer BO-Simulationssprache

Block-orientiertes (implizites) Modellierungsverfahren wird in eine Reihe von Schritten realisiert:

- ein Modell, z.B.

$$L(dY/dt) + X_1(t)Y|Y| = X_2(Y), \qquad (5.3)$$

wird zur Form des Simulationsmodells (implizite Funktion bezüglich Y)

$$dY/dt = [X_2(Y) – X_1(t)Y|Y|] / L \qquad (5.4)$$

umgewandelt;

- mit der Voraussetzung, dass die Variablen $X_2(Y)$ und $– X_1(t)Y|Y|$ der rechten Seite im Simulationsmodell (5.4) vorhanden sind, werden alle Operationen virtuell mit Hilfe von entsprechenden Funktionsblöcke erfüllt; damit etsteht ein Blockschaltbild der Blöcke, die miteinander entsprechend der Reihenfolge der Operationen in (5.4) verbunden werden (Abb. 5.3, Blöcke 1,...,6);

- es werden die Blöcke gesucht, die an den Ausgänge die Variablen haben, die wir am 1.Schritt als vorhandene vorausgesetzt haben (Block 4 – $X_2(Y)$, Block 6 – $-X_1(t)Y|Y|$, Abb. 5.3); mit der Berücksichtigung von +/- Zeichen werden diesen Variablen als Rückkopplungen nach Eingänge des Blocks 1 eingeführt (Abb.5.3);

- Blockschaltbild wird geprüft, möglicherweise korregiert und in PC eingeführt (die BO-Simulationssprachen SIMULINK, ISRSIM u.a. erlauben die Erstellung und Korregierung direkt am Bildschirm, ohne Zwischensprache der Blockschaltbilddarstellung zu nutzen). BO-Compiler wandelt Blockschaltbild in lauffähiger BO-Simulator um.

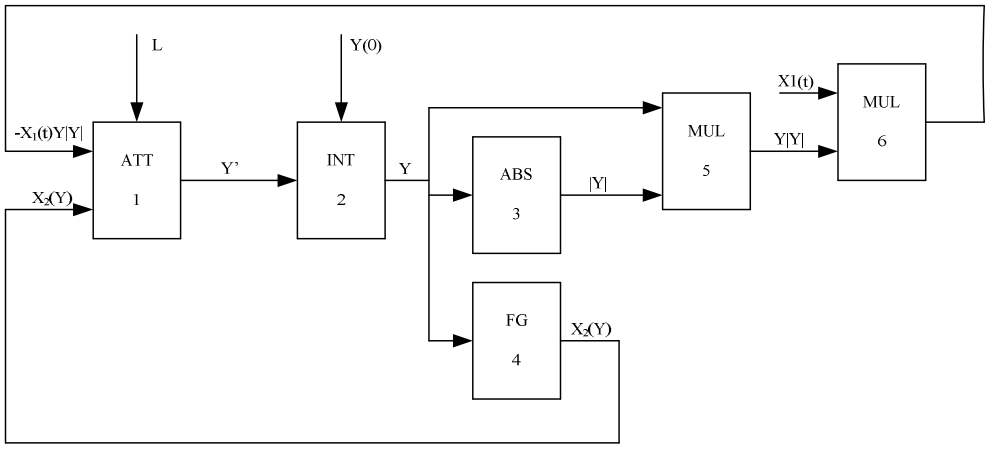

Abb. 5.3. Blockschaltbild des virtuellen Simulationsmodells, das mit den Funktionsblöcke einer BO-Simulationssprache erstellt wird

Analyse zeigt, dass BO-Lösungsprinzip entspricht der MIMD-Parallelität und kann als eine virtuelle Zuordnung „Funktionsblock – Prozess“ betrachtet werden (Abb.5.4).

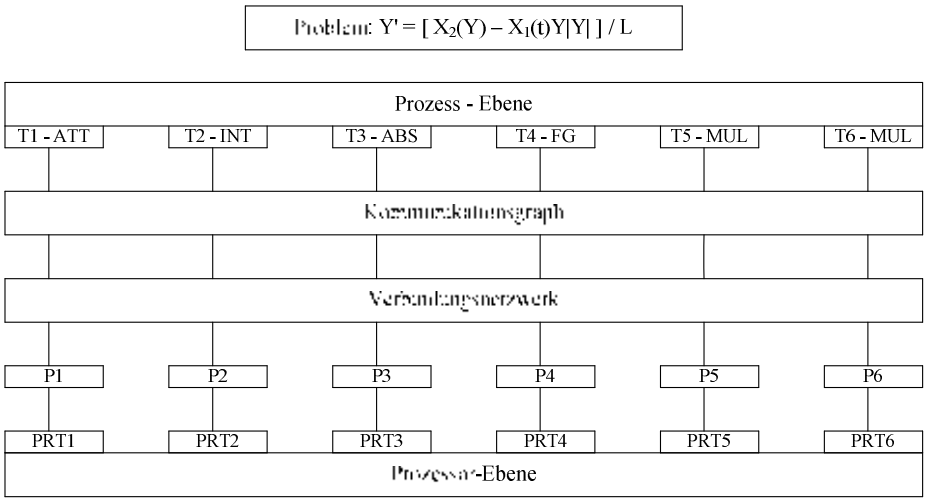

Abb.5.4 MIMD-Parallelität und BO-Verfahren

5.3 Defenition der virtuellen Paar „Funktionsblock – Prozess“

Virtueller Funktionsblock wird in Abb. 5.5 dargestellt. Dabei sind: $\overline{X}$ – Vektor der realen Eingabevariablen; KM – Kommunikationsmatrix $\overline{VE}$ – Vektor der virtuellen Eingabevariablen. MIMD-Prozess, der dem Funktionsblock entspricht, kann man mit dem Blockdiagramm darstellen (Abb.5.6).

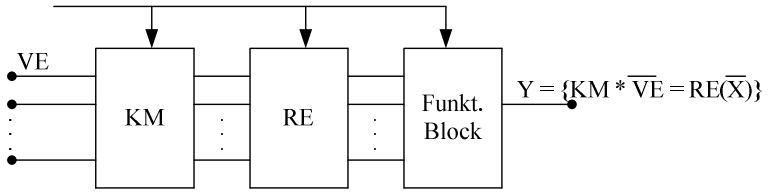

Abb. 5.5. Virtueller Funktionsblock der BO-Simulationssprache

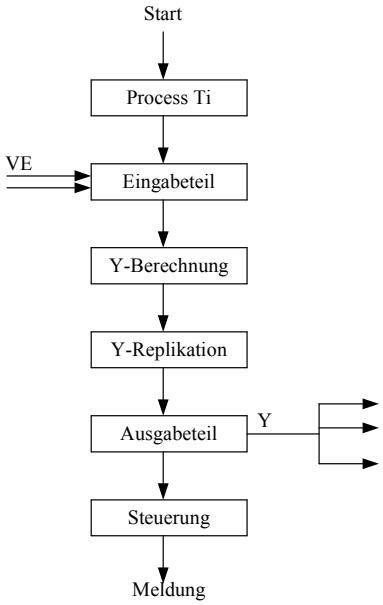

Abb. 5.6. Blockdiagramm des MIMD-Prozesses.

5.4 Abbildung der BO-Modellstruktur in virtuelle parallele Modellstruktur (VPMS)

VPMS ist eine virtuelle Prozess- und Kommunikationsstruktur. Führen wir Verbindungsvektor für Prozess Ti ein:

$$VSTi = (S_{i1}T_1\ S_{i2}T_2 \ldots S_{ik}\ T_k \ldots S_{in}\ T_n) = \underset{\substack{k=1 \\ i\in k}}{\overset{n}{V}} (S_{ik}T_k) \qquad (5.5)$$

Hier sind i – Prozessnummer, i =1,2,…,n; S_{ik} =1 – wenn die Verbindung Ti ↔ Tk vorhanden ist und 0 – wenn gibt es keine Verbindung Ti ↔ Tk.

Für k=i S_{ii} = 1, d.h. Zwischenergebnis wird im i-Prozess für weitere Berechnungen benutzt. Vektor VSTi charakterisiert virtueller Prozess Ti: einziger Ausgang und n Eingänge, die durch Sij bekommen die Variablen von Ausgängen der restlichen Prozesse.

Menge der Verbindungsvektoren für ganze BO-Struktur kann so dargestellt werden

$VST_1 = (S_{11}T_1\ S_{12}T_2 \ldots S_{1k}\ T_k \ldots S_{1n}\ T_n)$
$VST_2 = (S_{21}T_1\ S_{22}T2 \ldots S_{2k}\ T_k \ldots S_{2n}\ T_n)$
……..
$VST_k = (S_{k1}T_1\ S_{k2}T_2 \ldots S_{kk}\ T_k \ldots S_{kn}\ T_n)$ (5.6)
.,……
$VST_n = (S_{n1}T_1\ S_{n2}T_2 \ldots S_{nk}\ T_k \ldots S_{nn}\ T_n)$

Mit der Verwendung von (5.6) kommen wir zur Beschreibung der blockorientierten Struktur des MIMD-Models

$VART_1 = FUNT_1\,(S_{11}VART_1\ S_{12}VART_2 \ldots S_{1n}VART_n)$
$VART_2 = FUNT_2\,(S_{21}VART_1\ S_{22}VART_2 \ldots S_{2n}VART_n)$
::
$VART_k = FUNT_k\,(S_{k1}VART_1\ S_{k2}VART_2 \ldots S_{kn}VART_n)$ (5.7)
::
$VART_{n-1} = FUNT_{n-1}(S_{n-1,1}\,VART_1\ S_{n-1,2}VART_2 \ldots S_{n-1,n}VART_n)$
$VART_n = FUNT_n\,(S_{n1}VART_1\ S_{n2}VART_2 \ldots S_{nn}VART_n)$

Hier sind VARTi – die Ausgangsvariablen von Prozessen Ti, i = 1,2,...,n; FUNTi – die von Prozessen Ti realisierte Funktionen. Aus (5.7) folgt die Kommunikationsmatrix

$$KM = \begin{pmatrix} S_{11} & S_{12} & \ldots & S_{1n} \\ S_{21} & S_{22} & \ldots & S_{2n} \\ \ldots & \ldots & \ldots & \ldots \\ S_{k1} & S_{k2} & \ldots & S_{kn} \\ \ldots & \ldots & \ldots & \ldots \\ S_{n1} & S_{22} & \ldots & S_{nn} \end{pmatrix} \qquad (5.8)$$

die alle von Blockschaltbild bedingte Verbindungen formal darstellt.

Die rechte Seite von (5.6) lässt sich formal als die Matrix-Multiplikation generieren

$$\begin{pmatrix} S_{11}\ S_{12}\ \dots S_{1n} \\ S_{21}\ S_{22}\ \dots S_{2n} \\ \dots\dots\dots\dots\dots \\ S_{k1}\ S_{k2}\ \dots S_{kn} \\ \dots\dots\dots\dots\dots \\ S_{n1}\ S_{n2}\ \dots S_{nn} \end{pmatrix} * \begin{pmatrix} T_1 & 0 & \dots & 0 \\ 0 & T_2 & \dots & 0 \\ \dots\dots\dots\dots\dots & & & \\ 0 & 0 \dots T_k & \dots & 0 \\ \dots\dots\dots\dots\dots & & & \\ 0 & 0 & \dots & T_n \end{pmatrix} \qquad (5.9)$$

Die zusammengestellte Vektoren $VST_1 \dots VST_n$ sind die Zeile der Modellzustandsmatrix (MZM):

$$MZM = KM * DT \qquad (5.10)$$

Dabei ist DT eine Diagonalmatrix der T – Prozesse.

5.5 Devirtualisierung der VPMS

In Abb.5.7 sind qualitativ die Lastcharakteristiken der Funktionsblöcke gezeigt.

Abhängig von Art der mathematischen Operationen und numerischen Verfahren der Implementierung jeder Block besitzt eigenes Realisierungsprogramm, das für MIMD-Prozess bestimmter Last bedeutet. Prinzipiell ist Struktur der BO-MIMD-Prozesse in Abb.5.4 lauffähig, aber die Prozesse haben ungleichmäßige Arbeitsumfänge. Deshalb bei der Devirtualisierung soll umfangreiche Lastbalancierung des Blockschaltbildes durchgeführt werden. Dem minimalkörnigen Prozess (MKP) soll eine Funktionsblockkette zugeordnet werden, die eine Differentialgleichung der ersten Ordnung löst. Diesen Ketten werden vom Generator des BO-Lösers nach entsprechenden Algorithmen erstellt.

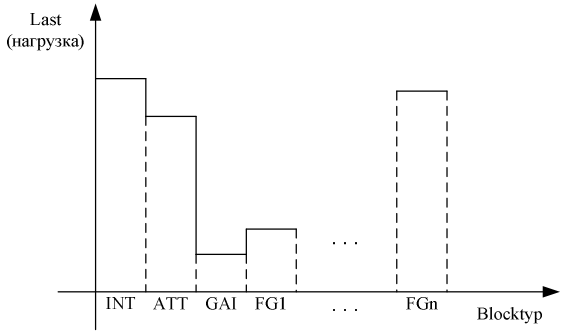

Abb. 5.7. Zur Lastanalyse der VM-Blockeinheiten

So entsteht die von dem BO-Lösungsprinzip bedingte virtuelle Blockschaltung, die das virtuelle parallele Simulationsmodell der ersten Parallelitätsebene (PE-1) mit minimal körnigen Prozessen „realisiert“. Das virtuelle Verbindungsnetzwerk (VNW) hat zwei Ebene: innere MKP-Verbindungen sowie äußere, die alle MKP entsprechend der Approximationschema verbinden.

Für die gut strukturierten Simulationsprobleme ist oft möglich eine Matrix-Vektor-Darstellung der Gleichungssysteme zu erstellen, z.B. für dynamisches Netzobjekt

$$X = -WY \qquad (5.11)$$

$$Y' = S_uH - S_uRZ - S_uRR(t)Z$$

mit folgenden Bezeichnungen: X,Y – Luftstromvektoren in Baum- und Antibaumzweigen; Su – umgewandelte Maschenmatrix; R, RR(t) – diagonale Parametermatrizen; Z – Vektor mit den Komponenten Xj|Xj|,Yk|Yk|, i =1,2, ..,n-1; k = 1,2,.., γ; γ = m-n+1; m, n – die

Kanten- und Knotenmenge; H – Vektor der Ventilatorcharakteristiken; W – berechnete Topologiematrix.

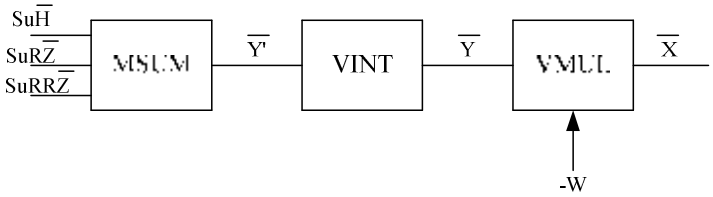

Abb.5.8. Matrix-Vektor-Funktionsblöcke im Einsatz für die Lösung des Gleichungssystems (5.11)

Suche der gleichmäßigen eindimensionalen und Matrix-Vektor-Lastketten beinhaltet folgende Operationen:

a)Auswahl der lastmaximalen Blöcken(LMB);

b)Lastanalyse der Blockketten;

c)Erstellung der Modellstruktur mit den lastgleichen Blockmodulen.

Dann wird eine Abbildung der Modellstruktur mit den lastgleichen Blockmodule in Prozessstruktur durchgeführt: Zuordnung “Blockmodule-Prozesse”; Erstellung der Prozessstruktur mit der Definition der Verbindungen zwischen den Prozesse. Struktur der Matrix-Vektor-Prozesse wird ähnlich erstellt. Die Verbindungen zwischen den Prozessen können durch Topologie-und Kommunikationsmatrizen (wie in (5.11), Matrix W) definiert werden.

Der Devirtualisierungsvorgang soll vom vorgeschlagenen in der VPSU-Simulationssoftware eingebauten BO-Devirtualisator durchgeführt werden. Als Randbedingungen hat er die Parallelitätsebene PE-1 und Zielrechnersystem ZRS. Die optimale Zuordnung dem ZRS wird mit der Berücksichtigung der höheren Parallelitätsebenen etwa PE-2 – PE-4 und der Abbildung ihren virtuellen Verbindungsnetzwerke VNW-2 – VNW-4 in dem ZRS-Verbindungsnetwerk realisiert. Diese Zuordnung benutzt Code-Generator, der ein lauffähiges MPI-Programm generiert.

6. Zusammenfassung und Ausblick

1. Stand der Entwicklung der parallelen Simulationstechnik und die im Rahmen der Zusammenarbeit von HLRS und FCWT erzielte Ergebnisse erlauben uns ein TEMPUS-Projekt zu konzipieren, das der Hauptziel „Modernisierung der Lehre an der ukrainischen technischen Universitäten durch Anwendung der Ansätze und Methoden der parallelen Simulationstechnik (ParSimTech)“ verfolgt. Es wurden die Projektarbeiten bezüglich der Infrastruktur und methodischen Unterlagen in eine Reihe von Teilprojekten gegliedert.

2. Entwicklung der verteilten parallelen Smulationsumgebung wird als zentrales ParSimTech-Problem angesehen. Deshalb im Teilprojekt TTP1 wird die VPSU-Dekomposition auf 10 Subsysteme betrachtet. Es wurde allgemeine Methodik der Subsystemerarbeitung und Inbetriebnahme vorgeschlagen. TTP1 wird in der TEMPUS-Vorbereitungsphase starten und der ParSimTech-Spezialisierung der FCWT-Absolventen-2011 beitragen.

3. Es wurden aktuelle Arbeiten der Entwicklung von parallelen DNOVP-Simulatoren fortgesetzt. Als erste Etappe der Aufbau von Gleichungslödern wurden die diskreten virtuellen Simulationsmodelle aufgrund des blockartigen numerischen Verfahrens (BAV-DSM) erstellt und die Aufgaben für den Gleichungsgenerator definiert.

4. Vorgeschlagenes Entwicklungskonzept der block-orientierten (BO) parallelen Simulationssprache wird die Lösung des Problems der Erhöhung von VPSU-Benutzerfreundlichkeit dienen.

Die weiteren Aktivitäten werden sich auf der Beantragung des TEMPUS-Projektes und systematischen Implemetierungen von VPSU-Subsystemen, parallelen BAV-Gleichungslöser, Apriori-Analyse von virtuellen parallelen BAV-DSM sowie Anwendungen in der Lehre und im Simulations-Service-Zentrum konzentrieren.

7. Literaturverzeichnis

1. Zeitz, M.: Simulationstechnik.In:Chemie-Ingenieur-Technik, 59, 1987, S.464 – 469
2. Anoprienko A.J., Svjatnyj V.A., Bräunl T., Reuter A., Zeitz M.: Massiv parallele Simulationsumgebung für dynamische Systeme mit konzentrierten und verteilten Parametern. 9. Symposium ASIM’94, Tagungsband, Vieweg, 1994, S. 183-188.
3. Schmidt B.: Simulationssysteme der 5. Generation. SiP, Heft 1, 1994, S. 5-6.
4. Feldmann L.P., Lapko V.V., Svjatnyj V.A., Trub I. I., Bräunl T., Reuter A., Zeitz M.: Algorithmen einer massiv parallelen Simulationsugebung für dynamische Systeme mit verteilten Parametern. 10. Symposium ASIM’96, Tagungsband, Vieweg, 1996, S. 519-524.
5. Gilles, E.-D.: Modellierungssystematik für verfahrenstechnische Prozesse. In: Abschlussbericht "Methoden zur Modellierung und Berechnung der Dynamik verfahrenstechnischer Prozesse" der DFG-Forschergruppe Universität Stuttgart, 1996, S.7-22.
6. Feldmann L.P. Svjatnyj V.A., Bräunl T., Reuter A., Zeitz M.: Implementierung und Effizienzanalyse der parallelen Simulationsalgorithmen für dynamische Systeme mit verteilten Parametern (Plenarvortrag). 11. Symposium ASIM’97, Tagungsband, Vieweg, 1997, S. 38-47.
7. Borchard J.: Newton-type decomposition methods in large-scale dynamic process simulation. In: Computers and Chemical Engineering, 25 (2001), 951-961.
8. Svjatnyj V.A., Nadeev D.V., Solonin O. M., Rothermel K., Zeitz M.: Subsysteme einer verteilten parallelen Simulationsumgebung für dynamische Systeme. 16. Symposium ASIM 2002, Tagungsband, 2002, S. 64 – 69.
9. Modelica – A Unified Object-Oriented Language for Physical Systems Modeling. Language Specification. Version 2.0, 2002.
10. Svjatnyj V.A., Moldovanova O.V., Cheptsov O.O., Rothermel K., Zeitz M. Generierung und parallele Lösung von Simulationsmodellen für Netzobjekte mit verteilten Parametern. In: R.Hohmann (Hrsg.), Tagungsband 17. ASIM-Symposium Simulationstechnik, Magdeburg 2003, SCS 2003, 193-198.
11. Svjatnyj V.A., Moldovanova O.V., Feldmann L.P.: Parallele Simulationsumgebung für dynamische Netzobjekte mit verteilten Parametern. In:

F.Hülsemann u.a. (Hrsg.), Tagungsband 18. ASIM-Symposium Simulationstechnik, Erlangen 2005, SCS 2005, 416-421.

12. Svjatnyj V., Beljaev O., Lapko V., Tschepzov O., Hohmann R.: Zur Entwicklungsorganisation des Simulations- und Servicezentrums für die Kohleindustrie. In: F.Hülsemann u.a. (Hrsg.), Tagungsband 18. ASIM-Symposium Simulationstechnik, Erlangen 2005, SCS 2005, 554-559.

13. Moldovanova O.V., Svjatnyj V.A., Feldmann L., Resch M., Küster U.: Problemorientierte parallele Simulationsumgebung. In: DonNTU, FRTI-Werke, Reihe “Informatik, Kybernetik und Rechentechnik”, Band 93. – Donezk, 2005. – S. 145–150.

14. Svjatnyj V., Moldovanova O., Smagin A., Resch M., Keller R., Rabenseifner R.: Virtuelle Simulationsmodelle und ein Devirtualisierungsvorgang für die Entwicklung der parallelen Simulatoren von komplexen dynamischen Systemen. In: DonNTU, FRTI-Werke, Reihe “Probleme der Modellierung und rechnergestützten Projektierung von dynamischen Systemen”, Band 5(116). – Donezk, 2006. – S. 36–43.

15. Bondareva, K., Svjatnyj, V.: Verfahrenstechnisch orientierte parallele Simulationsumgebung. In: DonNTU, FRTI-Werke, Reihe “Probleme der Modellierung und rechnergestützten Projektierung von dynamischen Systemen”, Band 5(116). – Donezk, 2006. – S. 28–35.

16. Bönisch Th., Alessandrini V.: DEISA overview: projekt status, strategies and perspectives. In: Sixth DEISA Training Session, HLRS, Stuttgart, March 5-7, 2008, P.1-49.

17. Nickols, J., Buck, I., Gerland, M. Scalable Parallel Programming with CUDA. *ACM Queue,6,* 2, 2008, 40-53

18. Feldmann L.P. Implementierung und Effizienzanalyse von parallelen blockartigen Simulationsalgorithmen für dynamische Systeme mit konzentrierten Parametern. In: Möller, D.P.F. (Hrsg.): Tagungsband 14. ASIM-Symposium Simulationstechnik in Hamburg, September 2000, SCS-Europe BVBA, Ghent/Belgium 2000, S. 241-246

19. Feldmann L.P., Dmitrieva O.A., Gerber S.. Abbildung der blockartigen Algorithmen auf Parallelrechnerarchitekture. In: Tavangarian,D., Grützner,R. (Hrsg.): Tagungsband 15. ASIM-Symposium Simulationstechnik in Rostock, September 2002, SCS-Europe BVBA, Ghent/Belgium 2002, S.359-364.

20. Feldmann, L.P.: Parallele Simulationsalgorithmen für die mit den gewöhnlichen Differentialgleichungen beschreibende dynamische Systeme. In: Elektronische Modellierung, Band 26, N1, 2004, S.19-30 (Russisch: Фельдман Л.П. Параллельные алгоритмы моделирования динамических систем, описываемых обыкновенными дифференциальными уравнениями. //Электронное моделирование, том 26, № 1, 2004.- С. 19-30).

21. Feldmann, L.P., Nasarova, I.A.: Parallele Algorithmen der numerischen Lösung von Caushy-Aufgabe für gewöhnlichen Differentialgleichungssysteme. In: Mathematische Modellierung, Band 18, N6, 2006, S. 17-31. (Russisch: Фельдман Л.П., Назарова И.А. Параллельные алгоритмы численного решения задачи Коши для систем обыкновенных дифференциальных уравнений. //Математическое моделирование, том 18, №6, 2006.- С.17-31).

22. Feldmann, L.P.: PARALLELE ALGORITHMEN DER NUMERISCHEN LÖSUNG VON CAUSHY-AUFGABEN FÜR GEWÖHNLICHEN

DIFFERENTIALGLEICHUNGEN. Die Vortragspräsentation für HLRS, Februar 2009. **(**Übersetzung von Prof. V.A. Svjatnyj**)**

23. Moldovanova , O.V.: Ansätze zur Organisation von parallelen Berechnungen in den Simulationsaufgaben der Grubenbewetterungsnetze. Dissertationsarbeit, Donezk, 2008, (Russisch)

24. Guseva, A.B.: MIMD-paralleler Gleichungslöser für dynamische Netzobjekte mit verteilten Parametern. Magisterarbeit. FRTI DonNTU, Donezk, 2007, (Russisch).

25. Guseva, G.B., Moldovanova, O.V.: MIMD-paralleler Gleichungslöser für dynamische Objekte mit verteilten Parametern. In: Probleme der Modellierung und rechnergestützten Projektierung von dynamischen Systemen: DonNTU-Werke, Band 6 (127) – Donezk, 2007. – S. 149-158 (Ukrainisch).

26. Feldmann L.P., Resch M., Svjatnyj V.A., Zeitz M.: Forschungsgebiet: parallele Simulationstechnik. In: ASIM'2009, Tagungsband der Vorträgekurzfassungen, Cottbus, 2009, S.

27. Feldmann L.P., Resch M., Svjatnyj V.A., Zeitz M.: Forschungsgebiet: parallele Simulationstechnik. In: DonNTU, FRTI-Werke, Reihe "Probleme der Modellierung und rechnergestützten Projektierung von dynamischen Systemen", Band 9(150). – Donezk, 2008. – S. 9-36.

28. Svjatnyj V.A.: Probleme paralleler Simulationstechnik. Realisierung von virtuellen parallelen DNOVP-Simulationsmodellen. Bericht über HLRS-Forschungsaufenthalt-2009, März 2009, HLRS, Stuttgart.