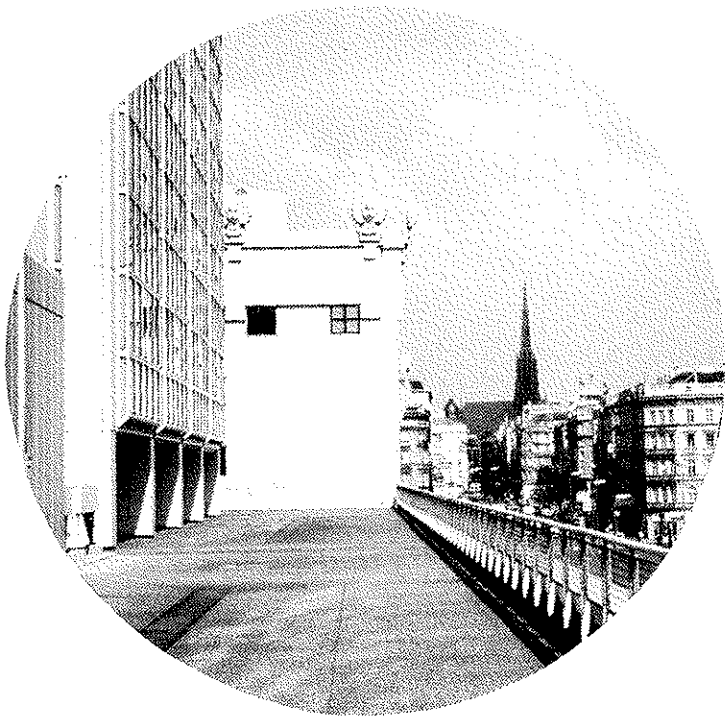
Interface

herausgegeben von der
Abt. Hybridrechenanlage (Simulationsrechenanlagen)
des EDV-Zentrums der
Technischen Universität Wien

Nummer 24
April 1988



Das neue Institutsgebäude der Technischen Universität
im Zentrum Wiens

Inhaltsverzeichnis

Redaktion: Irmgard Husinsky

Abt. Hybridrechenanlage (Simulationsrechenanlagen) des EDV-Zentrums der Technischen Universität Wien, A-1040 Wien, Wiedner Hauptstraße 8-10.

Herausgeber, Verleger, Hersteller: EDV-Zentrum der Technischen Universität Wien, Abteilung Hybridrechenanlage (Simulationsrechenanlagen), Leitung: Dipl. Ing. Dr. W. Kleinert, A-1040 Wien, Wiedner Hauptstraße 8-10. Tel.: (0222) 58801 5481. Ttx: (232) 3222467=TUW, Modem (Telefax): (0222) 5871020. Druck: Hochschülerschaft Technik, A-1040 Wien, Argentinierstraße 8.

Wir sind übersiedelt!

Die Übersiedlung der Simulationsrechenanlagen von der Gußhausstraße in das neue Gebäude auf den Freihausgründen fand im August 1987 statt.

Obwohl ein besonders großer und schwerer Teil des EAI SIMSTAR Simulationsmultiprozessors mit Hilfe eines Kranes durch ein ausgebautes Gangfenster aus dem Gebäude in der Gußhausstraße abtransportiert werden mußte, verlief der Transport ohne Zwischenfälle.

Die Demontage, der Transport und die Wiederinstallation im neuen Maschinenraum erfolgten plangemäß während der Sommerferien.

Im neuen Institutsgebäude stehen für das Personal und die Rechner endlich ausreichende Räumlichkeiten zur Verfügung.

Zu Beginn des Wintersemesters waren sowohl der EAI SIMSTAR als auch das hybride AutoPATCH System EAI PACER 600A wieder für alle Benützer über das lokale TUNET im Time-Sharing-Betrieb zu erreichen.

Unsere neue Adresse lautet:

Wiedner Hauptstraße 8-10
A-1040 Wien

Der Zugang zu unseren Räumen befindet sich im 2. Obergeschoß im gelben Bereich (Turm B).

Betrieb an der Hybridrechenanlage (Simulationsrechenanlagen)

Das Simulationsrechenzentrum bietet volle Unterstützung und Beratung bei der Lösung von Simulationsproblemen an: von der Modellbildung über die Implementierung (Wahl der Simulationswerkzeuge) bis zur Experimentierphase.

Unsere Abteilung umfaßt derzeit folgende Simulationsrechensysteme:

- *Das EAI PACER-AutoPATCH Hybridsystem mit der Simulationssprache HYBSYS. Dieses System, dessen erste Komponenten bereits 1969 installiert wurden, wird seit dem Frühjahr dieses Jahres nicht mehr verbessert. Es ist aber weiter betriebsbereit. Die Anzahl der User wurde sowohl für die Time-Sharing Sessions als auch für die Jobs begrenzt.*

- *Als wichtigstes Simulationswerkzeug wird nun der EAI SIMSTAR verwendet, welcher eine erheblich genauere Rechenleistung bietet. Als Software steht am SIMSTAR ein PTRAN-HYBSYS Simulationssystem zur Verfügung. Dem SIMSTAR vorgeschalten ist ein eigener UNIX-Rechner namens TUNIX, welcher den SIMSTAR von seinen nicht auf die Simulation bezogenen Tätigkeiten entlasten soll. Dieser Rechner übernimmt unter anderem das Aufbereiten der Modelle, den Transfer zum SIMSTAR sowie die Präsentation der Ergebnisse und den interaktiven Verkehr. Außerdem übernimmt TUNIX gewisse Spooler- und Backup-Funktionen. An diesem zentralen Frontendsystem stehen ein Banddrucker, ein Matrixdrucker und ein Laserdrucker für graphische Ausgabe zur Verfügung.*

- *Außerdem betreuen wir das digitale Simulationssystem ACSL auf der CYBER CDC 860 der Abt. Digitalrechenanlage. Für besonders rechenintensive Simulationen wird ACSL auf der NAS AS/9160 des IEZ installiert werden. Die Simulationssprache SYSMOD wird auf einer MicroVAX und auf der NAS AS/9160 installiert werden. Für einzelne Pakete wie ACSL und SIMAN gibt es PC-Versionen, welche an der Abt. Hybridrechenanlage (Simulationsrechenanlagen) als Hochschullizenzen erhältlich sind.*

Für die Betreuung der Lehr- und Forschungsanwendungen an den Simulationsrechenanlagen stehen die Mitarbeiter der Abteilung Hybridrechenanlage (Simulationsrechenanlagen) für Programmberatung, Kundenbetreuung und Problemanalyse zur Verfügung.

Umfangreiche Dokumentationen über die Betriebssysteme der einzelnen Rechenanlagen bzw. Benützerhandbücher für die Simulationssprachen liegen auf und werden laufend verbessert und erweitert.

Der Zugang zu den einzelnen Simulationsrechenanlagen erfolgt primär über das hochschulinterne TUNET, soferne der gewünschte Anschluß an das entsprechende Rechensystem hergestellt werden kann. Ansonsten besteht auch die Möglichkeit, im Terminalraum der Abt. Hybridrechenanlage (Simulationsrechenanlagen) eine der Anlagen zu benützen. Dort stehen vor allem auch die erforderlichen graphischen Terminals bereit. Für die Einbindung entsprechender externer analoger Daten an eines der Simulationsrechensysteme steht im Haus eine Anschlußmöglichkeit zur Verfügung.

An der Abt. Hybridrechenanlage (Simulationsrechenanlagen) wird an einem Forschungsprojekt LATOUR gearbeitet, dessen Ziel es ist, die Simulationsaufgaben mit der herkömmlichen Benutzeroberfläche auf einer speziell entwickelten Hardware bestehend aus mehreren digitalen Knotenrechnern, kontrolliert durch ein UNIX-System, aufzubringen. Neben der Multiprozessor-Hardware wird der entsprechende parallele Compiler dazu entwickelt.

A. Blauensteiner

Rechnen an den Anlagen der Hybridrechenanlage (Simulationsrechenanlagen)

Benützung der Simulationsrechensysteme

An der Abteilung Hybridrechenanlage (Simulationsrechenanlagen) des EDV-Zentrums der Technischen Universität Wien stehen derzeit als Simulationsrechensystem der Simulationsmultiprozessor EAI SIMSTAR mit dem Frontend-Prozessor TUNIX und das EAI PACER AutoPATCH-Hybridsystem zur Verfügung. Für die Benützung dieser Rechenanlagen ist eine Benützungsberechtigung erforderlich, die an der Abteilung Hybridrechenanlage (Simulationsrechenanlagen) beantragt werden muß. Ein entsprechendes Formular ist beim Operator (Wiedner Hauptstr. 8-10, 2.OG, Turm B, Zimmer DB02004, Tel. 588 01 - 5481 DW) erhältlich und auch dort ausgefüllt abzugeben. Die Benützungsbewilligung wird im allgemeinen noch am gleichen Tag im betreffenden System eingerichtet.

Öffnungs- und Betriebszeiten

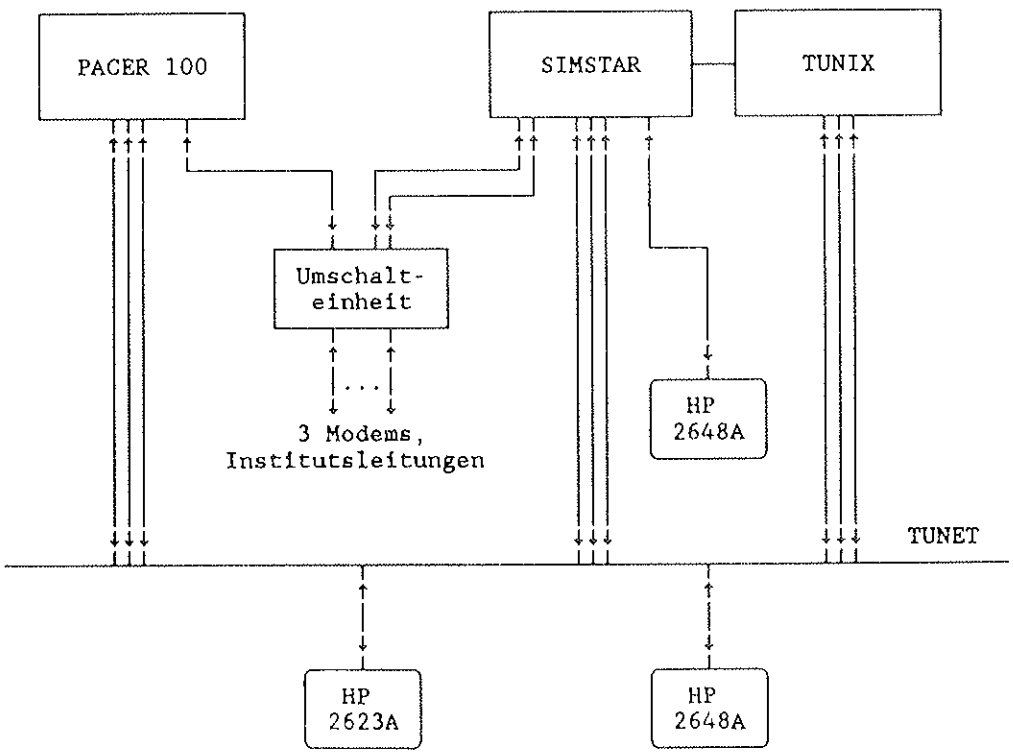
Das Simulationsrechenzentrum ist

Montag bis Donnerstag	von 8 bis 18 Uhr
Freitag	von 8 bis 16 Uhr

geöffnet. Während dieser Zeiten kann an allen Simulationsrechensystemen uneingeschränkt gerechnet werden. Bei Problemen mit der Herstellung der Verbindung zum Rechner kann der diensthabende Operator (Tel. 588 01 - 5481 DW) kontaktiert werden.

Terminal-, TUNET- und Modem-Anschlüsse

Die Rechenanlagen können über TUNET-Anschlüsse, direkteTerminalleitungen und Modem-Anschlüsse benützt werden.



Von institutseigenen Terminals, die an das TUNET angeschlossen sind, können die Rechner mit den Prozeduraufrufen

DO SIM	für den SIMSTAR
DO TUNIX	für den TUNIX
DO HYB	für den PACER

angewählt und während der oben genannten Betriebszeiten benützt werden.

Für Benützer, die nicht die Möglichkeit haben, an einem Institutsterminal zu rechnen, stehen während der Betriebszeiten im Terminalraum der Abteilung Hybridrechenanlage (Simulationsrechenanlagen) drei graphische Terminals zur Verfügung:

HP 2623A	am TUNET
HP 2648A	am TUNET
HP 2648A	am SIMSTAR

Benützer, die an einem dieser Terminals rechnen wollen, müssen beim Operator anläuten (Torsprechanlage Rufnummer 68), der ihnen dann den Zutritt zum Terminalraum ermöglicht. Nach dem Beenden der Arbeiten im Terminalraum sollte ebenfalls wieder der Operator verständigt werden. In Zeiten stärkeren Andrangs kann eine Reservierung der drei Benützerterminals beim Operator vorgenommen werden.

Zusätzlich zum TUNET und den direkten Terminalanschlüssen besteht die Möglichkeit, den SIMSTAR und PACER über einen Modem-Anschluß (300 Baud) zu verwenden. Dafür stehen die folgenden Telefonnummern zur Verfügung:

TU-interne Hausklappen	193, 194, 195
Wiener Telefonnummer	587 10 20

Vor dem Rechnen über Modem ist eine telefonische Anmeldung beim Operator erforderlich, da das betreffende Modem unter Umständen erst auf den gewünschten Rechner umgeschaltet werden muß.

F. Blöser

Diskrete Simulation mit SIMAN

F. Breitenecker
Institut für Analysis, Technische Mathematik und Versicherungsmathematik

I. Husinsky
EDV-Zentrum, Abt. Hybridrechenanlage (Simulationsrechenanlagen)

Verfügbarkeit

SIMAN von Systems Modelling Corporation ist als kombinierte Simulationssprache konzipiert (d.h. für diskrete und kontinuierliche Simulationen). Ihr Hauptanwendungsgebiet ist die prozeßorientierte diskrete Simulation. Derartige Modelle treten z.B. in der Fertigungstechnik, bei der Jobverfolgung in Rechensystemen, etc. auf.

SIMAN ist in der Microcomputer-Version an der Abt. Hybridrechenanlage (Simulationsrechenanlagen) des EDV-Zentrums der Technischen Universität Wien auf IBM-kompatiblem PC installiert. Für den internen Gebrauch in der Lehre an der Technischen Universität wird das Programmpaket in der Microlab-Version mit Dokumentation gegen einen Unkostenbeitrag weitergegeben.

Voraussetzungen für die Installation des SIMAN Microlab Package sind:

- IBM-kompatibler PC mit MS-DOS 2.0 oder höher
- min. 384 KB Memory
- Disketten-Drive und Hard Disk oder 2 Disketten-Drives

Empfohlen wird

- mathematischer Coprocessor
- Microsoft FORTRAN oder C-Compiler Version 4.0 oder höher, falls kombinierte diskrete und kontinuierliche Systeme modelliert werden sollen
- IBM CGA oder EGA-kompatible Graphikkarte für die graphische Eingabe-Utility BLOCKS

Das SIMAN Microlab Package enthält alle SIMAN-Prozessoren (siehe Softwarestruktur) sowie die Eingabe-Utility BLOCKS. Die Anzahl der Blöcke pro Modell ist auf 50 beschränkt. Ein Reference Manual und das Buch "Introduction to SIMAN" liefern ausführliche Dokumentation.

Prozeßorientierte diskrete Simulation mit SIMAN

Das Simulationsproblem wird in zwei Teilen vorgegeben: Modellrahmen und Experimentierrahmen.

Im Modellrahmen (Model Frame) erfolgt die Beschreibung der statischen und dynamischen Charakteristiken des Systems. Das System wird durch Teilprozesse modelliert, die durch vordefinierte Blöcke angegeben werden. "Entities" sind die interessierenden Größen, die sequentiell durch den Prozeß laufen (z.B.: Kunden, Werkstücke). Beispiele für Blöcke sind: Generierung von Entities, Attributzuweisung, Verzögerung, Warteschlangen, Belegen und Freigeben von Resourcen, Verzweigungen, Beobachtungspunkte, etc. Eine Utility ermöglicht einfache interaktive menügesteuerte Eingabe der Blöcke.

Der Experimentierrahmen (Experimental Frame) enthält die Experimentierbedingungen zur Exekution des Modells. Dies sind z.B. Werte wie Anfangsbedingungen, Kapazitäten von Resourcen, Bearbeitungszeiten von Entities, Tabellen, Prioritäten, Maximalwerte, Simulationszeit, etc.

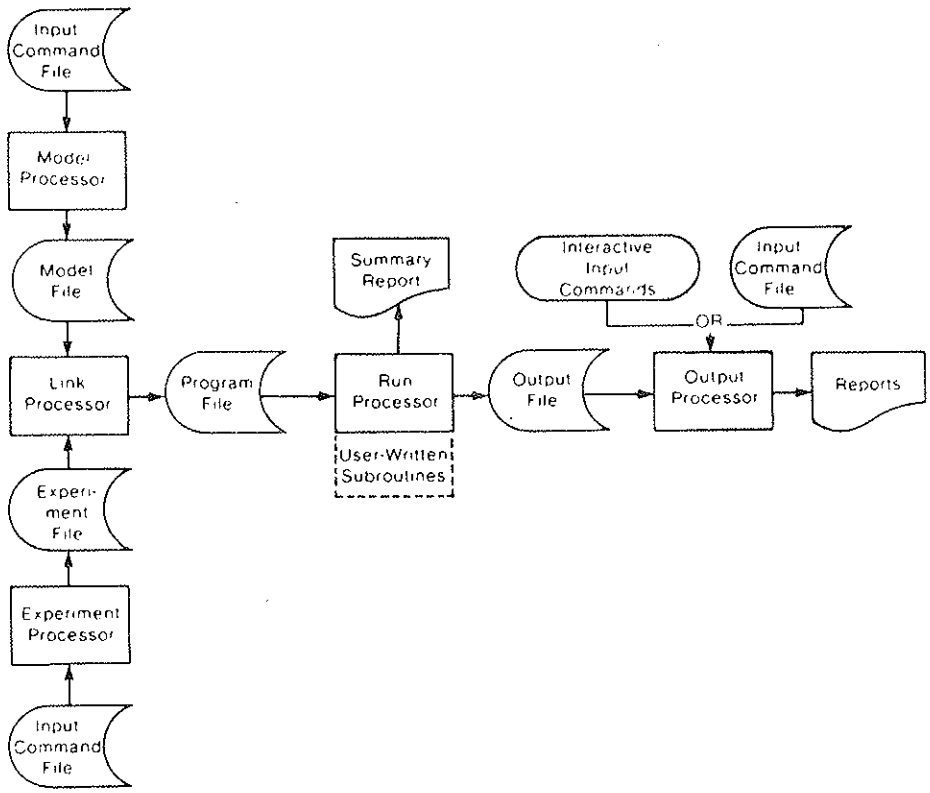
Ferner besteht die Möglichkeit, FORTRAN oder C-Subroutinen zum Modell zu laden, wo auch Modellgleichungen für kontinuierliche Systeme dazukodiert werden können.

Während des SIMAN-Laufes werden die interessierenden Variablen aufgezeichnet. Als Ergebnis liefert der SIMAN Summary Report Statistiken.

Mithilfe des interaktiven Output Prozessors können die Simulationsergebnisse in Form von Kurven, Tabellen, Histogrammen, etc. dargestellt werden.

Error Messages in Textform, Trace-Möglichkeiten und ein Debugger erleichtern das Auffinden von Fehlern bei der Programmierung.

Softwarestruktur



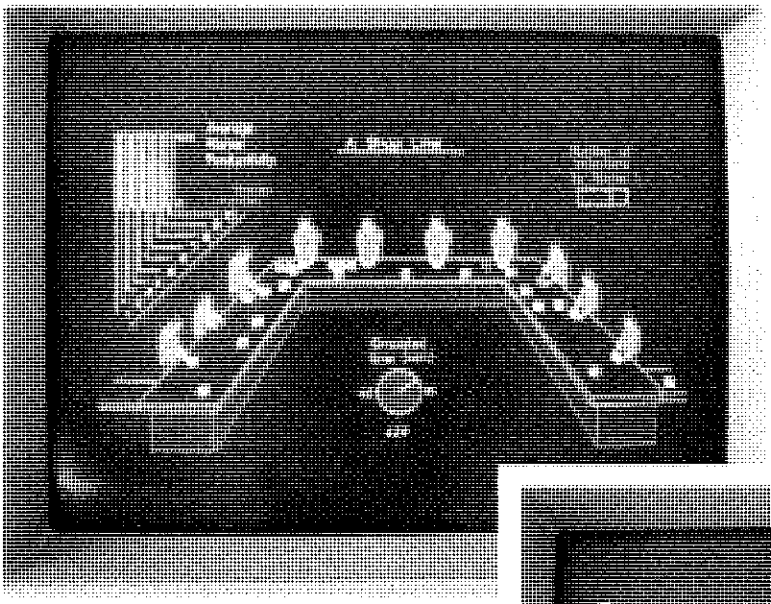
- Model Prozessor MODEL: verarbeitet Modellrahmen zu Model File
- Experiment Prozessor EXPMT: verarbeitet Experimentierrahmen zu Experiment File
- Link Prozessor LINK: kombiniert Model File, Experiment File und eventuelle Subroutinen zu Program File
- Run Prozessor SIMAN: führt Simulationsläufe durch, schreibt Ergebnisse auf Output File
- Output Prozessor OUTPUT: analysiert, formatiert und stellt Daten des Output Files dar

Animation von Simulationsergebnissen

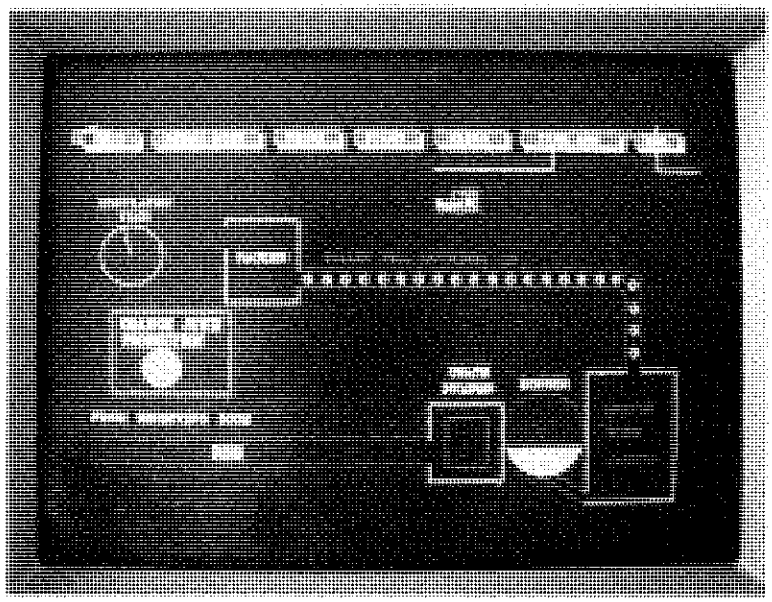
An der Abt. Hybridrechenanlage (Simulationsrechenanlagen) des EDV-Zentrums und an der Abt. für Regelungsmathematik, Hybridrechen- und Simulationstechnik des Instituts für Analysis, Technische Mathematik und Versicherungsmathematik ist auch eine erweiterte Version von SIMAN, das Professor's Package, installiert. Hier können Modelle mit bis zu 500 Blöcken gerechnet werden. Außerdem können SIMAN-Simulationen auf dem Bildschirm mithilfe des CINEMA/EGA Systems animiert werden. Dieses System enthält einen graphischen Editor, der es erlaubt, auf einem EGA-Schirm menü- und mausgesteuert ein Layout zur graphischen Repräsentation eines SIMAN-Modells zu erstellen.

Eine spezielle Version von SIMAN erzeugt dann aus dem SIMAN-Modell und dem Layout eine Echtzeit-Animation.

Z.B.:



Animation einer Fließbandarbeit



Animation einer Orangenjuiceproduktion

SIMAN in der Lehre

Die diskrete Simulationssprache SIMAN wurde in der Vorlesung und in den Übungen "Diskrete Simulationssysteme" (Wahlfach für Technische Mathematiker, ab 4. Sememester) und in der Vorlesung und in den Übungen "Simulationssprachen" (allgemeine Lehrveranstaltung) verwendet.

Einerseits wurde mit SIMAN das Prinzip der prozeßorientierten Simulation dargestellt, andererseits arbeiteten die Studenten in den Übungen mit dieser Sprache an einfachen Beispielen. Die Software (das Microlab Package) wurde dankenswerterweise vom Simulationsrechenzentrum zur Verfügung gestellt.

Während in der ersten Vorlesung das Modellieren mit SIMAN in ein allgemeines Konzept über diskrete Simulation eingebettet war, wurde in der zweiten der mit dem Microlab Package mitgelieferte SIMAN-Kurs verwendet. Dieser ist ein Skelettskriptum, das sich auch für Folien eignet, und ist sehr gut als Intensivkurs zu verwenden (die Vorlesung war eine Blocklehrveranstaltung). Zum Selbststudium ist das mitgelieferte Buch "Introduction to SIMAN" von C. Dennis Pegden sehr gut geeignet.

Nach den Erfahrungen ist festzustellen, daß sich SIMAN sowohl für Anfänger und für kleine Modelle eignet, als auch für "harte" aufwendige diskrete Simulationen (einige Studenten verwenden SIMAN in Praktika weiter). Allerdings ist hier sehr bald die Grenze des Microlab Package erreicht (50 Blöcke - das ist wenig, da die Blöcke teilweise sehr elementar sind).

SIMAN erhebt teilweise den Anspruch darauf, eine kombinierte Simulationssprache zu sein, also auch dynamische Prozesse simulieren zu können. Bei der Beschreibung der Differentialgleichungen muß man dabei allerdings auf die FORTRAN-Ebene hinabsteigen (wie übrigens auch bei der Ereignissimulation) und ist sehr starken Einschränkungen unterworfen. Mit SIMAN arbeitet man sehr gut im Bereich der prozeßorientierten Simulation, Ereignis- und vor allem kontinuierliche Simulation sind mehr oder weniger Zusatzteile auf FORTRAN-Level, die man zur prozeßorientierten Simulation mitverwendet, wenn nötig.

Wer von kontinuierlicher Simulation kommt, dem fällt zunächst die mangelnde Unterstützung im Experimentierrahmen auf. Dieser ist nicht interaktiv, man hat ein File mit der Beschreibung des Simulationslaufes anzulegen. Ebenso fällt auf, daß Graphiken nicht im Experimentierrahmen angefertigt werden, sondern nach der Simulationssitzung mit einem eigenen Postprozessor (unter Verwendung abgespeicherter Daten).

Diese Eigenheiten haben allerdings fast alle diskrete Sprachen, in SIMAN sind sie noch sehr komfortabel zu nennen. Außerdem wird für die neue Release ein wesentlich komfortablerer Experimentierrahmen versprochen.

Innerhalb des Professor's Package wird eine Animation angeboten, die parallel zur Simulation des Modells läuft und die Entities wunderbar färbig von Resource zu Resource bewegt, die man selbst wieder wunderbar grafisch (an-)malen kann. Diese Animation ist sicher wichtig zur Management-geeigneten Repräsentation von Simulationen, aussagekräftiger allerdings sind die statistischen Methoden, die der Postprozessor anbietet: er erlaubt nicht nur das Zeichnen diverser Grafiken, sondern in stärkerem Maße statistische Korrelationen zwischen simulierten Zeitreihen zu ermitteln.

Nach den guten Erfahrungen wird SIMAN auch in der Vorlesung "Diskrete Simulation" (für Informatiker) verwendet werden.

Beispiel

Das Einsatzgebiet von SIMAN auf dem Gebiet der diskreten Simulation ist sehr vielfältig. Hauptaufgabengebiet ist sicher die Fertigungstechnik, aber SIMAN kann prinzipiell jede Art diskreter Vorgänge simulieren. Ein kleines Beispiel, das in den vorher erwähnten Übungen gerechnet wurde, demonstriert dies.

Wenn ein Restaurantbesitzer sein Lokal vergrößern will, so sollte er sich fragen, ob das überhaupt nötig ist, bzw. ob eine Vergrößerung durch die vermehrte Anzahl von Gästen überhaupt rentabel ist.

Am besten beginnt er mit einer Simulation des Ist-Zustandes (50 Tische für je 2 Personen) zur Stoßzeit (17-21 Uhr). Erfahrungsgemäß kommen die Gäste in Gruppen zu 2 (40%), 3 (30%), 4 (20%) oder 5 (10%) Personen. Die Tische werden für Gruppen von mehr als 2 Personen zusammengestellt. Ankommende Gäste (Gruppen) warten (in einer Warteschlange), bis ein Tisch frei wird. Weiters geht die Erfahrung ein, daß Gäste, die fünf oder mehr Gruppen warten sehen, wieder gehen. Die Bedienungszeit für Essen und Getränke beträgt zwischen 15 und 20 Minuten (gleichverteilt). Durchschnittlich brauchen die Gäste 20 Minuten zum Essen (normalverteilt, Standardabweichung 2 Minuten). Nach dem Essen zahlen die Gäste bei einer zentralen Kasse, wo sie sich wieder anstellen müssen, das Bezahlen dauert zwischen 1.5 und 3 Minuten (gleichverteilt). Die Gäste selbst treffen im Mittelwert alle 1.6 Minuten ein (Exponentialverteilung).

Die folgende Abbildung zeigt das SIMAN-Modell für diesen Prozeß, die Entities sind die Gäste, die Resourcen sind die Tische und die Kassa. Das Modell kann bequem in einem semigraphischen Editor (BLOCKS) erstellt werden, so wie es die Abbildung zeigt.

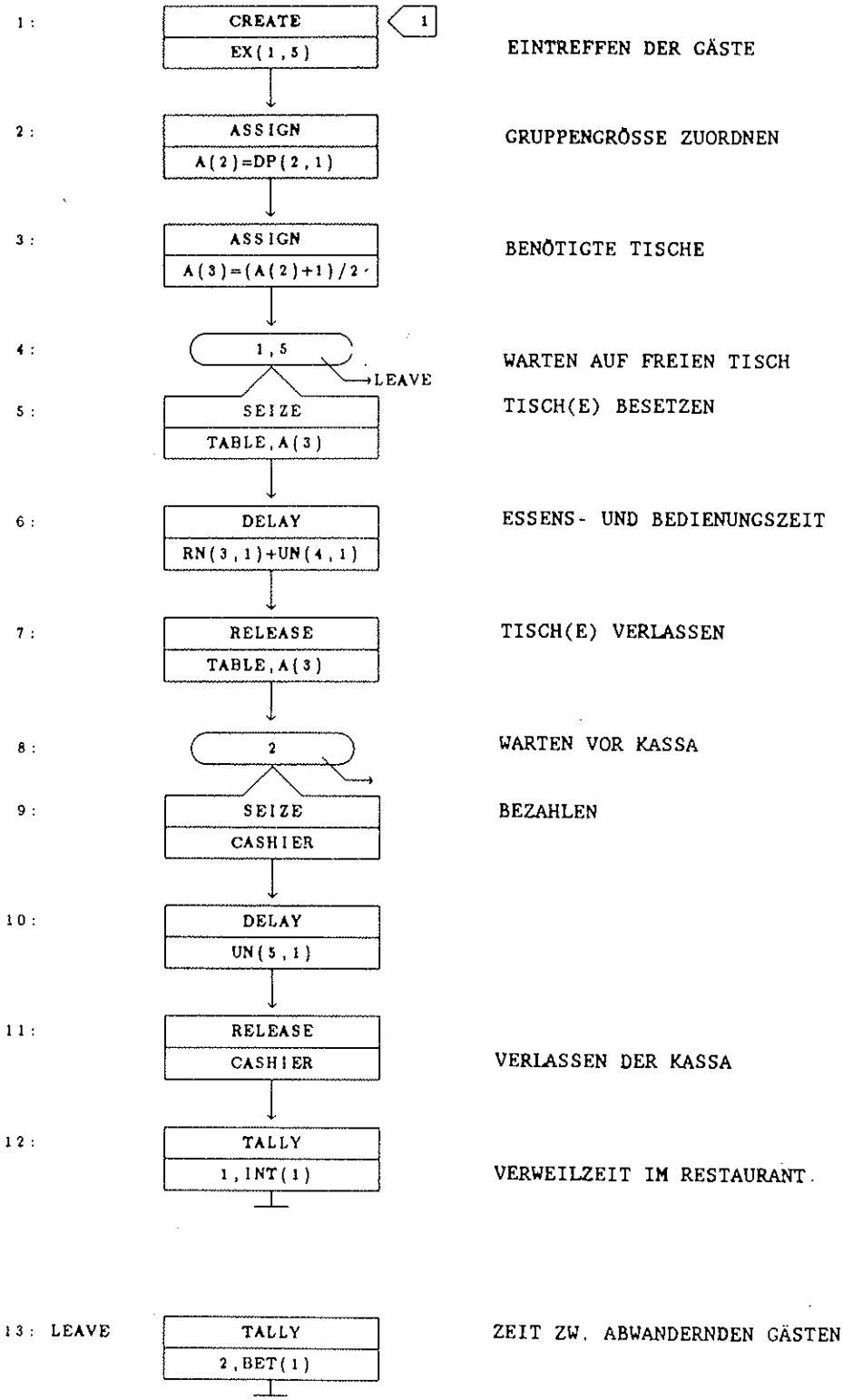


Abbildung 1
SIMAN Modell .

Im Experimentierrahmen wird festgelegt, daß folgende Größen bei der Simulation ermittelt werden sollen: die Anzahl und Verweilzeit der bedienten Gästegruppen, die Anzahl der abwandernden (Nicht-)Gästegruppen, die durchschnittliche Anzahl besetzter Tische, die durchschnittliche Anzahl wartender Gruppen und die durchschnittliche Auslastung der Kasse.

Abbildung 2 zeigt den SIMAN Summary Report, der diese Größen ausgibt.

Run ended at time .2400E+03

Tally Variables

Number	Identifier	Average	Standard Deviation	Minimum Value	Maximum Value	Number of Obs.
1	ZEIT IM REST.	53.05133	10.67583	36.47529	73.37518	81
2	ABWANDERNDE GÄSTE	.00000	.00000	.00000	.00000	0

Discrete Change Variables

Number	Identifier	Average	Standard Deviation	Minimum Value	Maximum Value	Time Period
1	BELEGTE TISCHE	32.63940	11.92986	.00000	50.00000	240.00
2	KASSENAUSNÜTZUNG	.77437	.41799	.00000	1.00000	240.00
3	WARTENDE GRUPPEN	.13366	.54762	.00000	4.00000	240.00

Run Time : 7 Second(s)

Abbildung 2
SIMAN Summary Report

Mit speziellen Anweisungen können Größen während der Simulation gesampelt und im Postprozessor weiterverarbeitet werden. Abbildung 3 und 4 zeigen Barcharts über die Auslastung der Tische (die Werte schwanken um 40 herum, später allerdings jedoch werden alle 50 besetzt) und über die Auslastung der Kassa (sie ist zu 100% ausgelastet!) - hier führt die im Summary Report angegebene Auslastung zu Trugschlüssen, denn in den ersten 20 Minuten ist die Kasse klarerweise arbeitslos - was aber in die statische Auslastungsstatistik eingeht.

Erste Schlußfolgerung aus dieser Simulation könnte sein, mit der Vergrößerung (mehr Tische) noch zu warten und statt dessen eine zweite Kassa aufzustellen.

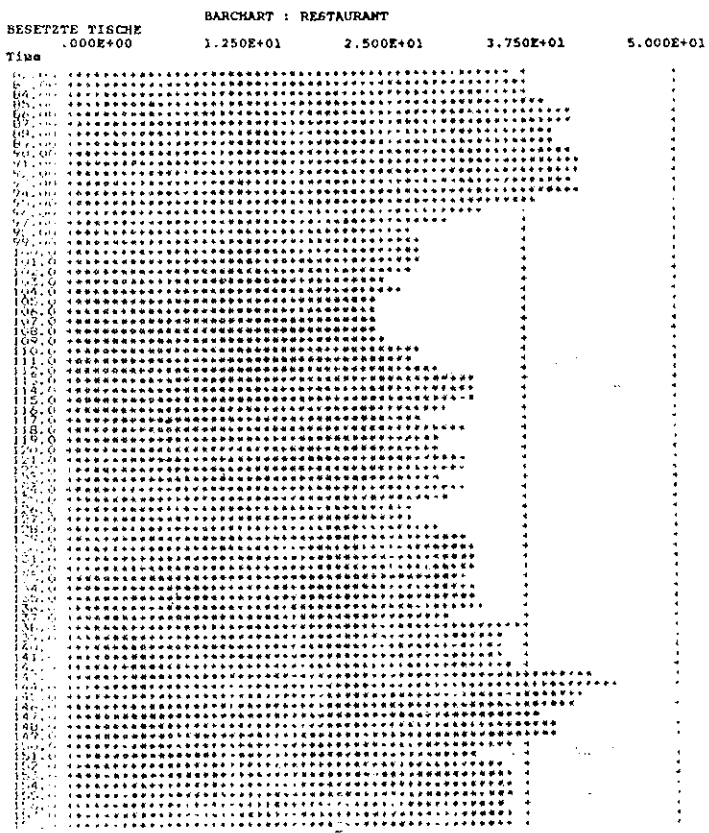


Abbildung 3

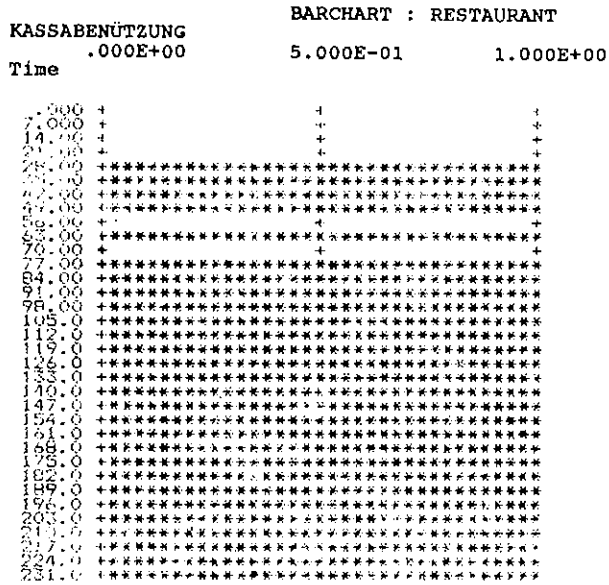


Abbildung 4

Das SIMULATIONSSYSTEM HYBSYS*)

D.Solar, F.Breitenecker
Institut für Analysis, Technische Mathematik und
Versicherungsmathematik

Die Simulationssprache HYBSYS am Simulationsrechenzentrum hat sich sehr gut bewährt. Vor allem die interaktive Modellbildung und die komfortable Experimentierumgebung erlaubten in kurzer Zeit die Analyse komplexer Prozesse. Sie ist gebunden an den Analogrechner EAI PACER 600 bzw. an den neuen Simulationsrechner EAI SIMSTAR ([1, 2, 3]). Bald tauchte daher der Gedanke auf, ein "rechnerunabhängiges, digitales HYBSYS" zu entwickeln. Zu bedenken ist dabei natürlich, daß die gesamte Software neu konzipiert werden muß. Zu diesem Zweck wurde von den Autoren ein Projekt beim Fonds zur Förderung der wissenschaftlichen Forschung beantragt, das Ende 1988 gewährt wurde. Das Projektvolumen finanziert zwei halbe Vertragsassistenten und den Ausbau von AT-PCs.

Simulationssprachen gibt es viele, und das SIMULATIONSSYSTEM HYBSYS versucht, nicht eine weitere Simulationssprache zu sein, sondern ein Modellentwicklungssystem, mit dem auch simuliert werden kann. Dazu ist ein modernes Sprach- und Implementierungskonzept nötig. Denn aufgrund der gewachsenen Struktur mit teilweise notwendiger Aufwärtskompatibilität von neuen Versionen und mit teilweise "krampfhaftem" Hinzufügen neuer Simulationselemente haben viele der herkömmlichen Sprachen, darunter vor allem die meistverwendeten und meistverbreiteten (ACSL, CSSL, CSMP,..) eine Reihe von Nachteilen.

Das Konzept des SIMULATIONSSYSTEMS HYBSYS verhindert bereits alle diese Nachteile. Grundlegend dabei ist eine Trennung von Modell und Experiment, zusätzlich wird noch die Methode als eigene Ebene (sprachlich und konzeptionell) angesehen ([4]). Dabei ist dann das Experiment als Anwendung einer Methode auf ein Modell aufzufassen (z.B.: Methode "Optimierung" + Modell "Schwingung" → Experiment "Minimierung eines Dämpfungsparameters").

Mit seiner Konzeption ist das SIMULATIONSSYSTEM HYBSYS nun eine Software, die folgende Ebenen der Modellbildung und Simulation durchführt bzw. unterstützt:

* Modellbildung
* Modellvalidierung
* Modellidentifikation
* Simulation

Die Grundlagen für das Simulationssystem sind die folgenden Eigenschaften, die allgemeine Konzepte, spezielle Eigenschaften und Implementierungsfragen betreffen:

* Allgemeine Eigenschaften:

  A1) Strikte Trennung zwischen Modell, Methode und Experiment ([4])
  A2) Interaktive Modelldeklaration und Modelländerungen auf strukturierten Modelldatenbasen mit Teilmodellen und Modellbibliotheken
  A3) Verwendung von geeigneten Datentypen, die eine konsistente Modellbeschreibung ermöglichen (semantische Unterscheidung von Parametern, Variablen, ...) mit einem sehr allgemeinen Datentyp "Tabellenfunktionen"
  A4) Trennung von Modell- und Experimentiervariablen unter Zuhilfenahme einer Experimentier-Datenbasis
  A5) Verwendung verschiedener Sprachebenen für Modell-, Methoden- und Experiment-Ebene (funktionale Sprache, prozedurale Sprache, Befehlssprache mit prozeduralen Eigenschaften)
  A6) Offenes System ! (d.h. beliebig erweiterbare Methodenbank [5])

---

*) Unterstützt vom Fonds zur Förderung der wissenschaftlichen Forschung Projekt Nr. P6553

* Spezielle Eigenschaften:

S1) Modellbeschreibung auch mit Vektor- und Matrizenoperationen, auch für Tabellenfunktionen
S2) Schnittstellen der Modelldatenbasis zu anderen Simulationssprachen wie z.B. ACSL ([6])
S3) Methoden zur (automatischen) Generierung spezieller Modelle (über Bondgraphen, Kompartments, Übertragungsfunktionen, etc.)
S4) Verwendung moderner Integrationsalgorithmen
S5) Möglichkeiten zur Behandlung von Unstetigkeiten
S6) Möglichkeit zur Skalierung des Systems
S7) Optimierungsmethoden (Rückführung auf Parameteroptimierung)
S8) Unterschiedliche Methoden zur Dokumentation von Simulationsergebnissen (2D-Plots, 3D-Plots, Tabellen, Plots mit Schichtenlinien) basierend auf der Ergebnisrepräsentation in Tabellenfunktionen
S9) Eigenwertanalyse und Linearisierung
S10) Frequenzbereichsanalyse (mit numerischen und analytischen Methoden)
S11) Sensitivitätsanalyse
S12) Erweiterungsmöglichkeiten für den Benutzer durch Einbindung eigener spezieller Methoden
S13) Auf Experimentebene komplexe Kontrollstrukturen, wie Befehlsschleifen und bedingte Befehle zum Aufbau komplexer Experimente u.a. mit Hilfe von Tabellenfunktionen
S14) Experimentelle Datentypen mit Durchführung impliziter Simulationsläufe mit/ohne Abspeicherung auf Tabellenfunktionen

* Implementierung:

I1) auf MS-DOS Ebene und auf UNIX-(XENIX-)Ebene
I2) Programmierung in C, Benutzererweiterungen auch mit kompatiblen Compilern
I3) Hardware: AT-PCs mit Graphikerweiterungen, Rechner mit UNIX/XENIX

Verwiesen sei nochmals auf die Punkte A6) und S2), die HYBSYS als offenes System ausweisen und Abbildungen auf andere Simulationssprachen vorsehen. Das SIMULATIONSSYSTEM HYBSYS sieht u.a. seine Stärke in der Modellentwicklung, Modellvalidierung und Modellidentifikation, Simulationen können auch dann mit anderen Sprachen fortgeführt werden. Von diesem Punkt aus betrachtet ist HYBSYS auch als Entwicklungssprache für dynamische Modelle zu sehen.

Ein kurzer Einblick in eine spezielle Eigenschaft soll die Mächtigkeit des Simulationssystems demonstrieren: in HYBSYS haben Tabellenfunktionen auch auf der Experimentierebene große Anwendungsmöglichkeiten, ferner wurde die Technik der impliziten Simulationsläufe beibehalten. Mit diesen Eigenschaften können nun in sehr einfacher Form komplexe Experimente durchgeführt und dokumentiert werden, vor allem können auch in einfacher Weise aufwendige dreidimensionale Zeichungen angefertigt werden.

Die folgenden Befehle definieren Variable, Parameter und Tabellenfunktionen. Die Variablen Y, DY und Z seien dabei Teile eines komplexen Modelles.

```
VAR REAL: Y, DY

PAR REAL: A, B

VAR REAL: Z [1:10]

FUN REAL: F(REAL)
```

Mit den nebenstehenden Definitionen von Tabellenfunktionen seien nun einige Beispiele für die Abspeicherung impliziter Simulationsläufe auf Tabellenfunktionen gegeben (die Simulationsläufe werden implizit durch das Experiment "Zuweisung einer Zustandsvariablen auf eine Tabellenfunktion" gestartet);

```
FUN REAL: G(REAL,REAL)
FUN REAL: H [1:10] (REAL)
```

Diese Tabellenfunktionen können dann später z.B. gezeichnet oder in weiteren Simulationsläufen weiterverwendet werden (soferne die Tabellenfunktion auch als "rechte Seite" im Modell auftaucht).

```
F = Y ( A=1,10,1 )
```

Dieses Experiment liefert eine Kurve; es speichert auf die Tabellenfunktion F als Funktionswerte die Werte der Zustandsvariablen Y zum Zeitpunkt TEND in Abhängigkeit von 11 äquidistanten Argumentwerten des Parameters A ab (es werden 11 Simulationsläufe durchgeführt, die Tabellenfunktion hat 11 Stützpunkte)

```
G = Y (T, A=1,10,1)
```

Dieses Experiment liefert eine Fläche; es speichert auf die zweidimensionale Tabellenfunktion G die Werte der Zustandsvariablen Y in Abhängigkeit von T und des Parameters A ab (11 Simulationsläufe, (NDT+1)*11 Stützpunkte)

```
G = Y (A=1,10,1; B=1,0,-.05)
```

Dieses Experiment liefert eine Parameterfläche; es speichert auf die zweidimensionale Tabellenfunktion G die Werte der Zustandsvariablen Y zum Zeitpunkt TEND in Abhängigkeit der Parameter A und B (11*21 Simulationsläufe und Stützpunkte)

```
H = Z
```

Dieses Experiment liefert 10 Kurven. Auf den Vektor H der 10 eindimensionalen Tabellenfunktionen H wird der zeitliche Verlauf des 10-dimensionalen Zustandsvektors Z abgespeichert (ein Simulationslauf, 10*(NDP+1) Stützpunkte)

Diese Tabellenfunktionen können nun mit dem Plot-Befehl gezeichnet werden, einige sind dann 3D-Plots !

Die Tabellenfunktionen auf Experimentierebene und die impliziten Simulationsläufe erlauben aber weitaus komplexere Experimente. Das folgende Beispiel, die iterative Lösung einer Integralgleichung, soll hier nun eine der Möglichkeiten kurz beleuchten:

Der Kern K(t,x) der Integralgleichung

$$x = \int_0^t K(s,x).x(s)ds$$

möge ein komplexer algebraischer Ausdruck abhängig von x sein.

Derartige Gleichungen können daher nur iterativ gelöst werden (nur in Sonderfällen des Kernes kann die Integralgleichung in eine Differentialgleichung umgewandelt werden).
Diese Iteration startet mit einer Näherung x_0, vermöge der dann die weiteren Näherungen x_i für x(t) durch

$$x_{i+1} = \int_0^t K(s,x_i).x_i ds$$

berechnet werden.

Mit Hilfe von Tabellenfunktionen kann nun diese Iteration sehr einfach als Experiment formuliert werden:

```
PAR REAL: EPS
VAR REAL: X, EX
FUN REAL: FX(REAL)
```

Die nebenstehenden Definitionen deklarieren das Iterationsmodell für die Integralgleichung.

```
EQU K  = f(FX,t)
EQU X  = INTEG ( K*FX, 0.)
EQU EX = MAXT( ABS(EX-X) )
```

Die Gleichung EQU K=.. berechnet den Kern mit der alten Näherung FX, die als Tabellenfunktion definiert ist; die Gleichung EQU X=... integriert Kern mal alter Näherung zur neuen Näherung X; parallel dazu errechnet sich die Differenz zwischen alter und neuer Näherung aus EQU EX=...

```
EX > EPS ! FX = X
```

Diese einfache Anweisung führt nun das doch komplexe Experiment der Iteration für die Integralgleichung durch. Das Einzelexperiment FX=X führt einen Simulationslauf durch (= eine Iteration) und speichert die neue Näherung gleich auf die Tabellenfunktion FX um, um bei der nächsten Iteration als alte Iteration zu dienen. Das Experiment FX=X steht in einer Schleife (angezeigt durch "!"), die durchgeführt wird, solange EX>EPS gilt - also wird die Iteration solange durchgeführt, bis der maximale Fehler zwischen zwei Iterationen kleiner einer Schranke EPS wird.

Dieses Beispiel zeigt die Mächtigkeit dieses Tabellenfunktionen-Konzeptes bereits ziemlich genau auf. Ähnliche einfach beschreibbare, doch sehr komplexe Experimente mit Iterationen wie bei der Integralgleichung können auch bei partiellen Differentialgleichungen (aufbereitet mit Liniemmethoden) zu einem raschen Erfolg führen. Ergebnisse von Simulationen mit einem Modell können nun auch in anderen Modellen für Iterationen, als Sollkurven, etc. verwendet werden.

Das Projekt ist auf zwei Jahre anberaumt, eine Testversion wird bis Sommer 1988 zur Verfügung stehen (Präsentation bei der European Simulation Multiconference im Juni in Nizza). Der erste Prototyp für AT-PC wird voraussichtlich Ende dieses Jahres laufen.

Literatur

1. Solar D., Berger F., Blauensteiner A.: HYBSYS - Interaktive Simulationssoftware für ein hybrides Mehrbenutzersystem. Informatik-Fachbericht 56, Springer-Verlag, Heidelberg (1982) 257-265.
2. Solar D.: HYBSYS-PTRAN - An experimentation tool for the EAI SIMSTAR Parallel Multiprocessor. Proc. 2nd European Simulation Conference, Antwerp, Sept.1986, Publ. SCS, San Diego (1986), 203-208
3. Embley R.W.: The technology behind SIMSTAR, an all new simulation multiprocessor. Informatik-Fachbericht 85, Springer-Verlag, Heidelberg, (1984) 317-327
4. Breitenecker F., Solar D.: Models, Methods, Experiments - Modern aspects of simulation languages. Proc. 2nd European Simulation Conference, Antwerp, Sept.1986, Publ. SCS, San Diego (1986) 195-199
5. Breitenecker F.: Optimierung in HYBSYS. Interface 22 (1985)
6. Mitchell and Gauthier Associates: Advanced Continuous Simulation Language (ACSL) - Reference Manual, Concord, Mass. USA, 4th Edition (1986)

EAI Computer Users' Group Meeting 1987 in Wien

I. Husinsky
EDV-Zentrum, Abt. Hybridrechenanlage (Simulationsrechenanlagen)

Die EAI Computer Users' Group umfaßt alle Benützer von EAI Geräten weltweit. Seit etwa 20 Jahren wird jährlich eine Tagung der Users' Group in Europa abgehalten. Zum dritten Mal in der Geschichte der EAI Computer Users' Group Meetings war die Technische Universität Wien der Gastgeber und Organisator. Seit dem ersten Users' Group Meeting in Wien im Jahre 1973 ist es zur Tradition geworden, daß nach der Eröffnung und der Vorführung der lokalen Rechenanlagen das Meeting in einem netten Ort außerhalb von Wien fortgesetzt wird. Im Jahre 1973 war dies in Baden bei Wien, im Jahre 1980 Gösing an der Mariazellerbahn.

Diesmal begann das Meeting am 27. Oktober in Wien, die Übersiedlung der Simulationsrechner in das neue Institutsgebäude auf den Freihausgründen war gerade abgeschlossen, und die neuen Räume standen für die Tagungsteilnehmer zur Verfügung.

Rektor Prof. Kraus und Vorstand Prof. Weinmann hielten Reden in der Eröffungssitzung.

Folgende Präsentationen stellten die Arbeiten und die Geräte des Simulationsrechenzentrums vor:

W. Kleinert:
The Hybrid Computation Centre of the Technical University of Vienna

W. Kleinert, M. Gräff:
Simulation of a Three-Phase Rectifier Bridge with SIMSTAR and ACSL

D. Solar:
Introduction to HYBSYS PTRAN

Anschließend wurde eine Anwendung des EAI SIMSTAR Multiprozessors on-line vorgeführt und die Räume des Simulationsrechenzentrums besichtigt.

Dann fuhren alle Teilnehmer nach Dürnstein in der Wachau, wo das Meeting im Hotel Richard Löwenherz fortgesetzt wurde.

In den nächsten zwei Tagen wurden die folgenden interessanten Vorträge von EAI-Anwendern gehalten:

E. Reisinger, H. Weiß, Technische Universität Graz:
Local Distribution of Magnetic Force and Current Density in the Slots of Deep Bar Cage Induction Machines

E. Reisinger, H. Weiß, Technische Universität Graz:
On EAI 2000 Integrator, Switch and Parallel Logic Performance (and Improvement)

A. Braun, Universität der Bundeswehr München:
Real-Time Computer Simulation of Turbofan Engines

W. Bär, Universität Erlangen-Nürnberg:
Modelling and Simulation of the Dynamic of a Continuous Unloading Crane

O. Krettek, RWTH Aachen:
Introduction to Hybrid Analog Studies of Vehicle Pneumatic Suspension

K.H. Fasol, Ruhr-Universität Bochum:
Application of a Program Package Using a New Approach to Identification, Model Reduction and Controller Design

M. Gräff, Technische Universität Wien:
Programming the SIMSTAR PLU

H. Schemmann, G. Diefenbach, Philips Aachen:
Application of the EAI 2000/PDP 11 System at Philips Forschungslabor Aachen

D. Solar, F. Breitenecker, Technische Universität Wien:
The Simulation System HYBSYS and its Table Function Concept

F. Breitenecker, D. Solar, Technische Universität Wien:
Case Studies with SIMSTAR and HYBSYS PTRAN: Vector Optimization and Environment for Heart Regulation Models

Repräsentanten der Firma EAI berichteten über:

O. Wright, EAI West Long Branch:
EAI Status Report

A. Markman, EAI West Long Branch:
EAI Current Product Presentation

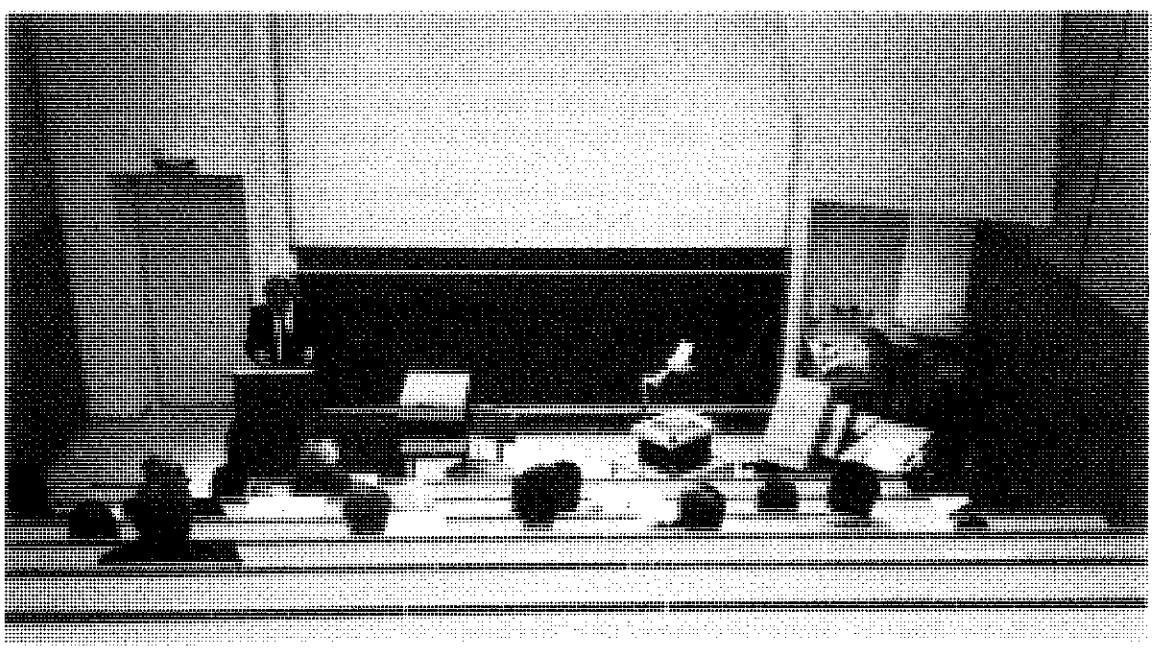
A. Markman, EAI West Long Branch:
6 Degree-of-Freedom Real Time Robot Simulation

In einem User Forum konnten offene Fragen und Probleme der einzelnen User sowie der Users' Group an sich mit EAI diskutiert werden.

Das technische Programm wurde durch einen Empfang des Landeshauptmannes von Niederösterreich, einen Heurigenabend, eine Führung durch die Kirche von Dürnstein und den traditionellen Abschlußabend, mit einem Zaubererauftritt, ergänzt.

Ein ausführlicher Bericht über die Tagung und schriftliche Ausarbeitungen aller Vorträge sind auf Anfrage erhältlich.

Im Jahre 1988 wird die 12. internationale IMACS Tagung in Paris der Treffpunkt der EAI User sein.



Eröffnungssitzung

Simulationsverfahren zur Berechnung der kollektiven Spontanemission im gedämpften Hohlraum

J. Seke
Institut für Theoretische Physik

F. Rattay
Institut für Analysis, Technische Mathematik und Versicherungsmathematik

Bei den Strahlungsübergängen in der Laserphysik spielen meistens nur zwei ausgewählte atomare Energieniveaus eine Rolle. Deswegen werden Zwei-Niveau-Atommodelle in Wechselwirkung mit dem elektrischen Strahlungsfeld behandelt.

Wir untersuchten einen Spezialfall der Spontanemission von mehreren gleichartigen Zwei-Niveau-Atomen, die sich im gedämpften Hohlraum befinden. Durch eine besondere Annahme über die Form des Hohlraumes, die auch experimentell verwirklicht wurde, bildet sich darin eine stehende Welle mit einer bestimmten Frequenz aus. Der Hohlraum sei so abgestimmt, daß diese Frequenz mit der von den darin befindlichen Atomen übereinstimmt. Im nichtidealen Hohlraum kommt es allerdings zu einem Strahlungsverlust durch die Wände und man spricht dann von einem gedämpften Hohlraum. Spontanemission bedeutet, daß am Anfang kein Strahlungsfeld im Hohlraum vorhanden ist. Obwohl die Atome nicht direkt miteinander in Wechselwirkung stehen, ergibt sich eine indirekte Wechselwirkung über das Strahlungsfeld und dies führt zu einem kooperativen Verhalten. Ein Kollektiv von Atomen strahlt daher wesentlich schneller ab als ein Einzelatom.

In der Literatur wurde bisher nur der ideale (ungedämpfte) Hohlraum exakt behandelt. Wir konnten erstmals exakte Gleichungen für den mehratomigen Fall aufstellen und die numerische Auswertung durchführen. In Vorarbeiten wurden am Hybridrechner der einatomige Fall, aber auch schon Spezialfälle der kollektiven Spontanemission gelöst (vgl [1-4], Bild 1).

Für den mehratomigen Fall wurde ein exaktes, rekursives System von Differentialgleichungen gefunden. Die vierdimensionalen Zustandsgrößen X sind die Dichtematrixelemente zur Bestimmung der Erwartungswerte der atomaren Inversionsbesetzungszahl, der Anzahl der Photonen im Strahlungsfeld, der Energiefluktuationen, der Atomfeldkorrelation usw. [5,6]. Für N Atome ergibt sich die Rekursion:

$$\begin{aligned}\frac{\mathrm{d}X_{n(k),l(m)}}{\mathrm{d}t} = (-1)^{n-l} g\{&[(N-n)(n+1)(k+1)]^{1/2} X_{n+1(k+1),l(m)} \\ &-[(N-l)(l+1)(m+1)]^{1/2} X_{n(k),l+1(m+1)} \\ &+[(N-n+1)nk]^{1/2} X_{n-1(k-1),l(m)} \\ &-[(N-l+1)lm]^{1/2} X_{n(k),l-1(m-1)}\} \\ &+2\mathbb{K}[(k+1)(m+1)]^{1/2}(1-\delta_{nk})(1-\delta_{lm}) X_{n(k+1),l(m+1)} \\ &-\mathbb{K}(k+m) X_{n(k),l(m)},\end{aligned}$$

$$n=0,1,\ldots,N, \quad k=0,1,\ldots,n, \quad l=n-k,n-k+1,\ldots,n, \quad m=l-n+k,$$

mit den Konstanten g und K und den Anfangsbedingungen

$$X_{n(k),1(m)}(0) = 2\delta_{n0}\delta_{10}\delta_{k0}\delta_{m0}$$

Leider steigt die Zahl der Differentialgleichungen mit N^3:

$$N_{tot} = \sum_{j=2}^{N+1} \frac{j!}{2(j-2)!} + (N+1)(1/2N+1) = 1/6(N+1)(N+2)(N+3),$$

Das schnelle Anwachsen der Gleichungsanzahl erlaubt ein paralleles Aufbringen am Analogrechner nur im Fall weniger Atome. Die Simulationen wurden daher mit ACSL ausgeführt. Obwohl ACSL die Matrixintegration erlaubt, sollte sie bei großen, schwach besetzten Matrizen nicht eingesetzt werden. In der oben angegebenen Differentialgleichung stehen auf der rechten Seite maximal 6 Summanden der Form Koeffizient$\cdot X_{index}$. Durch Einführen von 6 Koeffizienten- und 6 Indexvektoren, die in der Initial-Section berechnet wurden, konnte der Rechen- und Platzaufwand soweit reduziert werden, daß die Kapazitätsgrenze für den Einzelbenutzer auf der CYBER der TU-Wien (im NOS2 Betrieb) erst bei Systemen von 25 Atomen, die durch 3276 Differentialgleichungen beschrieben werden, erreicht wurde.

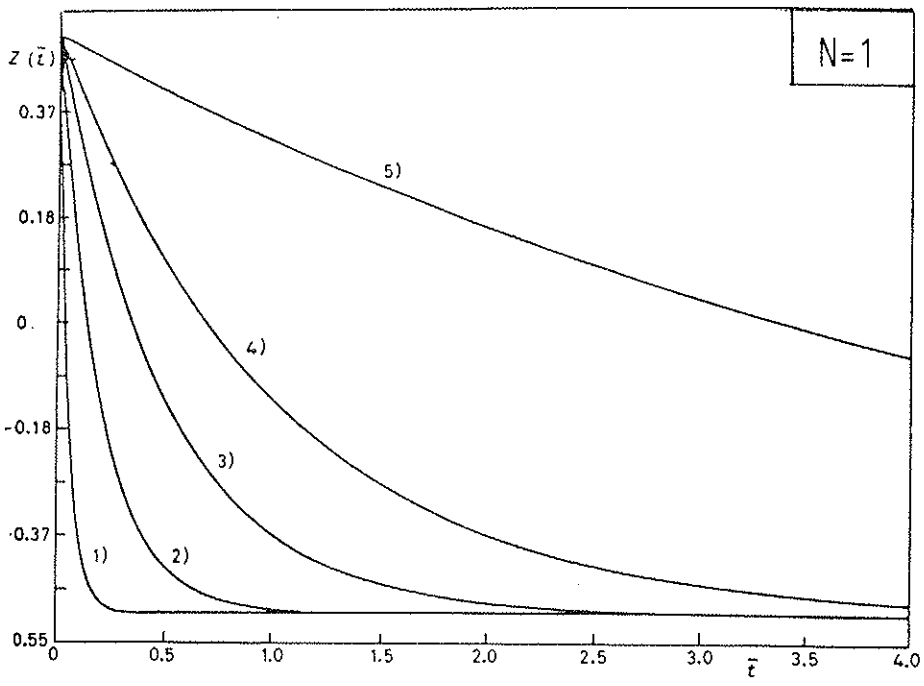
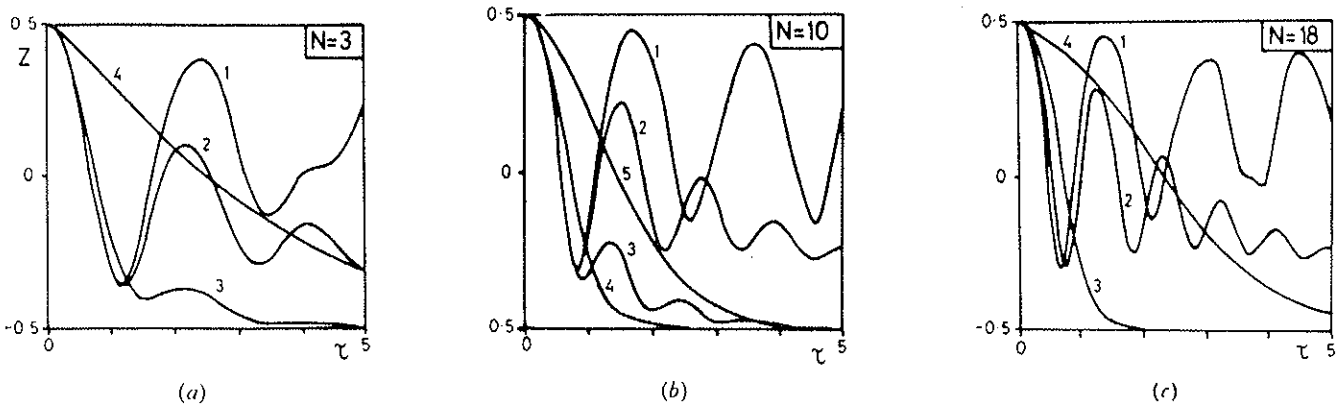
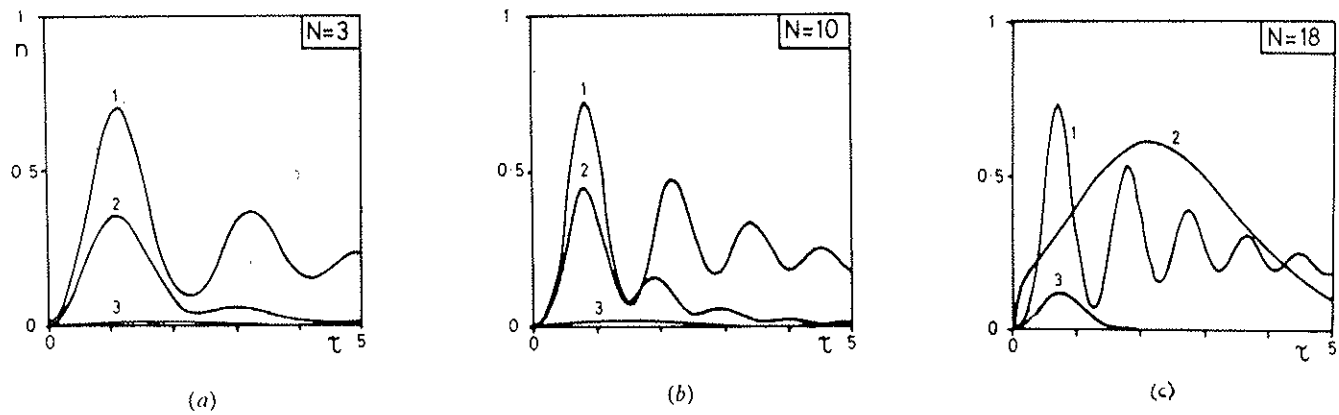


Abbildung 1
Die zeitliche Änderung des Erwartungswertes für die Besetzungszahlinversion (die Differenz der Wahrscheinlichkeiten für die Besetzung der atomaren Niveaus). Die Kurven 1-5 entsprechen der Parametervariation K/g = 0.1, 0.4, 1, 2, 10. Berechnung hybrid am EAI-PACER.



(a) (b) (c)

Abbildung 2
Änderung des Erwartungswertes für die Besetzungszahlinversion in Abhängigkeit von der Hohlraumdämpfung und von der Atomzahl N.
a) N=3, K/g=0, 0.2, 1, 10 entsprechend der Numerierung von 1-4,
b) N=10, K/g=0, 0.2, 1, 3, 10
c) N=18, K/g=0, 0.2, 5, 25



(a) (b) (c)

Abbildung 3
Zeitliche Änderung des Erwartungswertes für die Anzahl der Photonen im Hohlraum.
a) N=3, K/g=0.2, 1, 10
b) N=10, K/g=0.2, 1, 10
c) N=18, K/g=0.2, 5, 25

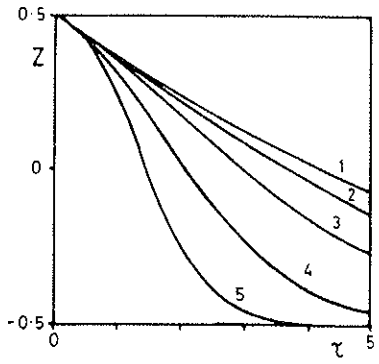


Abbildung 4
Änderung der Besetzungszahlinversion in Abhängigkeit von der Anzahl der Atome bei fester Dämpfung. N = 2, 3, 5, 10, 18 entsprechend zur Numerierung von 1-5.

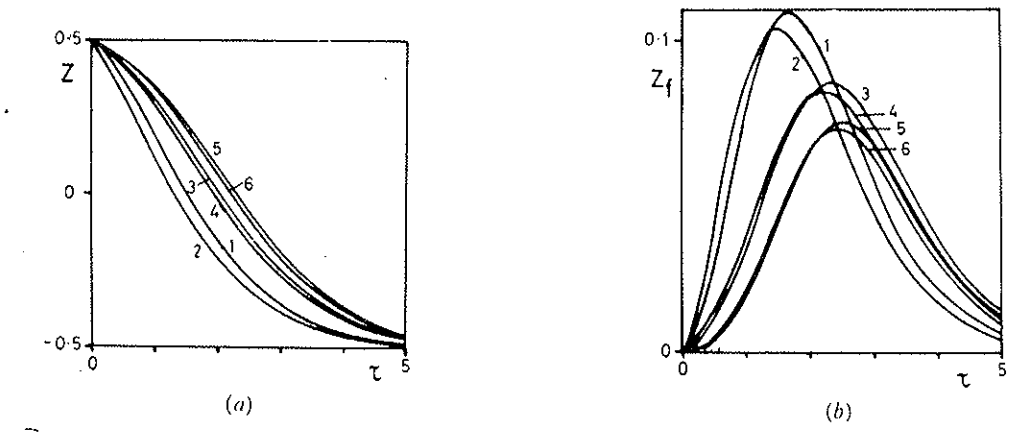


Abbildung 5
Vergleich der exakten Ergebnisse der Besetzungszahlinversion Z und der Energiefluktuation Z_f mit den Näherungen von anderen Autoren. Diese guten Übereinstimmungen ergeben sich jedoch nur für spezielle Werte, in denen deren Näherungsverfahren anwendbar sind, vgl. [6], [7].

Literatur

1. J. Seke, 1985, Lett. Nuovo Cim., 43, 103
2. J. Seke, 1986, Optics Lett., 11, 189
3. J. Seke, 1986, Phys. Rev. A, 33, 739
4. J. Seke, 1986, Phys. Rev. A, 33, 4409
5. J. Seke und F. Rattay, 1987, Opt. Soc. Am. B, 4, 380
6. J. Seke und F. Rattay, 1987, J. Mod. Optics, 34, 651
7. R. Bonifacio, P.Schwendimann und F. Haake, 1971 Phys. Rev. A, 4, 302, 854

Die Simulation elektrisch angeregter Nervenfasern

F. Rattay
Institut für Analysis, Technische Mathematik und
Versicherungsmathematik

Oft wird der Mensch mit einem Supercomputer verglichen: Jede der etwa 10^{11} Nervenzellen hat bis zu 10000 synaptische Endigungen und ist damit mit anderen Nervenzellen verbunden. 10^6 Nervenfasern sorgen für die parallele Ein- und Ausgabe des Zentralnervensystems und dienen damit hauptsächlich zur Sinneswahrnehmung und zur motorischen Steuerung des Muskelapparates. Technische Überlegenheit zeigen Computer allerdings in der Zuverlässigkeit und vor allem in der wesentlich kürzeren Zykluszeit, denn da die synaptischen Schaltvorgänge durch chemische Reaktionen ausgelöst werden, ergeben sich Schaltzeiten im Millisekundenbereich. Die große Packungsdichte der Nervenzellen ist nur auf Grund ihrer hochentwickelten Feinstruktur möglich.

Die Zellen werden durch eine Membran abgeschlossen, die bloß aus einer zweilagigen Molekülschicht besteht. Sie enthält Öffnungen mit einer sehr spezifischen Ionendurchlässigkeit. Im Inneren einer Nervenzelle herrschen deshalb andere Ionenkonzentrationen vor als außerhalb. Dadurch hat das Zellinnere ein etwa 70mV niedrigeres Potential als die extrazellulare Flüssigkeit. Wird diese Ruhespannung (z.B.: infolge einer chemischen Reaktion an einem synaptischen Ende einer Nervenfaser) um etwa 30mV angehoben, so führt dies zu einer Kettenreaktion, die durch den Na^+Ionen-Einstrom durch die Membran ausgelöst wird. Das Potential im Nerveninneren steigt dabei um etwa 100 mV an (Aktionspotential), bevor es wieder auf den stationären Wert zurückfällt. Diese lokale Störung läuft dann die Nervenfaser entlang und dient damit zur Signalübertragung. Die Kodierung höherer Reizintensität erfolgt aber nicht durch eine weitere Potentialerhöhung des Aktionspotentials sondern durch häufigeres "Feuern".

Diese Reizung einer Nervenfaser kann aber auch mit Hilfe von Elektroden erfolgen, die entweder implantiert werden oder als Hautelektroden bloß oberflächlichen Kontakt haben. Die medizinische Bedeutung der funktionellen Elektrostimulation begann 1959, als Senning in Schweden den ersten Herzschrittmacher implantierte.

Bei Querschnittlähmungen verhindert die Unterbrechung des neuralen Datenflusses die Aktivierung entsprechender Muskeln. Je nach Lage der Wirbelsäulenverletzung können dabei die unteren, alle vier Extremitäten und auch noch die Atmung betroffen sein. Durch Elektrostimulation ist es möglich, daß solche Patienten vom Rollstuhl aufstehen bzw. ohne Beatmungsmaschine leben können. Neben der neuromotorischen Stimulation ist auch die Elektrostimulation von sensorischen Nerven von großer Bedeutung. Durch Reizung des optischen und des akustischen Nerven können wieder entsprechende Wahrnehmungen erweckt werden. Leider ist es dabei technisch nicht möglich, jede einzelne Nervenfaser individuell anzusprechen und damit den (ohnedies nicht bekannten) natürlichen Kode nachzuahmen, sondern es werden im allgemeinen einzelne kleine Elektroden in der Nähe eines Nervenbündels implantiert und die Stimulationssignale erfassen dann bestimmte Faserpopulationen. Wie aber die einzelnen Fasern tatsächlich in Abhängigkeit von Elektrodenkonfiguration und Signalformen reagieren, läßt sich experimentell schlecht nachvollziehen. Einen besseren Zugang bietet die Computersimulation mit geeigneten Nervenmodellen [1-5].

Lokale Nervenmodelle

Im "space clamp" Experiment versorgt eine axial in die Nervenfaser eingesetzte Elektrode mit freier Oberfläche alle Punkte der Membran einer Nervenfaser gleichmäßig mit Strom. Durch Austauschen der inneren und äußeren Flüssigkeiten können die Membrandurchlässigkeiten bezüglich verschiedener Ionen spannungs- und zeitabhängig erfaßt werden. Hodgkin und Huxley fanden damit eine recht gute gleichungsmäßige Beschreibung der Nervenmembran (siehe Übersicht I) durch vier gewöhnliche Differentialgleichungen (HH), die durchaus auf einem Analogrechner gelöst werden können, aber wegen der vielen Nichtlinearitäten (die aus

Genauigkeitsgründen zum Teil durch Funktionsgeber abgedeckt werden sollen) relativ viele Komponenten belegen.

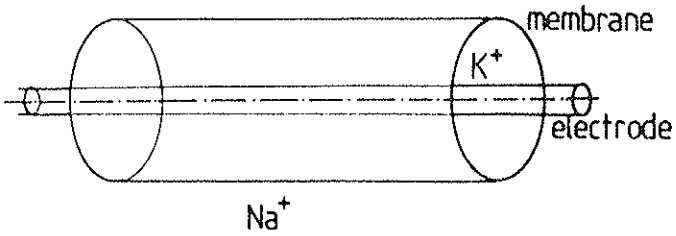
Im Falle der Elektrostimulation des Gehörnerven ist eine Untersuchung der Auswirkungen stochastischer Störungen auf das Feuerungsverhalten von besonderem Interesse für die Ausbildung der entsprechenden Nervensignalmuster. Wegen der Komplexität der HH-Gleichungen haben wir für unsere weltweit erste Simulation des elektrisch stimulierten Gehörnervens unter Berücksichtigung stochastischer Einflüsse zunächst ein einfacheres Modell verwendet [3],[6],[7]. Fitzhugh reduzierte die HH-Gleichungen auf ein System zweiter Ordnung, das für viele Anwendungen ausreicht. Bei der Auswertung von simulierten Feuerungsmustern, die ein Sprachsignal in der Gehörprothese hervorruft, konnten wir das Nervenmodell als Nebenrechnung (!) dreimal parallel am EAI-680 aufbringen [8]. Aufwendigere Modelle, wie das von Frankenhaeuser und Huxley für myelinierte Nervenfasern, oder das Modell von Scriven, wo auch die Kalziumabhängigkeit und die Ionenpumpen modelliert werden, sollten digital simuliert werden.

Die lokalen Modelle lassen keine direkten Schlüsse auf die von der Geometrie abhängigen Elektrodenströme zu. Trotzdem ergeben sich, wie Übersicht II zeigt, etliche Anwendungen, die zum Großteil auch hybrid lösbar sind. So wurde unter anderem der Einfluß von hochfrequenten Sinusschwingungen im Stimulus am EAI PACER simuliert und es zeigte sich, daß die Nervenfasern zwar beim Anschalten des Signals feuern - sonst aber nur innerhalb eines sehr engen Bereichs, der auch noch frequenzabhängig ist [9]. Dieses Verhalten war bisher nur von der Stimulation mit konstanten Strömen bekannt. Aus den Ergebnissen der Simulationen konnte auch das nachteilige Verhalten hochfrequenter Trägerfrequenzen bei Gehörprothesen erklärt werden [9].

Unter Verwendung der Simulationssprache ACSL wurde auch oft mit dem Frankenhaeuser-Huxley (FH) Modell gerechnet. Es konnte gezeigt werden, daß einige experimentelle Befunde (wie das Mehrfachfeuern bei niederfrequenter Reizung) wohl bei HH, nicht aber bei FH auftreten. Im Gegensatz zur bisher allgemein vertretenen Meinung zeigt sich u.a. damit, daß das HH-Modell auch im Falle myelinierter Fasern dem FH-Modell vorzuziehen ist [10,11].

Simuliert man mit einer Elektrode in der Nähe des Gehörnerven , so kann eine einzelne Faser - bedingt durch ihre Refraktärzeit - maximal etwa 1000 mal pro Sekunde feuern. Mit Hilfe des Fitzhugh-Modells konnte durch Simulation am EAI PACER gezeigt werden, welches Feuerungsmuster sich dabei im Nerven einstellt, und daß die Information über höhere Formanten als Zeitdifferenz zwischen den Reizsignalen an verschiedenen Nervenfasern übertragen wird [3] (Abb. 1). Dieses Phänomen tritt aber leider nur sehr sporadisch auf, kann aber durch Verstärken des Rauschsignals etwas verbessert werden. Tatsächlich hat leichtes Verrauschen des Sprachsignals eine leichte Verständnisverbesserung der Patienten gebracht. Hier hat also die Simulation erst zum Verständnis der Informationsübertragung geführt und niemand wäre auf die Idee gekommen, einem schlecht Hörenden noch Zusatzgeräusche anzubieten, um das Verständnis zu verbessern. Eine Strategie zur Verstärkungswirkung dieses sogenannten Volleyeffektes zeigt Abb. 2 [12]. Lokale Modelle (am EAI PACER und in ACSL gerechnet) konnten auch noch zur Erklärung anderer Phänomene führen [13].

LOKALES EXPERIMENT (space clamp) und SIMULATION nach HODGKIN-HUXLEY



Im Nerveninnern überwiegen K^+, außen Na^+ Ionen. Ein kurzer Stromimpuls an der axial angebrachten Elektrode kann ein Aktionspotential auslösen, wobei sich alle Punkte der Membran gleich verhalten.

Der Strom per cm^2 besteht aus einem kapazitiven und einem Ionenstromanteil und ist durch den Elektrodenstrom I_{el} (per cm) bestimmt.

$$C.\dot{V} + I_i = I_{el}/2\pi r$$

wobei der Ionenstrom aus Natrium-, Kalium- und Leckstromanteil besteht $I_i = I_{Na} + I_K + I_L$ Die kompletten HH-Gleichungen lauten damit

$$\dot{V} = [-g_{Na}m^3h(V-V_{Na}) - g_K n^4(V-V_K) - g_L(V-V_L) + I]/C$$

$$\dot{m} = [-(\alpha_m + \beta_m).m + \alpha_m].k$$

$$\dot{n} = [-(\alpha_n + \beta_n).n + \alpha_n].k$$

$$\dot{h} = [-(\alpha_h + \beta_h).h + \alpha_h].k$$

mit dem Temperaturkoeffizienten k

$$k = 3^{0.1T-0.68}$$

und den Koeffizienten α und β

$$\alpha_m = (2.5 - 0.1V)/(exp(2.5 - 0.1V) - 1)$$

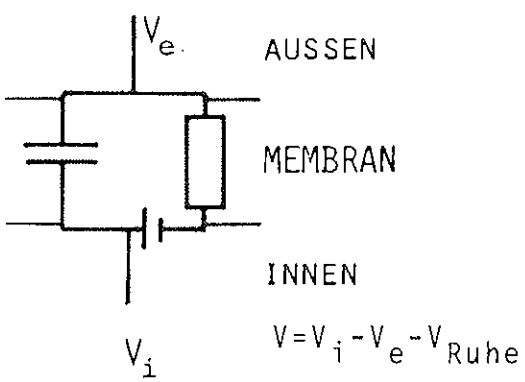
$$\beta_m = 4.exp(-V/18)$$

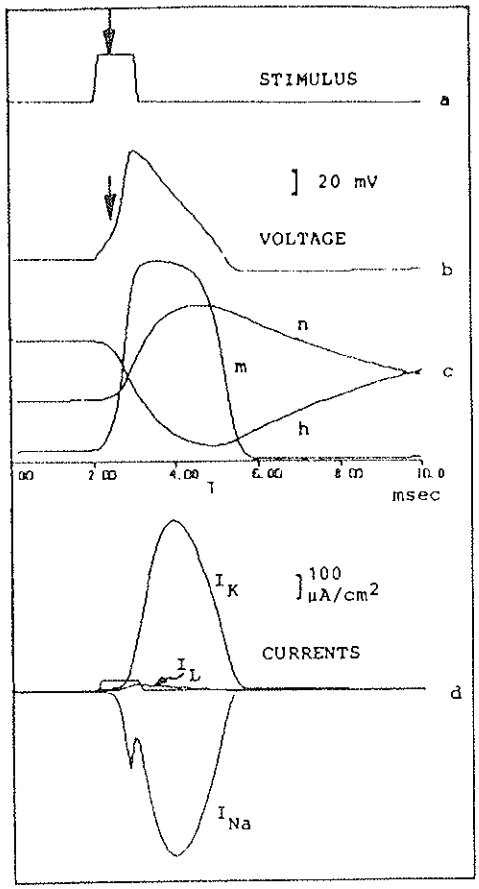
$$\alpha_n = (1 - 0.1V)/10.(exp(1 - 0.1V) - 1)$$

$$\beta_n = 0.125.exp(-V/80)$$

$$\alpha_h = 0.07.exp(-V/20)$$

$$\beta_h = 1/(exp(3 - 0.1V) + 1)$$



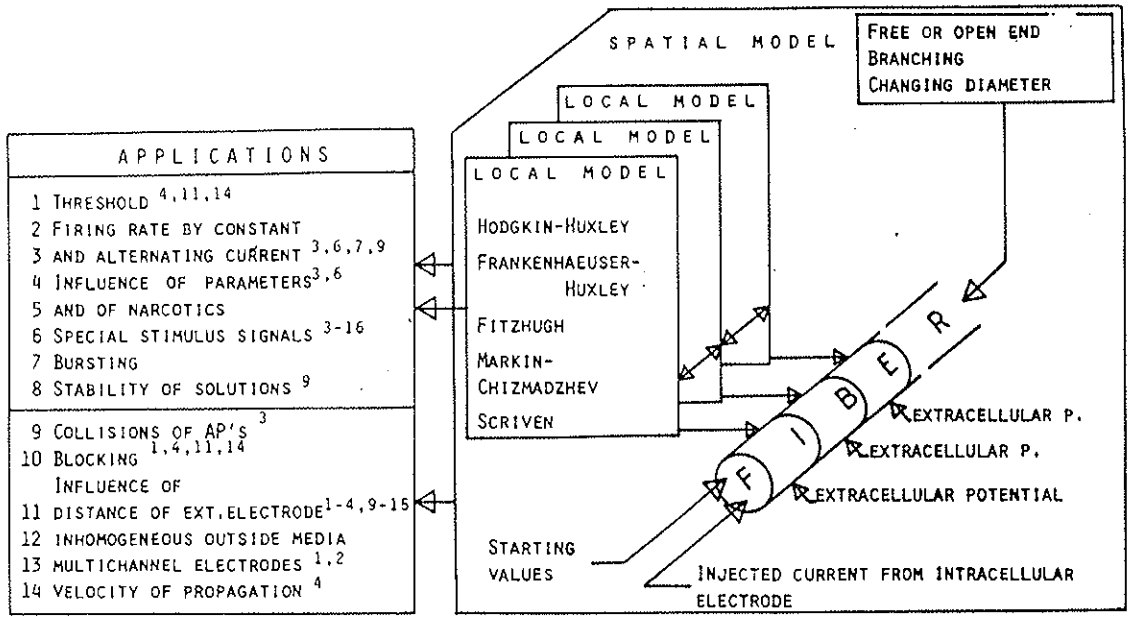


SYMBOLE KONSTANTEN u. [EINHEITEN]

V	reduzierte Membranspannung $V_{Ruhe} = 0$		$[mV]$
I_{el}	Elektrodenstrom per cm		$[\mu A/cm]$
I	Anteil des Elektrodenstroms durch $1cm^2$ der Membran. I ist eine Stromdichte		$[\mu A/cm^2$
I_i	Ionenstromdichte an der Membran		$[\mu A/cm^2]$
I_{Na}, I_K, I_L	Natrium- Kalium- und Leckstrom		$[\mu A/cm^2]$
C	Kapazität der Membran 1		$[\mu F/cm^2]$
r	Faserradius	0.024	$[cm]$
g_{Na}	max. Na-Leitfähigkeit	120	$[kohm/cm^2]$
g_K	max. K-Leitf.	36	$[kohm/cm^2]$
g_L	max. Leck-Leitf.	0.3	$[kohm/cm^2]$
V_{Na}	Spannung, hervorgerufen durch unterschiedl. Na-Ionenkonzentrationen an beiden Seiten der Membran	115	$[mV]$
V_K	Kaliumspannung	-12	$[mV]$
V_L	Leckspannung	10.6	$[mV]$
m, n, h	Wahrscheinlichkeiten für das Aktivieren und Blockieren der Ionenkanäle in der Membran		
k	Temperaturkoeffizient		
T	Temperatur		$[°C]$
	Zeit in		$[msec]$

ÜBERSICHT II

ANWENDUNG LOKALER und RÄUMLICHER MODELLE

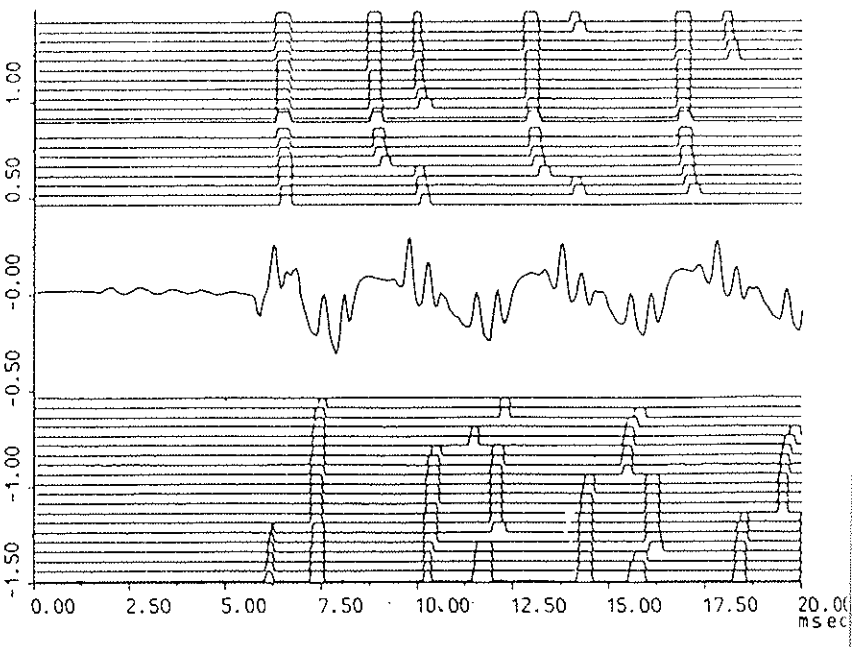


Übersicht II

Die hochgestellten Nummern in den lokalen (Fälle 1-8) und räumlichen Anwendungen (Fälle 1-14) beziehen sich auf die Literaturzitate. Alle diese Arbeiten gründen auf Simulationen, die mittels HYBSYS am EAI PACER oder mittels ACSL an der CYBER der Technischen Universität Wien gerechnet wurden. Anwendungen anderer Autoren werden insbesonders noch in Ref. 5 angeführt.

Abbildung 1

Reaktionen verschiedener Fasern des Gehörnerven auf die Elektrostimulation. Die Mittelkurve zeigt als Stimulussignal den Elektrodenstrom, der den Luftkompressionen des Sprachsignals "ü" entspricht. (Abgenommen als Potentialdifferenzen an der Kopfhaut eines Implantatträgers, vgl. Ref. 17). jede einzelne Linie des oberen Bildteils entspricht dem Feuerungsverhalten einer unterschiedlich entfernten Nervenfaser, wobei gegen oben die Intensität des Reizsignals zunimmt, d.h. die obersten Linien entsprechen den am nächsten liegenden Nervenfasern. Die Nervenantworten synchronisieren sich mit den Maxima des Stimulussignals. Zeitunterschiede, die kleiner als 1 msec sind, können von einer Einzelfaser nicht vermittelt werden, doch treten sie (relativ selten) zwischen verschieden entfernten Fasern auf. Solche Effekte sind bei diesem Sprachsignal bei negativer Elektrodenpolung erkennbar. Dreht man die Polarität eines Lautsprechers um, so kann der Normalhörende keinen Unterschied wahrnehmen. Bei der Elektrostimulation zeigt die Simulation jedoch ein deutlich verändertes Muster und das daraufhin vorgenommene Umpolen erbrachte tatsächlich eine Änderung in der Wahrnehmung beim Patienten.



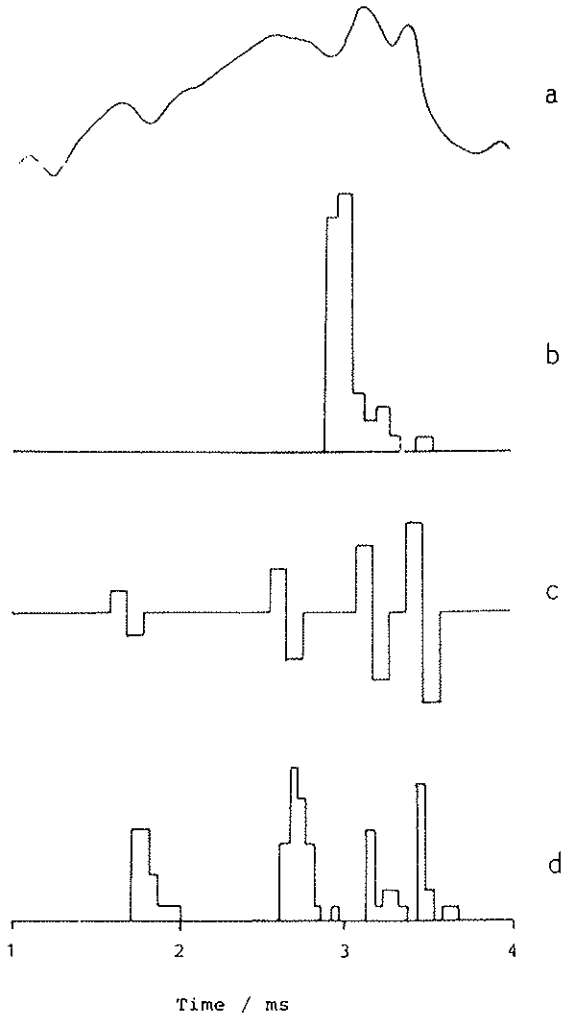


Abbildung 2
Verbesserungsvorschlag für die Kodierungsstrategie bei Einkanalstimulation von Gehörprothesen.
Ein Sprachsignal (a) hat leider sehr oft eine Feinstruktur im höherfrequenten Bereich, die nicht vernachlässigt werden darf, wenn es nicht zur Verwechslung mit anderen Phonemen kommen soll.
(b) zeigt anhand der Reaktionen von 60 verschieden weit entfernten Fasern als Histogramm den Informationsverlust der Feinstruktur.
(c) ist das Ergebnis einer Pulskodierung mit Verstärkungseffekt während einer Grundschwingung des Sprechers.
Das Gesamtfeuerungverhalten des stimulierten Nerven ist wieder als Histogramm in (d) dargestellt und gibt eine wesentlich reichhaltigere Information (Lösung mit dem Fitzhugh-Modell; 3 msec).

Räumliche Modelle

Die lokalen Modelle simulieren, wie es durch Elektrostimulation überhaupt zu einem Aktionspotential kommt. Dazu wird angenommen, daß (in Elektrodennähe) an allen Punkten der Membran dieselbe Spannung herrscht. Bei einer besseren Näherung (vor allem, wenn wie in den praktischen Anwendungen, die Elektrode außerhalb der Nervenfaser liegt) wird man jedoch die Nervenfaser segmentieren und jedes einzelne Segment durch ein lokales Modell beschreiben. Da die Spannungsänderungen während eines Aktionspotentials zum Großteil auf Kosten des intrazellulären Potentials gehen, kann diese Aktivität vom Einfluß der Elektrode auf das extrazelluläre Potential entkoppelt werden, d.h. das extrazelluläre Potential kann bereits auf Grund der Elektrodengeometrie, des spezifischen extrazellulären Widerstandes und des Elektrodenstromes bestimmt werden und die Reaktion der Nervenfaser wird dann in einem zweiten Schritt durch die Vorgabe des extrazellulären Potentials bestimmt. Die Teilnetze der Innenwiderstände sind dann gemäß Abb. 3 nur mehr durch die interzellulären Widerstände verkoppelt.

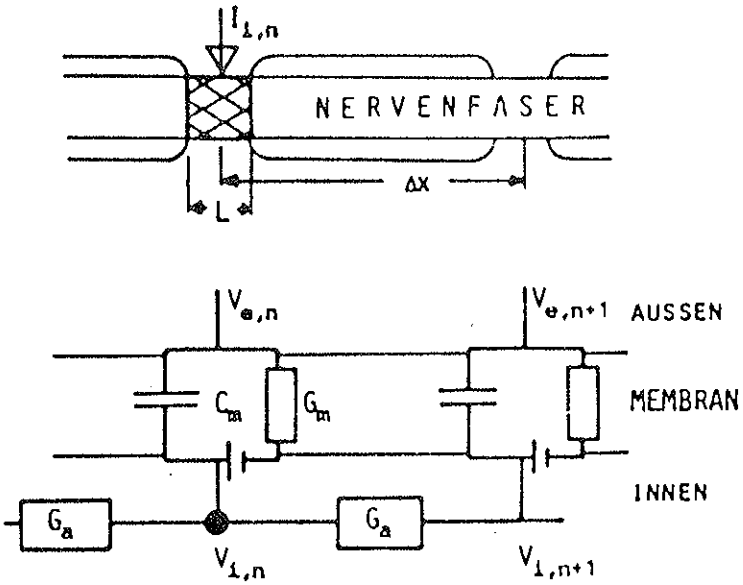


Abbildung 3

Nervenfaser und ihr elektrischer Ersatzschaltplan.

Die Membran von Nervenfasern wird nur an den myelinfreien Ranvier'schen Schnürringen der Länge L von den Ionenströmen I_i durchflossen (bei unmyelinierten Fasern ist L gleich der Segmentierungslänge Δx). Jedes aktive (freie) Membransegment läßt sich durch ein lokales Modell simulieren. In einer ersten Näherung kann das extrazelluläre Potential durch die Außenelektroden bestimmt werden bzw. ist V_e=0, wenn keine extrazelluläre Anregung erfolgt. Die Verkoppelung der lokalen Netze erfolgt durch die inneraxonalen Widerstände G_a.

Diese Vorgangsweise erleichtert nicht nur die Modellbildung, sondern auch das Verständnis der Elektrostimulation von langen Zellen, wie die der Nerven- und Muskelfasern. Wie in [14] erstmals gezeigt wurde, hängt die Reaktion einer elektrisch stimulierten Nervenfaser von der zweiten Ableitung des extrazellulären Potentials V_e entlang der Faser (x-Richtung) ab und wir nennen daher $f=d^2V_e/dx^2$ die Aktivierungsfunktion.

Implantierte Elektroden sind meist klein und in Nervennähe angebracht. Dadurch kann man sie durch punktförmige Stromquellen approximieren und das durch eine Elektrode im homogenen isotropen Medium hervorgerufene Potential ist

$$V_e = \rho_e I_{el}/4\pi r$$

wobei ρ_e der spez. Widerstand des extrazellularen Mediums, I_{el} der Elektrodenstrom und r der Abstand des betrachteten Punktes von der Elektrode ist. Das von mehreren aktiven Elektroden generierte Potential V_e läßt sich durch Superponieren der Einzeleinflüsse errechnen [1,2,4,14]. Komplizierter ist die Bestimmung von V_e für Oberflächenelektroden, da hier erst das Potential über die Laplacegleichung mit entsprechenden Randbedingungen (z.B. mit finiten Elementen) gelöst werden muß [11,14].

Sobald das extrazelluläre Potential als Funktion von x bestimmt ist, ergibt sich auch die Aktivierungsfunktion, und die Reaktion der Nervenfaser läßt sich aus den verkoppelten lokalen Modellen bestimmen. Wie die Übersicht II zeigt, können damit eine Reihe weiterer Anwendungsfälle simuliert werden, wobei bezüglich der Ergebnisse auf die angeführte Literatur hingewiesen werden muß.

Die räumlichen Modelle verlangen das parallele Aufbringen von 10 bis 100 lokalen Modellen, also von 20 bis 500 gewöhnlichen Differentialgleichungen. Ein Lösungsweg, der sich dabei recht gut bewährt hat, war die Simulation mit ACSL auf der CYBER. Durch Verwenden der Vektorintegration bleibt das Programm kurz und übersichtlich und es können die graphischen Ausgabemöglichkeiten genutzt werden. Selbstverständlich können die Modelle auch mit einem Standardprogramm für gewöhnliche Differentialgleichungen gelöst werden. Die Verkopplung der lokalen Modelle würde mit gegen Null gehender Segmentierungslänge zu einer Diffusionsgleichung (partielle Differentialgleichung vom parabolischen Typ) führen, die dann üblicherweise nach dem Cranc-Nicolson-Verfahren gelöst wird, während der oben besprochene Lösungsweg der Linienmethode zur Lösung dieser partiellen Differentialgleichung entspricht.

Literatur

1. Rattay F. und Mayr W.: Eine quantitative Abschätzung elektrisch aktivierter Fiberpopulationen am Beispiel der Karussellstimulation. Biomed. Technik 32 (1987), 184-190
2. Rattay F.: Simulation of nerve responses by extracellular electrostimulation. Proc. 2nd Vienna Int. Workshop FES (1986), 67-70
3. Motz H. and Rattay F.: Simulation of the functioning of electrostimulation prosthesis for the profoundly deaf. Proc. 2nd Vienna Int. Workshop FES (1986) 167-170
4. Rattay F.: Ways to approximate current-distance relations for electrically stimulated fibers. J. Theor. Biol. 125 (1987), 339-349
5. Rattay F.: Modelling and simulation of electrically stimulated nerve and muscle fibers. A review. Math. & Comp. in Simulation 29 (1987) 357-366
6. Motz H. and Rattay F.: Models of the electrostimulation of the nervus acusticus. Proc. 1st Vienna Int. Workshop FES (1983)
7. Hochmair-Desoyer I.J., Hochmair E.S., Motz H. and Rattay F.: A model for the electrostimulation of the nervus acusticus. Neuroscience 13 (1984), 553-562
8. Rattay F.: Vokalsynthese. Interface 23 (1986), 15-19
9. Rattay F.: High frequency electrostimulation of excitable cells. J. Theor. Biol. 123 (1986), 45-54
10. Motz H. and Rattay F.: A study of the application of the Hodgkin-Huxley and the Frankenhaeuser-Huxley model for electrostimulation of the acoustic nerve. Neuroscience 18 (1986), 699-712
11. Rattay F.: Modelling the excitation of fibers under surface electrodes. IEEE-Trans. BME-35 (im Druck)
12. Motz H. and Rattay F.: Signal processing strategies for electrostimulated ear prostheses based on simulated nerve response. Perception 16 (im Druck)
13. Rattay F. and Motz H.: Simulation of the response of a multichannel nerve array to pulse shapes produced by single channel electrostimulation. Perception 16 (im Druck)
14. Rattay F.: Analysis of models for external stimulation of axons. IEEE-Trans. BME-33 (1986), 974-977
15. Rattay F.: Simulation von Nervenreaktionen durch Elektrostimulation. ASIM Tagung Zürich, Informatik-Fachberichte Nr. 150, Springer Berlin (1987)
16. Rattay F.: Simulation der Reizentwicklung im Gehörnerv durch elektrische Anregung. Interface 20 (1983), 46-48
17. Rattay F. and Wallenberg E.L.: Investigation of the problem of vowel recognition by the profoundly deaf fitted with an electrostimulating prosthesis by means of simulation with nerve models (in Vorbereitung).

Eine Optimierungsumgebung für ACSL*)

A. Sauberer, R. Ruzicka, F. Breitenecker, I. Troch
Institut für Analysis, Technische Mathematik und
Versicherungsmathematik

Der vorliegende Beitrag beschäftigt sich mit der Integration von Optimierung in Simulationssprachen vom CSSL-Typ (wie z.B. ACSL). Dabei wird im wesentlichen das von der Simulationssprache erzeugte "Simulations-Hauptprogramm" erweitert. In der Folge wird der Optimierungs-Preprozessor "GOMA" vorgestellt, der automatisch für ein ACSL-Modell diese Programmerweiterungen generiert; im besonderen wird auf die komplexe und unter Umständen mehrdeutige Wertübergabe zwischen Simulations- und Optimierungsprogramm eingegangen.

Einleitung

Die Simulation technischer Systeme hat den Zweck, das Verhalten des Systems durch Analyse (und Rechnersimulation) eines mathematischen Modells dieses Systems zu untersuchen. Im Rahmen der Simulation taucht dann sehr bald die Frage nach "optimalen" Systemparametern (in gewissen Grenzen frei wählbare Konstante wie z.B. Zeitkonstanten von Reglern, etc.) und optimalen Steuerungen im Falle von geregelten bzw. gesteuerten Systemen (eine auf Parameteroptimierung rückführbare Frage) auf. Optimierung ist daher eine notwendige Ergänzung der Simulation.

Prinzipiell stehen drei Möglichkeiten zur Verfügung, Optimierung in Simulationssprachen einzubinden, nämlich

- Erweiterung des Runtime-Interpreters der Sprache,
- Programmierung der Optimierung in der Modellbeschreibung und
- Mischformen mit/ohne Eingriff in tiefere Ebenen.

Die einfachste Möglichkeit zur Optimierung in Simulationssprachen vom CSSL-Typ ist, die Parameteroptimierung im Modell zu definieren (zu beschreiben). Parameteroptimierung ist implementierbar, indem in der TERMINAL SECTION Parameter abhängig von Werten der Gütemaße geeignet geändert (=optimiert) werden und dann ein Rücksprung in die INITIAL SECTION erfolgt; darauf wird dann ein Simulationslauf mit den neuen Parametern durchgeführt, etc. Diese Methode ist allerdings nur sehr beschränkt brauchbar, da der Optimierungsalgorithmus selbst programmiert werden muß: es kann keine Optimierungsroutine einer Bibliothek verwendet werden, da derartige Algorithmen die Berechnung von Gütefunktionen (in diesem Fall ein Simulationslauf) in einem externen Unterprogramm verlangen - was hier unmöglich ist.

In [1] wird eine Möglichkeit angegeben, bei der Optimierung und Simulation "gleichberechtigt" arbeiten. Diese Methode ist aber maschinen- und betriebssystemabhängig.

Eine allgemein implementierbare und effiziente Methode zur Parameteroptimierung besteht in der Erweiterung des Simulations-Hauptprogramms, wie es CSSL-Sprachen erzeugen. Diese Methode, in [2] in Grundzügen vorgestellt und in [3, 4, 5, 6] weiterverfolgt, ermöglicht die Parameteroptimierung in ACSL.

Im Prinzip wird bei dieser Methode das Simulations-Hauptprogramm, das der ACSL-Precompiler erzeugt und das in einer Schleife den Runtime-Interpreter (ZZEXEC) und eine Simulation (ZZSIML) aufruft, um eine Optimierungsroutine OPTIM (z.B. aus einer Programmbibliothek) erweitert, die ihrerseits zur

---

*) Unterstützt durch das Forschungsprojekt S3207 "Praktische Berechnung optimaler Steuerungen und Regelungen" des Fonds zur Förderung der wissenschaftlichen Forschung

Auswertung der Gütefunktionen Simulationsläufe (ZZSIML) aufruft; sinnvoll ist weiters eine Abfrage, die zwischen "normaler" Simulation und Optimierung auswählt; Abbildung 1 zeigt diese Erweiterung.

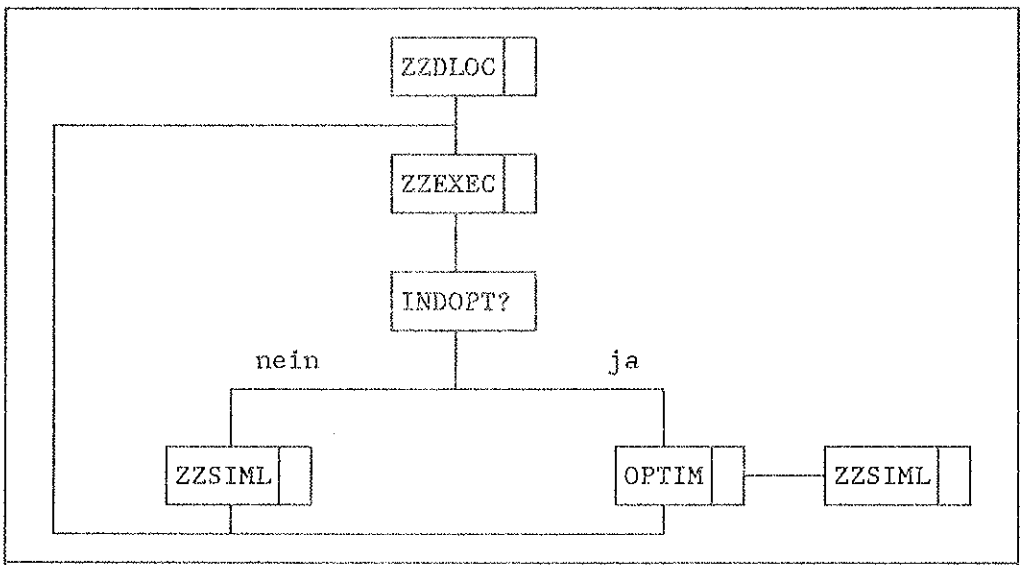


Abbildung 1

Die Parameterübergabe zwischen Optimierung OPTIM, Simulation ZZSIML und dem Simulations-Hauptprogramm erfolgt dabei über den von ACSL erzeugten COMMON BLOCK, der alle Variablen enthält und auch extern zugänglich ist.

Allerdings treten einige Probleme bzw. Unbequemlichkeiten auf:

1. Der Benutzer muß das Simulations-Hauptprogramm selbst ändern. Er muß auf FORTRAN-Ebene umfangreiche Parameterübergaben programmieren.

2. Das Modell selbst muß um die Berechnung der Gütefunktionen (meist in der DERIVATIVE SECTION) erweitert werden.

3. Beschreiben die Parameter eine Steuerung, so sind komplexe ACSL-Tabellen von FORTRAN aus aufzubauen und im Modell etwaige Interpolationen vorzusehen; sind die parametrisierten Steuerungen stückweise konstant, so ist das Problem der Synchronisation von Integrationsschrittweite und Unstetigkeiten in den Steuerungen zu lösen.

4. Es kommt zu jeder Menge von Seiteneffekten, da ja oft nicht bekannt ist, welche Variable die Simulation generiert und welche die Optimierung. Das Problem der Seiteneffekte wird umso größer, je allgemeiner eine derartige Optimierungsumgebung ist.

In [7] und [8] wird ein Grundkonzept angegeben, das einige der obigen Probleme löst. Es wird dem Benutzer ein "genormtes" um Optimierung erweitertes Simulations-Hauptprogramm zur Verfügung gestellt, das bereits die notwendigen Aufrufe zur Optimierung enthält. Zu ergänzen sind dann nur noch Routinen für die Parameterübergabe; diese Routinen "maskieren" die zu übergebenden Variablen und Parameter, um die Gefahr von Seiteneffekten zu vermindern.

GOMA - eine Optimierungsumgebung in ACSL

Das vorhin erwähnte Konzept wurde zu einer Optimierungsumgebung "GOMA" für ACSL weiterentwickelt, die u.a. die Berechnung optimaler (parametrisierter) Steuerungen erlaubt. GOMA leistet folgendes:

* automatische Generierung des ACSL-Simulations-Hauptprogramms
* automatische Erweiterung des Modells zur Auswertung der Gütefunktion

* automatische Erstellung von Tabellen und Feldern für parametrisierte Steuerungen
* Synchronisation von Integration und Unstetigkeiten in Steuerungen durch automatische Generierung von DISCRETE SECTIONS im ACSL-Modell
* automatisches Ersetzen aller Parameter für den Aufruf der Parameter
* Vermeidung aller Seiteneffekte zwischen Simulation und Optimierung durch Verwendung von Routinen, die Parameterwerte "maskiert" übertragen.

GOMA ist in PASCAL geschrieben und verwendet als Optimierungsprogramm derzeit die Routine E04VCF der NAG-Library. GOMA löst Parameteroptimierungsaufgaben und auch Probleme der Funktionenoptimierung (durch Rückführung auf Parameteroptimierung mit Beschränkungen unter besonderer Berücksichtigung von Steuerfunktionen der Regelungstechnik). Weitere Optimierungsalgorithmen sind in Vorbereitung.

Im folgenden wird GOMA näher beschrieben, wobei vor allem auf die Implementierung der diversen Teile eingegangen wird.

Für GOMA ist das Optimierungsproblem in der folgenden Form zu formulieren:

$$\dot{x} = f(t,x,u) \qquad x(t_0) = x_0 \tag{1}$$

mit $x \in R^n$ und der Steuerfunktion $u \in R^m$ (u tritt nur bei Steuerungsoptimierung auf).

Das Gütefunktional $J = J(t,x,u)$ möge ein Minimum werden unter den zusätzlichen Bedingungen: (t_f bezeichne die Endzeit)

$g_i(t_f,x(t_f)) = 0$ Endbedingungen (2)

$lo_i <= h_i(t,x,u) <= up_i$ nichtlineare Nebenbedingungen (3)

$u_{lo,i} <= u_i <= u_{up,i}$ Parameterbeschränkungen

Dieses Problem hat nun der Benutzer in einem Benutzerfile in Form einer Optimierungsbeschreibung (A; diese Eingabe kann auch interaktiv erfolgen) und in einem zweiten File, das die Modellbeschreibung (B; ACSL-Modell) enthält, zur Verfügung zu stellen:

A1) Anzahl der Steuerfunktionen u_i: m

A2) Anzahl der Stützstellen für die Parametrisierung

A3) Typ der Steuerung (Art der Parametrisierung):

KONSTANT
LINEAR
KUBSPLINE
BANGBANG

Die u_i können also als stückweise konstante, lineare, kubische Spline- oder vom Benutzer vorgegebene Funktionen parametrisiert werden (im letzen Fall sind die Intervallängen zwischen den einzelnen Teilstücken die Parameter der Optimierung).

A4) Name des Gütefunktionals (im ACSL-Modell (B) definiert)

A5) Angabe, ob t_f eine freie oder feste Endzeit darstellt. (Ist t_f frei, so ist dies ein weiterer Parameter der Optimierung)

A6) Anzahl der linearen Beschränkungen

A7) Angabe der Endbedingungen g_i (nach (2))

A8) Angabe der nichtlinearen Beschränkungsfunktionen h_i (nach (3))

A9) Ist der Typ BANGBANG: Angabe der BANGBANG-Funktionen

A10) zu optimierende Parameter

B) Ein ACSL-Modell, in dem die Gleichungen (1) und die Auswertung des Gütefunktionals implementiert werden.

Punkt A1) bis A3) tritt nur bei der Optimierung von Steuerfunktionen auf, Punkt A10) nur bei "reiner" Parameteroptimierung.

Die Arbeitsweise des Generators wird in Abbildung 2 wiedergeben:

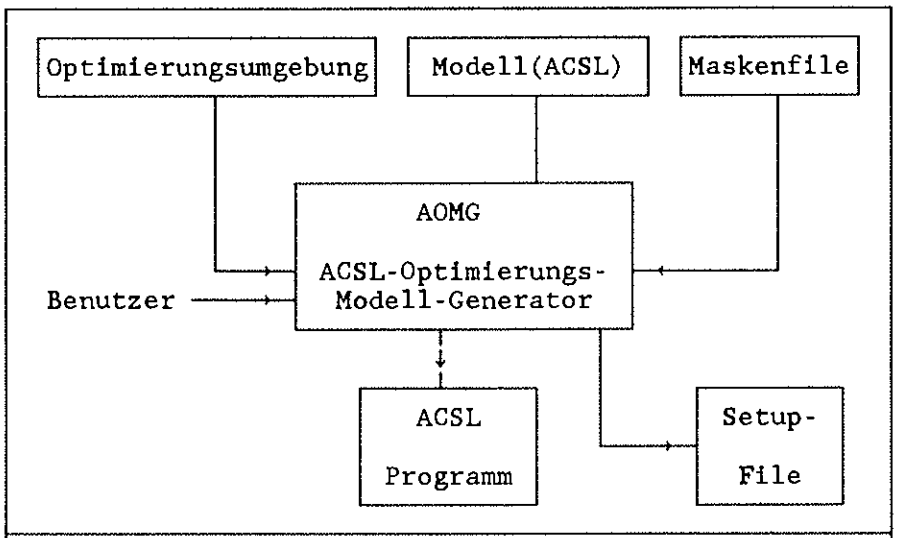


Abbildung 2 : Struktur von GOMA

Der Generator liest drei Files, das Modell der Optimierungsumgebungs-Beschreibung, das ACSL-Modell (diese beiden werden vom Benutzer zur Verfügung gestellt) und ein Maskenfile ein. Die Optimierungsumgebungs-Beschreibung kann auch interaktiv eingegeben werden. Der Benutzer wird hier von GOMA nach der Art und den Parametern der Optimierungsaufgabe gefragt.

Das Maskenfile enthält die Information über die Ergänzungen des ACSL-Modells und die hinzuzubindenden Fortran-Unterprogramme. Schließlich wird nach Angaben des Benutzers ein Setup-File für die Initialisierung einiger Optimierungsparameter erstellt.

Anhand des Beispiels einer Laufkatze mit Erzentlader [8] sollen zwei typische Benutzerfiles angebenen werden, wobei Kommentare unter Hochkomma stehen:

Das File ERZ.GOM zur Optimierungsumgebungs-Beschreibung:

```
" Eine Steuerfunktion U "
M  1
" Anzahl der inneren Stützstellen "
SZP 19
" U wird als eine stückweise lineare Funktion parametrisiert "
TYP  LINEAR
" Das Gütefunktional ist J = t_f "
FO TEND
" Die Endzeit t_f ist frei, also ein Parameter der Optimierung"
TEND FREI
" keine linearen Nebenbedingungen"
LINEAR 0
" 4 Endbedingungen  x_i(t_0) = x_if "
ENDE      4
```

```
X1 - X1F
X2 - X2F
X3 - X3F
X4 - X4F
" eine nichtlineare Nebenbedingung "
NICHTLINEAR    2
" Die Quadrate der negativen Anteile von X3 werden integriert"
                 tf
" h1(t,x,u) = ∫ [max(0 , - x3)]2dt
                 t0
I 2= -X3
" Die vierten Potenzen der Überschreitung der Schranke SCHR"
" durch X4 wird aufsummiert"
               szp+1
" h2(t,x,u) = Σ [max(0, x4(ti)-schr)]4
               i=1
S 4= X4 - SCHR
END
```

Das File ERZ.MOD zur Modellbeschreibung:

```
PROGRAM ERZ
INITIAL
CONSTANT ...
    "diverse Konstantenvereinbarungen"
END
  DYNAMIC
      DERIVATIVE
         " Modellbeschreibung"
         X1      =    INTEG(X2, X10)
         X2      =    INTEG(-CA2*X1 + CBETAQ*U, X20)
         X3      =    INTEG(X4, X30)
         X4      =    INTEG(-CC*X1 + CBETAQ*U, X40)
      END
      TERMT(T.GE.TEND-1.E-4)
  END
  END   $ "OF PROGRAM ERZ"
```

Der Generator GOMA erzeugt daraus nun ein ACSL-Modell mit einem in FORTRAN geschriebenen Simulationshauptprogramm, das alle nötigen Vereinbarungen für die Steuerungen enthält, und eine DISCRETE SECTION, die die Steuerung zeitlich updatet.

Nun kann mit ACSL weitergearbeitet werden, wobei beim Übersetzen anzugeben ist, daß das Simulationshauptprogramm vom Benutzer zur Verfügung gestellt wird.

Verfügbarkeit

Die ACSL-Lizenz wurde vom Simulationsrechenzentrum zur Verfügung gestellt.

Testversionen von GOMA sind an der Abt. für Regelungsmathematik, Hybridrechentechnik und Simulationstechnik des Instituts für Analysis, Technische Mathematik und Versicherungsmathematik verfügbar. Interessenten mögen sich bei Herrn Doz. F. Breitenecker, Klappe 5374, melden.

Literatur

1. Aymot J.R., van Blokland G.: Parameter optimization with ACSL, Proc. Summer Computer Simulation Conference 85, (1985) SCS, 63-68

2. Bausch-Gall I.: Parameteroptimierung bei technischen Modellen mittels einer kontinuierlichen Simulationssprache, Informatik-Fachbericht 56, Springer, Heidelberg (1982)
3. Breitenecker F.: Optimierung in kontinuierlichen Simulationssprachen - Aspekte bei Modellen technischer Systeme, Informatik-Fachbericht 85, Springer, Heidelberg (1984), 656-660

4. Breitenecker F.: Optimization in simulation packages and languages for continuous processes. In A. Sydow, M. Thoma, R. Vichnevetsky (eds.): System Analysis and Simulation 1985, vol. II, Mathematical Research vol. 28,(1985) Akademie-Verlag, Berlin, 78-81

5. Breitenecker F., Gräff M., Ruzicka R., Sauberer A., Troch I.: Optimierung in ACSL, GOMA - Ein Preprozessor zur automatischen Generierung von Optimierungsprogrammen, TU Wien, Institut f. Analysis, Techn. Mathematik u. Versicherungsmath., Abt. Regelungsmath., Hybridrechen- u. Simulationstechnik, Abteilungsbericht Nr.4 (1987)

6. Troch I.: Simulation and optimization, Proc. European Simulation Conference 87, Prague (im Druck)

7. Gräff M.: Berechnung von optimalen Steuerungen für dynamische Prozesse durch Parameteroptimierung, TU Wien, Institut f. Analysis, Techn. Mathematik u. Versicherungsmath., Abt. Regelungsmath., Hybridrechen- u. Simulationstechnik, Abteilungsbericht Nr.2 (1986)

8. Gräff M., Breitenecker F., Troch I.: Praktische Parameteroptimierung in ACSL, Interface 23 (1986)

CAPS - Compartment Analyse und Programm Synthese

Ein Programm zur Verarbeitung von Kompartmentsystemen

A. Sauberer, F. Breitenecker
Institut für Analysis, Technische Mathematik und
Versicherungsmathematik

Dieser Beitrag behandelt Modellbildung von dynamischen Systemen biologischer, biochemischer oder pharmakodynamischer Aufgabenstellungen mit Hilfe von Kompartmentsystemen. CAPS ist ein Programm, das die Beschreibung eines (nichtlinearen) Kompartmentsystems, das graphisch als Menge von Knoten (Kompartments, nichtlineare Elemente), welche durch Kanten (Raten) verbunden sind, gegeben ist, in eine Simulationssprache (hier ACSL) übersetzt. CAPS unterstützt die Formulierung von Optimierungs- und statistischen Aufgaben und das Umwandeln in eine spezielle Klasse von Differentialgleichungen: die Volterrasysteme. Das Programm CAPS wurde im Rahmen einer Dissertation entwickelt ([1]).

1. Einleitung

Unter Kompartments versteht man unter dem Gesichtspunkt des Anwenders einen idealisierten "Speicher" einer Substanz. Kommt eine betrachtete Substanz in einem biologischen System in verschiedenen Formen oder an verschiedenen Orten vor, dann möge jede Substanz in einer bestimmten Form und/oder an einem bestimmten Ort ein Kompartment bilden. Die Beschreibung der Funktion des Kompartments als Speicher erfolgt durch Massen-Bilanzen: Die zeitliche Veränderung der Substanz im Kompartment ergibt sich als Differenz der Summe der zu- und der Summe der abfließenden Mengen, die durch die Raten beschrieben werden.

Vom mathematischen Standpunkt ist ein Kompartment klarerweise eine Komponente der Lösung eines gewöhnlichen Differentialgleichungssystems 1.Ordnung.

Anwendung finden Kompartmentmodelle u.a. bei folgenden Problemen:

* Beschreibung chemischer Vorgänge
* Beschreibung von Materialflüssen
* Pharmakokinetik und -dynamik
* Ausbreitung von Krankheiten und Epidemien
* Wachstumsmodelle von Populationen (mit oder ohne Wechselwirkung)

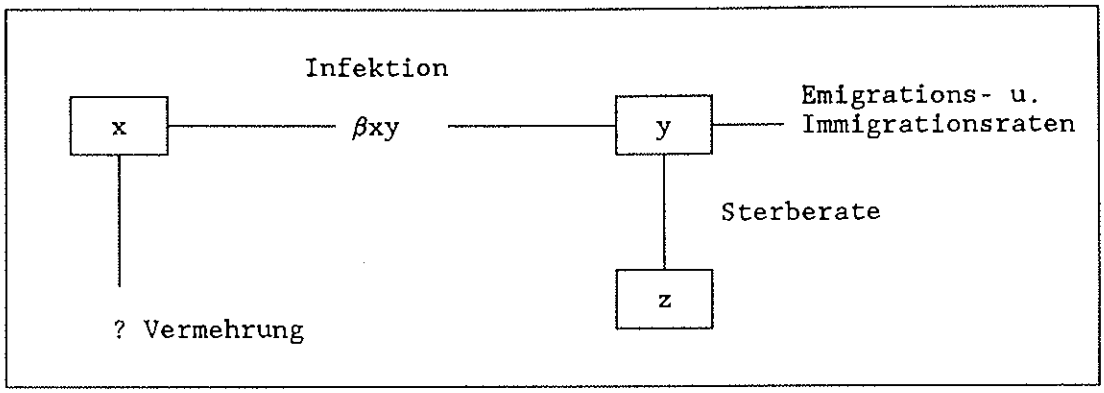
2. Das Programm CAPS

CAPS verlangt als Eingabe ein File, das die Modellbeschreibung eines dynamischen Systems in der "Sprache der Kompartments" enthält. Diese Sprache enthält Elemente zur Definition von Konstanten, Tabellen, Kompartments, nichtlinearen (auch logischen) Blöcken, Infusionen (idealisierte Zufuhren von Substanzen zu einem bestimmten Zeitpunkt oder zu einem bestimmten Ereignis in ein Kompartment, welche in der Zeit 0 durchgeführt werden, also die rechte Seite des entsprechenden Differentialgleichungssystems unstetig machen) und Raten, welche durch eine gerichtete Verbindung von höchstens zwei Kompartments (die Umwelt als Kompartment wird nicht explizit dargestellt) den Zu- bzw. Abfluß in bzw. aus Kompartments im linearen Fall beschreiben. Soll die Rate als Funktion des (der) Kompartments nichtlinear sein, so muß ein nichtlinearer Block dazwischengeschaltet werden, der die nichtlineare Funktion beschreibt.

Weiters können statistische oder Optimierungsaufgaben formuliert, ganze häufiger verwendete Teile als Macros definiert und verwendet, das Modell durch Aufteilung auf verschiedene Files übersichtlicher gestaltet und durch bedingte Anweisungen eine Übersetzung bestimmter Teile beim Aufruf ein- oder ausgeschaltet werden.

3. Ein Beispiel

In [2] findet man ein Modell für die regionale Ausbreitung der Tollwut bei Füchsen. Es werden einzelne Gebiete als Knoten betrachtet, in denen immer die gleiche Struktur der Krankheitsausbreitung zu finden ist. Bedeuten x die Anzahl der gesunden, y die Anzahl der an Tollwut erkrankten und z die Anzahl der gestorbenen Tiere, so läßt sich ein solcher Knoten durch folgendes nichtlineare Kompartmentsystem beschreiben:



Die Beschreibung eines solchen Knotens schlägt sich in folgender Macro-Definition wieder:

```
MACRO knoten(x,y,xicw,yicw);
" Die gestorbenen Tiere z selbst sind uninteressant "
CONST x_ic = xicw, y_ic = yicw;
COMP x(x_ic), y(y_ic);
BLOCK inf_x;
inf-x = beta*x*y;
RATE x_out, y_in, y_out;
x_out = x, inf_x, 1;
y_in  = inf_x, y, 1;
y_out = y, EMPTY, gamma;
"eventuell Vermehrung der gesunden Individuen"
IF verm1 INSERT <vermehr.cap>;
IF verm2 INSERT <vermehr2.cap>;
END;
```

Die Files vermehr.cap bzw. vermehr2.cap beinhalten die Beschreibung der Art der Vermehrung der gesunden Individuen (nur diese vermehren sich), je nach der gewünschten Modellierung: entweder gleichmäßig über das ganze Jahr verteilt, oder als Infusion in jedem Frühjahr modelliert.

Das eigentliche Modell-File enthält die Definitionen globaler Konstanter (z.B. beta, als Infektionsratenkonstante), mehrerer Knoten knoten und die entsprechenden Verbindungen zwischen diesen Knoten (Emigrations- bzw. Immigrationsraten, bedingt durch die Wanderung tollwütiger Füchse über Reviergrenzen hinweg und abhängig von geographischen Verhältnissen).

Einige Simulationen dieses Modells, an Hand eines fiktiven Knotennetzes von Revieren, konnten die Vermutung eines periodisch wiederkehrenden starken Ausbruchs der Seuche bestätigen (siehe [1]).

4. Umwandlung in Volterrasysteme

Volterrasysteme sind spezielle gewöhnliche Differentialgleichungssysteme 1.Ordnung, die häufig in der Populationsdynamik Verwendung finden. Sie zeichnen sich durch eine lineare rechte Seite aus, wenn an Stelle des gewöhnlichen Ableitungsoperators d/dt der logarithmische Ableitungsoperator d(ln)/dt verwendet wird (siehe dazu [3]).

CAPS verfügt auch über eine beim Aufruf angebbare Direktive, die es ermöglicht, eine große Klasse von Kompartmentmodellen (also gewöhnliche Differentialgleichungen 1.Ordnung) in Volterrasysteme (meist größerer Dimension) umzuwandeln, und diese dann auf die Simulationssprache ACSL abzubilden.

Als Beispiel werde das folgende hyperlogistische Wachstumsmodell fomuliert:

```
MODEL hyper;
CONST tend=3;
LOGIC ende;
ende = t>=tend;
TERMT ende;
CONST xic = 1, k=2, u1=1.5, u2=2.5, u3=3, b=100;
COMP x(xic);
BLOCK a;
a = k*(x**u1)*((b-(x**u2))**u3);
RATE xin;
xin = a, x, 1;
END;
```

Der wesentliche Anteil des in ein Volterrasystem übersetzten ACSL-Modells lautet (die Ci und Li sind abgeleitete Konstante, die in der Initial-Section stehen):

```
W7 = INTEG((C9*W5+C10*W5-C15*W7-C6*W5+C14*W7)*W7,L8)
W5 = INTEG((C10*W5-C15*W7-C6*W5)*W5,L6)
X = INTEG((C6*W5)*X,XIC)
```

5. Verfügbarkeit

Die ACSL-Lizenz wurde vom Simulationsrechenzentrum zur Verfügung gestellt.

Testversionen des Preprozessors CAPS sind an der Abt. für Regelungsmathematik, Hybridrechentechnik und Simulationstechnik des Instituts für Analysis, Technische Mathematik und Versicherungsmathematik verfügbar. Interessenten mögen sich bei Herrn Doz. F. Breitenecker, Klappe 5374, melden.

6. Literatur

1. Sauberer A.: Darstellung, Analyse und Simulation von Kompartmentsystemen unter Berücksichtigung von Nichtlinearitäten, Dissertation TU Wien (1987)

2. Timischl W.: Influence of Landscape on the Spread of an Infection, Bull. of Mathematical Biology Vol.46, No.5/6, pp 869-877, Pergamon Press (1984)

3. Peschel M., Mende W.: The Predator-Prey Model: Do We Live in a Volterra World ?, Akademie-Verlag Berlin (1986)

BAPS - Bondgraph Analyse und Programm Synthese

Ein Programmsystem zur Verarbeitung von Bondgraphen

R. Ruzicka, F. Breitenecker
Institut für Analysis, Technische Mathematik und
Versicherungsmathematik

Dieser Beitrag behandelt Modellbildung von dynamischen Systemen mit Hilfe von Bondgraphen. Bondgraphen sind eine graphische Darstellungsmethode für dynamische Systeme. BAPS, ein an der TU Wien aus einer Dissertation entwickeltes Programmsystem zur Analyse und Verarbeitung von Bondgraphen, wird vorgestellt ([1]). Es erleichtert die Erstellung von Bondgraphenmodellen, ermöglicht die Verwendung von nichtlinearen Bondgraph-Elementen und erzeugt aus der vom Benutzer erstellten Bondgraph-Beschreibung ein die Systemgleichungen enthaltendes Modell in der Syntax einer Simulationssprache (z.B. ACSL oder HYBSYS).

1. Einleitung

Bondgraphen sind eine graphische Darstellungsmethode für dynamische Systeme. Sie wurden in den 60er-Jahren von Henry Painter eingeführt und später von Dean Karnopp und Ronald Rosenberg weiterentwickelt.

Trotz ihres "Alters" sind sie gerade heute - da die Erforschung und einheitliche Beschreibung von dynamischen Systemen (nicht zuletzt in der Regelungs- und Robotertechnik) aufgrund der fortschreitenden Automatisierung besondere Geltung erlangt hat - wieder hochaktuell.

Zwei Punkte zeichnen die Bondgraphen-Modellbeschreibung aus:

Zuerst einmal lassen sich mit Bondgraphen dynamische Systeme aus diversen Gebieten der Technik beschreiben: mechanische Systeme, elektrotechnische, thermodynamische, hydrostatische, usw. Durch eine einheitliche Notation für Grundbausteine all dieser Gebiete geben Bondgraphen auch die Möglichkeit der Darstellung von interdisziplinären Systemen, wie z.B. das Zusammenwirken eines wasserbetriebenen Generators (hydrodynamisches System) und eines den erzeugten Strom nutzenden Verbrauchers (elektrotechnisches System).

Zweitens werden im Gegensatz zu anderen Notationshilfen Verbindungselemente als neue Atome der Darstellung eingeführt. Durch diese erreicht man, daß die das Verhalten des dynamischen Systems beschreibenden Gleichungen direkt aus den Elementen abgelesen werden können. Die Einfachheit dieser Notation erlaubt z.B. auch die Modellbildung dynamischer Systeme ohne Kenntnis der Differentialgleichungen.

2. Bondgraphen

Die Bondgraphen (laut [2] auch "Bonddiagramme") sind als Weiterentwicklung von Ersatzschaltbildern, Signalflußplänen und Ähnlichem zu verstehen. Die Vielfalt dieser mehr oder weniger praxisnahen Darstellungen für dynamische technische Systeme wird nun - vereinheitlicht und innerhalb gewisser Grenzen verallgemeinert - zur Bondgraphen-Beschreibungsmethode zusammengefaßt. Ein Bondgraph besteht aus Knoten (den "Elementen") und Kanten (den "Bonds"), die diese Knoten verbinden. Die Knoten, die jeweils ein technisches Grundelement symbolisieren, werden mit 0, 1, R, C, I, SE, SF, TR und GY bezeichnet.

Die Kanten werden zur Unterscheidung fortlaufend mit 1 beginnend numeriert. Jede Kante mit der Nummer i trägt eine Fluß- (f_i) und eine Spannungsvariable (e_i). Jeder Elementname kann mehrmals verwendet werden (die Unterscheidung der Elemente erfolgt aufgrund der Nummern der in ein Element mündenden Kanten).

Im folgenden wird kurz auf die physikalische Bedeutung dieser "Knoten" und "Kanten" und der von diesen getragenen Variablen eingegangen:

Neben den oben genannten Bondvariablen "effort" e (dies kann in der Praxis Kraft, elektrische Spannung, Druck, ... sein) und "flow" f (Geschwindigkeit, elektrischer Strom, Volumenstrom) werden auch die Zustandsvariablen Impuls p (= $\int$e dt; in der Praxis Impuls, Spannungsstoß) und Translation q (= $\int$f dt; Weg, elektrische Ladung, Volumen) verwendet.

Die einzelnen Elemente (0 bezeichnet ein sogenanntes Parallelelement - also eine Parallelschaltung von Elementen -, 1 ein Serienelement, R ein Widerstands-, C ein Kapazitäts-, I ein Trägheitselement, TR einen Transformator, GY einen Gyrator, SE eine Spannungsquelle und SF eine Flußquelle) besitzen folgende konstituierende Gleichungen (e und f beziehen sich auf die zum jeweiligen Element führende(n) Kante(n), Großbuchstaben bezeichnen die Elementkonstanten). Diese bilden dann auch die Basis für die Erzeugung der systembestimmenden Differentialgleichungen (hier sind zu jedem Element die entsprechenden Gleichungen angegeben):

R:	$e = R*f$	bzw.	$f = e/R$
C:	$e = \int f\,dt/C = q/C$	bzw.	$f = de/dt * C$
I:	$f = \int e\,dt/I = p/I$	bzw.	$e = df/dt * I$
SE:	$e = SE$		
SF:	$f = SF$		
TR:	$e_1 = e_2*\mu$,	$f_2 = f_1*\mu$	
GY:	$e_1 = f_2*\nu$,	$e_2 = f_1*\nu$	
0:	$e_1 = e_2 = \ldots = e_n$		
	$f_1 + f_2 + \ldots f_n = 0$		
1:	$f_1 = f_2 = \ldots = f_n$		
	$e_1 + e_2 + \ldots e_n = 0$		

(Bei 0- und 1-Elementen müßten noch die Vorzeichen je nach gegebener Energierichtung umgeändert werden).

3. Das Programmsystem BAPS

Wie bereits erwähnt, eignet sich die Bondgraphen-Modellbeschreibung sehr gut zur automatischen Erstellung der systembeschreibenden Differentialgleichungen eines mathematischen Modells. Für die automatische Modellerstellung ist es jedoch wesentlich, im Graphen die abhängigen und unabhängigen Größen (Variablen) zu erkennen.

Die Literatur über Bondgraphen behandelt in vielfältiger Weise lineare Bondgraphen (dies sind solche, bei deren Elementen die konstituierenden Gleichungen bzw. Differentialgleichungen linear sind) (siehe z.B. [2, 3]). In neuester Zeit werden aber auch schon Untersuchungen über nichtlineare Bondgraphen (siehe z.B. [4, 1]) veröffentlicht.

Viele Autoren geben Algorithmen zur Belegung von Bondgraphen-Kanten mit Kausalitäten (d.h. zur Wahl der abhängigen und unabhängigen Größen in jeder konstituierenden Gleichung) (siehe [3, 5, 6]) an. Mit Hilfe dieser Algorithmen (im speziellen unter Verwendung desjenigen für nichtlineare Bondgraphen) kann man nun darangehen, eine automatische Erstellung der Systemgleichungen (mittels Computer) in Angriff zu nehmen.

Die ersten - vor allem: theoretischen - Vorarbeiten zum hier vorgestellten Programm BAPS (Bondgraph Analyse und Programm Synthese) wurden im Rahmen einer Dissertation ([1]) am Institut für Technische Mathematik, Abteilung Regelungsmathematik, Hybridrechen- und Simulationstechnik, durchgeführt.

BAPS besteht aus zwei Teilen:

- dem "Compiler", der einen Bondgraphen und dessen Nichtlinearitäten in einem bestimmten Format einliest und in ein Zwischenformat (hauptsächlich in Präfixnotation) umwandelt, und
- einem "Linker", der das sehr allgemein gehaltene Zwischenformat in eine Simulationssprache übersetzt (z.B. in ACSL [7] oder in HYBSYS [8] bzw. geplant für das SIMULATIONSSYSTEM HYBSYS).

Im allgemeinen beginnt das BAPS-Programm mit dem Namen des Modells und setzt sich dann in drei Teilen fort:

- im 1. Teil wird der Bondgraph in einer z.B. auch in [9] verwendeten Form beschrieben (siehe Beispiel Tabelle 1, 3. Zeile)
- der 2. Teil enthält die Beschreibung von Konstanten (CONSTANT), von in der jeweiligen Ziel-Simulationssprache vorhandenen und im Modell verwendeten Standardfunktionen (EXTERN), zur Deklaration von unstetigen Zustandsänderungen, von Optimierungsumgebungen, von Terminierungsbedingungen, u.v.a.; hier können auch Systemgleichungen, die nicht direkt zur Bondgraphenbeschreibung gehören, angegeben werden (Tabelle 1, 6. bis 9. Zeile);
- der 3. Teil dient der Angabe der Konstanten bzw. - im nichtlinearen Fall - der konstituierenden Gleichungen der Bondgraph-Elemente (bei den Speicherelementen C und I wird auch der Anfangswert der Integration angegeben) und von sogenannten tdj- 0- und 1-Verbindungen, mit deren Hilfe die Bondgraphentopologie zeitabhängig gestaltet werden kann (Tabelle 1, 11. bis 13. Zeile).

Ein BAPS-Programm kann aus mehreren Teilmodellen bestehen, deren jeweilige Variablen lokal sind, die aber jederzeit gegenseitig auf Bond- und Zustandsvariable anderer Teilmodelle zugreifen können. Die Teilmodelle können auf verschiedenen Dateien stehen. Das erste vom Compiler erkannte Modell ist jeweils das "Hauptmodell".

Das Neue an BAPS ist, daß Nichtlinearitäten im Modell direkt verwendet werden. Es können:

- Element-Konstante nichtlinear angegeben und dabei beliebige algebraische Ausdrücke verwendet werden,
- direkt für ein Element die konstituierenden nichtlinearen Gleichungen angegeben werden,
- Ereignisse (in Abhängigkeit von der Zeit oder von Bondvariablen) definiert werden,
- in algebraischen Ausdrücken Varianten, die je nach Eintreffen oder Nichteintreffen des Ereignisses verschiedene Werte annehmen können, verwendet werden,
- mit Hilfe dieser Ereignisse Verbindungen im Bondgraphen zeitveränderlich definiert werden (tdj-Verbindungen),
- zusätzliche Gleichungen und Zuweisungen (z.B. Differentialgleichungen, die als Nichtlinearitäten in Bondgraphengleichungen eingehen) angegeben werden,
- unstetige Zustandsänderungen mithilfe obiger Ereignisse definiert werden,
- Tabellenfunktionen - auch mit Hysterese - definiert und in algebraischen Ausdrücken verwendet werden.

BAPS nimmt im Einklang mit den oben erwähnten Algorithmen während der Übersetzung die Belegung der Kanten mit Kausalitäten und die Überprüfung auf algebraische Schleifen vor.

Aufgrund der so erfolgten Zuweisung der Kausalitäten können nun die Systemgleichungen abgeleitet und in eine Zwischendatei geschrieben werden.

Auf Wunsch erstellt BAPS eine Liste der Kanten und Knoten des Bondgraphen samt den ermittelten Kausalitäten und der vom Benutzer durch die Reihenfolge der Angabe der Bondgraphelemente definierten Energierichtung (die Energierichtung, die sich auf die Vorzeichen in den Systemgleichungen auswirkt, verläuft auf einer Kante immer vom zuerst angegebenen Element zum später genannten).

BAPS unterstützt auch die schrittweise Entwicklung von Bondgraphsystemen, indem nicht näher spezifizierte Teilmodelle verwendet werden können (nur deren Ein- und Ausgangsgrößen müssen angegeben werden). Unbekannte nichtlineare Abhängigkeiten können durch einen einfachen Mechanismus beschrieben werden.

Die Linker-Programme übersetzen die Zwischendateien in Simulationssprachen und überprüfen dabei, ob die verwendeten Nichtlinearitäten in der jeweiligen Zielsprache auch definiert sind (so gibt es z.B. nicht in allen Simulationssprachen die Möglichkeit der Verwendung von mehrdimensionalen

Tabellen; auch sind oft nicht alle handelsüblichen mathematischen Funktionen - wie z.B. arcsin - vorhanden).

4. Anwendungsbereiche und Beispiele

Die Bondgraphenbeschreibungsmethode läßt sich in vielen Gebieten der technischen Wissenschaften anwenden:
Elektrotechnik, Mechanik, Hydrodynamik, Thermodynamik, usw. In diesen Gebieten gibt es jeweils Grundbausteine, die sich durch Bondgraph-Elemente darstellen lassen. Auch interdisziplinäre Systeme, also solche, die mehrere der oben genannten Gebiete berühren, können in einheitlicher Form mit Bondgraphen modelliert werden. BAPS unterstützt dies durch die Möglichkeit der Verwendung von Nichtlinearitäten in vielfältiger Art und Weise, denn "die Natur ist nichtlinear".

Andererseits werden Bondgraphen auch in der Medizin verwendet, wobei für medizinische Systeme mechanische oder elektronische Ersatzmodelle gebildet werden, die dann als Bondgraphen formuliert werden. Auch in der Chemie werden Bondgraphen verwendet (besonders hier sind nichtlineare Elemente an der Tagesordnung).

Ein weiteres Anwendungsgebiet erschließt BAPS durch die Möglichkeit der Angabe von Meta-Text-Befehlen. Darunter versteht man Anweisungen im Sourcetext, die die automatische Erstellung von Textteilen ermöglichen:

- Metavariable, also Variable, die im Laufe der Sourcetext-Abarbeitung verschiedene Werte annehmen können;
- Makros, auch rekursiv;
- if und ifnot;
- while und until;
- im Sourcetext verwendete Namen können mittels der Metavariablen dynamisch generiert werden.

Diese Meta-Text-Befehle ermöglichen u.a. die Generierung von "asymptotischen" Bondgraphmodellen. Darunter versteht man solche, bei denen das zu modellierende System asymptotisch durch eine Folge von Teilmodellen T_n approximiert wird (wobei diese Teilmodelle auch rekursiv definiert sein können; Anwendung: Lösung von partiellen Differentialgleichungen mit Linienmethoden).

Nicht zuletzt sei die Möglichkeit der Definition einer Optimierungsumgebung in BAPS erwähnt. Dadurch können Parameteroptimierungen und -anpassungen vorgenommen werden.

Als Abschluß ein kleines (lineares) Beispiel aus der Elektrotechnik: Eine geschlossene Serienschaltung von Spannungsquelle (sinusförmig), Widerstand und Kondensator.

Abb. 1 zeigt das elektronische Schaltbild, Abb. 2 den Bondgraphen, Tab. 1 den dazugehörigen BAPS-Sourcetext, Tab. 2 den Bondgraphen noch einmal als Kontrollausgabe von BAPS, Tab. 3 das erzeugte ACSL-File und Abb. 3 die Spannung e1 der Quelle, die Spannung e3 am Kondensator und den Strom f1 als ACSL-Plot.

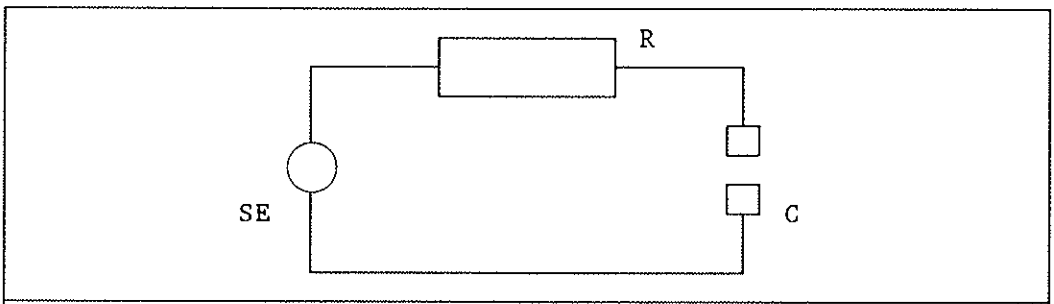


Abbildung 1: Schaltplan eines Schwingkreises

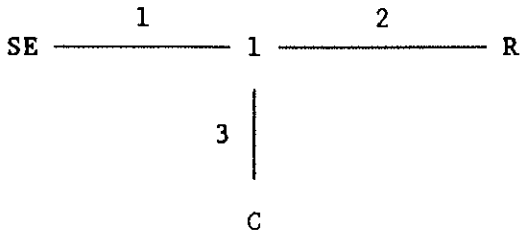


Abbildung 2: Bondgraph zum Schaltplan in Abbildung 1

```
RC_Glied:
     "Bondgraphenbeschreibung"
SE 1; 1 1 2 3; R 2; C 3;
{
     "Konstantendefinition"
CONSTANT TEND = 0.05, frequ=50, f = frequ*2*3.1416, ampl = 220;
EXTERN SIN(1);
     "Endbedingung"
TERMINATE T>=TEND;
     "Elementkonstanten und -gleichungen"
SE 1: SE1=ampl*SIN(T*f);
R  2: R2=100;
C  3: C3=0.0001;          q03=0;
}
```

Tabelle 1: BAPS-Sourcetext

```
     GRAPH RC_Glied:
             (von   Kausalitaet   nach)
     1:     SE 1                              --|  1 1 2 3
     2:     1 1 2 3                           --|  R 2
     3:     1 1 2 3                           |--  C 3
```

Tabelle 2: BAPS Bondgraphenbeschreibung

```
PROGRAM RCGLIE


  CONSTANT TEND = 0.05, FREQU = 50., AMPL = 220., R2 = 100.
  CONSTANT Q03 = 0., C3 = 0.0001

INITIAL
  F = FREQU*2.*3.1416
END   $   "INITIAL"

DYNAMIC

DERIVATIVE
  SE1 = AMPL*SIN(T*F)
  E1 = SE1
  F1 = F2
  F3 = F2
  E2 = E1-E3
  F2 = E2/R2

  E3 = Q3/C3
  Q3 = INTEG(F3,Q03)
END   $   "DERIVATIVE"

  TERMT((T.GE.TEND))
END   $   "DYNAMIC"

END   $   "RCGLIE"
```

Tabelle 3: ACSL-Programm (von BAPS generiert)

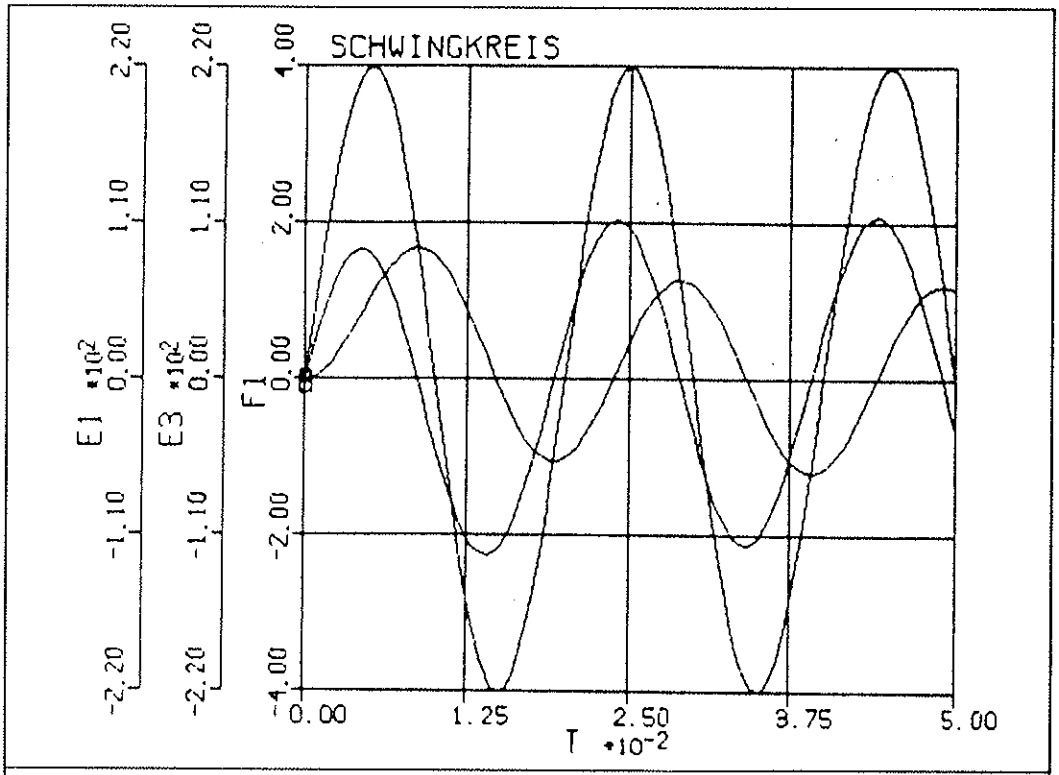


Abbildung 3: Quellspannung e1, Kondensator e3, Strom f1

5. Abschließende Bemerkungen, Verfügbarkeit

Aufgrund seiner sehr allgemeinen Konzeption (im speziellen: dem Format der Zwischendatei) ist es ohne großen Arbeitsaufwand möglich, Linkerprogramme für die verschiedensten Simulationssprachen zu entwickeln (wie oben erwähnt, sind Linker für das gleichungsorientierte ACSL gleichermaßen wie für das - in der vorliegenden Version - blockorientierte HYBSYS vorhanden). Die Tatsache, daß BAPS in C geschrieben ist, gewährleistet ein hohes Maß an Portabilität (es läuft auf Großrechnern genauso wie auf IBM-PC-kompatiblen oder z.B. einem Atari 1040ST).

Um die Eingabe des Bondgraphen noch weiter zu vereinfachen und den oft informatisch nicht so versierten Anwender zu unterstützen, ist derzeit ein graphisches Eingabesystem in Entwicklung.

BAPS wird neben der schon jetzt praktizierten Anwendung als Preprozessor für ACSL auch als Modul in das SIMULATIONSSYSTEM HYBSYS eingebunden werden. Das SIMULATIONSSYSTEM HYBSYS ist eine Weiterentwicklung von HYBSYS IV auf digitaler Basis, unterstützt durch ein Projekt des Fonds zur Förderung der wissenschaftlichen Forschung (siehe Seite 13).

Die ACSL-Lizenz wurde vom Simulationsrechenzentrum zur Verfügung gestellt.

Testversionen des Preprozessors BAPS sind an der Abt. für Regelungsmathematik, Hybridrechentechnik und Simulationstechnik des Instituts für Analysis, Technische Mathematik und Versicherungsmathematik verfügbar. Interessenten mögen sich bei Herrn Doz. F. Breitenecker, Klappe 5374, melden.

6. Literaturverzeichnis

1. R. Ruzicka: Eine Methode zur theoretischen und praktischen Darstellung und Analyse von Bondgraphen unter besonderer Beachtung ihrer Nichtlinearitäten (BAPS - Bondgraph Analyse Programm Synthese), Dissertation TU Wien (1987)

2. J. U. Thoma: Grundlagen und Anwendungen der Bonddiagramme, Verlag W. Girardet, Essen, (1974)

3. D. Karnopp, R. Rosenberg: System Dynamics: A Unified Approach, Wiley-Interscience, New York, (1975)

4. J. J. Granda: Computer Aided Modeling Program (CAMP): A Bond Graph Preprocessor for Computer Aided Design and Simulation of Physical Systems Using Digital Simulation Languages, PhD Thesis, Department of Mechanical Engineering, University of California, Davis (1982)

5. J. Barreto, J. Lefevre: Model Structure in Physiology: A Bondgraph Approach, Proc. of the IFIP-WC 7.1, Ghent, Belgium, North Holland (1982)

6. P. C. Breedveld: A Systematic Method to Derive Bond Graph Models, Proceedings of the 2nd European Simulation Congress, Sept. 9-12 (1986).

7. Mitchell and Gauthier Associates: Advanced Continuous Simulation Language (ACSL) - Reference Manual, Concord, Mass. USA, 4th Edition (1986)

8. D. Solar: Hybrid Simulation System - User Manual, EDV-Zentrum der TU Wien, Abt. Hybridrechenanlage, 4. Auflage, Release 5.3, (Nov. 1984)

9. R. Rosenberg: A Users Guide to ENPORT 4, John Wiley Inc., New York, (1974)

10. J. J. Van Dixhoorn, Simulation of Bond Graphs on Minicomputers, Journal of Dynamic Systems, Measurement and Control, (March 1977)

INTERFACE 24 April 1988