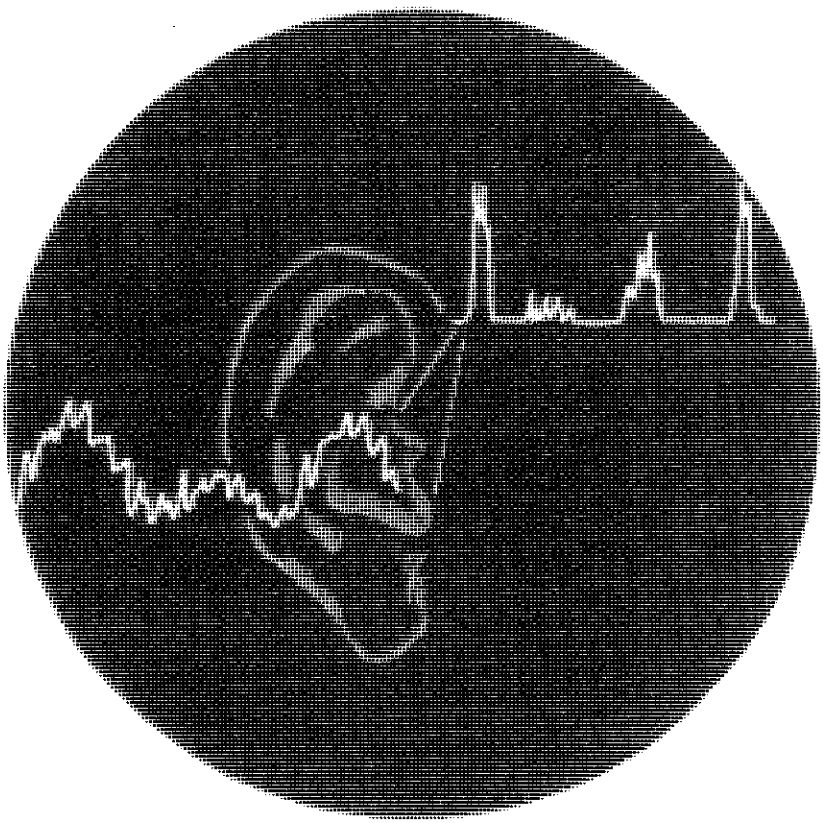
Interface

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

Nummer 20
Dezember 1983



Simulation der Umwandlung eines Tones
in ein Nervensignal

INHALTSVERZEICHNIS

Redaktion: Irmgard Husinsky
Abt. Hybridrechenanlage des EDV-Zentrums der TU Wien, A-1040 Wien, Gußhausstraße 27-29
Herausgeber, Verleger, Hersteller: EDV-Zentrum der TU Wien, Abt. Hybridrechenanlage,
Leitung: Dipl.Ing.Dr.W. Kleinert, A-1040 Wien, Gußhausstraße 27-29
Telephon: (0222) 56 01/3706 oder 3669, Telex: 136875 rzthw a, Modem: (0222) 65 98 17
Druck: Hochschülerschaft Technik, A-1040 Wien, Argentinierstraße 8

10 Jahre INTERFACE

Es ist nun 10 Jahre her, daß wir uns vorgenommen haben, mit der Herausgabe der Zeitschrift INTERFACE die Möglichkeiten und Einsatzgebiete unseres Hybridrechensystems aufzuzeigen. In den vergangenen 10 Jahren wurde bei uns umfangreiche Hardware- und Softwareentwicklung betrieben, sodaß wir zu dem heute laufenden komfortablen hybriden Time-Sharing System, basierend auf einem EAI PACER 600 mit AutoPATCH-Erweiterung, unterstützt durch das virtuelle Betriebssystem JCS/VS und die hybride Simulationssprache HYBSYS, gelangen konnten. Die bisher erschienenen Nummern von INTERFACE geben ein ausführliches Protokoll dieser Entwicklungsarbeit wieder.

Im "Benützerforum" konnten wir eine große Anzahl von Berichten unserer Benützer über ihre Arbeiten an unserem Rechner veröffentlichen. An dieser Stelle sei allen herzlich gedankt, die sich die Mühe nehmen, einen Bericht für uns zu verfassen. Für die Zukunft planen wir, mehr Wert auf praktische Anleitungen zur Anwendung unseres Hybridsystems zu legen. Unter der Rubrik "Simulationstechnik und HYBSYS" sollen anhand von Beispielen hybride Programmiertechniken bei Verwendung der Simulationssprache HYBSYS erläutert werden.

INTERFACE ist noch immer die einzige Publikation dieser Art an europäischen Hybridrechenanlagen. INTERFACE wird an alle Benützer der Hybridrechenanlage versandt und bietet für diesen Leserkreis auch Ankündigungen und technische Informationen. Ferner geht INTERFACE an Interessenten im Universitätsbereich und in Industrie und Wirtschaft. Auch viele ausländische Interessenten beziehen unser INTERFACE. Für unsere nicht-deutschsprachigen Leser versuchen wir immer, einige kurze englische Zusammenfassungen der Anwendungsartikel zu bringen.

Ich freue mich, daß ich von Anfang an in der Redaktion sein konnte, und danke allen Kollegen an der Hybridrechenanlage, ohne deren Mitarbeit es nicht möglich wäre, diese Zeitschrift zu machen.

Irmgard Husinsky
Redaktion

10 Years INTERFACE

10 years ago we started to document applications of our hybrid system by publishing the journal INTERFACE. In the past 10 years extensive hardware and software developments were carried out at our computation centre so that today we have a comfortable hybrid time-sharing system, based on an EAI PACER 600 system with AutoPATCH extension, supported by the virtual operating system JCS/VS and the hybrid simulation processor HYBSYS. The previous editions of INTERFACE give a detailed protocol of those developments.

In the users' section ('Benützerforum') we were able to publish a great number of papers dealing with interesting applications of our users whom we would like to express our thanks here. In the future we plan to put more emphasis on practical instructions how to use our hybrid system. In the section 'Simulationstechnik und HYBSYS' hybrid programming techniques will be demonstrated by means of examples using the simulation language HYBSYS.

INTERFACE is still the only publication of this kind on European hybrid computation centres. INTERFACE is sent to all our users and offers them announcements and technical information. Many people at universities and in business and industry are interested in INTERFACE. also have many readers from abroad. For those who don't understand German we try to offer short summaries of the application papers in English.

I am glad that I could work on INTERFACE from the beginning and would like to thank all my colleagues without whose help it would be impossible to make INTERFACE.

Irmgard Husinsky
Editor

DIGITALE INTEGRATION MIT HYBSYS

D. Solar

Die analoge bzw. hybride Methode zur Simulation dynamischer Systeme besticht durch ihre hohe Rechengeschwindigkeit infolge der parallelen Arbeitsweise und der Integration als elementare Operation und eignet sich daher besonders für die interaktive Untersuchung solcher Systeme. Andererseits wirft der beschränkte Operationsbereich das Problem einer Skalierung und die beschränkte Rechengenauigkeit (3-4 Stellen) sowie Phasenfehler in Abhängigkeit der Rechengeschwindigkeit die Frage nach der Gültigkeit der Lösung auf. Wenn auch Instabilitäten leicht erkannt werden können (jeder Lauf liefert eine andere Lösung), so sind die beiden genannten Probleme besonders bei einem hohen Grad an Nichtlinearitäten und in der Umgebung von instabilen Zuständen nicht trivial.

Die digitale Methode der Simulation dynamischer Systeme bedient sich verschiedener Näherungsverfahren (Integrationsverfahren), wobei die Genauigkeit der Lösung zwar im Rahmen der Maschinengenauigkeit wählbar ist, aber direkt proportional die Exekutionszeit des Verfahrens beeinflußt, welche im allgemeinen die der analogen Methode weit übersteigt. Je nach Eigenschaften des Systems (nichtlinear, steif, unstetig, etc.) bedingt die Wahl des Integrationsverfahrens auch die numerische Lösbarkeit, erreichbare Genauigkeit und benötigte Rechenzeit. Die wenigsten Verfahren liefern eine lokale und globale Fehlerabschätzung, wobei nur die letztere hinreichend für die Gültigkeit der Lösung ist. Ist nur eine lokale Fehlerabschätzung vorhanden, so kann die numerische Lösung beträchtlich von der tatsächlichen Lösung abweichen.
Eine Kombination beider Lösungsmethoden bietet eine Reihe von Vorteilen:

- Skalierung der analogen Schaltung durch Berechnung der Extremwerte aus der digitalen Lösung (zusätzlich zur automatischen On-line-Skalierung),
- Verifizierung der analogen Lösung durch Vergleich mit der digitalen Lösung (oder vice versa),
- Fortsetzen der Simulation auf digitaler Basis, falls die Genauigkeit des Analogrechners nicht ausreicht.

Im allgemeinen wird man die interaktive Untersuchung eines Systems aus Gründen der Rechenzeit analog durchführen und anschließend die wichtigsten Resultate digital verifizieren.

Die Implementation der digitalen Integration in das Simulationssystem HYBSYS ([1]) wurde von Dipl.Ing. Götzl im Rahmen einer Diplomarbeit ([2]) durchgeführt und mithilfe dreier Overlays realisiert, die mit folgenden Befehlen exekutiert werden können:

DPRP
DPRP+SKAL — Der Overlay DPRP generiert aus den gegebenen Modelldaten eine Subroutine, die aus den aktuellen Werten der Zustandsvariablen die rechte Seite des Differentialgleichungssystems (d.h. deren Ableitungen) berechnet. Wahlweise (+SKAL) kann auch ein Code zum Updaten der Extremwerte abgesetzt werden. DPRP muß vor dem ersten Aufruf von DRUN bzw. nach jeder Änderung der Schaltdaten exekutiert worden sein.

DRUN <varlist> — Der Overlay DRUN exekutiert ein (vorläufig festgelegtes) Integrationsverfahren, das unter Verwendung der von DPRP generierten Subroutine das Differentialgleichungssystem löst und den Lösungsverlauf der

in <varlist> angegebenen Variablen abspeichert. Falls DPRP+SKAL aufgerufen wurde, werden die Skalierungsfaktoren aller Variablen aus der digitalen Lösung berechnet.

DPLOT <varlist> Der Overlay DPLOT zeichnet die digitalen Ergebnisse der in <varlist> angegebenen Variablen und ermöglicht in Kombination mit dem PLOT,S Befehl den Vergleich von digitaler und analoger Lösung.

Als Integrationsverfahren wurde ein Runge-Kutta-Fehlberg-Verfahren der Ordnung 5 gewählt, das als Teil des Programmpakets GDP1 (Lösung von Differentialgleichungen mit graphischer Ausgabe, [3], [4]) als Unterprogramm verfügbar ist und eine Schätzung sowohl für den lokalen als auch für den globalen Diskretisierungsfehler liefert. (Eine detaillierte Beschreibung des Verfahrens ist der Beschreibung für das Programmpaket GDP1 ([3]) zu entnehmen.)

In der vorliegenden Implementation sind die relative und die absolute Fehlerschranke fest definiert. Die absolute Fehlerschranke ist mit 10^{-6} festgelegt, die relative Fehlerschranke wird aus der globalen Zeitkonstanten TSG und der globalen Zeittransformation BETA berechnet:

$$RELERR = 10^{TSG-6}/BETA$$

TSG	
1	Normal Seconds
2	Fast Seconds
3	Slow Milliseconds
4	Normal Milliseconds
5	Fast Milliseconds

d.h. es wird ein Bezug zur analogen Rechengeschwindigkeit derart hergestellt, daß die Genauigkeit mit steigender Rechengeschwindigkeit abnimmt.

Die Ausgabepunkte für nachfolgende graphische Darstellungen sind durch die Diskretisierungsanzahl POINTS festgelegt, wobei zu beachten ist, daß bei zu engen Abständen der Ausgabepunkte der Integrationsverlauf behindert werden kann.

Die folgenden zwei Beispiele wurden aus der Beschreibung des Programmpakets GDP1 ([3]) genommen und sollen die Unterschiede von analoger (a) und digitaler (d) Lösung aufzeigen. In beiden Fällen stimmt die analoge Lösung für BETA=10 besser mit der digitalen Lösung überein als für BETA=1, was auf eine Verringerung des Phasenfehlers infolge der langsameren Rechengeschwindigkeit der Integrierer zurückzuführen ist. Für BETA>10 konnte keine zusätzliche Verbesserung erreicht werden.

Die geringere Abweichung der analogen Lösung von der digitalen Lösung im ersten Beispiel ist darauf zurückzuführen, daß dieses nur mit linearen analogen Komponenten (Sinus und Cosinus wurden mithilfe einer Differentialgleichung erzeugt) aufgebaut wurde, während im zweiten Beispiel echte nichtlineare Komponenten auftreten (Quadrat, Multiplikation), deren Spezifikationen bezüglich Rechengenauigkeit von vornherein schlechter sind als die der linearen Komponenten. Weiters tritt im zweiten Beispiel ein gewisser Grad an Steifheit auf, sodaß infolge der erhöhten Genauigkeitsanforderungen die Fehler der nichtlinearen Komponenten stärker ins Gewicht fallen.

Die (analog ermittelten) Phasenbilder zeigen jedoch, daß der qualitative Lösungsverlauf nicht beeinträchtigt wird. In Beispiel 1 konvergiert das Phasenbild gegen eine Ellipse ($\pi^2 y^2 + y'^2 = \pi^2$), in Beispiel 2 bleibt der Grenzzyklus erhalten.

Beispiel 1: Die Differentialgleichung

$$y'' + 2y' + (4\pi^2+1)y = ef(t)$$

mit $f(t) = (3\pi^2+1)\cos\pi t - 2\pi\sin\pi t$

soll für e=1 mit den Anfangsbedingungen

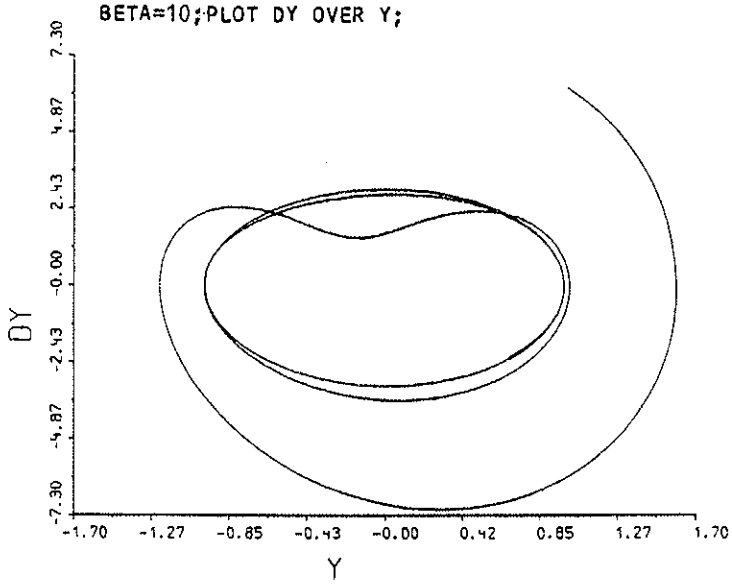
$$y(0) = 1,\ y'(0) = 2\pi,\ t \in [0,2\pi]$$

gelöst werden.

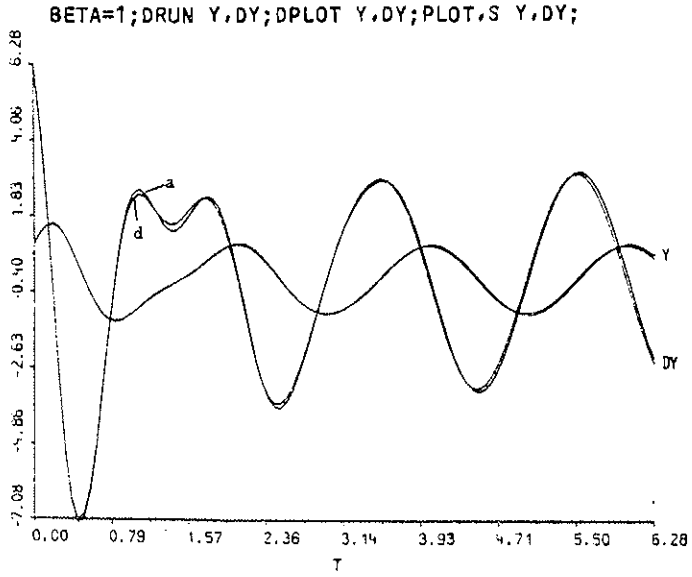
```
MODEL
  PAR:Y0,DY0,E,PI=3.1415
  PAR MULT:PI2
  PI2=2,PI
  PAR SQR:QPI
  QPI=PI
  PAR SUM:A,B
  A=4*QPI,1
  B=3*QPI,1
*
  INT:U,V
  U=PI*V
  V,IC=1,-PI*U
*
  SUM:FT
  FT=B*V,-PI2*U
*
  INT:Y,DY
  Y,IC=Y0,DY
  DY,IC=DY0,-2*DY,-A*Y,E*FT
  END

PREPAR;DPRP;
E=1;Y0=1;DY0=PI2;
SCUNIT=.1;SCALE;
```

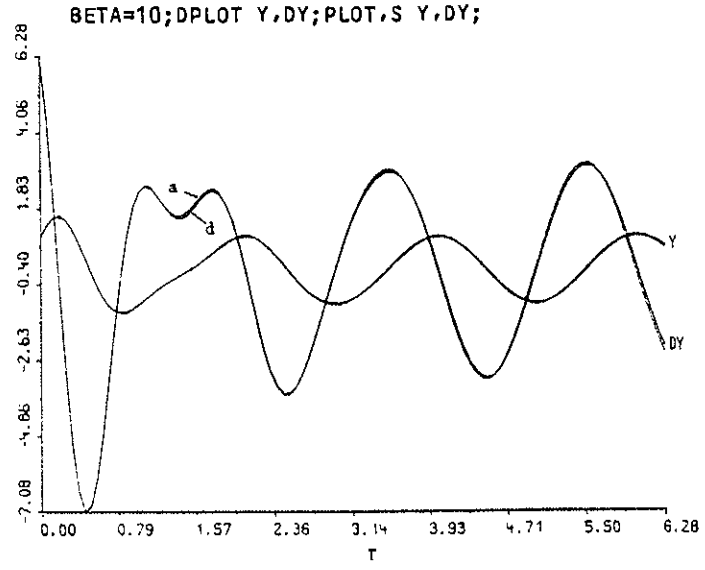
```
BETA=10;PLOT DY OVER Y;
```




```
BETA=1;DRUN Y,DY;DPLOT Y,DY;PLOT,S Y,DY;
```




```
BETA=10;DPLOT Y,DY;PLOT,S Y,DY;
```



Beispiel 2: Die Van der Pol'sche Gleichung

$$y'' - n(1-y^2)y' + y = 0$$

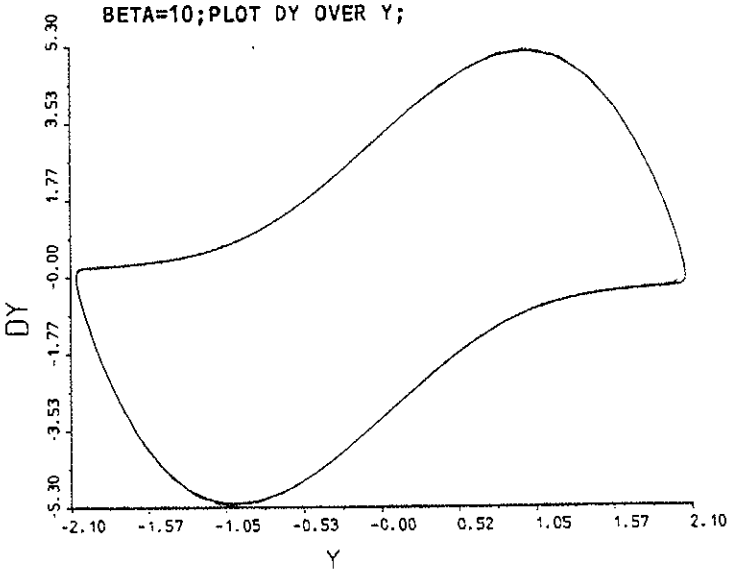
soll für n=3 mit den Anfangsbedingungen

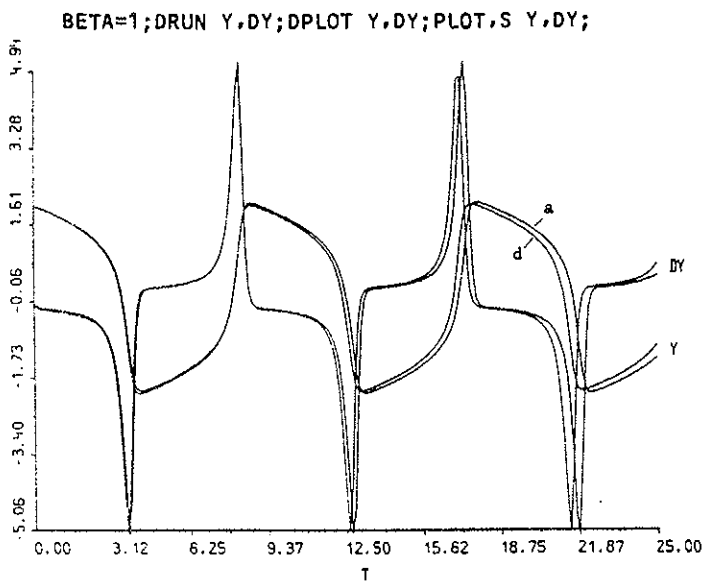
$$y(0) = 2,\ y'(0) = 0$$

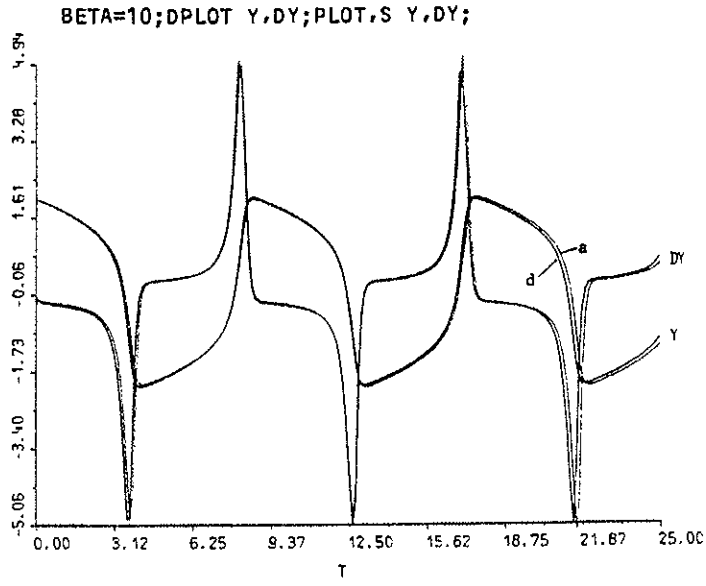
im Intervall $t \in [0,25]$ gelöst werden.

```
MODEL
  PAR:N,Y0,DY0
  INT:Y,DY
  SQR:Y2
  SUM:SY2
  MULT:MDY
  Y,IC=Y0,DY
  DY,IC=DY0,N*MDY,-Y
  Y2=Y
  SY2=1,-Y2
  MDY=SY2,DY
  END

PREPAR;DPRP;
Y0=2;DY0=0;N=3;
SCUNIT=.1;SCALE;
```







Literatur

1. Solar, D.: HYBSYS User Manual, Version 5 TS. Abt. Hybridrechenanlage des EDV-Zentrums der Technischen Universität Wien, Juli 1983
2. Götzl, P.: Integration der digitalen Simulation in das Softwarepaket HYBSYS. Diplomarbeit, Technische Universität Wien, 1983
3. Blöser, F., Mitstorfer, E., Überhuber, C.W.: GDP1 - Programmpaket zur graphischen Darstellung von Lösungen gewöhnlicher Differentialgleichungen. Teil I: Allgemeine Beschreibung. Bericht Nr. 42/79, Inst. f. Num. Math., TU Wien, 1979
4. Blöser, F., Mitstorfer, E., Überhuber, C.W.: GDP1 - Ein Programmpaket zur numerischen Berechnung und graphischen Darstellung der Lösungen von Anfangswertproblemen gewöhnlicher Differentialgleichungen. INTERFACE 15/16, Juni 1980

aktuelle mitteilungen

Modemanschluss

Es ist nun möglich, Remote Terminals über Modems (300 Baud) an die Hybridrechenanlage anzuschließen. Für TU-interne Verwendung stehen die Hausklappen 194 und 195 zur Verfügung. Extern sind wir über die Wiener Rufnummer 65 98 17 ansprechbar. Darüberhinaus können mit einem Akustikkoppler bei Instituten On-line-Vorführungen von HYBSYS durchgeführt werden.

Neuer Drucker

Seit Dezember 1982 steht ein Dataproducts B600 132-Spaltendrucker zur Verfügung (FORTRAN-Einheitennummer 16, SPOOLING-Option erforderlich).

Neues Abrechnungssystem

Seit September 1983 gilt am Hybridrechenzentrum ein neues, verfeinertes Abrechnungssystem für den Betriebsmittelverbrauch, das die tatsächlich verbrauchten Systemresourcen besser wiedergibt.

Die nun verrechneten Systemsekunden enthalten zusätzlich zu den tatsächlich verbrauchten CPU-Sekunden auch Systemservices wie Filemanagement, I/O Services und die Taskladezeiten. Zusätzlich werden jetzt auch alle Accountinginformationen für den Input/Output berücksichtigt, sodaß nun je nach Gerätetyp zwischen 4 g und S 1,20 pro Kilowort Output verrechnet werden, wobei die Ein/Ausgabe auf papierverbrauchenden Geräten am stärksten gewichtet wird. Im Zuge der Neukalkulation wurde der Preis für eine Systemsekunde von S 1,- auf 50 g reduziert.

Für alle Jobnummern, die für wissenschaftliche Arbeiten und Praktika vergeben wurden, wird monatlich eine formale Rechnung erstellt, die an die entsprechenden Institutsvorstände geschickt wird.

EUROPEAN SIMULATION MEETING ON SIMULATION IN RESEARCH AND DEVELOPMENT

27. - 30. August 1984, Eger, Ungarn

organisiert von IMACS/Ungarn, unterstützt von der ungarischen wissenschaftlichen Gesellschaft für Meßtechnik und Automatisierung.

Auskünfte und Anmeldungen bei

Dr. A. Jávor, Central Research Institute for Physics of the Hungarian Academy of Sciences, P.O.Box 49, H-1525 Budapest 114

Neu im Betriebssystem JCS/VS 8

Im folgenden sind kurz die Neuerungen und Verbesserungen im Betriebssystem JCS/VS 8 aufgezählt, die im letzten Jahr durchgeführt wurden.

Es wurde ein neues, virtuelles I/O System realisiert, wobei als logische Ein/Ausgabe-Einheit das Autoterminal eingeführt wurde. Die auf Autoterminal programmierte Ein/Ausgabe erfolgt dabei immer auf dem Terminal, von dem man den Job gestartet hat.

Die Control-Options wurden um einige Options erweitert. Hervorgehoben sei hier die EVENT-Option, die es ermöglicht, sogenannte Events zu setzen, zurückzusetzen oder abzufragen und dadurch die zeitlich logische Abfolge von Programmen (Tasks oder Jobs) zu steuern.

Zur Bearbeitung von virtuellen Feldern wurde ein neues Konzept verwirklicht, wobei die Felder nicht mehr auf Datenfiles, sondern auf eigens dafür bestimmten Massenspeicherbereichen abgespeichert werden, die vom System verwaltet werden. Das Wechseln der Pages wurde wesentlich beschleunigt, wobei eine möglichst optimale Pagegröße durch genaue Messungen ermittelt wurde. Für den FORTRAN-Benützer, der wie bisher die Routinen DEFINE, TOCORE und OFCORE aufruft, ändert sich nur die Identifikation des virtuellen Feldes (Ein-Wort-Identifikation anstelle des 6stelligen Feldnamens). Bei Verwendung des FORTRAN 77 Compilers FTU (siehe auch Seite 28) genügt die Typisierung eines Feldes mit VIRTUAL.

Ein virtuelles Terminalsystem ermöglicht eine komfortablere Benützung der verschiedenen, an die Hybridrechenanlage angeschlossenen Terminals (siehe auch Seite 16 und 18). Es erlaubt eine terminalunabhängige Programmierung der terminalspezifischen Eigenschaften (z.B. Cursorbewegung, Semigraphics, etc.). Das Programmpaket CUR/SR enthält FORTRAN-aufrufbare Unterprogramme zur Programmierung von Cursorbewegung und Bildschirmgestaltung.

Im Oktober 1983 wurde ein neues Districtkonzept eingeführt, das die Run-Time Districts für 8 Batch User vorsieht. Insgesamt 77 Districts sind physikalisch auf 10 Platten verteilt. Die zum Abspeichern von Benützerfiles vorgesehenen Districts sind wie bisher die Districts 13, 14 und 19 - 24.

kurse

Folgende Kurse werden von der Hybridrechenanlage nach Vereinbarung abgehalten:

RH1 Gerätetechnik EAI PACER 600A AutoPATCH-System

RH2 Benützung des Betriebssystems JCS/VS 8

RH3 Hinweise für FORTRAN-Programmierer an der Hybridrechenanlage

RH6 Assembler-Programmierung

RH7 Softwareunterstützung für die Benützung des PACER 600 als Plotter System

RH11 Assembler-Programmierung für Fortgeschrittene mit Übungen

RH13 Bedienung des hybriden Prozessors HYBSYS
RH14 zur Simulation dynamischer Systeme am AutoPATCH-System, Teil 1 bzw. Teil 2

RH16 Programmentwicklung am Terminal

Nähere Auskünfte und Anmeldungen zu den Kursen telephonisch oder persönlich bei Herrn M. Schandl (1040 Wien, Gußhausstraße 27-29, 4. Stock, Zimmer 1404/05, Tel: 56 01 / 3706DW).

Kurse der Abteilung "Regelungsmathematik und Hybridrechen- und Simulationstechnik" des Instituts für Analysis, Technische Mathematik und Versicherungsmathematik:

Kurs AH1 16.1. - 18.1.1984
Simulationen an den Klein-Analogrechnern EAI 1000

Kurs AH2 18.1. - 20.1.1984
Einführung in die hybride Simulationssprache HYBSYS

Kurs AH3 23.1. - 25.1.1984
HYBSYS für Fortgeschrittene

Kurs AH4 25.1. - 27.1.1984
Einführung in die digitale Simulationssprache ACSL (zur allgemeinen Verwendung an CYBER implementiert)

Auskünfte: Dr. F. Breitenecker, Dr. F. Rattay (Klappen 3746, 3747,3754)

EAI Computer Users' Group Meeting 1983

I. Husinsky

Das diesjährige EAI Computer Users' Group Meeting begann am 19. September am Institut für Automatisierungstechnik der Ruhr-Universität Bochum (Prof. K.H. Fasol). Es nahmen 31 Personen aus 7 europäischen Ländern sowie den USA teil, unter ihnen der Präsident der nordamerikanischen EAI Users' Group. Zuerst wurde die Universität vorgestellt und das Institut besucht.

Das Meeting wurde in Altastenberg im Sauerland fortgesetzt. Es gab 7 Präsentationen von EAI Benützern. W. Kleinert legte seine Pläne für ein neues Rechnerkonzept (LATOUR) vor. I. Husinsky berichtete über die Erfahrungen, die im vergangenen Jahr mit dem hybriden Time-Sharing System MACHYS gemacht wurden.

Die Firma EAI präsentierte ausführlich ihren neu entwickelten Simulationsprozessor SIMSTAR. Ferner gab es Working Groups und Diskussionen.

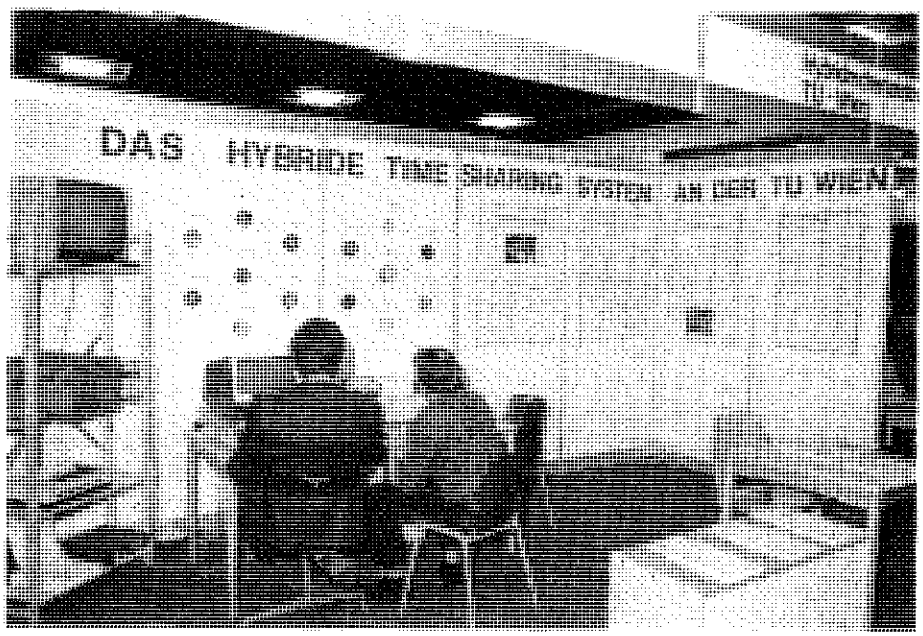
Österreichische Wissenschaftsmesse 1983

F. Berger

Das Hybridrechenzentrum der Technischen Universität Wien war auf der österreichischen Wissenschaftsmesse vertreten, die vom 23. - 26. Februar 1983 im Wiener Messepalast stattfand.

Auf dem Stand wurde das weltweit erste hybride Time-Sharing System MACHYS vorgestellt. Zu diesem Zweck wurde eine direkte Datenleitung über Modem vom Messepalast zum Hybridrechner hergestellt. Für interessierte Besucher konnten kleinere Probleme sofort on-line am Terminal in HYBSYS modelliert und untersucht werden.

Es lagen Informationsblätter über unsere Hardware und Software sowie HYBSYS-Dokumentationsbeispiele auf. Den Stand des Hybridrechenzentrums besuchten schätzungsweise 300 - 400 Messebesucher, von denen etwa 50 Personen sich genauer informieren ließen bzw. einer Vorführung beiwohnten.



First European Simulation Congress

12.9. - 16.9.1983, Aachen

F. Breitenecker, W. Kleinert

In Aachen fand vom 12.9. - 16.9.1983 der "First European Simulation Congress" (ESC 83) statt. Der "European Simulation Congress" ist eine alle drei Jahre stattfindende internationale Tagung, die von den europäischen Simulationsvereinigungen ASIM/GI (Fachausschuß 4.5 Simulation in der Gesellschaft für Informatik), DBSS (Dutch Benelux Simulation Society), SIMS (Skandinavische Simulationsvereinigung) und UKSC (United Kingdom Simulation Council) unterstützt wird.

Die erwähnten Simulationsgesellschaften haben es sich zur Aufgabe gestellt, sich mit allen Aspekten von Simulation und Modellbildung in Theorie und Praxis zu beschäftigen und den Wissens- und Erfahrungsaustausch zwischen Forschern und Anwendern auf diesem breiten interdisziplinären Gebiet zu fördern. Die Gründung dieser "nationalsprachigen" Simulationsgesellschaften, die mit den großen Simulationsgesellschaften IMACS und SCS über Kooperationsverträge zusammenarbeiten, hat sich bisher sehr bewährt, weil durch die räumliche Konzentration der Mitglieder und durch dieselbe Sprache eine intensive Kommunikation möglich ist. Unter den erwähnten Simulationsgesellschaften ist ASIM die deutschsprachige und umfaßt somit Wissenschaftler und Anwender aus der Industrie und den Universitäten in Deutschland, Schweiz und Österreich (Näheres über ASIM siehe Seite 14).

ESC 83 wurde im Rahmen von ASIM von Prof. Dr. W. Ameling (Lehrstuhl für Allgemeine Elektrotechnik und Datenverarbeitungssysteme, Rogowski-Institut für Elektrotechnik, RWTH Aachen) sehr sorgfältig und erfolgreich organisiert.

Das wissenschaftliche Programm gliederte sich in eingeladene Hautvorträge, Beiträge von Tagungsteilnehmern und Round-Table-Diskussionen und wurde durch Vorführungen von Hard- und Software für Simulation ergänzt.

Die eingeladenen Hauptvorträge von G. A. Korn, K. Waldschmidt, D. Popovic, F. E. Cellier und G. C. Vansteenkiste umfaßten den "State-of-the-Art" in Modellbildung und Simulation.

Die Tagungsteilnehmer präsentierten ihre Beiträge in bis zu vier Parallelsitzungen. Die Beiträge waren wie folgt zusammengefaßt: Modellbildung und Identifikation (8), Simulationssoftware (19), Anwendungen in Ingenieurwissenschaften und Mechanik (22), Anwendungen in Medizin (12), Modellbildung und Simulationstechniken (9), Simulationshardware (8), Anwendungen in Chemie (4), Anwendungen in Biologie (3), Anwendungen in Ökonomie (5).

In der Sektion "Simulationssoftware" wurden unter anderem Beiträge über bekannte Simulationssprachen und Neuentwicklungen vorgetragen:

ACSL, CSSL, HYBSYS, GPSS-FORTRAN, SYSMOD, MICRODARE - als Vertreter "kontinuierlicher" Sprachen,

SDS, DYNAMO, SLAM, A68SIM, ADA - als Vertreter "diskreter" Sprachen und

OSCAR, FORCASD, PAWS, COSMOS, MAPLIS - als Vertreter spezieller Sprachen.

Die Sektion "Simulationshardware" beinhaltete die neuen Entwicklungen für Simulationsrechner. Erwähnt sei vor allem das neue Simulationsrechnerkonzept SIMSTAR (P. Landauer, EAI West Long Branch) auf Hybridrechnerbasis, das eine neue Ära für die Simulation von Modellen mit sehr hohen Frequenzen einleiten wird.

Die meisten Beiträge in der Sektion "Anwendungen in Medizin" befaßten sich mit Modellen für das Lang- und Kurzzeitverhalten des Blutkreislaufes und mit Modellen für die Nervenerregung (z.B. Muskelaktivierung).

Die Sektion "Anwendung in Ingenieurwissenschaften und Mechanik" bot ein sehr breites Spektrum, angefangen von Simulationen der Flugzeugdynamik und Rad-Schiene-Dynamik über Simulation hochfrequenter digitaler Regelungskreise bis zu Simulationsaufgaben auf klassischem mechanischem Gebiet (Membran, Brücken,etc.). Diese Aufzählung erhebt keinen Anspruch auf Vollständigkeit; bemerkenswert ist allerdings aber, daß viele der Simulationen technischer Systeme mit Standard-simulationssoftware (ACSL, CSSL,etc.) und/oder mit einem Hybridrechner durchgeführt wurden. In der Sektion "Modellbildung und Simulationstechnik" trugen die Autoren einerseits über Modellbildungsalgorithmen und andererseits über spezielle Simulationstechniken vor.

Vier Beiträge beschäftigten sich bzw.benutzten die hybride Programmiersprache HYBSYS. W. Kleinert stellte den Entwicklungsstand des hybriden Time-Sharing Systems MACHYS vor (W. Kleinert, D. Solar, F. Berger: Status Report on TU Vienna's Hybrid Time-Sharing System). Zwei Arbeiten verwendeten HYBSYS als Simulationssoftware, und zwar für ein medizinisches Modell (F. Breitenecker, W. Kaliman: Simulation of Pathological Blood Pressure Behaviour after Treadmill Test in Patients with Coarctation of the Aorta) und für einen Software-Leistungsvergleich zur Simulation einer direkten Regelung für ein Magnetschwebefahrzeug (F. Breitenecker: Comparison of Digital und Hybrid Simulation of Regulator Plants with Alternatively Continuous and Discrete Feedback). Ein weiterer Beitrag verwendete HYBSYS zur Simulation von allgemeinen Volterrasystemen (M. Peschel, W. Mende, F. Breitenecker: On a New Concept for the Simulation of Dynamic Systems).

Die sehr gut besuchten Round-Table- und Panel-Diskussionen ergänzten das wissenschaftliche Programm und ergaben interessante Aufschlüsse über zukünftige Entwicklungen auf dem Gebiet der Simulation. Diskutiert wurde unter anderem über:

"The Teaching of Modelling and Simulation Techniques at University Level",

"Parallel Computation" und

"The Future of Simulation Software"

In einer ASIM-Mitgliederversammlung wurde die Gründung von drei Arbeitskreisen beschlossen: "Simulation technischer Systeme", "Simulation in Biologie und Medizin" sowie "Simulation asynchroner Systeme". Hauptziele der Arbeitskreise für die nähere Zukunft sind die Ausrichtung eigener Sektionen (mit Schwerpunktbildung) auf künftigen ASIM-Tagungen (ESC und Symposien) und zwischen den Tagungen ein formloses ein- bis zweitägiges Treffen bei einem Arbeitskreismitglied.

Ebenfalls beschlossen wurde, das nächste deutschsprachige Symposium von ASIM in Wien zu veranstalten. Diese Tagung, das "2. Symposium Simulationstechnik" wird an der Technischen Universität vom 25.9. bis 27.9.1984 stattfinden (Näheres siehe Seite 55).

Abschließend muß noch erwähnt werden, daß Prof. Ameling und sein Mitarbeiterstab nicht nur den wissenschaftlichen Ablauf hervorragend organisierten, sondern auch mit einem umfangreichen Rahmenprogramm die persönlichen Kontakte zwischen den Tagungsteilnehmern förderten.

A S I M

FACHAUSSSCHUSS 4.5 SIMULATION IN DER GI

AUFGABEN UND ZIELE

ASIM, der Fachausschuß 4.5 Simulation in der Gesellschaft für Informatik (GI) ist eine Vereinigung, die sich der Förderung und der Weiterentwicklung der Simulation in allen Fachrichtungen widmet.

o ASIM beschäftigt sich mit den folgenden Arbeitsgebieten:

- Methodologie: Modellbildung, Modellvalidierung • Modellbeschreibungsverfahren • Mathematische Verfahren • Systemidentifikation • Optimierungsverfahren
- Verfahren und Methoden der Simulation: Simulation auf Analog- und Hybridrechnern • Digitale Simulation für diskrete, kontinuierliche und kombinierte Systeme • Simulationssprachen • Softwareunterstützung • Künstliche Intelligenz in der Simulation • Datenverwaltung in der Simulation • Hardwareunterstützung: Rechnersysteme und Rechnerarchitektur
- Anwendungsgebiete: Es werden besondere Verfahren und Methoden auf den Anwendungsgebieten entwickelt. Zu den Anwendungsgebieten der Simulation gehören insbesondere: Ingenieurwissenschaften • Mathematik • Naturwissenschaften wie Physik, Chemie usw. • Medizin, Biologie, Ökologie • Wirtschaftswissenschaften • Verwaltung, Planung, Operations Research • Sozialwissenschaften • Ausbildung

o ASIM unterstützt und fördert die Arbeit all derer, die auf dem Gebiet der Simulation tätig sind.

o ASIM organisiert Fachgespräche, Fachtagungen, Arbeitskreise und Projektgruppen.

o ASIM beteiligt sich an nationalen und internationalen Konferenzen.

o ASIM wirkt bei der Herausgabe des Informationsblattes SIMULATION NEWS EUROPE mit.

o ASIM arbeitet als Fachausschuß 4.5 der GI zur Zeit eng mit den folgenden Gesellschaften zusammen:

- SCS (Society for Computer Simulation),
- IMACS (International Association for Mathematics and Computers in Simulations)
- OBSS (Dutch Benelux Simulation Society)
- SIMS (Scandinavian Simulation Society)
- UKSC (United Kingdom Simulation Council)

Der Fachausschuß 4.5 hat zur Zeit die folgende Zusammensetzung:

Prof.Dr.W. Ameling (stellvertretender Sprecher des Fachausschusses)
Lehrstuhl für Allgemeine Elektrotechnik
Rogowski-Institut für Elektrotechnik, RWTH Aachen

Dr. Ingrid Bausch-Gall (Sprecherin der Fachgruppe "Simulationsanwendungen")

Dr. Francois Cellier (Sprecher der Fachgruppe "Simulationstechnik")
ETH Zürich

Dr.H. Fuss
GMD Bonn

Dr. Wolfgang Kleinert (Sprecher der Fachgruppe "Simulationstechnik"),
Technische Universität Wien

Dr. Dietmar Möller (Sprecher der Fachgruppe "Simulationsanwendungen"),
Universität Mainz

Prof.Dr. Bernd Schmidt (Sprecher des Fachausschusses)
Universität Erlangen

ORGANISATION VON ASIM

ASIM gliedert sich in zwei Fachgruppen, die ihrerseits Arbeitskreise für spezielle Simulationsaufgaben einsetzen:

Fachgruppe 4.5.1 Simulationstechnik
Sprecher: Dr. Francois Cellier, Dr. Wolfgang Kleinert
Tätigkeitsfeld: Allgemeine Methodologie der Modellbildung und Simulationstechnik

Fachgruppe 4.5.2 Simulationsanwendungen
Sprecher: Dr. Ingrid Bausch-Gall, Dr. Dietmar Möller
Tätigkeitsfeld: Simulation in der Anwendung

Die bisher gegründeten Arbeitskreise, an denen auch ohne ASIM-Mitgliedschaft mitgearbeitet werden kann, sind:

Arbeitskreis 4.5.1.1 Simulation asynchroner Systeme
Kontaktadresse: Dr. H. Fuss, GMD, Postfach 1240, D-5205 St. Augustin 1

Arbeitskreis 4.5.2.1 Simulation in Biologie und Medizin
Kontaktadresse: Dr.D. Möller, Universität Mainz, Postfach 3980, D-6500 Mainz

Arbeitskreis 4.5.2.2 Simulation technischer Systeme
Kontaktadresse: Dr.F. Breitenecker, TU Wien, Gußhausstraße 27-29, A-1040 Wien

Weitere Arbeitskreise sind in Vorbereitung.

MITGLIEDSCHAFT

Die Mitgliedschaft kann formlos beantragt werden, es wird ein Mitgliedsbeitrag von 20 DM jährlich erhoben.

Vorteile der Mitgliedschaft sind:
- Aufnahme in die Adressliste von ASIM, dadurch laufende Information über alle geplanten Veranstaltungen.
- Ermäßigung des Teilnehmerbeitrags für Veranstaltungen von ASIM und von kooperierenden Gesellschaften.
- Erhalt des Mitteilungsblattes SIMULATION NEWS EUROPE.
- Möglichkeit zur aktiven Mitarbeit in den Fachgruppen.

Auskunft und Antragsformulare bei

Dr. Ingrid Bausch-Gall
Wohlfartstr. 20
D-8000 München 45

BENÜTZUNG DES VIRTUELLEN TERMINALSYSTEMS

F. Blöser

Das virtuelle Terminalsystem im Betriebssystem JCS/VS 8 (siehe Seite 18) gestattet die problemlose Verwendung von Kontrollfunktionen eines Terminals. Diese Funktionen (Setzen von visuellen Attributen, Cursorpositionierung, Ausgabe von Semigraphic-Zeichen, etc.) können geräteunabhängig durch virtuelle Character programmiert werden. Erst bei der tatsächlichen Ausgabe auf ein bestimmtes Terminal werden diese Character durch die für das spezielle Gerät erforderlichen Kontrollzeichen ersetzt. Zu diesem Zweck ist jedem physikalischen Unit Code (d.h. jedem angeschlossenen Controller) über eine Tabelle (den Virtual Terminal Class Vector) eine Virtual Terminal-Klasse zugeordnet.

Alle auftretenden Virtual Terminal-Klassen müssen in der Virtual Terminal Character-Tabelle definiert sein. Wird nun bei der Ausgabe auf einem bestimmten Gerät ein virtueller Character ('170-'237) im Outputpuffer entdeckt, so wird statt dieses Characters die in der Virtual Terminal Character-Tabelle für die betreffende Terminalklasse und den betreffenden Character festgelegte Folge von Zeichen bzw. Operationsvorschriften ausgegeben.

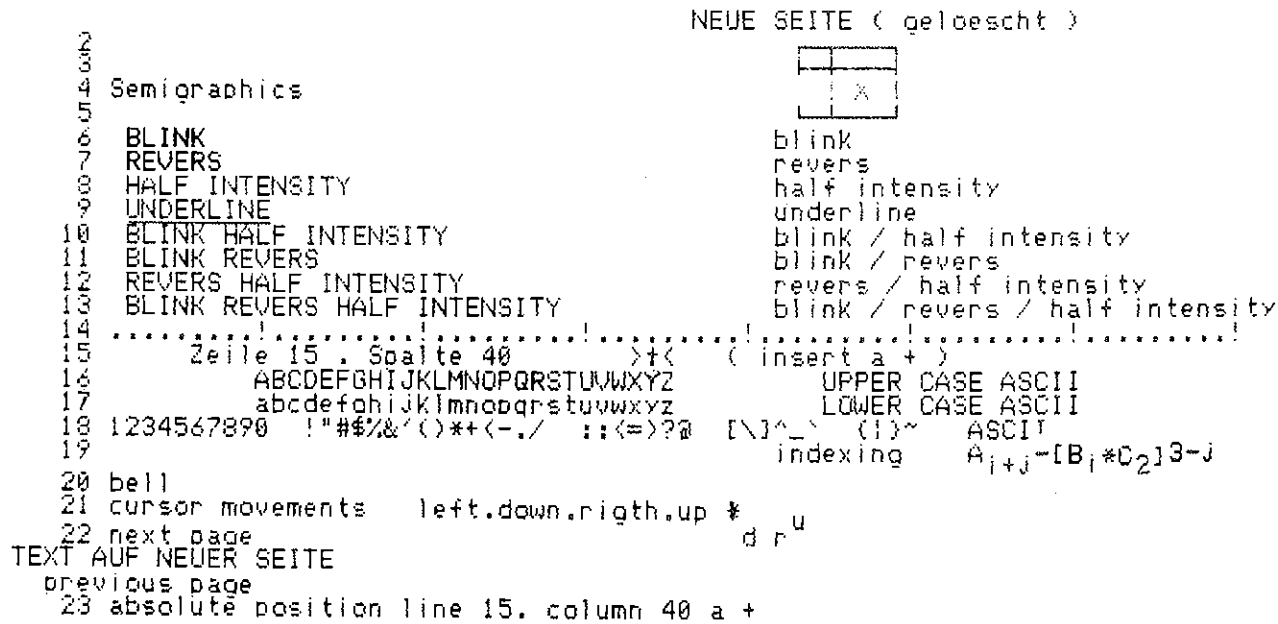
Für alle standardmäßig an der Hybridrechenanlage angeschlossenen Ausgabegeräte sind die Terminalklassen und die den virtuellen Charactern entsprechenden Zeichenfolgen bereits festgelegt. Die Funktionen der virtuellen Character wurden dabei bestmöglich für die einzelnen Geräte realisiert. Für die Benützer eines standardmäßig angeschlossenen Terminals gibt es daher keine Probleme bei der Verwendung von virtuellen Charactern.

Anders ist die Situation für Benützer von Terminals, die nur fallweise an einem Controller, der gerade frei ist, an der Hybridrechenanlage angeschlossen sind. Grundvoraussetzung dafür, daß die virtuellen Character nicht nur bei der Ausgabe wirksam werden, sondern die entsprechenden Funktionen am Bildschirm unmittelbar nach Eintippen der entsprechenden Tasten angezeigt werden, ist der Betrieb des Terminals im System Echo-Mode (Full Duplex-Mode). Außerdem muß im Virtual Terminal Class Vector eine geeignete Terminalklasse für den Controller, an dem das Terminal betrieben wird, gesetzt sein. Diese beiden Aktionen (Versetzen des Controllers in den System Echo-Mode und Zuordnung einer Terminalklasse) können vom Operator an der Systemkonsole durchgeführt werden. Der Full Duplex-Mode muß hingegen vom Benützer auf dem Terminal eingestellt werden.

Derzeit sind 11 Terminalklassen in der Virtual Terminal Character-Tabelle definiert:

```
Class  1: STANDARD CLASS
Class  2: LINE PRINTER
Class  3: HP 2623A / 2648A
Class  4: ADDS REGENT 40
Class  5: ISC INTECOLOR 8001
Class  6: LEAR SIEGLER ADM 42
Class  7: TELEVIDEO 910
Class  8: TALLY T1612
Class  9: ITOH 8510
Class 10: ADDS VIEWPOINT/3A+
Class 11: ADDS VIEWPOINT
```

Handelt es sich bei dem Benützerterminal um eines der in der Tabelle angeführten Geräte, so ist vom Operator nur die entsprechende Klasse für den Benützer-Controller zu setzen. Zur Kontrolle, ob eine Terminalklasse auf das Benützerterminal zutrifft, gibt es den Source File VTCHAR auf District 26, der virtuelle Character enthält und auf dem Benützerterminal gelistet werden kann.



VTCHAR für Klasse 9

Falls keine Terminalklasse zutrifft, so kann eventuell eine neue Klasse definiert werden. Die Virtual Terminal Character-Tabelle ist auf dem Virtual Terminal File gespeichert, von dem sie nur jeweils beim Coldstart geladen wird. Zum Editieren dieses Files gibt es das interaktive Virtual Terminal File-Kontrollprogramm VTFCON. Damit kann eine neue Terminalklasse definiert werden, wobei die Belegung der Zeichen zunächst wie die der Standardklasse oder irgendeiner anderen bereits existierenden Klasse gewählt werden kann. Im Anschluß können dann einzelne Zeichen, die von dieser Grundbelegung abweichen, entsprechend geändert werden.

DAS VIRTUELLE TERMINALSYSTEM

A. Blauensteiner

Im Betriebssystem JCS/VS 8 der Hybridrechenanlage ist ein virtuelles Terminalsystem implementiert. Jedes angeschlossene Terminal wird mit einer speziellen Terminalklasse identifiziert, welche dem Gerätetyp des Terminals entspricht. Dadurch können alle Ausgabezeichen auf dieses Terminal je nach Gerätetyp und der gewünschten Funktion virtualisiert werden. Das heißt, daß der Programmierer seine gewünschten Kontrollfunktionen auf dem Bildschirm unabhängig vom Gerätetyp programmieren kann, und diese Funktion auf jedem beliebigen Terminal durchgeführt werden kann.

VIRTUAL TERMINAL CHARACTER (VTC)

octal	Taste CONTROL	SYMBOLIC CODE	Beschreibung	Typ
170		FH	Form Feed Home	SC
171		BP	Back Page	SC
172		SCRH	Reverse Half Intensity	SC
173		SCBRH	Blink Reverse Half Intensity	SC
174		CL	Clear Rest of Line	SC
175		RM	Roll Mode	SC
176		PM	Page Mode	SC
177		BD	Backspace & Delete	CC
200				
201	A	SGLM	├	SG
202	B	SGMD	┴	SG
203	C	BS	← Cursor left	CC
204	D	SGRM	┤	SG
205	E	SCB	Blink	SC
206	F	FP	Form Feed next Page	SC
207	G	BELL	Audio Tone	GC
210	H	FS	→ Cursor right	CC
211	I	TAB	Tabulator	CC
212	J	LF	↓ Cursor down (Line Feed)	CC
213	K	HLF	Half Line Feed	CC
214	L	FF	Form Feed Clear	SC
215	M	CR	Carriage Return	CC
216	N	SGRO	┘	SG
217	O	UP	↑ Cursor up	CC
220	P	SCR	Reverse	SC
221	Q	CCP	Cursor Position absolute	CC
222	R	SCN	Normal Display	SC
223	S	SGMM	┼	SG
224	T	SGLU	┌	SG
225	U	SGRU	┐	SG
226	V	SGLD	└	SG
227	W	SCH	Half Intensity	SC
230	X	HUP	Half Cursor up	CC
231	Y	SGMU	┬	SG
232	Z	SCU	Underline	SC
233	[	ESC	Escape	GC
234	\	SGH	─	SG
235	]	SGV	│	SG
236	^	SCBR	Blink Reverse	SC
237	<-	SCBH	Blink Half Intensity	SC

Es gibt bei den virtuellen Charactern vier verschiedene Gruppen:

- Die SEMIGRAPHICS (SG), das sind Zeichen, aus denen sich Rechtecke und deren Unterteilungen zusammensetzen lassen,
- SCREEN CONTROL-Funktionen (SC), welche generelle Funktionen am Bildschirm durchführen wie Löschen des Bildschirmes oder alle visuellen Attribute: Blinken, halbe Intensität, invertierter Hintergrund und Unterstreichen nachfolgender Texte,

- GENERAL CONTROL -Zeichen (GC), welche allgemeine Terminalfunktionen darstellen und
- CURSOR CONTROL-Zeichen (CC), welche den Cursor positionieren bzw. kontrollieren können.

Manche dieser Funktionen können nicht bei allen Geräten implementiert werden und werden dann zu einer entsprechenden Default-Funktion umdefiniert. Wieder andere Funktionen können nur über Umwege realisiert werden, dies geht jedoch nur zu Lasten der Output-Geschwindigkeit.

Die virtuellen Terminalcharacter sind derzeit nur für Outputfunktionen realisiert und nehmen hier einen bestimmten Teil des möglichen programmierbaren Codespektrums ein.

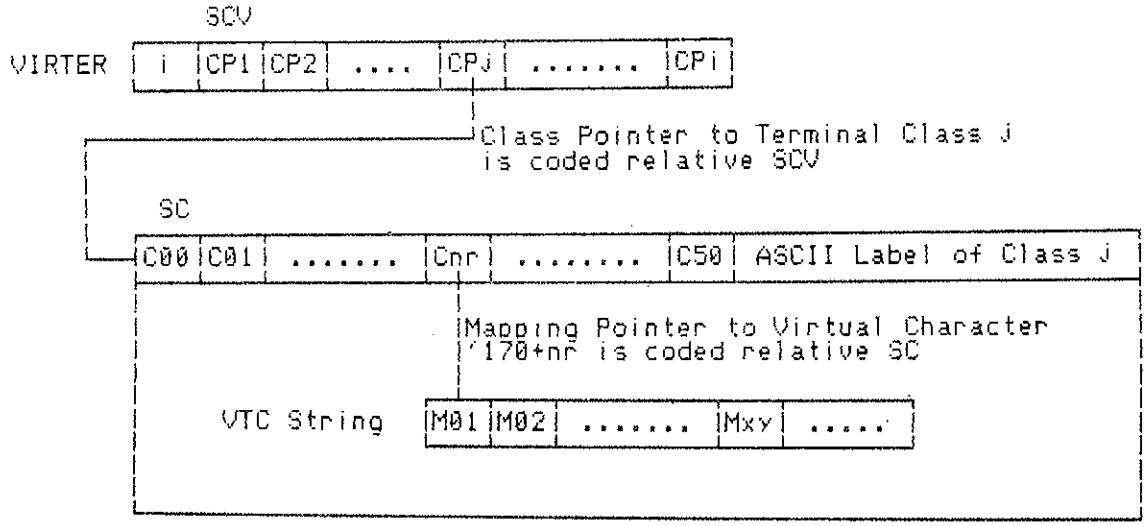
OCTAL VALUE	DEFINITION	
	OUTPUT	INPUT
Ø - 167	BINARY	
17Ø - 237	VTC	
24Ø - 377	ASCII	

I/O BYTE

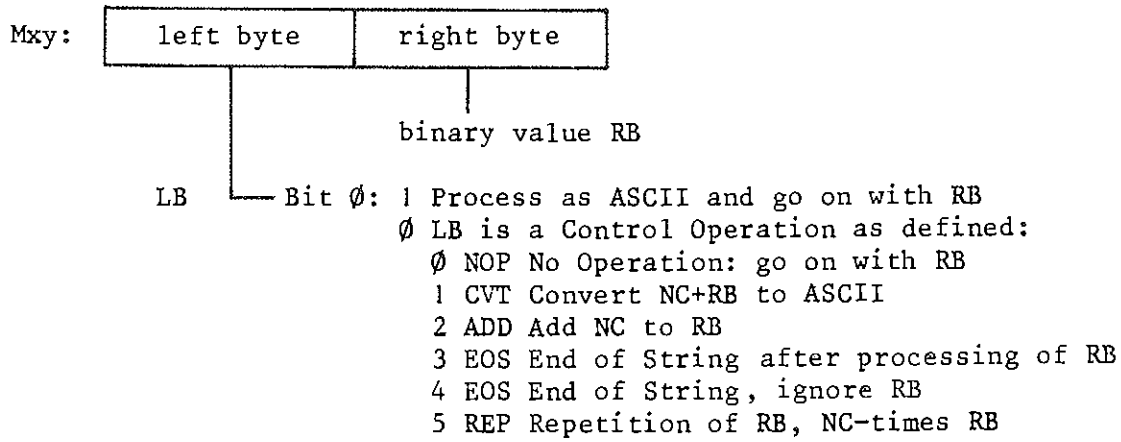
Die restlichen Character werden nach dem normalen ASCII-Standard interpretiert bzw. sind binär. Der Programmierer kann sowohl in seinem Assembler- als auch in seinem FORTRAN-Programm die in der Tabelle fix definierten Funktionen durch ein Byte programmieren. So kann er in seinem FORMAT-Befehl alle ASCII-Zeichen, also auch die Kleinbuchstaben und die Sonderzeichen genauso codieren wie die virtuellen Terminalfunktionen, indem er die entsprechenden Kontrolltasten drückt. Dies hat auch den Vorteil, daß unmittelbar nach Eintippen der entsprechenden Tasten durch System-Echo am Bildschirm die gleichen Funktionen angezeigt werden wie beim Auflisten des Programmes.

Die Realisierung der virtuellen Character im System beruht auf einer dynamisch strukturierten Tabelle, in der alle implementierten Terminalklassen verschlüsselt dargestellt sind. Diese Tabelle wird jeweils nur beim Coldstart geladen. Bei der Ausgabe eines virtuellen Characters wird dieser bis zu seiner tatsächlichen Bearbeitung in seiner virtuellen Darstellung in den Puffern mitgeführt, erst bei seiner tatsächlichen Ausgabe wird in der Interrupt-Routine des Betriebssystems dieser Character durch einen durch die Tabelle definierten String anderer Character bzw. Operationsvorschriften ersetzt. Dadurch belegen auch komplizierte Konstruktionsvorschriften nur ein Byte Puffer-Platz. Außerdem kann der unsynchronisierte Prozeß der ankommenden Output-Daten und deren tatsächliche Ausgabe auf das entsprechende Terminal in eindeutiger Weise abgewickelt werden. Es kann also die Durchführung der entsprechenden Funktionsvorschriften eines virtuellen Characters nicht mehr unterbrochen werden. Andererseits kann diese Konstruktionsvorschrift sehr wohl auf nachfolgende Bytes des Output-Puffers zugreifen, dann nämlich, wenn die Funktionsvorschrift von Parametern abhängig ist, wie etwa beim Cursor-Positionieren in bestimmte Spalten bzw. Zeilen.

STRUKTUR DER VIRTUAL TERMINAL CHARACTER-TABELLE



Der Label einer Virtual Terminal-Klasse wird gepackt in 1Ø Worten dargestellt. Insgesamt sind i Klassen vorgemerkt. Jede Klasse hat einen Class Pointer CPj (j=1,2,...i), der relativ SCV codiert ist. Dieser Pointer zeigt in der Reihenfolge der Klassen völlig unabhängig auf ein Paket mit den codierten VTCs. Dieses an SC beginnende Paket beginnt mit '5Ø Pointern Cnr entsprechend den virtuellen Charactern '17Ø+nr (nr=Ø,1,...'47), welche relativ SC codiert sind. Jeder Pointer Cnr zeigt auf einen String von Worten Mxy, welche byteweise sequentiell interpretiert werden. Statt des Characters '17Ø+nr wird am Terminal der Klasse j dieser String prozessiert.



NC ist der nächste Output Character, der binär [1-'167] in den Output-Puffer des angesprochenen Terminals gelangt. Dieser Wert wird sodann entsprechend der Vorschrift mit dem rechten Byte (RB) verarbeitet. Man beachte, daß das virtuelle Terminalsystem erst dann weiterarbeitet, wenn dieser Character NC eingelangt und prozessiert ist. LB ist das linke (eventuell Control) Byte. Bei der CVT-Funktion wird eine zweistellige Zahl in das entsprechende ASCII-Paar gewandelt.

Der hybride Time-Sharing Algorithmus

W. Kleinert, D. Solar, F. Berger

Der Time-Sharing Algorithmus des hybriden Time-Sharing Systems MACHYS ist als die Synchronisation verschiedener Kopien des interaktiven Compilers HYBSYS implementiert, die als verschiedene Tasks im virtuellen Betriebssystem JCS/VS 8 laufen. Daher basiert der Synchronisationsmechanismus auf dem Schema der rotierenden Prioritäten des Betriebssystems. Bis zu 18 Tasks können vom Betriebssystem gleichzeitig bedient werden, bis zu 8 Tasks können hybride sein. Das Betriebssystem gibt die Kontrolle an den nächsten Task, wenn der Cycle für den gerade laufenden Task verstrichen ist oder falls der Task auf die Beendigung eines I/O-Transfers wartet. HYBSYS verwendet zusätzlich Semaphoren und globale Status-Tabellen, um einen eigenen Level rotierender Prioritäten aufzubauen, der immer nur dem Task den Zugang zur hybriden Simulationshardware erlaubt, der sie gerade benötigt.

Der erste Task, der einen hybriden Lauf anfordert, sperrt die hybride Simulationshardware für alle anderen Tasks, lädt dessen Load/Run Datenbasis, startet den hybriden Lauf und gibt die Kontrolle an den nächsten Task. Jeder Task, der im folgenden einen hybriden Lauf anfordert, gibt nun die Kontrolle sofort an den nächsten Task, da die hybride Simulationshardware noch gesperrt ist. Alle anderen Tasks werden davon nicht betroffen. Der aktive hybride Task gibt die Kontrolle an den nächsten Task solange der Lauf noch nicht beendet ist. Sobald das Ende des Laufs erkannt wird, werden die benötigten Daten von der hybriden Simulationshardware ausgelesen, die Sperre der hybriden Simulationshardware wird aufgehoben und die Kontrolle geht an den nächsten Task. Die hybriden Resourcen sind nun frei für den nächsten Task, der einen hybriden Lauf angemeldet hat.

Dieser Algorithmus bewirkt, daß

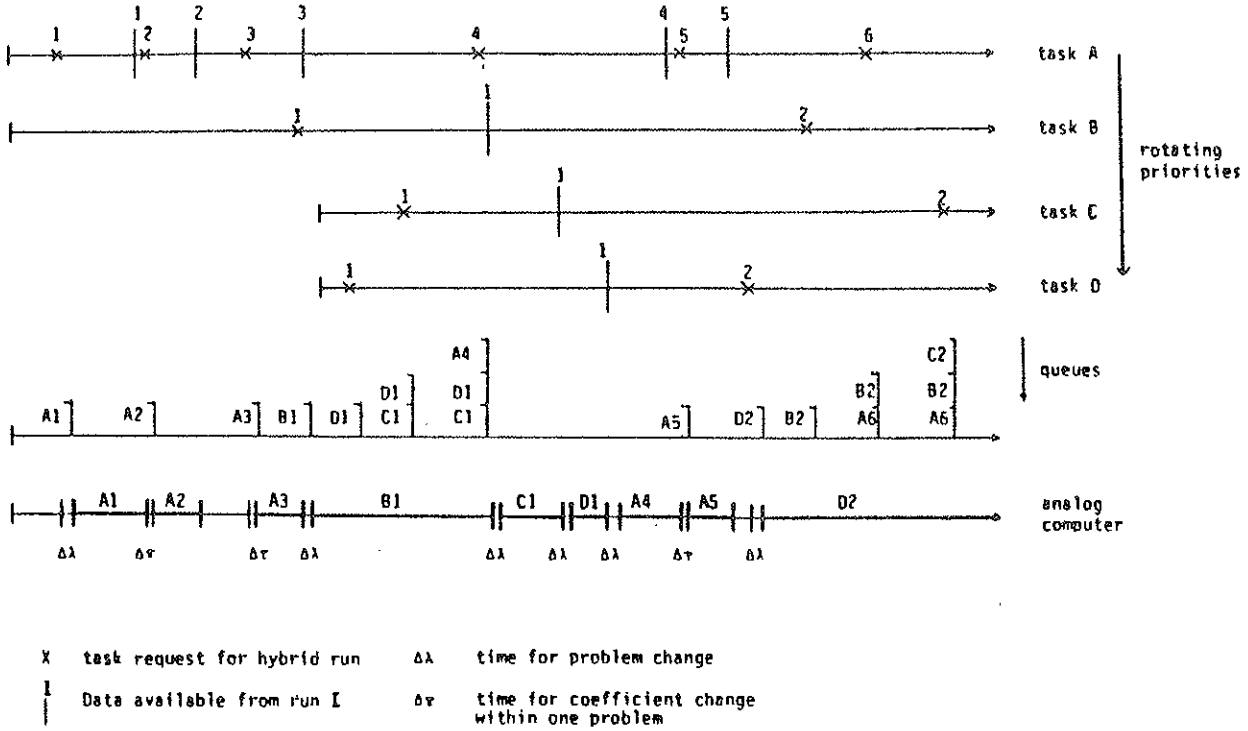
- jeder Task das gesamte Zeitintervall bekommt, das er für einen hybriden Lauf benötigt (Anfangswertübernahme und Operationszeit der Integrierer),
- verschiedene hybride Tasks ihre Läufe in der Reihenfolge der rotierenden Priorität des Betriebssystems erhalten, unabhängig von der Reihenfolge ihrer Laufanforderungen. Daher bekommt kein Task zwei oder mehrere aufeinanderfolgende Läufe, außer wenn keine anderen Anforderungen bestehen oder wenn er privilegiert ist.

Für spezielle Anforderungen wurden zwei andere Arten von Prioritäten implementiert:

Master Mode: Ein Task in diesem Prioritätsmode ist der einzige, der Zugang zur hybriden Simulationshardware hat. Alle anderen Tasks erhalten eine entsprechende Meldung, ihre Anforderung eines hybriden Laufs wird nicht durchgeführt. Dieser Mode wurde für Real-Time-Anwendungen, Tests und Wartungszwecke implementiert.

Privileged Mode: In diesem Prioritätsmode hat ebenfalls kein anderer Task Zugriff zur hybriden Simulationshardware, jedoch wird dieser Mode automatisch vom Compiler an- und abgeschaltet, um zwei oder mehrere aufeinanderfolgende Läufe durchzuführen und dadurch zeitaufwendige Problemwechsel während der Exekution laufintensiver Algorithmen zu vermeiden.

Auszugsweise Übersetzung des beim 1. Europäischen Simulations-Kongreß (ESC 83) präsentierten Vortrages: W. Kleinert, D. Solar, F. Bergér: Status Report on TU Vienna's Hybrid Time-Sharing System. First European Simulation Congress ESC 83, Informatik Fachberichte 71, Springer Verlag 1983, pp 193-200



Zeitdiagramm für rotierende Prioritäten

Die Datenmenge, die bei einer Laufanforderung zur hybriden Simulationshardware übertragen werden muß, ist abhängig von

- den Änderungen, die der Benützer seit dem letzten Lauf an seinem Modell vorgenommen hat,
- den Änderungen, die andere Benützer in der Zwischenzeit (seit dem letzten Lauf des aktuellen Benützers) an der hybriden Simulationshardware verursacht haben.

Die Zeit, die für einen Problemwechsel benötigt wird, hängt offensichtlich von der Datenmenge ab, die bei jedem Lauf übertragen werden muß. Um diese Problemwechsel zu minimieren, wurde die Load/Run Datenbasis in drei Gruppen unterteilt:

Prepare-Daten: sie enthalten die Daten für den Set-Up der analogen und logischen Switch-Matrizen und die Wahrheitstabellen für den Pipelined Boolean Processor. Diese Daten ändern sich bei einer Modelländerung.
Untergruppen: analoge und logische Modelldaten.

Set-Daten: sie enthalten hauptsächlich die analogen und logischen Koeffizienten. Diese Daten ändern sich, wenn ein Parameter geändert wird.
Untergruppen: analoge und logische Koeffizienten, je eine Untergruppe für jede hybride DA-Variable und jeden hybriden Funktionsgeber (HYFU).

Run-Daten: sie enthalten die angeforderten Resultate des hybriden Laufs. Diese Daten werden nach jedem Lauf übertragen.
Untergruppen: je eine Untergruppe für jede hybride AD-Variable und für jeden Selektor-Kanal.

Für jede der erwähnten Untergruppen besitzt jeder hybride Task einen USE-Indikator, der gesetzt wird, falls die entsprechenden Teile der Hardware verwendet werden, und einen SET-Indikator, der gesetzt wird, falls eine quantitative Änderung gemacht wurde, wodurch angezeigt wird, daß diese Daten neu geladen werden müssen, obwohl kein Task-Wechsel erfolgte.

Eine globale CHANGE-Status-Tabelle gibt für jede Untergruppe den Identifikationscode des Task an, der diese Komponenten zuletzt verändert hat. Diese Tabelle wird nach jedem Lauf auf den aktuellen Stand gebracht. Durch folgende Formel kann angegeben werden, ob eine Untergruppe i für eine Laufanforderung neu geladen werden muß:

$$LOAD_i = USE_i \wedge (SET_i \vee CHANGE_i)$$

Die folgende Tabelle zeigt gemessene Werte für die Set-Up-Zeit unter verschiedenen Voraussetzungen (einschließlich der Änderung der Status-Tabelle). Die extrem langen Zeiten für Probleme, die den Pipelined Boolean Processor (PBP) verwenden, resultieren daraus, daß diese Daten auf Platte gespeichert sind und für jeden Lauf übertragen werden müssen (Wahrheitstabellen: 16 Blöcke zu je 256 Worten). Da nur diejenigen Blöcke der Wahrheitstabellen übertragen werden, die tatsächlich benötigt werden, ist diese Zeit im allgemeinen wesentlich kürzer.

Required Set up Data	Set up time in ms	Average run delay time in ms	Used Hardware
Prepare Data (Model Change)	6.4 560.0 560.0	60 - 80 1000 - 1500 1000 - 1500	without PBP, HYBMEM with PBP with PBP, HYBMEM
Set Data (Parameter Change)	3.2 3.6 7.0	50 - 80	without PBP, HYBMEM with PBP with PBP, HYBMEM
Run Data (Run Repetition)	2.2	20 - 60	

Die durchschnittliche Laufverzögerungszeit bedeutet die Zeit zwischen dem Ende des letzten Laufs bis zum Beginn des nächsten Laufs, wenn mittlere Maschinenbelegung angenommen wird (2 hybride Tasks und 2 FORTRAN-Compilationen mit einer Cycle-Zeit von je 20 ms, 4 Editor Tasks mit einer Cycle-Zeit von je 6 ms). Diese Zeit beinhaltet auch das Auslesen einer analogen Variablen.

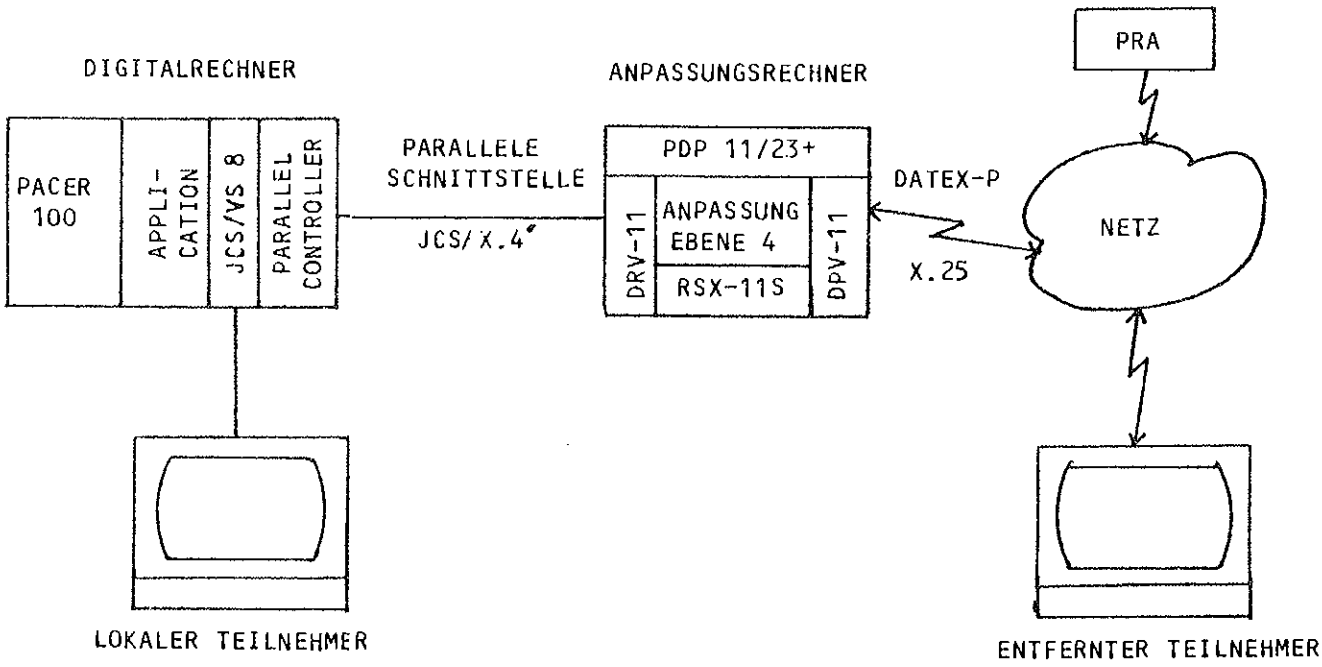
Da der hybride Time-Sharing-Mechanismus von jedem Task selbst durchgeführt wird, kann der gerade aktive hybride Task das Ende seines Laufs nur dann erkennen, wenn er innerhalb des rotierenden Prioritätsschemas an der Reihe ist. Dadurch wird die erwähnte Laufzeitverzögerung verursacht, innerhalb welcher die hybride Simulationshardware inaktiv ist.

Um die zwei schwachen Stellen dieses Systems zu überwinden - die Laufzeitverzögerung und die Lange PBP Set-Up-Zeit - wird der Time-Sharing-Mechanismus in das Betriebssystem eingebaut werden. Das Betriebssystem wird den nächsten angeforderten hybriden Lauf unmittelbar nach dem Ende des vorhergehenden Laufs starten. Dies wird durch einen Interrupt angezeigt werden. Die PBP-Tabellen werden als virtuelle Felder gespeichert werden. Dadurch wird die Verzögerung eliminiert werden.

Anschluss an das DATEX-P Netz

A. Blauensteiner

Im Rahmen eines vom Bundesministerium für Wissenschaft und Forschung, Abteilung II/6 (Dr. Rozsenich, Zahl 60.017/1-26/82) erteilten Forschungsauftrags "Pilotprojekt ACONET Wien" (Projektleitung: o.Prof.Dr. H. Stimmer) wurde die Hybridrechenanlage an das öffentliche DATEX-P Netz angeschlossen. Ziel dieses Projekts war es, in engster Koordination mit ähnlichen Forschungsprojekten in Linz und Graz, die Hybridrechenanlage mit der Prozeßrechenanlage über das öffentliche Datenpaketvermittlungsnetz zu koppeln. Die Implementierung erfolgte in enger Zusammenarbeit mit der Prozeßrechenanlage über einen Anpassungsrechner PDP 11/23 direkt in unserem Betriebssystem JCS/VS 8 und basiert, aufbauend auf das X.25 Protokoll, auf dem ISO-Architekturmodell Schicht 4.



Der Anpassungsrechner kann über einen X.25 Bootstrap direkt vom DATEX-P Netz von der Prozeßrechenanlage (PRA) urgeladen werden. Dadurch ist an unserer Anlage keinerlei Peripherie für den Anpassungsrechner nötig und durch das urgeladene Betriebssystem und die dazugehörige Software für die Ebene 4 und deren Anpassung für unseren Digitalrechner minimiert sich die Software-Wartung. Der Anpassungsrechner wurde über ein synchrones Parallelinterface, das im Rahmen einer Diplomarbeit am Institut für elektrische Regelungstechnik (Vorstand: o. Univ. Prof. Dr. A. Weinmann) entwickelt wurde, an den PACER 100 gekoppelt.

Es werden insgesamt 4 virtuelle Circuits unterstützt, das heißt, daß von allen an der Hybridrechenanlage laufenden Programmen insgesamt maximal 4 Verbindungen von und zum DATEX-P Netz parallel unterhalten werden können. Diese können von einem oder mehreren Usern aufgebaut bzw. verwendet werden.

Der Programmierer sieht seine Schnittstelle zum DATEX-P Netz wie eine normale I/O Schnittstelle und verwendet auch die entsprechenden Betriebssystemaufrufe. Er muß sich nur um den Netzaufbau bzw. Abbau kümmern oder einen entsprechenden Aufbau eines externen Users abwarten. Auch dafür hat er Betriebssystemunterstützung. Die Daten laufen dann über einen Systemmultiplexer über eine einzige synchrone parallele Datenleitung eines entsprechenden Interfaces zum Anpassungsrechner. Dabei wird ein eigenes Datenprotokoll JCS/X.4 verwendet, welches in minimaler Weise den entsprechenden Anforderungen genügt. Im Anpassungsrechner werden die 4 virtuellen Benützer verwaltet und der Datenaustausch mit der entsprechenden Ebene-4-Schnittstelle durchgeführt, welche dann den eigentlichen Netztransfer bewerkstelligt.

Für potentielle Benutzer würden sich vorerst 2 Hauptanwendungsarten ergeben:

1. Time-Sharing-Betrieb mit der Hybridrechenanlage über DATEX-P

Falls der Teilnehmer über ein entsprechendes intelligentes Terminal (ISO Transportprotokoll Klasse Ø) oder einen Rechner mit der entsprechenden Software verfügt, kann er seine Workstation über DATEX-P an unseren Digitalrechner anschließen, indem er die entsprechende LOG-IN Prozedur seiner Netzanschlußprozedur folgen läßt. Danach ist der Teilnehmer regulär an unserem Time-Sharing-Betrieb beteiligt, er kann Source Files editieren und Programme zur Exekution bringen. Durch das virtuelle Terminalsystem hat er vollen Character Support und sein Terminal wird wie ein lokaler Teilnehmer identifiziert. Er erhält eine der physikalischen I/O Devices U16 - U19 und kann logisch jede beliebige Ein-/Ausgabenummer wählen. Entsprechend sind seine FORTRAN-Programme nicht einmal zu modifizieren. Diese werden an seinem Arbeitsplatz die visuellen Attribute, Semigraphics, etc. entsprechend darstellen.

2. Datenaustausch über DATEX-P

An der Hybridrechenanlage steht ein Softwarepaket zur Verfügung, welches den Blockdatenaustausch zwischen einem Programm, welches am Digitalrechner PACER 100 läuft und einem weiteren synchronisierten Programm eines anderen angeschlossenen Teilnehmers über DATEX-P ermöglicht. Dadurch wird die HOST TO HOST Kommunikation ermöglicht bzw. ein Datenaustausch auf Betriebssystemebene denkbar.

Nähere Unterlagen bzw. Informationen über diese Anwendungsmöglichkeiten können direkt an der Hybridrechenanlage eingeholt werden.

FILE-DOKUMENTATION UND FILE-TRANSFER IN JCS/VS 8

F. Blöser

Für die Benützer der Hybridrechenanlage stehen 8 Districts zur Verfügung, auf denen Files (Source Files, übersetzte bzw. gebundene Programme, Datenfiles, etc.) abgespeichert werden können. Derzeit sind auf diesen Benützerdistricts knapp 300 Files gespeichert, die ca. zwei Drittel des verfügbaren Platzes belegen. Um den Benützern und den Operatoren die Übersicht über die Files und die Belegung der Districts zu erleichtern, wurde das Programm FILLIP entwickelt, das Filenamen nach bestimmten Kriterien auswählen und auflisten kann. Mit Hilfe des Programmes TRAFIL können dann diese ausgewählten Files auf einen anderen District transferiert werden, wobei der Transfer verifiziert wird. Auf diese Weise werden etwa jedes Monat automatisch Benützerfiles, die länger als zwei Monate nicht angesprochen wurden, auf eine Wechselplatte ausgelagert.

FILE-DOKUMENTATION

Das Programm FILLIP zur Dokumentation der unter dem Betriebssystem JCS/VS 8 generierten Files unterscheidet zwei Betriebsarten, die beliebig oft und in beliebiger Reihenfolge auftreten können:

- Auswahl von Files nach bestimmten Kriterien
- Auflistung von Files (eventuell sortiert)

Im Normalfall wird der Auswahl von Files jeweils immer eine Auflistung dieser Files folgen.

Bei der Auswahl von Files wird für jeden File, der die gewünschten Kriterien erfüllt, eine ASCII-Fileinformationszeile aufgebaut, die dann auf einen Source File geschrieben wird. (Die Fileinformationszeilen haben dieselbe Gestalt wie die einzelnen Zeilen beim Auflisten eines District mit dem File Controller JCSCON.) Der Auswahl-Modus wird durch eine Kontrollzeile angewählt, auf der auch ein Name und District für den Source File für die Ausgabe gewählt werden kann. An diese Zeile anschließend müssen die Auswahlkriterien durch Selektorzeilen angegeben werden. Mögliche Auswahlkriterien sind z.B. District, Filename, Jobnummer, Filetyp, Generierungsdatum und letztes Verwendungsdatum. Es können bestimmte Werte und zum Teil auch Wertebereiche vorgegeben werden. Mehrere Selektoren können durch Vorzeichen miteinander kombiniert werden, wobei ein Plus ein logisches "Und", ein Minus ein logisches "Und Nicht" bedeutet. Logische "Oder"-Verknüpfungen können durch mehrere Selektorkarten realisiert werden. Ein File wird dann ausgewählt, wenn die Auswahlkriterien mindestens einer Selektorkarte auf den File zutreffen.

Bei der Auflistung von Files wird ein Source File mit Fileinformationszeilen am Short Printer gelistet. Der List-Modus wird ebenfalls durch eine eigene Kontrollzeile angewählt, wobei daran anschließend noch eine Zeile mit den gewünschten Sortierkriterien folgen kann. Mögliche Sortierkriterien sind alle Auswahlkriterien (Selektoren). Die Reihenfolge der Selektoren bestimmt die Sortierreihenfolge. Die Selektoren können mit einem Plus (steigende Sortierreihenfolge) oder einem Minus (fallende Sortierreihenfolge) verknüpft werden.

Im folgenden soll an Hand einiger Kontroll- und Selektorzeilenbeispiele die Verwendung des Programmes FILLIP gezeigt werden. (Für die Befehle würde der jeweils erste Buchstabe ausreichen. Für ein leichteres Verständnis können die Befehle aber auch ausgeschrieben werden, das Programm überliest die entsprechenden Zeichen.)

- Liste aller Source Files auf District 13 und 14, deren Name mit den Buchstaben "HYB" beginnt:

```
SUCHE
TYP(SO)+DIST(13-14)+NAME(HYB###)
LISTE
```

- Liste aller HYBSYS-Overlays mit der Jobnummer 123, die nicht im November 1983 generiert wurden:

```
SUCHE
JNR(123)-GENDATUM(83/11)+TYP(OV)
LISTE
```

- Sortierte Liste aller Files mit der Jobnummer 123, wobei die Files nach District, innerhalb eines Districts alphabetisch sortiert sein sollen. Für eine spätere Verwendung soll der Source File mit den Fileinformationszeilen auf District 14 unter dem Namen MFILES abgespeichert werden:

```
SUCHE MFILES 14
JNR(123)
LISTE MFILES 14
DIST+NAME
```

Eine genaue Beschreibung des Programmes ist im JCS/VS 8 User Manual zu finden.

FILE-TRANSFER

Files können mit dem File Controller JCSCON von einem District auf einen anderen transferiert werden. Für den automatischen Transfer eines einzelnen Files im Verlauf der Exekution eines Batch Jobs gibt es das Programm TRANSI, bei dem der zu transferierende File mittels der FILE-Option und der Zieldistrict mittels der DISTRICT-Option auf der /RUN-Controlline anzugeben sind.

Mit Hilfe des Programmes TRAFIL können mehrere Files transferiert werden, wobei die Namen und Districts der zu transferierenden Files in Fileinformationszeilen auf einem Source File (wie er z.B. von FILLIP erzeugt wird) vorgegeben werden müssen. Auf Wunsch können die Files nach erfolgreichem Transfer auf dem ursprünglichen District gelöscht werden. Am Short Printer wird ein Protokoll über die Transfers gelistet, auf dem auch eventuell aufgetretene Probleme (Namensgleichheit, zuwenig Platz am District,etc.) vermerkt sind.

Unter Verwendung von FILLIP und TRAFIL wurde zu Beginn dieses Jahres damit begonnen, Benützerfiles, die über einen Zeitraum von zwei Monaten nicht angesprochen wurden, automatisch auf eine Depotplatte auszulagern. Von Jänner bis Oktober 1983 wurden Files mit einem Gesamtspeicherbedarf von knapp 40000 Sektoren ausgelagert, das ist etwa das Doppelte des Platzes, der ständig von Benützerfiles belegt ist. Sollte ein ausgelagerter File doch wieder benötigt werden, so können die Operatoren auf Wunsch des Benützers diesen File nachladen.

F T U - FORTRAN 77 COMPILER

I. Husinsky

Der an der Hybridrechenanlage entwickelte FORTRAN 77 Compiler FTU, über den bereits im letzten INTERFACE berichtet wurde, steht nun in seiner ersten Version FTU.Ø den Benützern zur Verfügung. Es gilt die Syntax von FORTRAN 77. Der Befehlssatz ist in FTU.Ø fast vollständig implementiert.

In weiteren Versionen (FTU.1 und FTU.2) werden noch folgende Statements implementiert werden: CHARACTER, COMPLEX und DOUBLE PRECISION Type Statements, IMPLICIT Statement, Character Assignment, Block IF, ELSE IF, ELSE und END IF Statement, Statement Label Assignment, Assigned GOTO und die Statements BACKFILE, ENDFILE und REWIND.

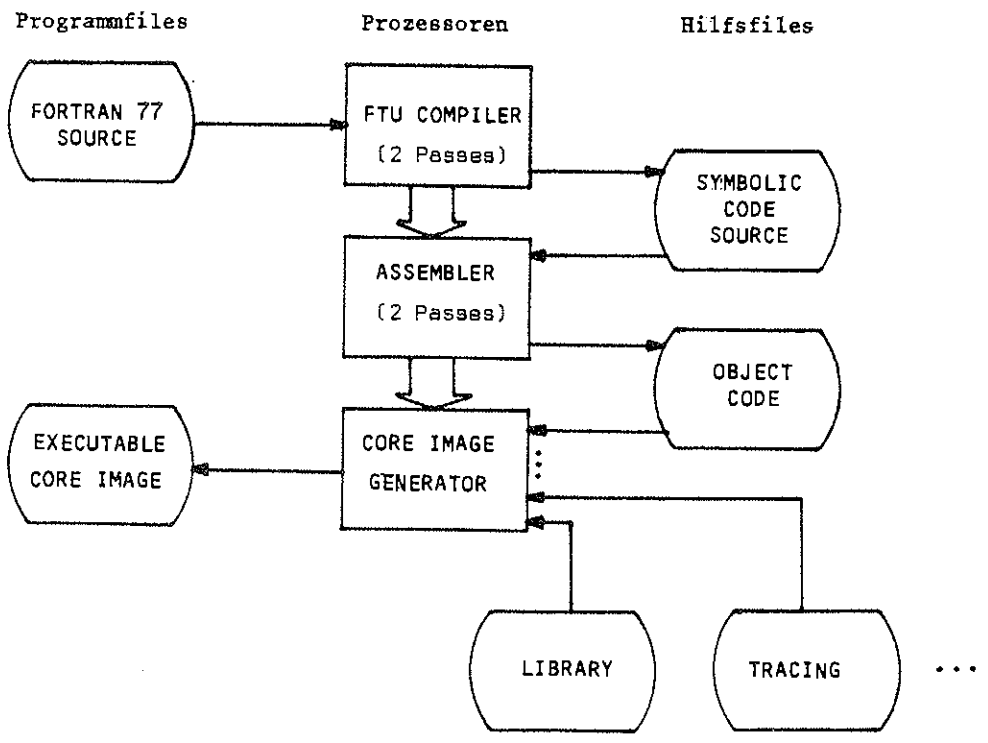
VERWENDUNG

Der FTU-Compiler wird durch eine /RUN-Steuerline aktiviert, mit den FORTRAN Source Lines als Input. Über Options lassen sich verschiedene spezielle Eigenschaften des Compilers aktivieren. Falls Run-Time Tracing gewünscht wird, müssen beim Binden des Programmes die Trace-Routinen dazugeladen werden.

Eine detaillierte Beschreibung zur Verwendung des FTU-Compilers ist an der Hybridrechenanlage erhältlich. Ein ausführliches Manual ist in Arbeit.

GENERIERUNG EINES EXEKUTABLEN FORTRAN-PROGRAMMES MIT FTU

Der Compiler FTU verarbeitet den FORTRAN 77 Source Code in zwei Durchläufen (Pass 1: Label Processing, Pass 2: Code Generation) zu einem symbolischen Code, den er auf einem Source File zwischenspeichert. Aus diesem Code wird vom Assembler ein Objekt generiert, das dann unter Hinzufügen von Library und anderen benötigten Files vom Core Image Generator zu einem exekutablen Programm gebunden werden kann.



EINIGE SPEZIELLE EIGENSCHAFTEN

Jede Feldvariable kann durch die Deklaration mit dem Typ VIRTUAL außerhalb des Programmbereiches, auf Massenspeicher, angelegt werden. Dadurch kann für große Programme Memoryplatz gewonnen werden (Beschränkung 32k pro Programm in JCS/VS 8). Beliebig viele Felder (maximal 32k pro Feld) können VIRTUAL deklariert werden. Der Zugriff zu diesen als virtuell deklarierten Feldelementen erfolgt über einen Zwischenpuffer im Memory über das Betriebssystem.

Die Befehle LABEL und REWIND ermöglichen das Markieren und Positionieren von bestimmten Positionen auf Systemfiles (Inputfile bzw. Spoolfiles).

Oktale Konstante können durch Anhängen von B definiert werden (z.B.: K= 177B). Zur Ausgabe von oktalen Werten steht das O-Format zur Verfügung (z.B.: O6).

Der Compiler liefert eine ausführliche Dokumentation des übersetzten Programmes: jede Zeile wird numeriert (dem Source File entsprechend), Fehlermeldungen erfolgen in Textform (englisch) im Anschluß an die fehlerhafte Zeile (mit Markieren des ersten fehlerhaften Zeichens), jede Printerseite erhält einen Seitenkopf mit Datum, Uhrzeit, Programmname, etc.. Nach jedem Programmteil wird eine Map-Liste ausgegeben: Argumente, Common-Blöcke, Variable, externe Referenzen, Statementnummern.

Auf Wunsch können weitere Eigenschaften aktiviert werden:

- Exekution von Kommentarzeilen, die mit C* beginnen,
- Ausgabe des vom Compiler generierten symbolischen Codes,
- Variableninitialisierung mit Null,
- Erweiterung der Map-Liste (Konstante, Hilfsvariable),
- Ausgabe am Long Printer (breitere Map-Liste),
- Run-Time Tracing,
- Listenunterdrückung,
- Übersetzen ohne Code- und Objekt-Generieren.

RUN-TIME TRACING

Zum Run-Time Tracen eines durch FTU übersetzten Programmes gibt es folgende Möglichkeiten:

- Line Tracing: Tracing vor jeder exekutierten Source-Zeile,
- Label Tracing: Tracing vor jeder exekutierten Source-Zeile, die eine Statementnummer besitzt,
- Subroutine Tracing: Tracing bei jedem Subroutine-Einsprung,
- Do Loop Tracing: Tracing vor jedem Do Loop-Durchlauf,
- Symbolic Tracing: Tracing an Checkpoints des Symbolic Codes.

Die Ausgabe der Trace-Informationen erfolgt auf dem Terminal, wo man das Programm gestartet hat (Autoterminal). Je nach Art des Tracings werden verschiedene Parameter bzw. Systemzellen und Registerinhalte ausgegeben. Interaktiv kann der Benützer zwischen verschiedenen Aktionen wählen:

- Ausschalten und Reaktivieren der einzelnen Trace-Arten,
- Überspringen der Traces, generell oder bis zu einer bestimmten Zeile,
- langsamer oder rascher Ablauf der Traces,
- Abbruch des Programmes.

R A T F O R

EIN FORTRAN-PREPROZESSOR

AM HYBRIDRECHENZENTRUM DER TU WIEN

H. Hummer

Mit dem Vordringen strukturierter Sprachen wurden viele Preprozessoren zum alten Eisen geworfen - möchte man jedoch bei FORTRAN bleiben oder hat man keine andere Sprache zur Verfügung, scheint diese Entscheidung überdenkenswert.

RATFOR (RATional FORtran, entwickelt von B. Kernighan an den Bell Laboratories, zusammen mit einer großen Zahl nützlicher FORTRAN-Tools) ist FORTRAN plus RATFOR-Schlüsselwörter, die bei der Vorübersetzung ausgewertet werden. Das Ergebnis des Vorübersetzerlaufes ist ein Standard-FORTRAN-Quellprogramm, und wer ganz sicher gehen will, kann dieses bei jedem Lauf ebenso archivieren, wie er sein RATFOR-Programm archiviert. Damit wird dem Benützer, der aus Gründen langfristiger Kompatibilität sonst zögern würde, sein Programm in einer Preprozessor-Syntax zu schreiben, zu jedem beliebigen Zeitpunkt ein Umsteigen ermöglicht, denn

- es gibt immer ein Standard-FORTRAN-Quellprogramm zu dem RATFOR-Quellprogramm.

WAS BIETET RATFOR DEM BENÜTZER?

- Bereits existierende FORTRAN-Programme können durch minimale Änderungen in die RATFOR-Syntax übergeführt werden. Diese Änderungen sind im Editor in wenigen Minuten durchzuführen.
- Von diesem Zeitpunkt an stehen dem RATFOR-Benützer leistungsfähige Konstruktionen wie

```
IF ( ) Statementblock ELSE Statementblock    und ELSE IFs,
DO     Statementblock,
WHILE ( ) Statementblock,
BREAK, NEXT
```

  zur Verfügung, die sich besonders zum Durchlaufen verketteter Listen zum Ausschalten umständlicher Abfragen wegen der Möglichkeit nullmaligen Schleifendurchlaufs, zum Initialisieren von Schleifen, für komplizierte Abbruchbedingungen, etc. eignen. BREAK und NEXT bieten elegante Möglichkeiten, Schleifen zu verlassen bzw. das nächste Element zu verarbeiten, ohne Statementnummern beachten zu müssen.

  Außer bei FORMAT kann man in RATFOR gänzlich auf Statementnummern verzichten, sie werden automatisch generiert.

- Das Programm kann formatfrei (d.h. ohne Beachten von Spaltennummern) geschrieben werden, Fortsetzungszeilen werden automatisch erkannt.

- Auftretende Statementnummern werden in die richtige Zone transportiert.
- Pro Zeile sind mehrere Anweisungen möglich - das Quellprogramm benötigt weniger Speicherplatz.
- Nach den Anweisungen ist in derselben Zeile noch Kommentar möglich - das Quellprogramm benötigt weniger Speicherplatz und läßt sich sehr leicht lesen und umfangreich dokumentieren.
- Makromöglichkeiten - Ersetzen von Strings, wie etwa Feldgrenzen, durch zum Prozessierungszeitpunkt definierte Werte. Die Feldgrenzen werden damit zwar nicht dynamisch, aber doch flexibel.
- Umschalten der Eingabe auf andere Source Files. Treten etwa in vielen Unterprogrammen COMMONs auf, können diese auf extra Files einzeln abgelegt werden. Zum Prozessierungszeitpunkt werden diese Programmteile gerufen und eingefügt. Die Übersichtlichkeit des Quellprogrammes steigt damit erheblich, der Platzbedarf sinkt.
- Ob FORTRAN ANSI 66 oder 77, RATFOR bietet seine Eigenschaften als Übermenge über das jeweils verwendete FORTRAN an, dessen jeweilige Leistungsfähigkeit natürlich erhalten bleibt.

Ein Nachteil sei nicht verschwiegen: die Vorübersetzungszeit für ein RATFOR-Programm ist mindestens so groß wie die eigentliche Übersetzungszeit, meist jedoch wesentlich größer. Aus eigener Erfahrung sei hinzugefügt, daß in einem mit RATFOR strukturiert geschriebenen Programm viel Zeit und Mühe bei Fehlersuche, Wartung und Änderung durch die leichte Lesbarkeit des praktisch selbst dokumentierenden Programmes erspart wird. Soviel, daß nach den ersten Anlaßfällen die Wartezeit auf den Vorübersetzer nicht mehr als störend, sondern eher als vorgreifende Entlastung von mühsamer Fehlersuche empfunden wird.

Eine ausführliche Beschreibung liegt an der Hybridrechenanlage auf.

Der Preprozessor wurde uns vom Leiter der Abt. Digitalrechenanlage des EDV-Zentrums der TU Wien, Herrn Dipl. Ing. Dieter Schornböck, zur Implementierung überlassen, wofür wir an dieser Stelle unseren Dank aussprechen möchten.

METAASSEMBLER - ERWEITERUNGEN

F. Berger

In den letzen beiden Ausgaben von INTERFACE wurde der an der Hybridrechenanlage implementierte Metaassembler METASM vorgestellt, dessen Eigenschaften mittlerweile wiederum wesentlich erweitert wurden.

26 OPERANDEN

Ein neuer Code wird durch wahlweises Markieren durch einen Kleinbuchstaben (kb) oder ein Blank und einen Definitionsstring festgelegt. Als Kleinbuchstabe ist der maximal auftretende Operand (a,b,...,kb) anzugeben. Dies bedeutet, daß maximal 26 Operanden definiert werden können.

KEINE SPALTENDOMINANZ

Die für den Benutzer oft störende Spaltendominanz, wie es bei der Assemblerprogrammierung praktisch immer der Fall ist, wurde eliminiert. Die einzelnen Zonen sind durch mindestens ein Blank oder aber beliebig viele Blanks anzuzeigen.

KONSTANTE

Neben den dezimalen Konstanten stehen nun auch oktale (durch vorangestelltes Hochkomma (') gekennzeichnet), hexadezimale (durch vorangestelltes Doppelhochkomma (") gekennzeichnet) und Zeichenkettenkonstante (eingeschlossen durch zwei $-Zeichen) zur Verfügung.

Neben der Bibliotheksroutine META, die einen bestimmten Meta-Objektfile lädt und exekutiert, und dem Loader METCIG, mit dem ein komplett im Metaassembler erstelltes Programm am PACER 1ØØ zur Exekution gebracht werden kann, steht nun auch ein Metaassemblerfile für die Emulation des PACER 1ØØ Codes auf einer PDP 11/23+ und ein Loader LSICIG zur Verfügung. Ebenso gibt es für die PDP 11/23+ einen Cross Assembler und für den INTEL 8086 ist sowohl ein Cross Assembler als auch ein Metaassemblerfile für die PACER 1ØØ-Emulation in Arbeit.

Ein ausführliches Manual ist an der Hybridrechenanlage erhältlich.

Nachfolgendes Beispiel hat ausschließlich demonstrativen Character. Für Frames wurden Namen von Bäumen, für Makros Städtenamen und als Tags Vornamen verwendet.

```
METAASSEMBLER  METASM 8.10.11  JCS/VS 8 * PACER 132  83/11/04 14.13  PAGE      2
LIST OF DEFINITIONS STORED IN META FILE:  EXAMPL    47

     1:   TANNE          Frame       12   LLLL HHHH HHHH
     2:   FICHTE   a     Frame       16   Aaaa LLLL HHHH HHHH
     3:   ESCHE    b     Frame       24   Aaaa LLLL HHHH HHHH Dbbb bbbb
     4:   FOEHRE   a     Frame        8   HLHL XXHH
     5:   HASE          LI      1    8   LLLL HHHH HHHH XXXX XXXX LLLL HHHH HHHH
     6:   IGEL    a     LI      1    8   Aaaa LLLL HHHH HHHH FFFF LLLL HHHH HHHH
     7:   GANS    b     LI      2    8   Aaaa LLLL HHHH HHHH Dbbb bbbb LLLL HHHH
                                        HHHH XXXX Abbb LLLL HHHH HHHH Daaa aaaa
     8:   HUND    c     LI      3    8   Abbb bbbb bbbb Accc LLLL HHHH HHHH Daaa
                                        aaaa FFFF FFFF Aaaa LLLL HHHH HHHH XXXX
                                        XXXX XXXX Abbb LLLL HHHH HHHH Dbbb bbbb
     9:   WIEN              Macro       2   HASE
                                          HASE
    10:   GRAZ    b         Macro       4   HASE
                                          IGEL          a
                                          GANS          ba
    11:   LINZ    c         Macro       7   HUND          bba
                                          HUND,*        acb
                                          IGEL,FOEHRE   b
```

```
METAASSEMBLER  METASM 8.10.11  JCS/VS 8 * PACER 132  83/11/04 14.13  PAGE     1
DEFINITION PASS      META FILE:  EXAMPL     47


 2: ******************************************
 3: *                                        *
 4: *  METAFILE  E X A M P L  DEFINITIONSTEIL  *
 5: *                                        *
 6: ******************************************
 7: *
 8: *  CONTROL PSEUDO CODES
 9: *
10:                      MODE:  N        Normal Mode
11:                      WORD:          8        Wortlaenge: 8 Halbbytes
12:                      DUMMY: H        Don't care Bits sind high
13: *
14: *  FRAME-DEFINITIONEN
15: *
16: <Frame>              TANNE           :4L8H   kein Operand
17: <Frame>              FICHTE a        :4Aa(TANNE)    ein Operand
18: <Frame>              ESCHE  b        :(FICHTE a)8Db         2 Operanden
19: <Frame>              FOEHRE a        :HLHLXX2H
20: *
21: *  META-DEFINITIONEN
22: *
23: *  METAs ohne oder mit Operanden mit Hilfe obiger Frames definiert
24: *  Ein- und Mehrwort-Codes
25: *
26:        <META>         HASE            1:(TANNE)8X(TANNE)
27:        <META>         IGEL    a       1:(FICHTE a)4F(TANNE)
28:        <META>         GANS    b       2:(ESCHE ab)(TANNE)4X(ESCHE ba)
29:        <META>         HUND    c       3:12Ab(ESCHE ca)8F(FICHTE a)12X(ESCHE bb)
30: *
31: *  MACRO-DEFINITIONEN
32: *
33: *  MACROs ohne oder mit Operanden aus obigen METAs zusammengesetzt
34: *  mit frei waehlbaren oder fix definierten Affixes
35: *
36:             <MACRO>   WIEN            (HASE)(HASE)
37:             <MACRO>   GRAZ    b       (HASE)(IGEL a)(GANS ba)
38:             <MACRO>   LINZ    c       (HUND bba)(HUND,* acb)(IGEL,FOEHRE b)
39:                       END


METAASSEMBLER  METASM 8.10.11  JCS/VS 8 * PACER 132  83/11/04 14.13  PAGE     3
ASSEMBLER PASS       2  META FILE:  EXAMPL    BINARY OBJECT FILE:  EXA08     47


 2: ******************************************
 3: *                                        *
 4: *  METAFILE  E X A M P L  ASSEMBLIERUNGSTEIL  *
 5: *                                        *
 6: ******************************************
 7: *
 8: *  Tags werden folgenden Adressen gleichgesetzt:
 9: *
10:                                      0080             ADAM   EQU             128     dezimal
11:                                      0200             WILLI  EQU             '1000   oktal
12:                                      0FFF             CAESAR EQU             "FFF    hexadezimal
13: *
14:            <MACRO>                   0000             XAVER  WIEN
15: *
16: *  Die Adresse der zweiten Instruction des Macros GRAZ wird um 1 erhoeht
17: *
18:            <MACRO>                   0002                    GRAZ            ADAM,*+1
19: 0006  60FF 00FF                      0006                    IGEL            *
20: 0007  00FF F80F F0F0 FFF9             0100 00FF               GANS            '400,CAESAR-WILLI
21:            <MACRO>                   0009                    GRAZ            XAVER,*-XAVER
22: 000D  0FF0 00FF                                       MARKUS HASE
23: *
24: *  Die Adresse des Codes HUND wird quadriert und als dritter
25: *  Operand eingesetzt. Da kein Affix programmiert wurde,
26: *  werden die F-Bits wie Don't care Bits behandelt, in diesem
27: *  Fall also high gesetzt.
28: *
29: 000E  08C4 0FF0 000E 0FF0 0C00 FF72    000E 0080 00C4          HUND            *,ADAM,***
30:                                      0027             ALBERT EQU             WILLI/MARKUS
31:            <MACRO>                   0011                    LINZ            WILLI,ALBERT,ADAM
32: 0018  10FF 00FF                      01F1                    IGEL            $Qa$
33:            <MACRO>                   0019                    LINZ,TANNE      XAVER,MARKUS,FRANZ
34: *
35: *  Da das Affix FICHTE beim Code HUND selbst parameterisiert
36: *  ist, wird an dieser Stelle der erste definierte Operand,
37: *  also FRANZ/2, eingesetzt.
38: *
39: 0020  00CE 0FFF 1101 0FF0 0C00 FFE0    0011 0000 013E          HUND,FICHTE     FRANZ/2,0,"FF+'77
40:                                                       FRANZ  END


                  METAASSEMBLER  METASM 8.10.11  JCS/VS 8 * PACER 132  83/11/04 14.13  PAGE     4
                  TAG TABLE    META FILE:  EXAMPL    BINARY OBJECT FILE:  EXA08     47

                  TAGNAM   ADR   REF      TAGNAM   ADR   REF      TAGNAM   ADR   REF


                  ADAM     0080    3      WILLI    0200    4      CAESAR   0FFF    1
                  XAVER    0000    3      MARKUS   000D    3      ALBERT   0027    1
                  FRANZ    0023    2
```

A C S L

ADVANCED CONTINUOUS SIMULATION LANGUAGE

BEMERKUNGEN UND HINWEISE ZUM TÄGLICHEN GEBRAUCH

RELEASE 7.C I

H. Hummer

Im folgenden sollen einige Erfahrungen weitergegeben werden, die teils von uns selbst, teils von der internationalen Benützergemeinde von ACSL stammen.

- Wenn Sie ein ACSL Handbuch besitzen, ersetzen Sie bitte auf Seite 4-2Ø das Feldelement A(18) durch A(13) , das Element (3,3) aus dem Feld A(5,5) ist richtig A(5*2+3) und damit A(13).

- Fehlende Klammern beim Übersetzen eines ACSL-Programmes werden normalerweise entdeckt (vorausgesetzt, es handelt sich nicht um Klammern in einer Hollerith-Zeichenkette).

  Suchen Sie Klammern, die laut Fehlernachricht fehlen, im Quellprogramm jedoch vorhanden zu sein scheinen, zu allererst ab Spalte 73 - ACSL liest nur bis Spalte 72.

- Wenn Sie irgendeine Line-Plot-Device benützen (sei es ein Plotter oder ein Tektronix-kompatibles Terminal via PLOT1Ø-Software), so denken Sie bitte daran, daß nur 3 Ziffern zur Achsenbeschriftung verwendet werden.

  Wenn mit automatisch skalierten Achsen gezeichnet wird, werden die Start-punkte und Unterteilungen so gewählt, daß bei der Beschriftung keine Information verloren geht. Die Sache sieht anders aus, wenn Fenster mit "(X)LO" und "(X)HI" erzwungen werden, deren Grenzen als Zahlenwerte gegeben sind, die sich nicht durch 3 Ziffern darstellen lassen.

  Wird z.B. eine Achse von 13.75 bis 14.25 gezeichnet, lautet die Beschriftung 13.7 bis 14.2 (durch Abschneiden), und die Unterteilungen 13.85, 13.95, 14.Ø5, etc. werden mit 13.8, 13.9 und 14.Ø beschriftet - es sieht also so aus, als wäre der ganze Darstellungsbereich um Ø.Ø5 verschoben, obwohl die Zeichnung korrekt und nur die Beschriftung falsch ist.

  Da drei Ziffern Genauigkeit in fast allen Fällen ausreichen, ist mit einer Behebung dieses Fehlers vorerst nicht zu rechnen. Benutzen Sie bitte Zahlen-werte, die sich mit drei Ziffern darstellen lassen, wenn Sie Daten in Fenster-bereichen zeichnen.

- Vielleicht haben Sie schon bemerkt, daß im ACSL-Handbuch die Beispiel-Modelle bei Kommentaren, Keywords und Titeln den Doppelapostroph benutzen - an anderen Stellen im Text jedoch wird der einfache Apostroph verwendet. Auf CDC-Maschinen muß in der Modelldefinition der Doppelapostroph verwendet werden (aufgrund der CDC-Software), der Laufzeitinterpreter akzeptiert beide Arten von Apostrophen - er wurde so konzipiert und benötigt keine Fremdsoftware zur Syntaxanalyse.

Mit Level 8 (und einem neuen Input-Mapper) sollen diese Inkompatibilitäten behoben werden. Einfach- und Doppelapostroph werden dann gleichwertig sein (u.a. wird dann auch Groß- und Kleinschreibung zugelassen werden).

- Wenn Sie Macros verwenden, versuchen Sie bitte nicht, ersetzbare Symbole darin mit DO beginnen zu lassen, wenn diese auf der linken Seite eines "=" Zeichens stehen.

  Beispiel:
  ```
  MACRO TT (OUT,DOUT,A,B)
  OUT=A
  DOUT=B
  MACRO END
  ```

  Aufruf mit der Parameterliste (F,C=X,Y) liefert erstaunlicherweise nach dem Vorübersetzen

  ```
  F=X            und nicht      F=X
  DOUT=Y                        C=Y
  ```

  Es sieht so aus, als wäre DOUT ein reservierter Name. Tatsächlich orientiert sich der Syntax-Analysator an der Zeichenkette "DO" und dem nachfolgenden Beistrich, um DO-Schleifen zu erkennen und Ersetzungen des "DO" zu verhindern.

  Man könnte einwenden, daß "DO" einfach von einer mit "DO" beginnenden Zeichenkette für einen Namen unterschieden werden kann - doch überlegen Sie bitte, daß in FORTRAN Leerzeichen keine Rolle spielen und somit auch

  ```
  DO100I=1,100
  ```

  zugelassen ist.

  Beachten Sie also bis zu einer tiefergreifenden Änderung der Syntax-Analyse die oben genannte Einschränkung.

- Eine Leerzeile nach der Modelldefinition wird von ACSL als Anzeichen für nachfolgende Unterprogramme gewertet. Ist diese Leerzeile aber die tatsächlich letzte Zeile des Modells, erhalten Sie die Fehlermeldung

  ```
  OUTSIDE TABLE LIMIT
  ```

  in diesem Falle zu Unrecht. Leerzeile löschen, und der Fehler ist behoben.

  Dieses Problem soll mit Level 8 ebenfalls behoben werden.

- Durch die Verwendung der FTN5-Version von ACSL finden die Benutzer mit CM100000 nicht mehr das Auslangen - dies trifft vor allem die interaktiven Benutzer, und hier speziell jene Gruppe, die auf diesen Wert limitiert ist und nicht mit RFL mehr Speicherplatz verlangen darf. Das sind vor allem jene Studenten, die mit Praktikumsnummern rechnen.

  Bei anderen Kunden kann es zu Platzproblemen kommen, wenn z.B. große Tabellen zur Funktionserzeugung verwendet werden.

  Dieses Problem wurde - wie vom ACSL-Hersteller vorgeschlagen - angegriffen, indem zu segmentieren versucht wurde.

  Trotz fleißiger Bemühungen und Mithilfe insbesondere der Systemgruppe des IEZ an der Universität konnte bis jetzt nur die Fehlerquelle gefunden werden, die eine solche Lösung verhindert - es ist die derzeitige Version des CYBER-Loaders.

  Mit der neuen Version 573 (deren Installation, wie uns gesagt wurde, nicht mehr allzu lange auf sich warten lassen wird) wird eine Segmentierung möglich sein.

- Wenn Sie Mißfallen an der im Dayfile erscheinenden Fehlernachricht bezüglich des PL*-Parameters finden sollten, haben Sie bitte noch etwas Geduld. Der Standardwert für das Laufzeit-Zeilenlimit sollte derzeit für alle in Frage stehenden Probleme ohnedies ausreichen.

- Man hat für Level 8 eine Anwendbarkeit der RANGE-, PLOT-, PRINT-Commands auf INTEGER- und LOGICAL-Variablen versprochen.

- Die Variable NRWITG (No-Rewind-Integer) bewirkt derzeit nicht das, was man gerne mit ihrer Hilfe erreichen möchte - das Darstellen von in mehreren Runs gewonnenen Daten auf einer Zeichnung (betrifft auch PRINT, RANGE). Leider scheinen nur die Daten des ersten Runs auf dem RRR (Scratch) File vorhanden zu sein.

  Der Anschein trügt - die Daten sind vorhanden, allerdings wird bei jedem START (oder CONTIN oder PLOT etc.) CDC-systemintern ein EOF hinter die gerade geschriebenen Daten gesetzt. Damit sind sie (außer nach PACK)"unerreichbar".

  Level 8 wird eine Korrektur bringen, allerdings sollte bis zum Erscheinen dieses Artikels auch die von uns durchgeführte Modifikation schon zur Verfügung stehen.

- In der Steuerung des Variable-Schrittweiten-Algorithmus (IALG=1,2) gibt es einen Fehler, der sich zwar nicht in der Genauigkeit, wohl aber in der Rechenzeit auswirken kann.

  Wir werden die von der Firma empfohlene Korrektur durchführen.

- Wenn Sie mit PLOT 'ALL' und graphischem Terminal arbeiten, werden Sie feststellen, daß nach Fertigstellung von Zeichnungen diese wieder gelöscht und dann die nächsten gezeichnet werden - letztlich bleibt nichts am Bildschirm, was man betrachten könnte, da man den Zeichenvorgang nicht anhalten kann.

  PLOT 'ALL' ist eben für off-line Plotter (oder Printer) gedacht. Man könnte auch die Graphik-Treiber-Routinen ändern.

- Ein ACSL Benützer berichtet von Schwierigkeiten, wenn er versucht, eine Zustandsgröße vom Wert ∿3.ØE-293(!) zu zeichnen.

  Die FORTRAN-Exponentiationsroutine versagt hier beim Skalieren. Eine einstweilige Lösung: nehmen Sie eine sehr kleine Konstante (z.B. 1.ØE-3Ø) zur Zustandsgröße dazu und nehmen Sie diese Summengröße mit anderem Namen in die PREPAR-Liste auf. Die Herstellerfirma wartet ab, ob noch jemand Gefallen an diesen Wertebereichen findet - man würde den Fehler dann eventuell beheben. Vielleicht haben Sie Lust, ein (sinnvolles) Modell zu entwerfen, das derartige "Größen" benötigt - ich kann aber nicht versprechen, daß die Herstellerfirma daran Freude haben wird.

  Vielleicht haben Sie aber auch ein anderes Modell oder sonst Interesse an ACSL, dann nehmen Sie bitte Kontakt mit uns auf, damit wir Ihnen diese Simulationssprache zeigen können.

simulationstechnik und hybsys

Unter dieser Rubrik werden anhand von Beispielen hybride Programmiertechniken bei Verwendung der Simulationssprache HYBSYS erläutert.

BEISPIEL: LÖSUNG EINES WÄRMELEITUNGSPROBLEMS MITHILFE DER ZEITDISKRETISIERUNG (CONTINUOUS SPACE DISCRETE TIME)

W. Kleinert

Zu lösen sei die folgende partielle Differentialgleichung samt den zugehörigen Anfangs- und Randbedingungen:

$$\frac{\partial^2 u}{\partial x^2} = \begin{cases} A * \frac{\partial u}{\partial t} & 0 \leq x \leq \frac{\pi}{2} \qquad A = \frac{C.\rho}{K_1} \\ B * \frac{\partial u}{\partial t} & \frac{\pi}{2} < x \leq \pi \qquad B = \frac{C.\rho}{K_2} \end{cases} \tag{1}$$

$u = u(x,t) \quad t \in [0,T]$

$u(x,0) = \sin x$

$u(0,t) = u(\pi,t) = 0$

K_1, K_2 ... Wärmeleitkoeffizienten

C ... Wärmekapazität

ρ ... Dichte

D.h. ein dünner Stab der Gesamtlänge π mit unterschiedlichen Wärmeleitkoeffizienten für jede der beiden Stabhälften hat zu Beginn einen sinusförmigen Temperaturverlauf. Gesucht sind die mit der Zeit veränderlichen Temperaturprofile.

Die Methode der Zeitdiskretisierung von parabolischen Differentialgleichungen besteht darin, in (1) die partiellen Ableitungen nach der Zeit durch den Differenzenquotienten zu ersetzen:

$$\left.\frac{\partial u}{\partial t}\right|_{t=t_i} = \frac{u(x,t_i) - u(x,t_{i-1})}{\Delta t} \qquad u_i(x) = u(x,t_i) \tag{2}$$

Das führt zu folgendem Zweipunkt - Randwertproblem mit unstetigen rechten Seiten.

$$u_i(0) = 0 \ , \ u_i(\pi) = 0 \iff u_i(\pi, u_i'(0)) = 0 \qquad (3)$$

$$u_i'' = \begin{cases} \frac{A}{\Delta t} & (u_i = u_{i-1}) \qquad 0 \leq x \leq \frac{\pi}{2} \\ \frac{B}{\Delta t} & (u_i = u_{i-1}) \qquad \frac{\pi}{2} < x \leq \pi \end{cases}$$

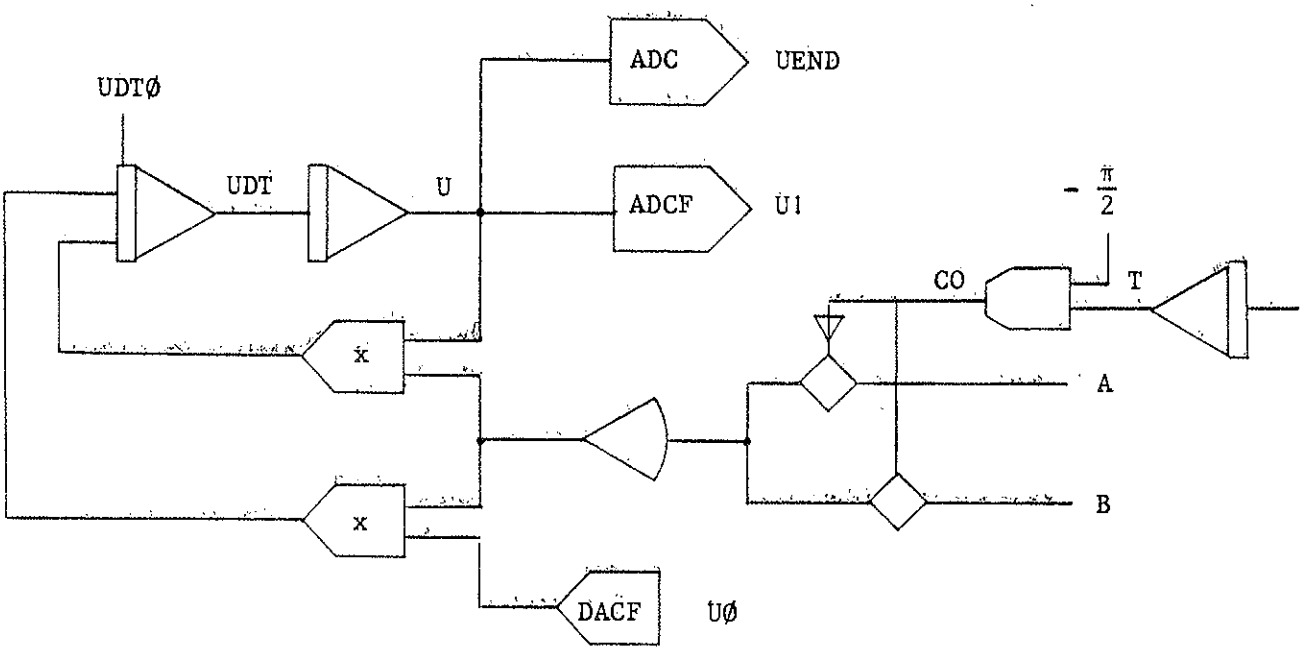
Die kontinuierliche Integrationszeit des Analogrechners wird in diesem Fall für die Integration nach der Ortsvariablen x verwendet.

Es muß nun der geeignete Anfangswert $u_i'(0)$ bestimmt werden, der zu der gewünschten Randbedingung $u_i(\pi)=0$ führt. Das kann z.B. mit dem in HYBSYS als Systembefehl zur Verfügung stehenden Nullstellenprogramm ZERO1 erreicht werden.

Zur Darstellung der Funktion u_{i-1} wird eine hybride DACF-Variable UØ verwendet, mit der das Randwertproblem gelöst wird. Die Lösung wird auf die hybride ADCF-Variable U1 gelegt und nach Lösung des Randwertproblems für die nächste Zeititeration umgespeichert.

Der Randwert $u_i(\pi)$ wird durch den ADC UEND realisiert.

BLOCKDIAGRAMM

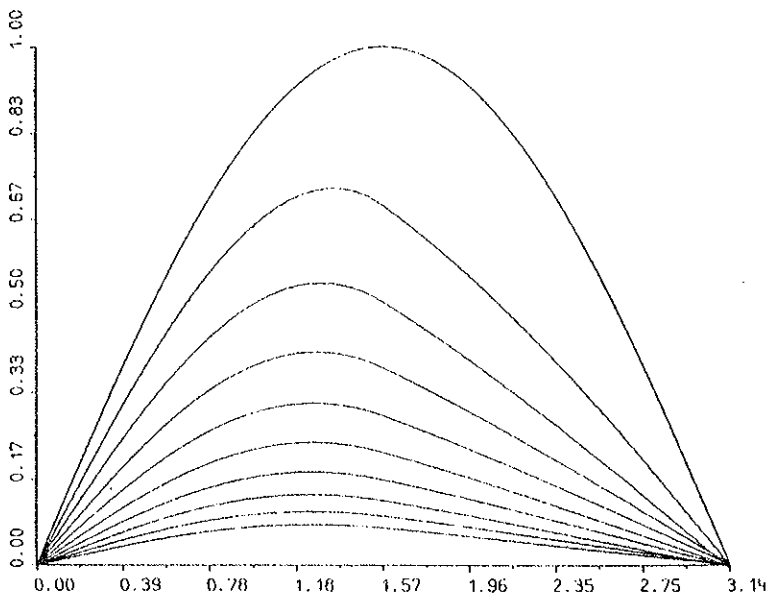


MODELLDEKLARATION IN HYBSYS

```
MODEL
  PAR:A,B,XS=1.57,ARG,UDTØ
  INT:U,UDT
  MULT:M1,M2
  DSWI:D
  COMP:CO
  DACF:UØ
  ADCF:U1
  ADC:UEND
  PAR SIN:SI
  SI=ARG
  U=UDT
  UDT,IC=UDTØ,M1,-M2
  CO=T,-XS
  D=_CO*A,CO*B
  M1=D,U
  M2=D,UØ
  U1=U
  UEND=U
  END
```

MODELLUNTERSUCHUNG UND ERGEBNISSE

`UØ=SI(ARG);`	Initialisierung von u(x,Ø)
`ZERO1 UEND BY UDTØ;`	Lösung des Randwertproblems (beinhaltet automatische Skalierung)
`PLOT UØ,U1;`	Zeichnen von u(x,tØ) und u(x,t1)
`U>Ø.1?UØ=U1;ZERO1;PLOT,S U1!`	Wiederholung bis u<.1



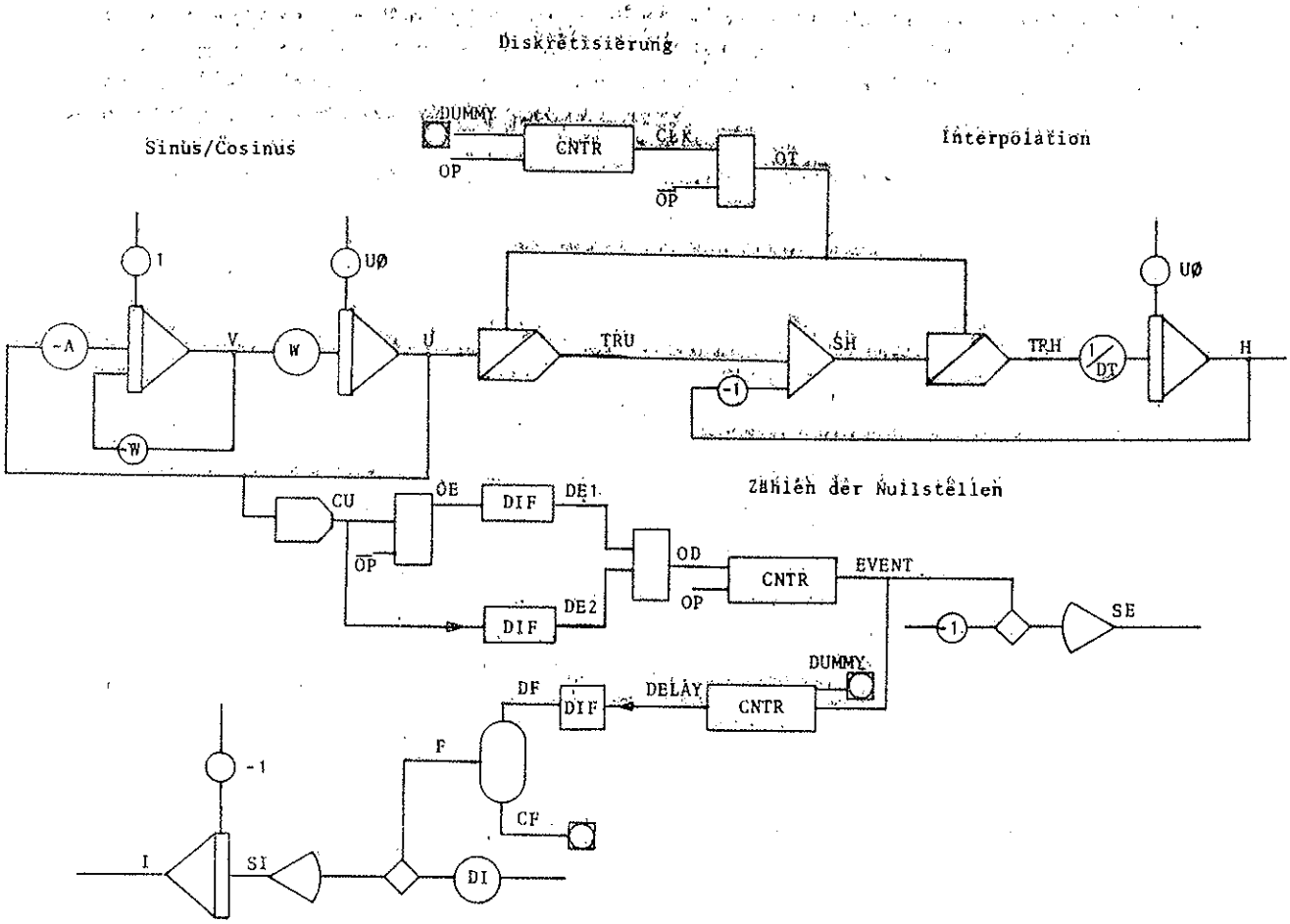
BEISPIEL: VERWENDUNG DER COUNTER IN HYBSYS

E. Wittek

Dem Benützer stehen in HYBSYS 5 Counter zur Verfügung, die mithilfe eines speziellen Commands auf 5 verschiedene Modes gesetzt werden können: Monostabile, Frequenzgeber, Impulsgeber, Ereigniszähler und Real-Time Clock.

Gegeben sei ein gedämpfter Sinus (U), der mithilfe einer Track/Store-Einheit (TRU) und eines als Frequenzgeber verwendeten Counters (CLK) abgetastet wird. Mit der gleichen Frequenz wird ein Track/Store für die Interpolation (H) gesteuert. Der Counter EVENT zählt die Nulldurchgänge des Sinus. Mit diesem Counter und einer nachgeschalteten Monostabilen (DELAY) wird der Anfangszeitpunkt der Integration der Geraden I festgelegt.

BLOCKDIAGRAMM

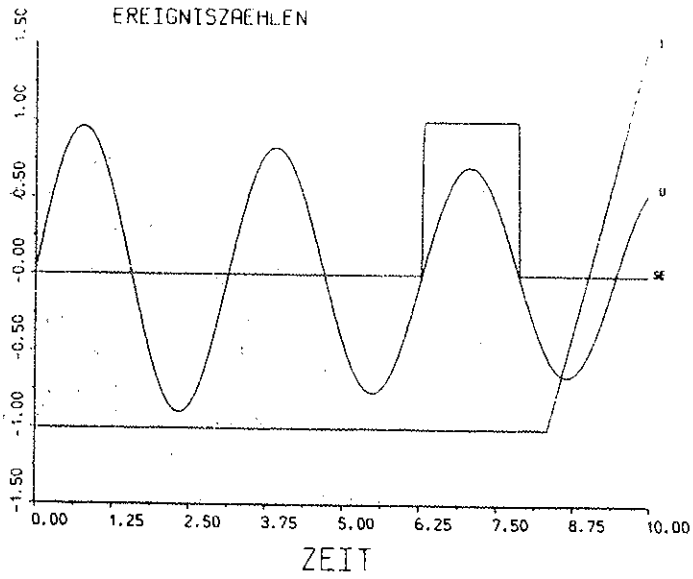


MODELLDEKLARATION

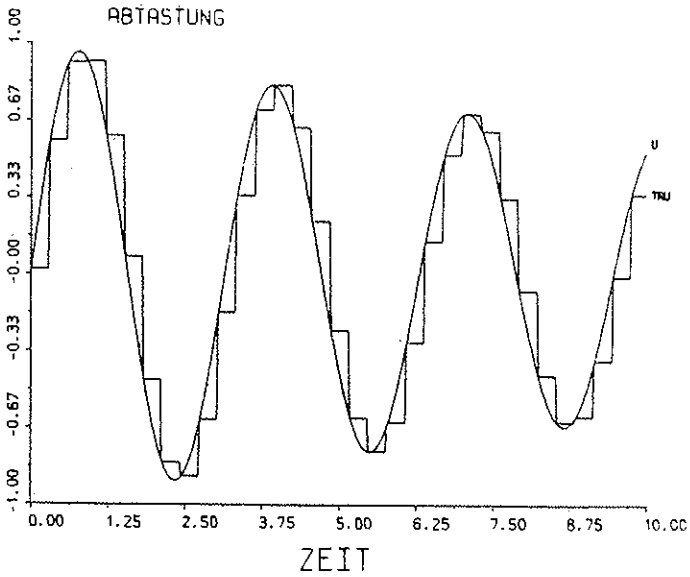
```
MODEL
CNTR:CLK,EVENT,DELAY
CL:DUMMY
"
"GEDAEMPFTER SINUS/COSINUS
"
INT:U,V
PAR:A=.1,W=2,U0=.02
U,IC=U0,W*V
V,IC=1,-A*V,-W*U
"
"ABTASTUNG
"
CLK=DUMMY,OP
OR:OT
OT=CLK,_OP
TRST:TRU
TRU=OT*U
"
"INTERPOLATION
"
PAR:DT=.3
PAR DIV:DH
DH=1,DT
SUM:SH
INT:H
TRST:TRH
SH=TRU,-H
TRH=OT*SH
H,IC=U0,DH*TRH
"
"EREIGNISZAEHLER
"UM EIN TRIGGERN DES COMPARATORS IM IC ZU VERHINDERN
"HAT SINUS KLEINEN ANFANGSWERT
"
COMP:CU
CU=U
OR:OE,OD
OE=_OP,CU
DIF:DE1,DE2
DE1=OE
DE2=_CU
OD=DE1,DE2
EVENT=OD,OP
SWI:SE
SE=EVENT*1
"
"INTEGRATION IN ABHAENGIGKEIT DES EREIGNISSES
"
DELAY=DUMMY,EVENT
DIF:DF
DF=_DELAY
CL:CF
FF:F
F=DF,CF
PAR:DI=1.5,TM=2
SWI:SI
INT:I
SI=F*DI
I,IC=-1,SI
END
"
POINTS=501
CNT CLK=PULSE(DT:.01),EVENT=RTC(4),DELAY=MONO(TM)
SCALE
```

ERGEBNISSE

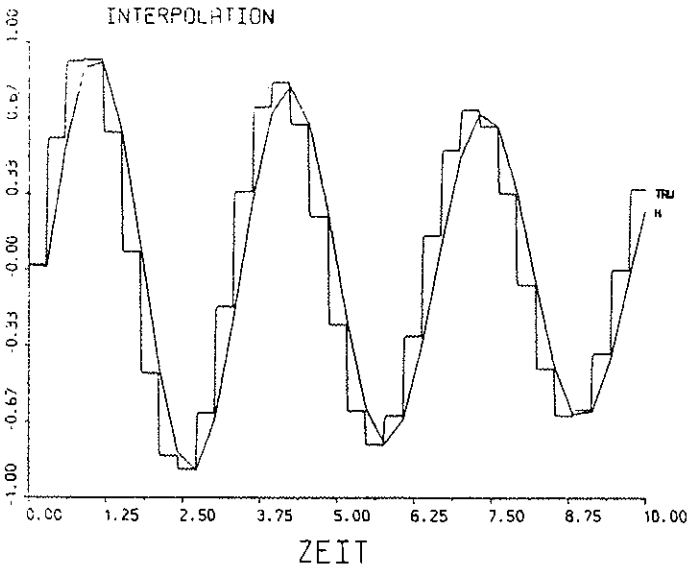
```
XTEXT=ZEIT
TEXT=EREIGNISZAEHLEN
PLOT U,SE,I
```




```
TEXT=ABTASTUNG
PLOT U,TRU
```




```
TEXT=INTERPOLATION
PLOT H,TRU
```



SIMULATION VON PATHOLOGISCHEM BLUTDRUCK-VERHALTEN BEI VERENGUNG DER AORTA*)

F. Breitenecker
Institut für Analysis, Technische Mathematik
und Versicherungsmathematik
Abt. Regelungsmathematik und Simulations- und
Hybridrechentechnik
Technische Universität Wien

J. Kaliman
Kardiologische Universitätsklinik
Universität Wien

EINLEITUNG

Die betrachtete Verengung der Aorta ist eine angeborene Mißbildung in der Aorta nach dem Abzweigen der linken Armarterie. Diese Verengung resultiert in einer Blutdruckdifferenz (Gradient) zwischen oberen und unteren Extremitäten, die bei Belastung stark ansteigt. Nach genormter Belastung auf dem Laufbandergometer geht normalerweise der systolische Druck in der arteria bracchialis (Handarterie) und in der arteria tibialis posterior (Knöchelarterie) innerhalb von 30 Sekunden auf den Systemdruck zurück, wobei beide Druckkurven quantitativ gleich sind (Gradient=0). Bei der betrachteten Arterienverengung zeigt der Druck in der arteria tibialis posterior ein pathologisches Verhalten: er schwingt und erreicht den Systemdruck erst nach fünf oder mehr Minuten; da der Druck in der arteria bracchialis nicht schwingt, schwingt der Gradient ebenfalls, bis er nach fünf oder mehr Minuten den "Ruhegradienten" erreicht (Abb. 1).

Die Simulation dieses pathologischen Blutdruckverhaltens nach einer Belastung basiert auf Messungen bei 66 Patienten (durchschnittlich 15 Jahre alt), deren Aorta verengt ist. Die Blutdruckmessungen wurden einmal vor der Belastung und jede Minute nach der Belastung mit der Doppler-Ultraschall-Methode durchgeführt.

MODELL

Der (gemessene) Blutdruckverlauf $\bar{m}(t)$ in der arteria bracchialis nähert sich dem Ruhedruck $\bar{m}_R$ in Form einer Exponentialfunktion (Abb.1), woraus folgendes Modell gewonnen werden kann:

$$T_m \dot{\bar{m}} = -\bar{m} + k_m u_m + k_m s_m \qquad (1)$$

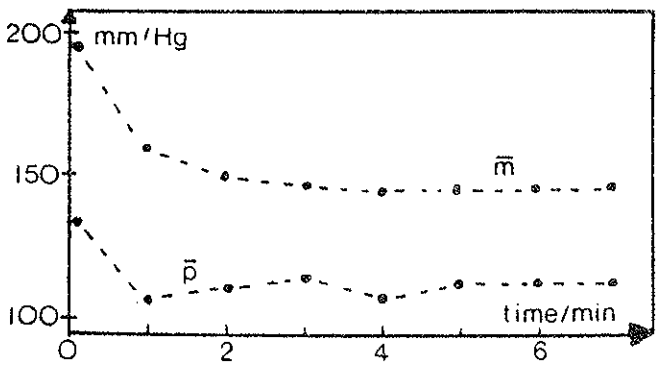


Abbildung 1
Gemessener Blutdruck $\bar{m}(t)$, $\bar{p}(t)$

Die Simulation startet am Ende der Belastung zum Zeitpunkt $t_0=0$ mit $\bar{m}(t_0)=\bar{m}_0=\bar{m}_0$, sodaß sich für den Verstärkungsfaktor k_m, für den stationären Zustand s_m und für die Belastung $u(t)$ die Werte

$$k_m=\bar{m}_0,\quad s_m=\bar{m}_R/\bar{m}_0,\quad u_m=\delta(t_0)$$

ergeben. Der physiologisch entscheidende Parameter ist die Zeitkonstante T_m.

Der (gemessene) Blutdruckverlauf $\bar{p}(t)$ in der arteria tibialis posterior schwingt um eine (fiktive) Exponentialkurve, bis er den Ruhedruck $\bar{p}_R$ erreicht. Als Ansatz dient daher ein Modell dritter Ordnung:

$$\begin{aligned} T_p \dot{p}_1 &= -p_1 + k_p u_p + k_p s_p \\ \dot{v} &= -d\dot{v} - fv \qquad (2) \\ p &= p_1 + A\,v \end{aligned}$$

Die Konstanten und Anfangswerte ergeben sich aus $p_1(t_0)=\bar{p}_0=k_p$, $u_p=\delta(t_0)$, $s_p=\bar{p}_R/\bar{p}_0$, $d=2.D$, $f=D^2+F^2$, $v(0)=D$, $\dot{v}(0)=1$; die physiologisch entscheidenden Parameter sind die Zeitkonstante T_p des fiktiven nichtpathologischen Verhaltens und Amplitude A, Frequenz F und Dämpfung D des überlagerten pathologischen Schwingungsverhaltens.

*) Deutsche Kurzfassung von F. Breitenecker, J. Kaliman: Simulation of Pathological Blood Pressure Behaviour after Treadmill Test in Patients with Coarctation of the Aorta. First European Simulation Congress ESC 83. Informatik Fachberichte 71, S480-486, Springer Verlag 1983.

```
MODEL
 PAR: MØ,MR,PØ,PR,TM,TF,A,D,F
 INT: M,P1,V,DV
 SUM: P
 DACF: MM,MP
 M,IC=MØ,-TM*M,MR
 P1,IC=PØ,-TP*P1,PR
 V=DV
 DV,IC=1,-D*DV,-F*V
 P=P1,A*V
END
```

Tabelle 1
Modelldeklaration in HYBSYS

SIMULATION

Simulation und Identifikation des Modelles wurden am Hybridrechenzentrum der TU Wien in HYBSYS durchgeführt. Die Simulation (Modelldeklaration in Tabelle 1) wird durch drei Overlays unterstützt. Der Overlay SETPFH,n lädt die DACFs mit den Meßkurven des Patienten n; STOPAR,n speichert die identifizierten Parameter ab; GETPAR,n liest die identifizierten Parameter zur weiteren Verwendung ein.

Die Identifikation der Parameter T_m, T_p, A, F, D wurde mit dem HYBSYS-Standardmakro ZERO in drei Stufen ausgeführt, die der physiologischen Seite des Problems angepaßt sind:

1) Identifikation von T_m durch Minimierung von E_m^A (für $T_m \leq 1{,}5$) oder E_m^M ($T_m > 1{,}3$)
2) Identifikation von T_p durch Minimierung von E_p^M
3) Identifikation von A, F, D durch Minimierung von E_p^A und E_p^M

Die verwendeten Gütefunktionale sind

$$E_m^A = \int_0^T |m-\bar{m}|\,dt, \quad E_m^M = \left|\int_0^T (m-\bar{m})\,dt\right|$$

$$E_p^A = \int_0^T |p-\bar{p}|\,dt, \quad E_p^M = \left|\int_0^T (p-\bar{p})\,dt\right|$$

Abbildung 2 faßt die Simulation und Identifikation des Modelles in HYBSYS zusammen.

ERGEBNISSE

Das lineare Modell (1), (2) zeigt eine sehr gute Anpassung an die Meßkurven. Abbildung 3 zeigt gemessenen und modellierten Blutdruckverlauf für einen Patienten. Tabelle 2 zeigt Maximalwert, Minimalwert, Mittelwert und Streuung der wesentlichen Modellparameter T_m, T_p, A, F, D, wobei die identifizierten Parameter von 64 Patienten zugrundegelegt wurden.

	T_m	T_p	A	D	F
MINIMAL VALUE	0.6	0.6	10.	0.2	0.350
MAXIMAL VALUE	2.8	2.8	35.	0.6	2.6
MEAN VALUE	1.688	1.730	19.60	0.389	1.139
STANDARD DEVIATION	0.062	0.059	0.770	0.014	0.075

Tabelle 2
Parameterwerte

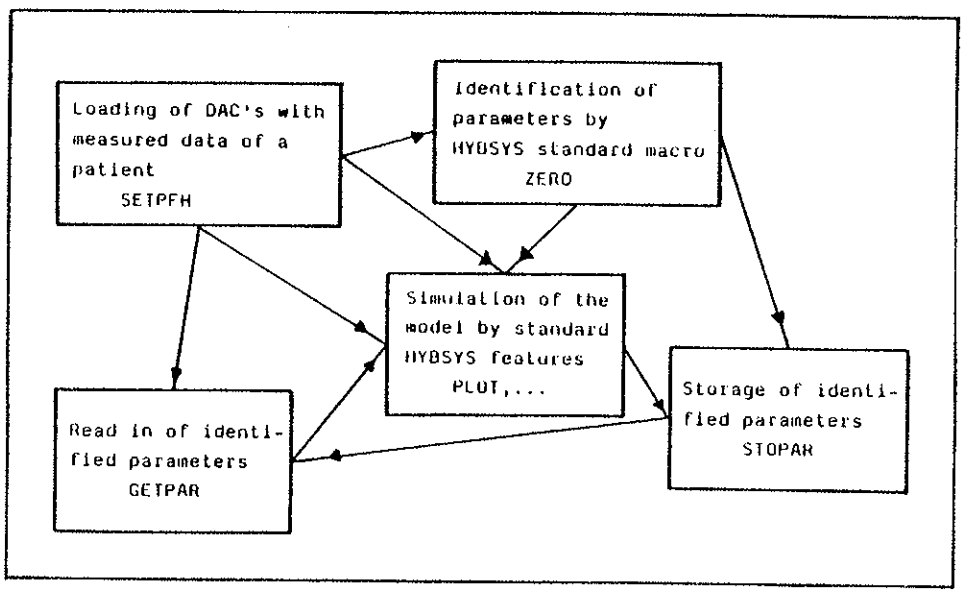


Abbildung 2
Simulation und Identifikation
des Modells in HYBSYS

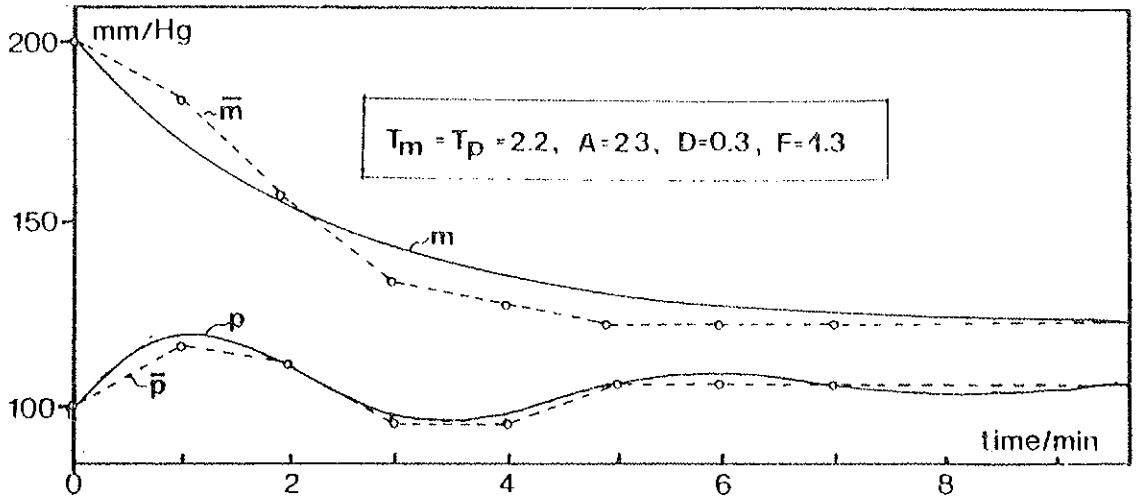


Abbildung 3
Gemessener und simulierter Blutdruck
Patient Nr. 1

Als interessantes Ergebnis können die Parameter noch weiter klassifiziert werden:

- T_m und T_p hängen nicht von A, F, D ab (und umgekehrt)
- T_m und T_p sind nahezu identisch
- Frequenz F und Amplitude A gehorchen annähernd der Beziehung (Abb. 4)

$$F = 3{,}75 \cdot e^{-0{,}07 \cdot A} \qquad (3)$$

- Dämpfung D und Amplitude A gehorchen annähernd der Beziehung

$$D = 0{,}75 \cdot e^{-0{,}04 \cdot A} \qquad (4)$$

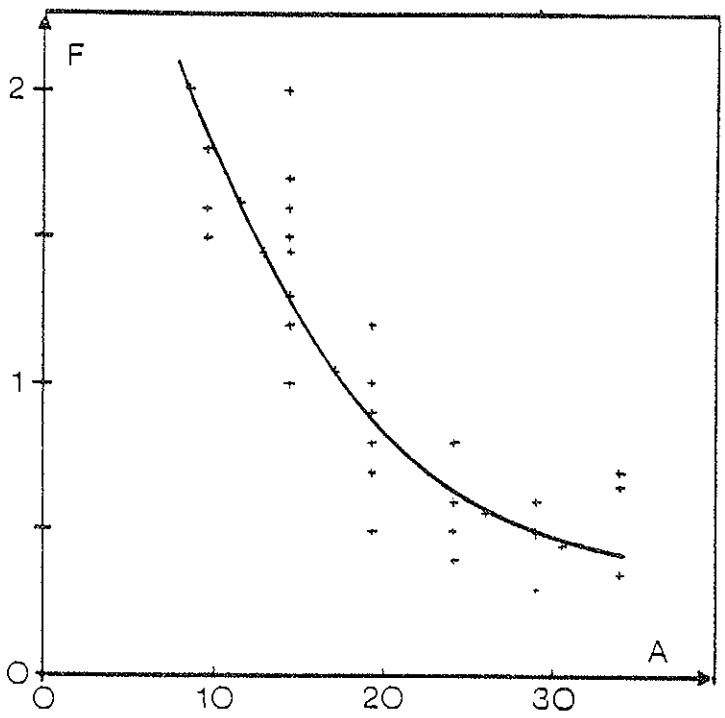


Abbildung 4
Beziehung zwischen F und A

Davon ausgehend können folgende Schlüsse gezogen werden:

- Die Parameter T_m und T_p charakterisieren die "Fitness" des Patienten und sind (im wesentlichen) unabhängig von der Arterienverengung.
- Das pathologische Blutdruckverhalten in der arteria tibialis posterior wird modellmäßig nur durch die Schwingungsparameter A, F, D bestimmt. Für eine grobe Näherung reicht die Kenntnis der Amplitude A, von der sich dann Frequenz F und Dämpfung D nach (3), (4) berechnen lassen.

BEMERKUNGEN

1) Ein nichtlineares Modell (Frequenz und Dämpfung zeitabhängig vom Unterschied zum Ruhedruck) brachte keine Verbesserung.
2) Derzeit wird weiter untersucht, ob die Schwingungsparameter A, F, D mit dem Gradienten in Ruhe oder mit dem Gradienten nach Absetzen der Belastung (t_0) zusammenhängen.

SUMMARY

A model is developed and simulated, which describes the pathological blood pressure behaviour after treadmill test in patients with coarctation of the aorta. The model is verified by a set of measured data. Five parameters characterize the model and consequently the pathological blood pressure behaviour. Interesting correlations between the parameters are shown.

SIMULATION DER REIZENTWICKLUNG IM GEHÖRNERV DURCH ELEKTRISCHE ANREGUNG

F. Rattay
Institut für Analysis, Technische Mathematik
und Versicherungsmathematik
Technische Universität Wien

Am Institut für allgemeine Elektrotechnik und Elektronik der Technischen Universität Wien wurde eine Gehörprothese entwickelt, die es Menschen, die seit Jahren völlig taub sind, ermöglicht, wieder zu hören [1]. Die Ursache der Gehörlosigkeit dieser Patienten ist ein organischer Defekt, der es nicht erlaubt, Schallwellen so zu übertragen, daß es zu einer Nervenreizung im Innenohr kommt. Eine relativ einfache Operation gestattet das Einsetzen einer speziellen Elektrode, die durch das runde Fenster eingeschoben wird und sich den Windungen der Schnecke anpaßt. Ist der Gehörnerv selbst noch nicht abgestorben, so können dann seine Fasern angeregt werden, entsprechend den sich zeitlich ändernden Potentialen der Elektrode. Diese Elektrostimulation wirkt nur lokal auf den nervus acusticus, also ohne Nebenwirkungen an anderen Nervenbahnen hervorzurufen.

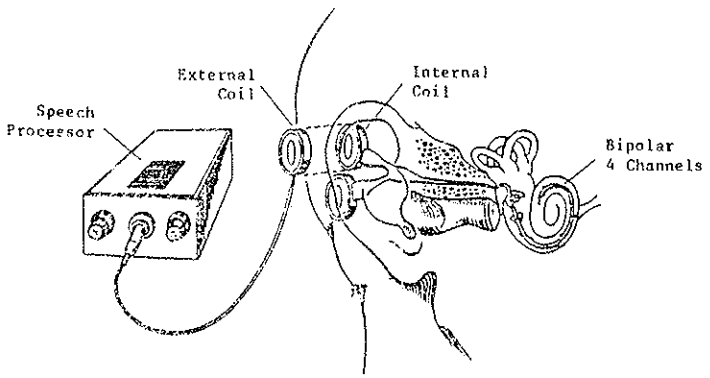


Abbildung 1

Über einen Sprachprozessor wird das ursprüngliche akustische Signal bezüglich der Amplituden komprimiert und drahtlos an eine eingepflanzte Spule übermittelt, die dann die in der Schnecke liegende Elektrode mit den entsprechenden Potentialen versorgt. Die Elektrode liegt eng an den Nervenbahnen, die zu den Haarzellen führen.

Obwohl diese Pionierleistung zu großartigen Ergebnissen - wie das Erkennen von ganzen Sätzen, ohne von den Lippen abzulesen - geführt hat, kommt es doch relativ oft zu Verwechslungen von Lauten. Um das Gesamtverhalten noch weiter verbessern zu können, wurde eine Simulation der Elektrostimulation am Hybridrechner durchgeführt.

Leider sind aus der Literatur keine einschlägigen Arbeiten bekannt, die als Modell des Gehörnervs herangezogen werden konnten, doch gibt es etliche allgemeine Modelle für die Nervenreizung, wobei das klassische Modell von Hodgkin und Huxley (HH-Modell: 4 DG 1. Ordnung) besonders hervorzuheben ist. Fitzhugh beschreibt ein daraus abgeleitetes zweidimensionales Modell (BVP-Bonhoeffer-Van-der-Pol-Modell), das ebenfalls in HYBSYS durchgerechnet wurde und zumindest qualitativ zu richtigen Ergebnissen geführt hat ([2], [3]), während die quantitativ besseren Resultate aus dem HH-Modell erzielt wurden.

Die HH-Gleichungen (1-4) dienen zur Berechnung des Potentials V an der Membran (dünne Haut), die eine einzelne Nervenfaser umgibt. Die Membran kann von Natrium-Kalium- und anderen Ionen durchflossen werden, wobei die Wahrscheinlichkeiten der jeweiligen Ionenstromstärke wieder potentialabhängig sind und durch drei DG - basierend auf den berühmten Messungen von Hodgkin und Huxley - beschrieben werden [4]. Durch genügend starke Anregung erreicht man ein plötzliches starkes Ansteigen von V, das nach kurzer Zeit (Refraktärperiode) wieder abfällt. Dieses "Aktionspotential" eilt dann die Nervenfaser entlang und vermittelt den Nervenreiz an das Gehirn.

Die HH-Gleichungen wurden für den elektrostimulierten Akustiknerv durch die Terme S und ma_j·noise erweitert:

$$I = C.\dot{V} + I_1 + S$$

$$I_1 = g_{Na} m_1^3 m_2 (V - V_{Na}) + g_K . m_3^4 (V - V_K) + g_L (V - V_L) \qquad (1)$$

$$\dot{m}_j = \alpha_j(V)(1 - m_j) - \beta_j(V) . m_j + ma_j . noise$$

$$j = 1,2,3 \qquad (2\text{-}4)$$

dabei bedeutet:

I:	Gesamtstrom durch die Membran
C:	Kapazität der Membran
V:	Potential an der Membran
I_i:	Ionenstrom (Natrium + Kalium + Restionen)
S:	von der Elektrode kommendes, stimulierendes Signal, das die akustische Information trägt
g_{Na}, g_K, g_L:	Leitfähigkeit bezüglich der drei Ionenströme
V_{Na}, V_K, V_L:	zugehörige Nernst-Potentiale
m_j:	Wahrscheinlichkeiten für das Öffnen und Schließen der entsprechenden ionendurchlässigen Kanäle in der Membran
α_j, β_j:	Funktionen von V mit exponentiellen Anteilen (genauer z.B. in [4])
ma_j:	Koeffizienten für zusätzliches Rauschsignal
noise:	weißes Rauschen

Obwohl sowohl aus einer analytischen Betrachtung als auch von den Experimenten her die Maximalwerte aller Größen leicht abschätzbar sind, zeigt sich durch die vielen Nichtlinearitäten, insbesondere bei den Variablen (α_j, β_j) mit exponentiellem Wachstum, eine gewisse Empfindlichkeit des Modells, wodurch z.B. auch das übliche Skalieren durch HYBSYS nicht zielführend ist. (Dazu kommt auch noch die Verwendung des Rauschgenerators.) Die Genauigkeit konnte jedoch durch Verwendung von Funktionsgebern für die $\alpha_j(V)$ und $\beta_j(V)$ in einen brauchbaren Bereich gebracht werden.

Rose [5] hat unter anderem am Katzennormalohr an Nervenfasern die Aktionspotentiale von reinen Sinustönen gemessen und die Ergebnisse statistisch ausgewertet. Es zeigte sich dabei eine gewisse zeitliche Synchronität zwischen dem Auftreten der Maxima des akustischen Sinussignales und den Aktionspotentialen in der betrachteten Nervenfaser. Die im Experiment vorhandene Streuung konnte aber zunächst in den Simulations-Modellen nicht gefunden werden. Erst das Einführen eines Rauschanteiles (ma_j·noise) brachte dann Übereinstimmung mit den Messungen und es zeigte sich paradoxer Weise, daß der Informationsgehalt des Nervensignals durch Hinzunahme des Rauschens vergrößert wird: Betrachtet man dann etwa die Summe der Aktionspotentiale mehrerer Nervenfasern, die ja parallel die akustischen Signale verarbeiten, so erhält man ein Signal, das in etwa den oberen Halbschwingungen des Tons entspricht (Abb. 2), [6]. Das rechnerische Ergebnis wurde natürlich aus einem seriellen Vorgang ermittelt, indem immer wieder das Modell für eine Nervenfaser aufgerufen wird und danach ein Histogramm erstellt wird (Overlay HIST für Histogramme 2. Art).

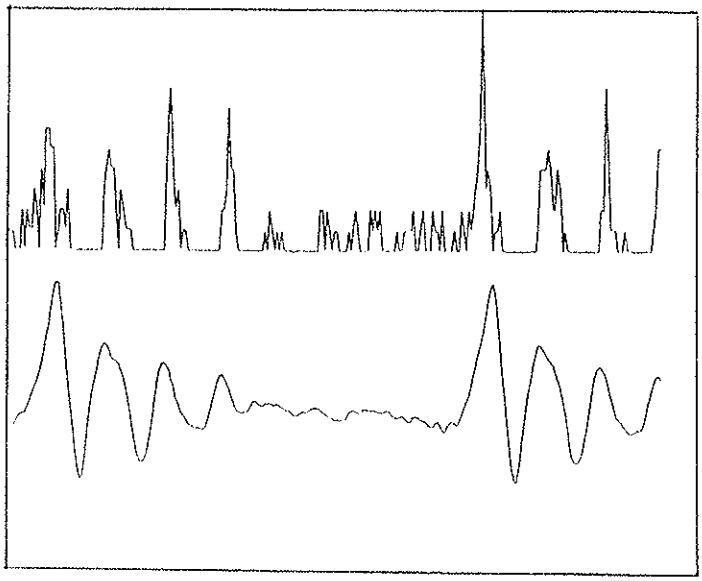


Abbildung 2

Das Bild zeigt im unteren Teil 1.5 Perioden eines (nicht komprimierten) a-Vokals. 40 Nervenfasern werden im Modell gleichzeitig durch ein äquivalentes elektrisches Signal angeregt, das von einer Elektrode ausgeht. Das Entstehen von Aktionspotentialen in jeder einzelnen Faser wird zeitlich vermerkt. Die obere Kurve gibt als Histogramm (2. Art) an, wieviele Nervenfasern momentan gerade Aktionspotentiale liefern. Das Histogramm bildet in etwa den positiven Teil des akustischen Signals nach, wobei aber in anderen Konfigurationen durchaus auch "Buckeln", die im negativen Bereich liegen, durchkommen - übereinstimmend mit experimentellen Resultaten.

Ein reiner Sinuston bewirkt nicht, daß bei jedem Maximum des Sinus ein Aktionspotential ausgelöst wird, sondern es gibt die Anzahl der "fehlenden" Aktionspotentiale Aufschluß über die Lautstärke des Tones. Dies wird in der Simulation durch Aufruf des Overlays HIS erwirkt und bringt in einem bestimmten Frequenzbereich ebenfalls Übereinstimmung mit dem Experiment (Abb. 3).

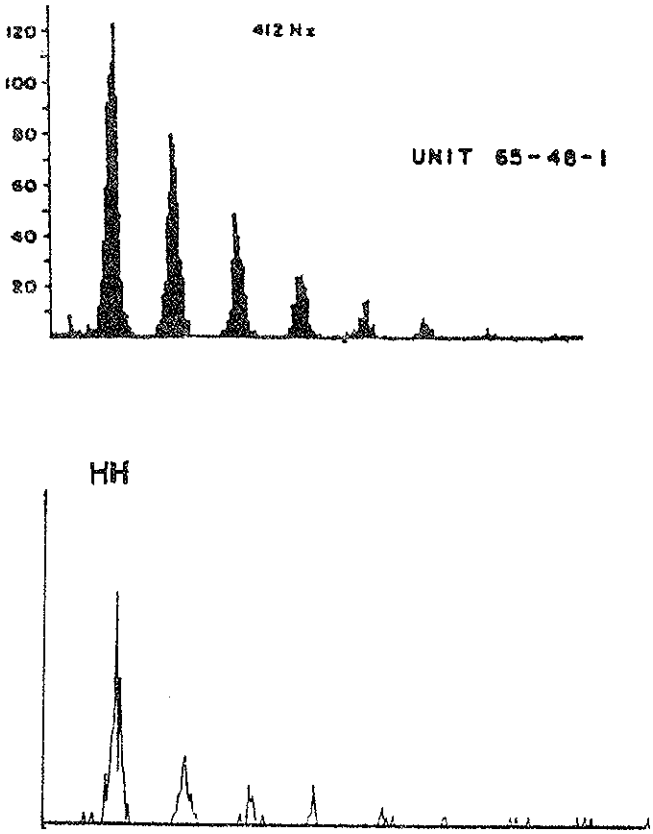


Abbildung 3

Wird eine Nervenfaser mit einem reinen Sinuston versorgt, so zeigt sich bei etwa 1900 angebotenen Perioden, der in den Histogrammen (1. Art) dargestellte zeitliche Abstand zwischen zwei Aktionspotentialen. Die Kurven stellen also die sich im Experiment (oben) und in der Simulation ergebenden Häufigkeiten der Zeitdifferenzen aufeinanderfolgender Aktionspotentiale dar. Der zeitliche Abstand von zwei Spitzen im Histogramm entspricht der Periodenlänge der Sinusschwingung. Auch in diesem Histogramm bilden sich wieder die positiven Teile des Sinus ab, doch verringert sich die Breite der einzelen Berge bei Verringern des Rauschsignales. Ein Vergrößern der Lautstärke bewirkt häufigeres Auftreten der Aktionspotentiale und dies führt zu einem Anwachsen der ersten Histogrammgipfel. Im Grenzfall liefert die Simulation - zumindest bei dieser niedrigen Frequenz - einen einzigen großen ersten Gipfel.

Durch Verwenden des Revox-Magnetophons ist es auch möglich, beliebige akustische Signale (Vokale, Silben,...) über Trunks abzunehmen, zu speichern (Overlay PHO) und die digitalisierten Werte (1001 Stützwerte pro Record) wieder zu verarbeiten. Der Overlay DPHO erlaubt auch die Erzeugung von komplexeren Tönen auf analytischem Weg. Eine Übersicht des gesamten Simulationspaketes bringt die Abbildung 4.

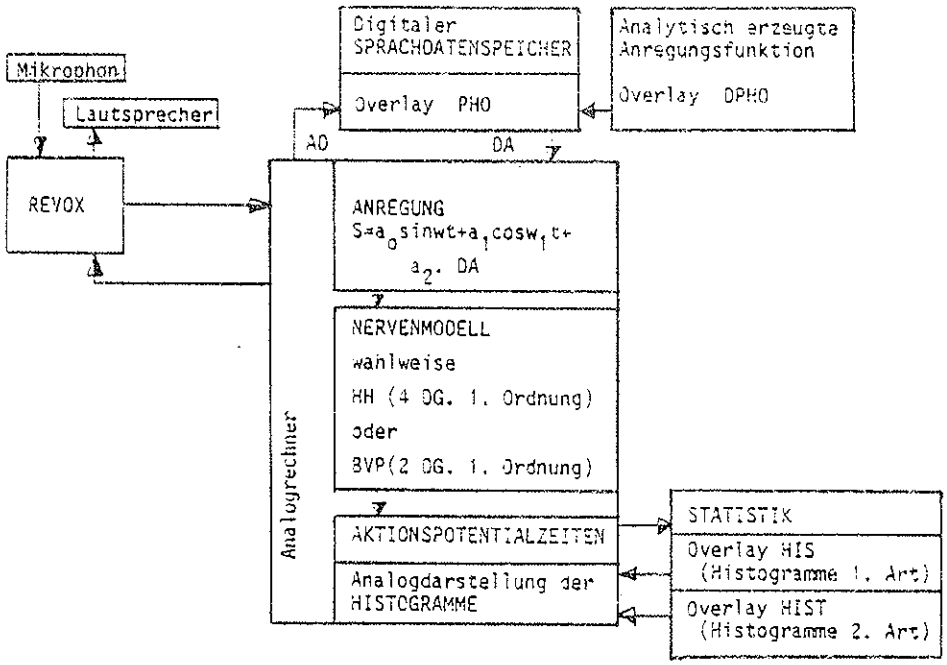


Abbildung 4

Wirkungsweise des Simulationspaketes der Elektrostimulation

Untersucht wurde vorwiegend der Hörbereich bis 2000 Hz, der auch in der Gehörprothese ausschlaggebend ist. Die Analogschaltung für das HH-Modell erfordert allerdings eine Zeittransformation mit dem Faktor 5, sodaß man die Daten leider nicht in Echtzeit verarbeiten kann. - Für die Erzeugung der Histogramme ist ein wiederholtes Abarbeiten desselben Akustiksignals erforderlich, sodaß dies für eine Echtzeitsimulation ohnedies nicht in Frage kommt. Trotzdem ist die Rechengeschwindigkeit für die

- Filterung bzw. Erzeugung des Akustiksignals,
- Lösung des Simulationsmodells für die Aktionspotentiale,
- Vorbereitung der Histogramme

im Analogrechner zufriedenstellend und erlaubt wesentlich größere Arbeitsgeschwindigkeiten als sie bei rein digitaler Simulation erwartet werden können*).

Leider ist das direkte Bearbeiten von Silben oder anderen zusammenhängenden Sprachsignalen nicht möglich, da bei einem Run nur 1001 Datenwerte abgespeichert werden können und damit nur Sprachsignale von etwa einer halben Sekunde erfaßt werden können.

Durch den Umfang und den Rahmen dieses Artikels bedingt, kann weder auf den Aufwand zur Modellfindung noch auf die Interpretation gewisser Resultate wie etwa des signifikanten Amplitudenbereichs oder der Frequenzabhängigkeit des Stimulationssignals eingegangen werden. Die umfangreichen Arbeiten konnten nur deswegen in relativ kurzer Zeit durchgeführt werden, weil hier am Hybridrechenzentrum ein weltweit einmaliger Komfort für hybride Simulationen zur Verfügung steht, der noch durch die "kundenfreundliche" Betreuung durch das Personal in so vorbildlicher Weise ergänzt wird, daß man auch beim Auftreten von Schwierigkeiten stets zuversichtlich sein Ziel vor Augen sah.

*) Das Programm für das HH-Modell umfaßt folgende Analogkomponenten: 11 INT, 10 MULT, 2 TRST, 16 SUM, 2 COMP, 1 TRI, 1 TRD, 3 DCFG, 2 ADCF, 1 DIV, 3 SQR, 1 SWI, 1 NOISE.

SUMMARY

Electrostimulation of the auditory nerve is simulated by general nerve models of fourth or second order. Two types of histograms show results in accordance with experiments.

LITERATUR

[1] Hochmair-Desoyer I. et al, 1981
Four Years of Experience with Cochlear Protheses.
Med. Progr. Technol. 8, 107-119

[2] Hochmair-Desoyer I., Hochmair E., Motz H., Rattay F.
A Modell for the Electrostimulation of the Nervus Acusticus.
In preparation

[3] Fitzhugh R., 1961
Impulses and Physiological States in Theoretical Models of Nerve Membrane
Biophys. J. 445-466

[4] Scott A., 1977
Neurophysics, Interscience
Wiley & Sons, N.Y.

[5] Rose J. et al, 1970
Frequency Analysis and Periodicity in Hearing.
A. W. Sijthoff, 176-187

[6] Motz H., Rattay F., 1983
Models for the Electrostimulation of the Nervus Acusticus
Proc. 1st Vienna International Workshop on Functional Electrostimulation

HYBRIDE UND DIGITALE SIMULATION VON REGELSTRECKEN MIT WAHLWEISE KONTINUIERLICHER ODER DISKRETER RÜCKKOPPLUNG *)

F. Breitenecker
Institut für Analysis, Technische Mathematik und Versicherungsmathematik
Abt. Regelungsmathematik und Simulations- und Hybridrechentechnik
Technische Universität Wien

EINLEITUNG

Die fortgeschrittene Mikrocomputertechnologie ermöglicht es, einen kontinuierlichen Regler einer Regelstrecke durch einen Mikrocomputer zu ersetzen. Das führt zu einer Diskretisierung der Rückkopplung, die das Regel- und Stabilitätsverhalten entscheidend ändern kann. Bei einem linearen Regelkreis kann der durch diese Diskretisierung entstehende Fehler explizit ausgedrückt werden; man sieht sofort, daß bei zu geringer Abtastfrequenz optimal ausgelegte Regelalgorithmen nicht mehr optimal arbeiten oder sogar zur Instabilität des Kreises führen. In diesem Beitrag wird nun untersucht, welche Möglichkeiten hybride und digitale Simulation zum Studium der auftretenden Phänomene bieten, wobei auf einen direkten Vergleich zwischen den Ergebnissen bei kontinuierlicher und bei diskretisierter Rückkopplung Wert gelegt wird (Simulation des Diskretisierungsfehlers).

SIMULATIONSMETHODEN UND SIMULATIONSSOFTWARE

Die Simulation muß drei Aufgaben erledigen:
- Simulation der kontinuierlichen Regelstrecke
- Simulation der "abgetasteten" Regelstrecke
- Simulation des Diskretisierungsfehlers.

Zwei verschiedene Simulationsmethoden bieten sich an:
1) Parallele Simulation von kontinuierlicher und diskretisierter Regelstrecke mit paralleler Simulation des Diskretisierungsfehlers
2) Serielle Simulation von kontinuierlicher und diskretisierter Regelstrecke mit (anschließender) Simulation des Diskretisierungsfehlers.

Generell ist Methode 2) zu bevorzugen, denn ein- und dasselbe programmierte Modell simuliert beide Regelstrecken, indem zwischen kontinuierlicher und diskretisierter Rückkopplung umgeschaltet wird.

Hybride Simulation

Abbildung 1 zeigt die Implementation der vorgeschlagenen Methode auf einem Hybridrechner (unter Verwendung der hybriden Simulationssprache HYBSYS). Die Abtastung der Ausgänge der Regelstrecke erfolgt mit Speicherpaaren, die von einem Zähler mit hochfrequentem Eingangssignal gesteuert werden.

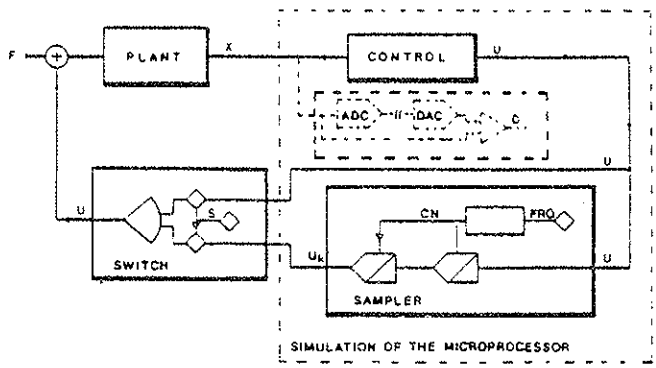


Abbildung 1

Ist das Regelgesetz zeitunabhängig, so kann die Rückkopplung noch kontinuierlich berechnet und erst dann diskretisiert werden, was zu einer Einsparung von Abtastelementen führt. Ein analoger Schalter wählt zwischen kontinuierlicher und diskretisierter Rückkopplung. Zur Simulation des Diskretisierungsfehlers werden hybride Elemente verwendet: in einem ersten Simulationslauf werden die interessierenden Ausgänge der Strecke über ADC s abgespeichert, in einem zweiten Simulationslauf (nach "Umschalten" der Rückkopplung) werden sie über DAC s zur Berechnung des Fehlers (in einem Summierer) herangezogen.

Digitale Simulation

Verwendet man zur Simulation eine digitale Simulationssprache vom CSSL-Typ (z.B. ACSL), so kann die Implementation lt. Abbildung 1 beibehalten werden. Der Abtaster wird direkt mit zwei gekoppelten ZHOLD-Macros modelliert, die von einer PULSE-Funktion gesteuert werden. Die Umschaltung zwischen den Rückkopplungen erfolgt mit einem RSW-Macro. Zur Berechnung des Diskretisierungsfehlers werden die Werte der interessierenden Ausgänge der Regelstrecke im ersten Simulationslauf zu den Auslesezeitpunkten in einem Feld abgespeichert, um dann im zweiten Lauf den Fehler zu berechnen.

ACSL bietet allerdings noch eine komfortablere Art, um Abtastaktionen zu realisieren, nämlich eine DISCRETE SECTION. Die in dieser Sektion programmierten Anweisungen (im vorliegenden Fall die Berechnung der Rückkopplung) werden zu vorgegebenen Zeitpunkten (unabhängig von den Auslesezeitpunkten) durchgeführt, wobei eine laufende Integration unterbrochen und erst nach Ausführung der DISCRETE SECTION fortgesetzt wird.

*) Deutsche Kurzfassung von F. Breitenecker "Comparison of Digital and Hybrid Simulation of Regulator Plants with Alternatively Continuous and Discrete Feedback" First European Simulation Congress ESC 83. Informatik Fachberichte 71, S 261-267, Springer Verlag 1983.

VERGLEICH DER SIMULATIONSMETHODEN BEI DER REGELUNG EINES MAGNETSCHWEBEFAHRZEUGES MIT STÖRGRÖSSEN-KOMPENSATION

Das dynamische Verhalten eines Magnetschwebefahrzeuges kann in erster Näherung durch zwei (lineare) Masse-Feder-Modelle (Fahrzeugunterbau, Wagenkasten) beschrieben werden. Fügt man die Gleichung für den Magneten hinzu, so erhält man in Zustandsraumbeschreibung ein lineares System fünfter Dimension.

$$\dot{x} = Ax + bu + f(t),\ x\varepsilon\mathbb{R}^5,\ u\varepsilon\mathbb{R},\ f\varepsilon\mathbb{R}$$

Die skalare Steuerung u repräsentiert die Ansteuerspannung für den Magneten und kann mit

$$u = k_1x_1 + k_2x_2 + k_3x_3 + k_4x_4 + k_5x_5$$

so ausgelegt werden, daß der Luftspalt zwischen Magnet und Trägerschiene nach einer isolierten Störung wieder den Normalwert annimmt, und daß die permanente Störung f(t) (verursacht durch das Schwingen der Trägerschiene) kompensiert wird.

Hybride Simulation mit HYBSYS

Tabelle 1a zeigt die Definition des Reglers mit wahlweiser Diskretisierung der Steuerung. Die Eingangsfrequenz des Zählers beträgt 50 kHz, sodaß die beiden Track/Store-Einheiten TS und UDIS sehr kleine Abtastraten erreichen können. Mit einer Abtastfrequenz von 42 Hz (0.0235 s) führt die Befehlsfolge

```
S=0;PLOT X1;S=1;PLOT,S X1          (1)
```

zwei Simulationsläufe durch, den ersten mit kontinuierlicher, den zweiten mit diskretisierter Rückkopplung und zeichnet beide Male die sehr empfindliche Zustandsgröße X1 (im selben Achsenkreuz; Abb. 2). Wie man sieht, gleicht die diskretisierte Steuerung UDIS die isolierte Störung aus, die dauernde Störung allerdings resultiert in einer ungewünschten Amplitudenverstärkung von X1. Bei einer Abtastfrequenz von 43 Hz (0.023 s) arbeitet die diskretisierte Rückkopplung ebenso gut wie die kontinuierliche, bei 41 Hz (0.024 s) ist das abgetastete System instabil (Abb. 5).

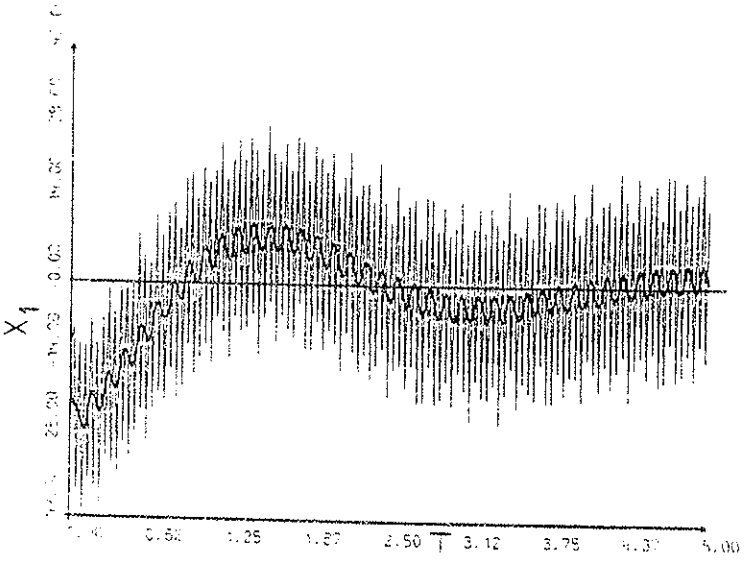


Abbildung 2

Zur Simulation des Diskretisierungsfehlers in X1 wird das Modell um ADC, DAC und Summierer erweitert (Tabelle 2a). Die Befehlsfolge

```
S=0;RUN;S=1;DAX1=ADX1;PLOT DIFX1     (2)
```

berechnet und zeichnet (im zweiten Simulationslauf) den Diskretisierungsfehler (Abb. 3).

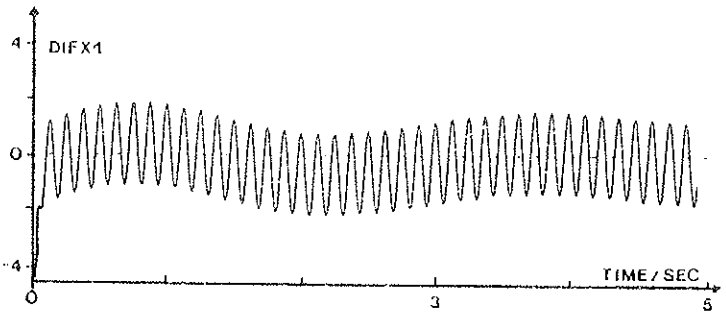


Abbildung 3

Digitale Simulation mit ACSL

Tabelle 1b zeigt die direkte Modellierung des Reglers mit wahlweiser Diskretisierung in ASCL, die HYBSYS sehr ähnlich ist.

Der Parameter SAMP legt die Abtastfrequenz fest. Die zu (1) äquivalente Befehlsfolge für das Simulieren des kontinuierlichen und des abgetasteten Systems lautet

```
SET ST=-1.$START$PLOT X1$SET ST=0.$START$PLOT X1
```

Legt man ein Ausleseintervall von 0.01 sec zugrunde (wodurch die Integrationsschrittweite mit 0.001 festgelegt wird), so arbeitet die diskretisierte Rückkopplung nur für Abtastfrequenzen größer gleich 58 Hz (0.017 s) ebenso gut wie die kontinuierliche, bei ca. 55 Hz (0.018 s) erhält man gleiche Resultate wie bei hybrider Simulation mit 42 Hz (Abb. 2), unter 53 Hz (0.019 s) wird das abgetastete System instabil. Eine Ausnahme bildet hier die Abtastfrequenz 50 Hz (0.02 s). Da Auslesezeitpunkt und Abtastzeitpunkt übereinstimmen, wird gleichzeitig abgetastet und eine neuerliche Integration über ein Teilintervall gestartet, was zu einem wesentlich kleineren globalen Fehler führt und somit nahezu dasselbe Ergebnis für kontinuierliche und diskretisierte Rückkopplung liefert (Abb. 5).

Realisiert man den Regler mit wahlweiser Abtastung mit einer DISCRETE SECTION, so wird die Definition sehr einfach (Tab. 1c). Zu beachten ist, daß nun die Integration alle SAMP Sekunden unterbrochen und die neue Steuerung berechnet wird. Die Ergebnisse bei Verwendung der DISCRETE SECTION sind ähnlich zu den hybriden Simulationsergebnissen mit HYBSYS; mit dem zugrundegelegten Integrationsalgorithmus mit der (konstanten) Schrittweite von 0.001 (Ausleseintervall 0.01 s) beträgt die niedrigste Frequenz, bei der kontinuierliche und diskretisierte Rückkopplung dasselbe Ergebnis liefern, 47 Hz (0.0215 s), bei 45 Hz (0.022 s) wird das abgetastete System instabil, bei 46 Hz (0.0215 s) bleibt es stabil, ohne allerdings die permanente Störung hinreichend zu kompensieren.

a) HYBSYS	b) ACSL, ZHOLD	c) ACSL,DISCRETE SECTION
MODEL SUM:UCON; IRST:T1;UDIS CNTR:CN; DSW1:U; CL:S UCON=-K1*X1,....,-K5*X5 T1=CN*UCON UDIS=_CN*T1 U=S*UCON,_S*UDIS END	DERIVATIVE ⋮ UCON=-K1*X1-....-K5*X5 CN=PULSE(0.,SAMP,SAMP/2.) T1=ZHOLD(UCON,.NOT.CN,UCON) UDIS=(T1,CN,T1) U=RSW(S1,UCON,UDIS) END	DISCRETE SAMPLER INTERVAL SAMP=0.022 UDIS=-K1*X1-....-K5*X5 END

Tabelle 1

a) HYBSYS	b) ACSL
MODEL ADCF:ADX1 DACF:DAX1 SUM:DIFX1 ADX1=X1 DIFX1=X1,-DAX1 END	DYNAMIC ⋮ I=I+1 DIFX1=X1-SX1(I) SX1(I)=X1 ⋮ END

Tabelle 2

Zur Simulation des Diskretisierungsfehlers werden in einem ersten Lauf die Werte in ein Feld abgespeichert und im zweiten mit "umgeschalteter" Rückkopplung zur Berechnung des Fehlers verwendet; diese Abspeicherung und Berechnung erfolgt sinnvollerweise nun in der DYNAMIC SECTION und nicht in der DERIVATIVE SECTION (Tab. 2b).

Die zu (2) korrespondierende Befehlsfolge lautet

```
SET ST=-1.$START$SET ST=0.$START$PLOT DIFX1
```

Zu Vergleichszwecken wurde (durch Erweiterung des Modelles) auch der Unterschied in X1 bei Diskretisierung mit ZHOLD und Diskretisierung mit der DISCRETE SECTION berechnet (Abb. 4).

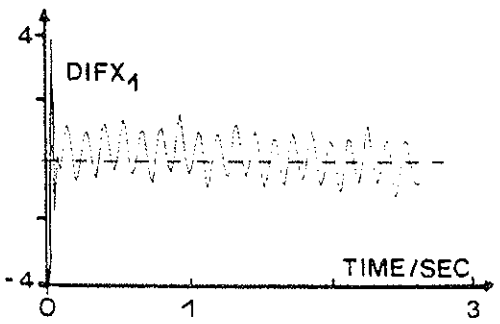


Abbildung 4

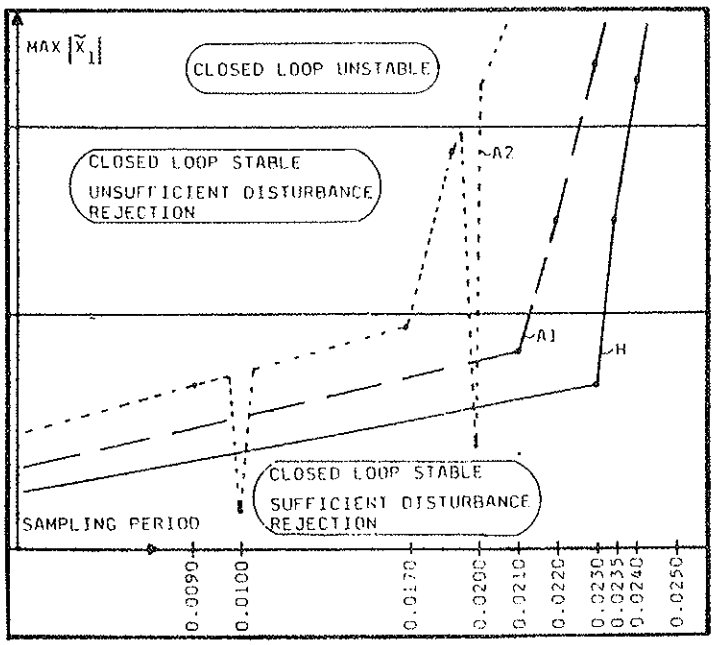


Abbildung 5

Es zeigt sich, daß die "natürliche" Simulation des Abtastens in ACSL mit ZHOLD-Elementen zu teilweise unzufriedenstellenden Ergebnissen führt: die Resultate hängen von der Synchronisation zwischen Abtast- und Ausleseintervallen ab. Die Verwendung der DISCRETE SECTION führt zu besseren Resultaten; allerdings muß der Benutzer gut über ACSL Bescheid wissen, um dieses sehr fortgeschrittene Sprachelement richtig zu verwenden. Um mit der digitalen Simulation mit ACSL genauso niedrige Abtastfrequenzen wie mit hybrider Simulation zu erreichen, muß die Integrationsschrittweite verkleinert werden, was zu einer starken Rechenzeiterhöhung führt. Die Rechenzeit bei hybrider Simulation ist unabhängig von der Abtastfrequenz und (in Grenzen) verkürzbar.

ZUSAMMENFASSUNG

Abbildung 5 vergleicht die "Güte" der Simulation des abgetasteten Systems bei Verwendung von hybrider Simulation mit HYBSYS (H), von digitaler Simulation in ACSL (A1) und ZHOLD-Elementen und in ACSL mit DISCRETE SECTION (A2).

SUMMARY

The aim of this paper is to compare the features of a digital (ACSL) and of a hybrid simulation language (HYBSYS) for the simulation of regulator plants with alternatively continuous and discrete feedback. Proposals for the implementation are given. As example the disturbance rejection control of a magnetically levitated vehicle with continuous or discrete feedback is considered.

NICHTLINEARE ZWANGSSCHWINGUNGEN VON GLEITGELAGERTEN ROTOREN

Eine Anwendung der Simulationssprache ACSL

H. Ecker und H. Springer

Institut für Maschinendynamik und Meßtechnik,
Technische Universität Wien

In einer der letzten Ausgaben von INTERFACE [1], wurde über die digitale Simulationssprache ACSL (Advanced Continuous Simulation Language) berichtet, welche seit etwa einem Jahr den Benutzern der CDC CYBER 720 an der TU-Wien sowohl im Batch- als auch im Time-Sharing-Modus zur Verfügung steht. Diese Simulationssprache wird in einer Reihe von Forschungsaufgaben des oben angeführten Instituts, etwa auf dem Gebiete der Rotordynamik sowie der Simulation von Kraftfahrzeugunfällen mit Erfolg angewendet. Im folgenden soll der Einsatz von ACSL für die Berechnung von zwangserregten Schwingungen von Rotoren mit nichtlinearen Lagercharakteristika gezeigt werden.*)

BEWEGUNGSGLEICHUNGEN DES ROTORS UND DEREN INTEGRATION DURCH ACSL

Bild 1 zeigt einen Mehrmassen-Rotor mit vorgekrümmter Welle und Lagerfluchtungsfehlern. Die zugehörigen Bewegungsgleichungen dieses Systems haben die Form [2]

$$\underline{M}\,\ddot{\underline{z}} + (\underline{D}_E+\underline{D}_I+\underline{G})\,\dot{\underline{z}} + (\underline{C}_S+\underline{C}_I)\,\underline{z} =$$

$$= \underline{F}_E(t) + \omega^2\underline{U}(t) + \underline{C}_S\underline{\delta}(t) + \underline{F}_B(\underline{z}-\underline{m},\dot{\underline{z}},\omega) \qquad (1)$$

Hierin sind $\underline{M}$, $\underline{D}$, $\underline{G}$, $\underline{C}$ die Massen-, Dämpfungs-, Gyroskopie-, und Steifigkeitsmatrizen des Rotors und $\underline{z}$ ist der Lagevektor, der die Verschiebungen und Neigungswinkel der Wellenachse in sämtlichen Knotenpunkten des Rotors enthält. Auf der rechten Seite von Gl.(1) bedeutet $\underline{F}_E(t)$ eine externe beliebige nichtsynchrone Krafterregung (im einfachsten Fall ist das eine statische Last, wie zum Beispiel das Eigengewicht des Rotors). $U(t)$ und $\delta(t)$ sind synchrone Erregungen zufolge Unwucht bzw. Wellenvorkrümmung des Rotors. $\underline{F}_B$ ist eine komplizierte nichtlineare Vektorfunktion, die das Verhalten der hydrodynamischen Lager beschreibt. Sie hängt von den Verschiebungen $\underline{z}$ und den Verschiebungsgeschwindigkeiten $\dot{\underline{z}}$ der Wellenzapfen in den Lagern sowie von den Fluchtungsfehlern $\underline{m}$ der Lager ab.

Für eine nichtsinguläre Massenmatrix $\underline{M}$ geht das System Gl.(1) in ein System 1. Ordnung von der Form

$$\dot{\underline{p}} = \underline{H}(\underline{p},\underline{z},t)$$
$$\dot{\underline{z}} = \underline{p} \qquad (2)$$

über. Damit wird eine unmittelbare Anwendung von ACSL bei gegebenen Anfangsbedingungen

$$\underline{z}(t_o) = \underline{z}_o$$
$$\underline{p}(t_o) = \dot{\underline{z}}_o \qquad (3)$$

möglich. Im Abschnitt DERIVATIVE des ACSL-Programms, welcher die zu integrierenden Bewegungsgleichungen bereitstellt, hat man die folgenden Anweisungen

```
p1 = INTEG (H1, ż01)
p2 = INTEG (H2, ż02)
.
pn = INTEG (Hn, ż0n)          (4)
z1 = INTEG (p1, z01)
z2 = INTEG (p2, z02)
.
zn = INTEG (pn, z0n)
```

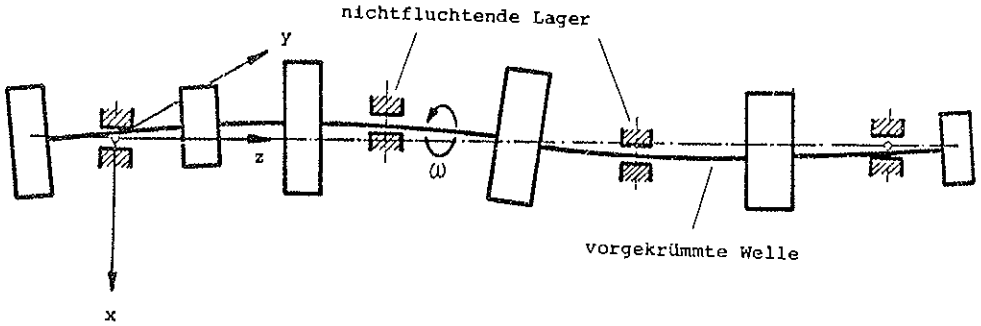


Bild 1
Schematische Darstellung eines Rotor-Lager-Systems

*)Für die stets vorhandene Unterstützung von seiten des ACSL-Betreuers an der Abteilung Hybridrechenanlage des EDV-Zentrums der TU Wien sei an dieser Stelle besonders herzlich gedankt.

aufzulisten. Die nichtlinearen Kraftfunktionen je Masseneinheit

$$H_i = f_i(p_1,\ldots,p_n,\ z_1,\ldots,z_n,t) \qquad (5)$$

mit i = 1,2,...,n, enthalten die hydrodynamischen Lagerkräfte, die Steifigkeits- und Dämpfungskräfte der Welle, sowie die Summe der externen Erregerkräfte und können unmittelbar im Abschnitt DERIVATIVE oder in Form einer FORTRAN-SUBROUTINE oder durch eine ACSL-MACRO-SUBROUTINE bereitgestellt werden.

BERECHNUNGSBEISPIEL

Mit der derzeitigen Ausbaustufe des Programms wurde als Testfall ein symmetrischer Einmassenrotor mit starrer Welle und zwei symmetrisch angeordneten Gleitlagern untersucht. Die Lagerungskräfte des hydrodynamischen Schmierfilms wurden Hilfe der Theorie des kurzen Lagers in analytischer Form bereitgestellt. Als Erregung wurden eine Unwucht (synchron) und eine nichtsynchrone umlaufende Kraft angenommen. Der Rotor ist horizontal gelagert und daher wirken als statische Lasten stets die Eigengewichtskräfte.

Bild 2 zeigt zwei Einschwingvorgänge des Wellenmittelpunkts eines starren Laval-Läufers nach sprunghafter Unwuchterregung. Im Fall (a) ist die Anfangslage so gewählt, daß sie nicht mit der stationären Gleichgewichtslage zusammenfällt, und man erkennt, wie sich das Wellenmittel unter der Wirkung der Unwucht und der Gewichtskräfte von außen dem stationären Unwuchtorbit annähert. Im Falle (b) ist die Ausgangslage die statische Ruhelage des unwuchtfreien Rotors, und das Wellenmittel nähert sich von innen her dem stationären Orbit. In beiden Fällen erkennt man die Nichtlinearität der hydrodynamischen Lagerkräfte am Abweichen der stationären Orbits von der Ellipsenform.

Bild 3 zeigt zwei stationäre Orbits des Wellenzapfens nach abgeklungenem Einschwingvorgang. Neben dem Eigengewicht wirken eine synchron umlaufende Unwuchterregung und eine asynchron rotierende konstante Kraft vom selben Betrage wie die Unwuchterregung. Im Bild 3 (a) läuft die asynchrone Kraft mit einem Viertel der Drehzahl um, im Bild 3 (b) mit dem Vierfachen der Drehzahl. Auch hier sind die Nichtlinearität und Anisotropie der Ölfilmkräfte an der Orbitform zu erkennen. Der in den Bildern angedeutete Viertelkreis gibt das radiale Lagerspiel an, wobei der Wellendurchmesser auf Null zusammengezogen ist.

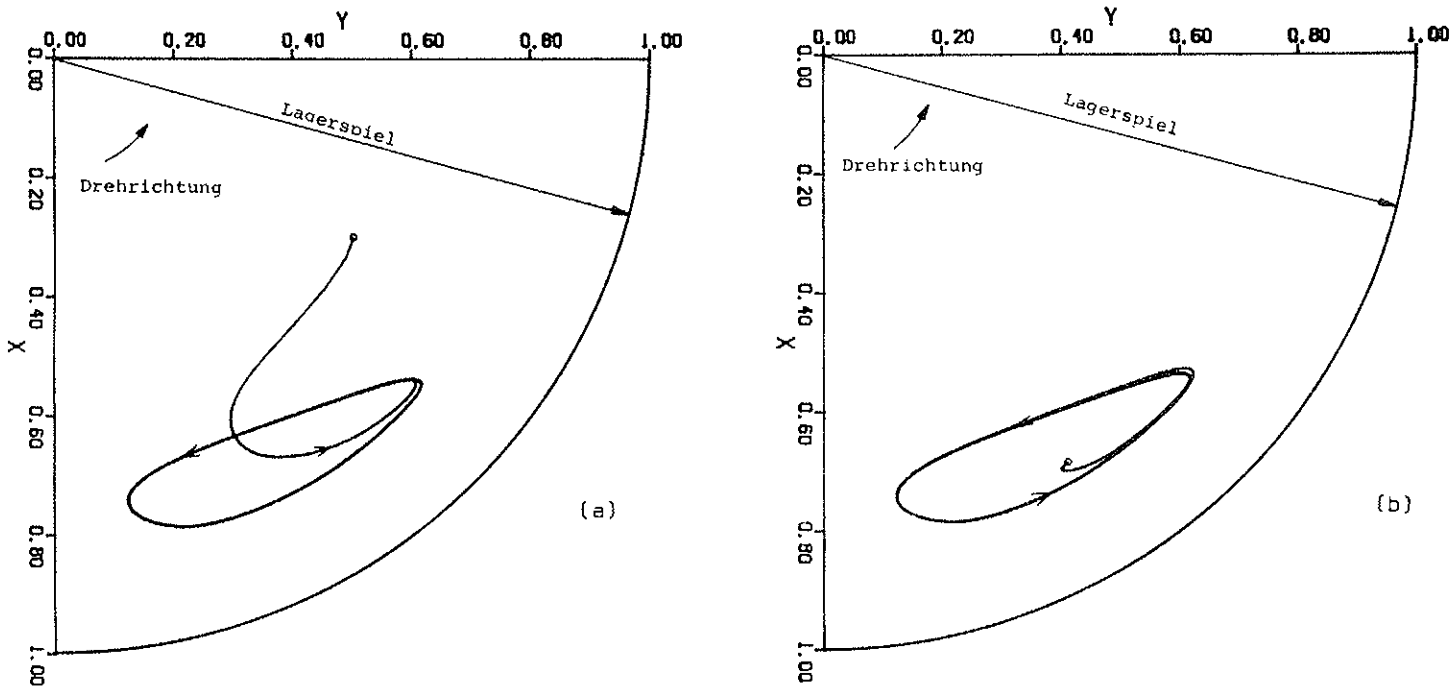


Bild 2
Instationäre Lagerzapfenorbits eines starren gleitgelagerten Laval-Läufers nach sprunghafter Unwuchterregung.
Rotormasse = 16.8 kg
Unwuchtexzentrizität = 200 µm
Drehzahl = 20 Hz
Radiales Lagerspiel = 15 µm
(a) ... Anfangslage ≠ stationäre Gleichgewichtslage
(b) ... Anfangslage = stationäre Gleichgewichtslage

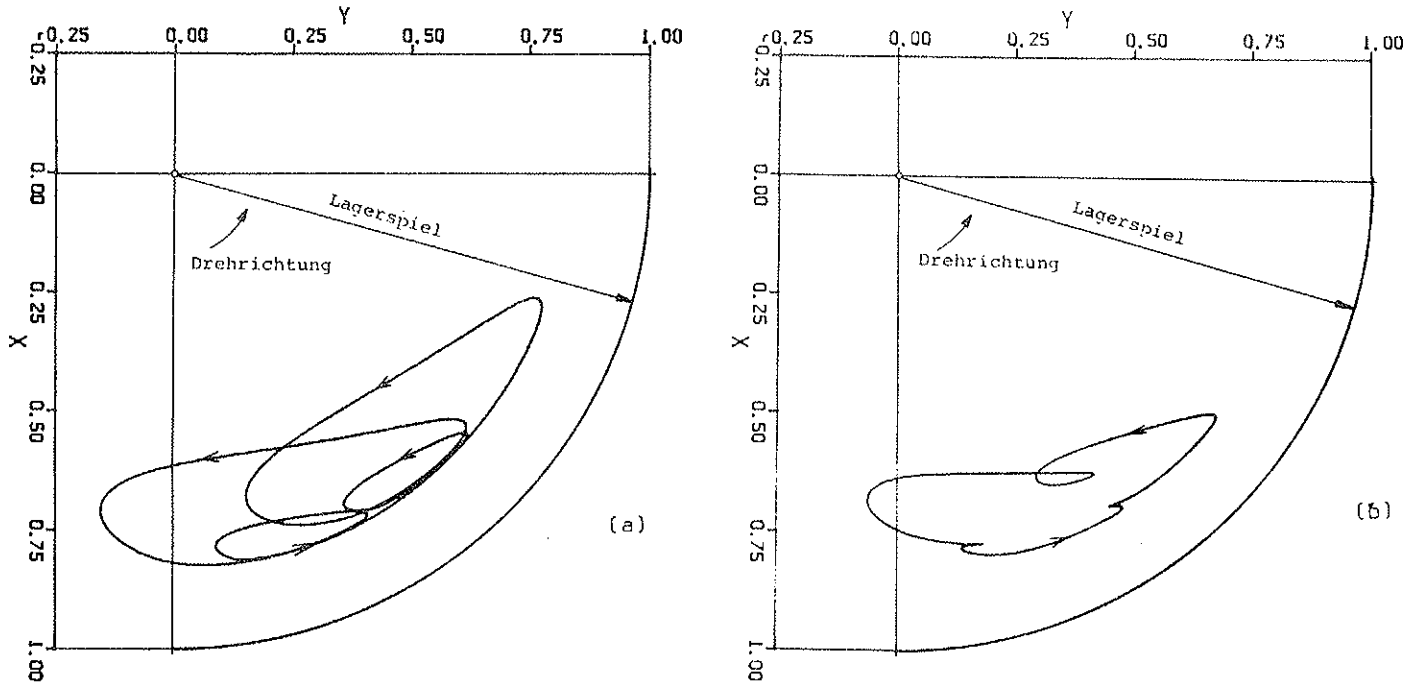


Bild 3
Stationärer Lagerzapfenorbit zufolge umlaufender Synchronerregung (Unwucht) und umlaufender Asynchronerregung.
Systemdaten wie Bild 2.
Kraftamplitudenverhältnis $F_a/F_s = 1$
Phasenwinkel = 0^0
Frequenzverhältnis : Bild (a) $f_a/f_s = 1/4$
Bild (b) $f_a/f_s = 4$

Es sei noch erwähnt, daß es durch Implementierung einer Treiberroutine für einen WATANABE WX4635R Zweifarben-Plotter des Instituts für Maschinendynamik ermöglicht wurde, die interaktive Plotter-Software des ACSL-Programms zu benützen und in rascher und bequemer Weise eine direkte Graphikausgabe zu erzielen. Die obigen Bilder sind ein Beispiel dafür.

Weiters ist zu erwähnen, daß durch den ACSL-Befehl "ANALYZ" eine Linearisierung der Bewegungsgleichungen an einer bestimmten Stelle durchgeführt werden kann und zugehörige lokale Eigenwerte und -vektoren ermittelt werden können. Damit ist es unter anderem möglich, durch die höchste auftretende Eigenfrequenz im System eine Abschätzung der größtmöglichen Integrationsschrittweite vorzunehmen.

LITERATUR

[1] Hummer H., ACSL (Advanced Continuous Simulation Language). Interface 19, Dez. 1982

[2] Springer H., Gunter E.J., Nonlinear Unbalance Response of Multimass Rotor-Bearing Systems with Initial Shaft Bow and Bearing Misalignment. Report No. UVA/643078/MAE 83, Rotor Dynamics Lab, University of Virginia, USA.

SUMMARY

The equations of motion for a multimass rotor supported by a misaligned multi-bearing system are presented. Since the characteristics of hydrodynamic bearings are highly nonlinear the rotor response due to synchronous and nonsynchronous excitation, in general, cannot be evaluated analytically. However, for nonsingular mass matrices of the system the equations of motion may easily be transformed into a first order system and a direct numerical integration is made possible by ACSL. This digital simulation language is extremely convenient to use and provides even graphical output on a plotter which has to be provided and programmed by the user. As an example, transient and steady state as well as synchronous and nonsynchronous orbits of a single-mass Jeffcott rotor, evaluated by ACSL-integration, are presented in the paper.

ASIM 84

Erste Ankündigung und Einladung zur Vortragsanmeldung

2. SYMPOSIUM SIMULATIONSTECHNIK

Wien, 25.-27. September 1984

Veranstalter

ASIM - Fachausschuss 4.5 Simulation in der Gesellschaft für Informatik (GI)
ASIM ist eine Vereinigung, die sich der Förderung und der Weiterentwicklung der Simulation in allen Fachrichtungen widmet.

Tagungsziel

Ziel des 2. SYMPOSIUMS "SIMULATIONSTECHNIK" ist es, den Austausch von Ideen und Erfahrungen von Fachleuten und Interessenten zu fördern, die auf dem Gebiet der Modellbildung und Simulation in Theorie und Praxis tätig sind.

Alle Gebiete der Simulation werden im Symposium behandelt:

-Methodologie
Modellbildung, Modellvalidierung * Modellbeschreibungsverfahren * Mathematische Verfahren * Systemidentifikation * Optimierungsverfahren

-Verfahren und Methoden der Simulation
Simulation auf Analog- und Hybridrechnern * Digitale Simulation für diskrete, kontinuierliche und kombinierte Systeme * Simulationssprachen * Softwareunterstützung * Künstliche Intelligenz in der Simulation * Datenverwaltung in der Simulation * Hardwareunterstützung: Rechnersysteme und Rechnerarchitektur

-Anwendungsgebiete
Ingenieurwissenschaften * Mathematik * Naturwissenschaften wie Chemie, Physik, usw. * Medizin, Biologie, Ökologie * Wirtschaftswissenschaften * Verwaltung, Planung, Operations Research * Sozialwissenschaften * Ausbildung

Tagungsleitung

F. Breitenecker, W. Kleinert, Technische Universität Wien

Teilnehmerbeitrag

Es wird ein Teilnehmerbeitrag von DM 150.-- erhoben (ASIM- und GI-Mitglieder DM 120.--).

Vortragsanmeldung

Bitte richten Sie Ihre Vortragsanmeldung bis zum 31. März 1984 an die Tagungsleitung. Benötigt werden eine ein- bis zweiseitige Zusammenfassung, Titel, Name und Anschrift des Verfassers.
Eine Mitteilung über die Annahme oder die Ablehnung der Vorträge durch das Programmkomitee erfolgt bis 16. April 1984.
Die endgültige Form des Vortrages in Form eines direkt reproduzierbaren Manuskriptes wird bis spätestens 30. Juni 1984 benötigt.
Die Tagungssprache ist Deutsch. Der Tagungsband in der Reihe "Informatik-Fachberichte" wird bis zu Beginn des Symposiums zur Verfügung stehen.

Auskunft

W. Kleinert, Tel. 0222/5601-3702 DW * F. Breitenecker, Tel.0222/5601-3747 DW

Adresse

Hybridrechenzentrum
Technische Universität Wien
Gusshausstrasse 27-29
A-1040 Wien

INTERFACE Dezember 1983