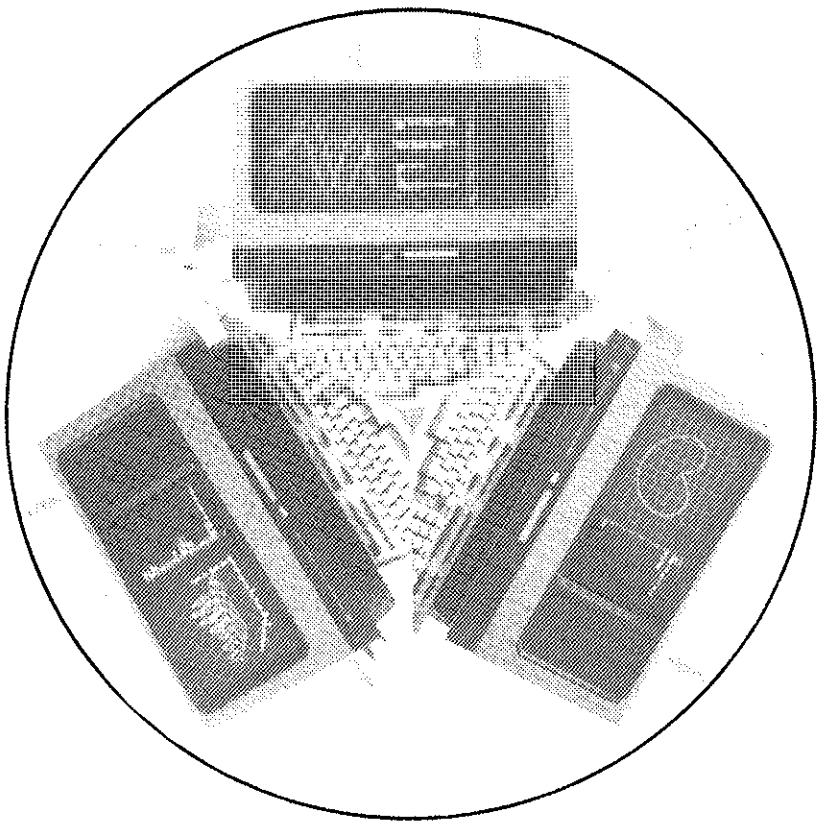
Interface

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

Nummer 19
Dezember 1982



Neu: Hybrides
Time-Sharing-System MACHYS

INHALTSVERZEICHNIS

Redaktion: Irmgard Husinsky
Eigentümer, Herausgeber, Verleger: EDV-Zentrum der Technischen Universität Wien, Abteilung Hybridrechenanlage
Vervielfältigung: Österreichische Hochschülerschaft Technik
Alle: Gußhausstraße 27-29, A-1040 Wien
Telex: 76875 rzthw a

Vorwort

Wir können wieder eine, wie wir glauben, interessante Nummer der Zeitschrift INTERFACE vorlegen. Auf einige der Artikel möchten wir besonders hinweisen:

Über das an der Technischen Universität Wien entwickelte hybride Time-Sharing-System MACHYS ist auf Seite 8 die neueste und umfassendste Dokumentation des aktuellen Standes dieses neuen Hybridrechensystems veröffentlicht.

Da wir sehr für die Verbreitung des Hybridrechnens eintreten, freuen wir uns, in diesem Heft die Hybridrechenanlage der Technischen Universität Graz vorstellen zu können, und hoffen, daß wir in Zukunft regelmäßig über Aktivitäten an Hybridrechnern in ganz Österreich berichten können.

Dr. D.J. Murray-Smith, Senior Lecturer an der Universität Glasgow, stellte uns freundlicherweise zwei Originalarbeiten zur Veröffentlichung zur Verfügung. Es handelt sich um zwei Themen, die Dr. Murray-Smith in zwei Vorträgen behandelte, als er im Dezember 1981 auf Einladung der Hybridrechenanlage des EDV-Zentrums der Technischen Universität Wien und des Instituts für Medizinische Statistik und Dokumentation der Universität Wien, sowie im Rahmen des Außeninstituts der Technischen Universität Wien, des ÖPWZ, der Arbeitsgemeinschaft für Automatisierung, und der International Federation of Automatic Control (IFAC), in Wien war. Die beschriebenen Simulationen wurden bei uns an der Hybridrechenanlage mithilfe des Simulationssystems HYBSYS nachvollzogen und demonstriert.

Wir hoffen, daß dieses INTERFACE Ihr Interesse findet.

Irmgard Husinsky
Redaktion

aktuelle mitteilungen

NEU: HYBRIDES TIME-SHARING-SYSTEM MACHYS

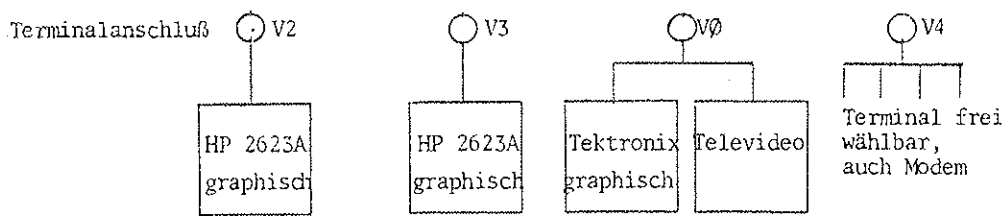
Mitte September 1982 konnte der hybride Rechenbetrieb auf das an der Technischen Universität Wien entwickelte hybride Time-Sharing-System MACHYS (Multi-ACcess HYbrid System) umgestellt werden (siehe auch Seite 8). Dieses neuartige, automatisierte Hybridrechensystem bringt für den Benützer der Hybridrechenanlage folgende Vorteile:

- Es können vorerst bis zu drei (1983 bis zu acht) Hybridbenützer parallel und unabhängig voneinander rechnen.
- Der hybride Sprachprozessor HYBSYS liegt in einer neuen, stark verbesserten und komfortableren Version (5 TS) vor.
- Die Macro-Datenbasis der verwendbaren Analogkomponenten im AutoPATCH-System wurde erweitert.

Das neue HYBSYS User Manual, Version 5 TS, dokumentiert alle Befehle des hybriden Sprachprozessors HYBSYS und ist zum Preis von öS 50.- an der Hybridrechenanlage erhältlich.

RECHENBETRIEB, TERMINALRESERVIERUNG

Durch die Installierung des neuen hybriden Time-Sharing-Systems MACHYS ist eine Open-Shop-Rechenzeitreservierung für Hybridbenützer nicht mehr nötig. Es muß jedoch eine Reservierung eines Terminals beim Operator vorgenommen werden. Dafür stehen vier Terminals zur Verfügung, siehe unten stehende Übersicht. Benützer, die HYBSYS rechnen wollen und graphische Ausgabe wünschen, müssen ein graphisches Terminal reservieren.



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. Eine ausführliche Dokumentation stellt das JCS/VS 8 User Manual dar, das in einer neuen Bearbeitung vorliegt und zum Preis von öS 50.- an der Hybridrechenanlage erhältlich ist.

Es wurde ein neuer Pass-Code für das Rechnen im Terminal-Betrieb eingeführt. Dieser dreibuchstabige Code wird nur dem Benützer (bei der Abgabe des Jobnummernansuchens) bekanntgegeben und kann diesem später nicht mehr mitgeteilt werden. Jeder Benützer kann seine Daten- und Programmfiles vor dem Lesen bzw. Schreiben durch andere Benützer schützen.

Der Source Prozessor wurde generell beschleunigt und benützerfreundlicher gestaltet. Die Source-Prozessor-Befehle beginnen nicht mehr mit dem Zeichen "\", die ersten zwei Zeichen eines Befehls genügen als Eingabe. Ein Tabulator kann in Texten voll verarbeitet werden. Es kann eine linke und rechte Spaltengrenze (Margin) für die Bearbeitung von Texten angegeben werden. Der Befehl COPY ermöglicht das Kopieren von Textbereichen. Diese Neuerungen wurden auch in die Routinen zum programmgesteuerten Editieren von Source Files übernommen.

Das Utility-Programm FILLIP ermöglicht das Erstellen von Filelisten nach verschiedenen wählbaren Kriterien (Filename, Typ, Jobnummer, Datum, District, etc.).

Es stehen FORTRAN-aufrufbare ASCII-Manipulations- und Konversionsroutinen zur Verfügung. Zur Bearbeitung ungelabelter, ungeblockter Magnetbänder mit 800 CPI Schreibdichte gibt es komfortable Unterprogramme. Intern können nun 8 logische Tapestationen angeschlossen werden.

Für Tischrechnerfunktionen kann der interaktive digitale Operations-Interpreter DOI verwendet werden.

BENÜTZERVERSAMMLUNG

Am 14. Oktober 1982 wurde anläßlich der Umstellung auf das hybride Time-Sharing-System MACHYS eine Benützerversammlung der Hybridrechenanlage abgehalten, bei der das neue System an zwei Terminals demonstriert wurde. Anschließend fand eine Diskussion statt.

ACSL-INFORMATIONSVERANSTALTUNG

Das digitale Simulationsprogramm ACSL (Advanced Continuous Simulation Language) mit der derzeit weltweit größten Zahl von Installationen wurde von der Hybridrechenanlage zur allgemeinen Verwendung an der CDC CYBER 72Ø installiert und wird in einer Informationsveranstaltung am 13. Jänner 1983 um 1Ø Uhr im kleinen Seminarraum des Instituts für Analysis, Technische Mathematik und Versicherungsmathematik mit einer On-line-Demonstration vorgestellt. Interessierte Benützer können auch im Rahmen des Kurses AH3+4 (siehe unten) die Anwendung der Simulationssprache ACSL erlernen.

KURSE des Arbeitsbereiches "Regelungstheorie und Hybridrechentechnik" des Instituts für Analysis, Technische Mathematik und Versicherungsmathematik:

Kurs AH 1+2 17.1. - 21.1.1983
Simulationen an den Kleinrechnern EAI 1000 - Einführung in das AutoPATCH-System - Prozessor HYBSYS

Kurs AH 3+4 24.1. - 28.1.1983
AutoPATCH-System mit Prozessor HYBSYS und Plotterunterstützung - Hybride Verfahren und Anwendungen: gewöhnliche Differentialgleichungen, partielle Differentialgleichungen, Integral- und Funktionalgleichungen, Optimierung (Overlayunterstützung in HYBSYS) - Einführung in die Simulationssprache ACSL (26. und 27. 1.).

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

kurse

RH3 Hinweise für FORTRAN-Programmierer an der Hybridrechenanlage
Termin: 1983/03/15
Vortragender: Dipl. Ing. F. Blöser

RH7 Softwareunterstützung für die Benützung des PACER 600 als Plotter System
Termin: 1983/03/15 und 1983/03/16
Vortragender: Dipl. Ing. F. Blöser

Die folgende Kurse werden nach Vereinbarung abgehalten:

RH1 Gerätetechnik EAI PACER 600A AutoPATCH-System

RH2 Benützung des Betriebssystems JCS/VS 8

RH6 EAI Assembler

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).

Das neue hybride Time-Sharing-System MACHYS

W. Kleinert

ZUSAMMENFASSUNG: Das hybride Time-Sharing-System MACHYS (Multi-ACcess HYbrid System) wurde an der Hybridrechenanlage der Technischen Universität Wien mit Unterstützung des Fonds zur Förderung der wissenschaftlichen Forschung entwickelt. Durch die Verwendung einer analogen Schaltmatrix sowie eines selbstentwickelten parallelen logischen Prozessors besteht die Möglichkeit, alle Rechenkomponenten des hybriden AutoPATCH-Systems EAI PACER 600A softwaregesteuert zu verwenden und einen vollständigen Problemwechsel im Millisekundenbereich durchzuführen. Bis zu acht Benutzer des interaktiven hybriden Simulationsprozessors HYBSYS 5 TS haben simultanen Zugriff zu allen Resourcen des Hybridrechners.

SUMMARY: The hybrid time-sharing system MACHYS (Multi-ACcess HYbrid System) is presented which has been developed at the Hybrid Computation Centre of the Technical University of Vienna with the support of the Austrian Science Foundation. An analog switch matrix and a selfdeveloped parallel logic processor allow high level software control of all computing components of the hybrid AutoPATCH system EAI PACER 600A. A complete problem change can be done within the range of milliseconds. The software is based on the interactive hybrid simulation processor HYBSYS 5 TS and the virtual time-sharing operating system JCS/VS 8, both developed at the Technical University of Vienna. Up to 8 hybrid users can have simultaneous access to all hybrid computing resources.

1. Einleitung

Seit Mitte September dieses Jahres ist am Hybridrechner der Technischen Universität Wien eine erste Version des hybriden Time-Sharing-Systems MACHYS im Einsatz. MACHYS ist ein Projekt, das von der Hybridrechenanlage gemeinsam mit dem Institut für elektrische Regelungstechnik (Vorstand: o.Prof.Dr.A. Weinmann) seit Ende 1980 durchgeführt und vom Fonds zur Förderung der wissenschaftlichen Forschung unterstützt wird. Die ersten Ergebnisse dieses Forschungsprojekts wurden im April 1982 auf der Simulationstagung ASIM '82 in Erlangen vorgestellt [1-3]. Im folgenden wird auf die Motivation und die Hardwarekomponenten, insbesondere die notwendigen Neuentwicklungen, näher eingegangen. Ferner werden die wichtigsten Neuerungen der mit der Umstellung auf hybrides Time-Sharing installierten Version 5 TS des hybriden Simulationsprozessors HYBSYS und der derzeit implementierte Time-Sharing-Mechanismus dargestellt.

2. Warum hybrides Time-Sharing?

Das Projekt MACHYS ist das vorläufig jüngste in einer Reihe von Forschungsvorhaben an der Technischen Universität Wien, die sich mit der Entwicklung und Verbesserung von hybriden Hard- und Software-Werkzeugen für die Simulation dynamischer Systeme befaßten.

Seit mehr als 10 Jahren ist an der Technischen Universität Wien ein Hybridrechner EAI PACER 600 installiert, der seit 1973 im Rahmen des universitären EDV-Zentrums als Dienstleistungsbetrieb geführt wird. Der an sich ungewöhnliche Versuch, hybride Rechenkapazität als Simulationshilfe im Rahmen einer EDV-Serviceleistung einer Vielzahl unterschiedlicher Benutzer anzubieten, war sehr erfolgreich, erforderte allerdings ständige Modernisierungen des Rechners und eigenständige Entwicklungen. Der Anstoß für alle diese Entwicklungen war weniger in theoretischen Überlegungen, als in der notwendigen Befriedigung eines gestiegenen Benutzerbedarfs zu suchen. Alle Entwicklungen gingen in die Richtung eines immer vollständiger automatisierten Simulationssystems. Ende 1979 wurde eine AutoPATCH-Erweiterung des Analogteils von der Firma EAI geliefert und installiert. Nach einer Übergangsphase, in der neben der traditionellen Programmierung des Analogrechners unter Verwendung von auswechselbaren Problembrettern auch ein Testbetrieb des AutoPATCH-Systems [4] durchgeführt wurde, wurde im Oktober 1980 ein ausschließlich AutoPATCH-orientierter hybrider Open-Shop Betrieb eingeführt. Eine Hardwarevoraussetzung dafür war die Entwicklung eines Pipelined Boolean Processors [5] als Ersatz für die diskrete parallele Logik des Analogrechners EAI 680. Damit wurden die Nachteile der manuellen Eingriffe beim Programmieren konventioneller Hybridrechner eliminiert.

Seit 1974 wurde an der Hybridrechenanlage an der Entwicklung des interaktiven Sprachprozessors HYBSYS gearbeitet [2], dessen Ausgangspunkt ein Algorithmus zur automatischen Online-Skalierung von analogen Rechenschaltungen war. Heute ist er ein sprachliches Simulationswerkzeug, das eine vollständig maschinenunabhängige Programmierung des gesamten hybriden AutoPATCH-Systems erlaubt.

Seit 1972 wird auch an der Entwicklung von Betriebssystemen für den Digitalteil des Hybridrechners, den Minicomputer EAI PACER 132 gearbeitet. Die neueste Version, JCS/VS 8 [6], ist ein völlig virtuelles System, d.h. neben virtuellen Massenspeichern gibt es virtuelle Terminals, virtuelle Files und virtuelle Systemprozessoren, die unabhängig von der Anzahl der parallel arbeitenden Benutzer arbeiten. Im Multiprogramming werden 18 Level - 8 Terminal User (reentrant Source Processor und File Controller), 8 Batch User (die auch real-time hybrid rechnen können) sowie 2 Systemlevel - parallel bedient. Von jedem Arbeitsplatz kann die Programmentwicklung und Programmexekution durchgeführt werden. Die virtuelle Architektur stellte eine ausgezeichnete Voraussetzung für die Einbettung des hybriden Time-Sharing-Systems in das Betriebssystem dar.

Die angeführten Hard- und Softwareentwicklungen waren günstige Voraussetzungen für die Implementierung eines hybriden Time-Sharing-Systems. Da sie den Zugang zum Hybridrechner enorm erleichterten, schufen sie aber auch indirekt die Notwendigkeit einer weiteren Automatisierung und besseren Nutzung der Resourcen des Hybridrechners durch Time-Sharing.

Die große Zahl der verschiedenen Benutzerprojekte, von denen sich erfahrungsgemäß jeweils 10 - 15 in der "Produktionsphase" befinden, führte in der Vergangenheit dazu, daß pro Benutzer im Schnitt nur 2-6 Stunden Open-Shop-Zeit pro Woche reserviert werden konnten. Obwohl mit der Einführung des AutoPATCH-Systems die Produktivität für Experimente mit Simulationsmodellen gegenüber dem Zustand des händischen Steckens und Problembrettwechsels drastisch gesteigert werden konnte, führte diese Einschränkung besonders in den kritischen Zeiten wie zu Semesterschluß oder bei Blockseminaren zu Engpässen und behinderte in einem gewissen Ausmaß auch die wissenschaftlichen Arbeiten.

Eine von uns durchgeführte Analyse der häufigsten Simulationen hat gezeigt, daß der Hauptteil der verbrauchten Echtzeit am Analogrechner für einen repetierenden Betrieb zur Erzeugung stehender Bilder von skalierten Variablen am Oszillographen verwendet wurde. Ein relativ großer Teil der Einschaltzeit verstrich durch wiederholte Überlegungsphasen des Benutzers (was übrigens keinerlei Kritik darstellt, sondern nur die Forderung nach höchstmöglicher Interaktivität unterstreicht), während die tatsächlich notwendige Belegung des Hybridrechners für Parametervariation, Online-Skalierung und andere Experimente mit dem Modell weniger als 10% der realen Anschlußzeit im Open-Shop-Betrieb entsprach.

Viele dieser Probleme können durch ein Time-Sharing-System gelöst werden. Darüber hinaus schafft ein Time-Sharing auch die prinzipielle Möglichkeit, über Remote-Anschlüsse, d.h. unter Verwendung von Modems, den Hybridrechner von graphischen Terminals aus, die nicht unbedingt mehr an der Hybridrechenanlage selbst stehen müssen, auch außerhalb der Operator-betriebenen Öffnungszeiten zu benützen.

3. Zielsetzung des Projektes MACHYS

Aufbauend auf der an der Hybridrechenanlage der Technischen Universität Wien vorhandenen Hard- und Software sollte ein hybrides Mehrbenutzersystem mit parallelem und Time-Sharing-Zugriff konzipiert und realisiert werden.

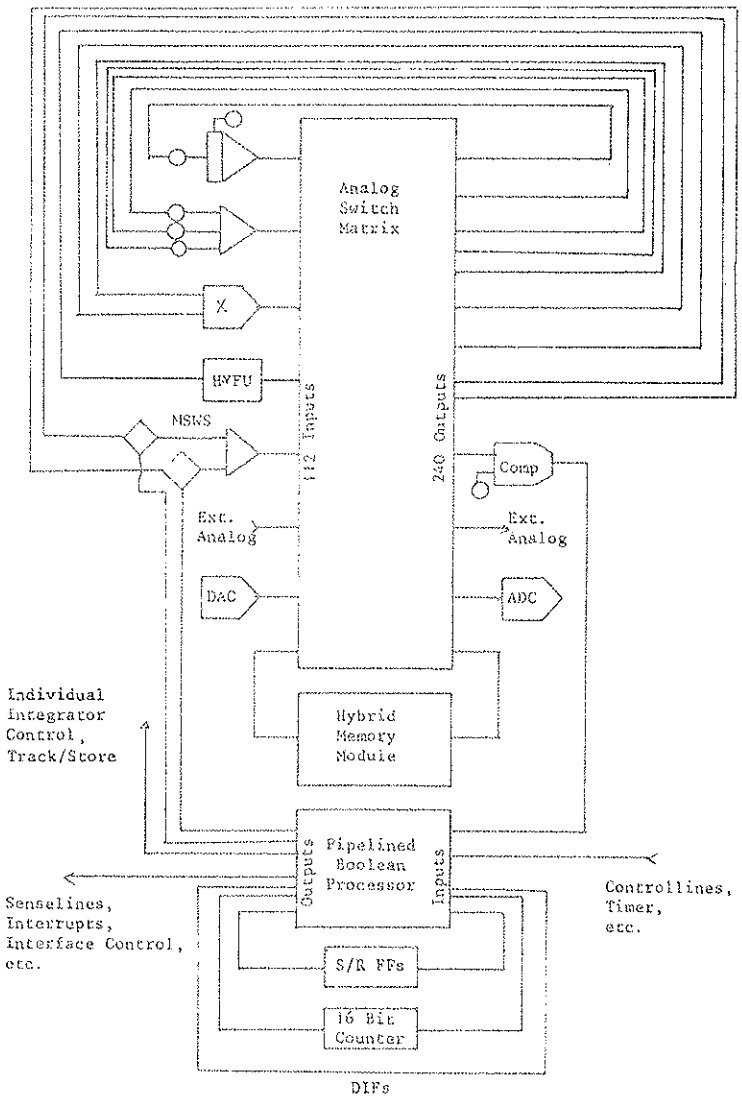


Abbildung 1
AutoPATCH-System

Der Zugriff auf alle analogen und hybriden Rechenkomponenten des AutoPATCH-Systems (Abb.1) sollte ausschließlich softwaregesteuert auf der Ebene einer höheren Simulationssprache erfolgen. Der Problemwechsel sollte ohne explizite Benutzeranforderung und so rasch erfolgen, daß er jedem Benutzer den Eindruck der alleinigen Verfügung über den Hybridrechner vermittelt. Die Programmierung sollte möglichst keine detaillierten Kenntnisse über die zugrundeliegende Hardware voraussetzen und ohne die zahlreichen notwendigen Tricks der konventionellen hybriden Programmierungstechniken auskommen. Diese Zielsetzungen wurden für einen ersten, einfachen seriellen Time-Sharing-Mechanismus durch die Implementierung der Version 5.1 TS von HYBSYS realisiert.

4. Die zugrundeliegende Hardware

Abbildung 2 stellt ein Blockschaltbild der Hardwarekonfiguration des hybriden Time-Sharing-Systems MACHYS dar. Die wesentlichen Teile des AutoPATCH-Systems - die analoge, programmierbare Schaltmatrix mit 5120 integrierten CMOS-Schaltern und der Pipelined Boolean Prozessor (PBP) - wurden bereits in früheren INTERFACE-Artikeln ausführlich beschrieben [4,5]. Sie bleiben auch das Kernstück der Hardware des neuen hybriden Time-Sharing-Systems. Die Forderung nach einem möglichst raschen Problemwechsel brachte es aber mit sich, daß unser betagter Analogrechner EAI 680 ein gehöriges "Facelifting" über sich ergehen lassen mußte.

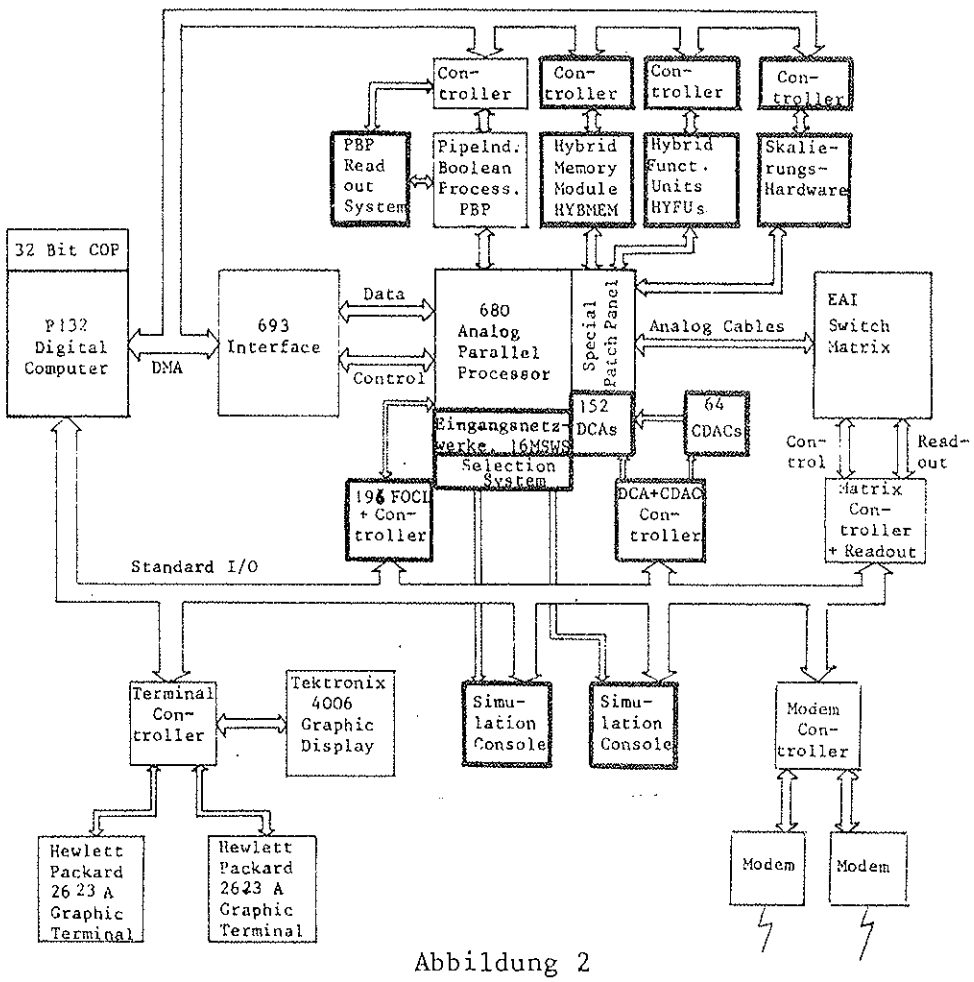


Abbildung 2
Hardware-Konfiguration

Eine Analyse der Engpässe im Zeitverhalten oder in der Kapazität der Hardware zu Beginn des Projektes führte zur notwendigen Behebung durch neuentwickelte, dedizierte Hardware, die in Abbildung 2 stark umrandet dargestellt ist.

So waren ursprünglich nur 60 digitale Potentiometer, aber 60 Servopotentiometer vorhanden, die wegen ihrer im Sekundenbereich liegenden Einstellzeit durch neuentwickelte steckerkompatible DCAs ersetzt werden mußten. Ebenso war es zum Aufbau komplexer Makros notwendig, 32 weitere DCAs, 64 digitale Konstantengeber sowie 192 zusätzliche digitale Kontrolleitungen zur Steuerung von neu entwickelten Eingangsnetzwerken für Integrierer und Summierer mit softwareumschaltbaren Verstärkungsfaktoren zu bauen. Da die Übertragung von zeitdiskretisierten hybriden Variablen zeitkritisch und nur in einer Richtung über den direkten Speicherkanal von oder zum Digitalrechner möglich ist, wurden sogenannte hybride Memories (HYBMEMs), d.h. lokale RAMs mit eigenem ADC bzw. DAC, entwickelt, die zeitsynchron tabellierte Variable abtasten bzw. dem Analogrechner anbieten können (Abb.3).

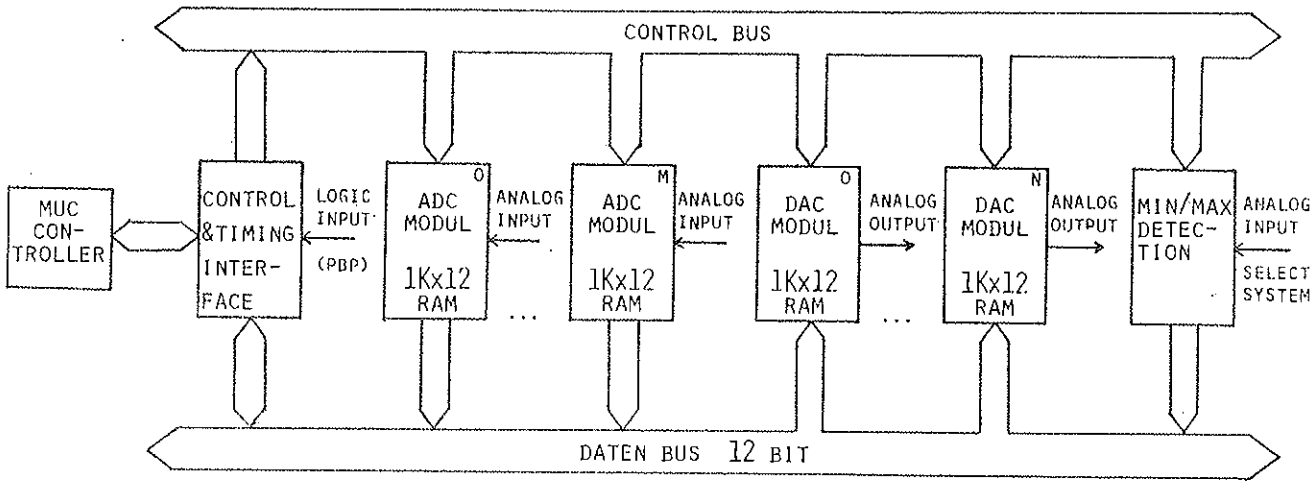


Abbildung 3
Hybride Memories, Blockdiagramm

Derzeit sind vier ADC- und sechs DAC-Moduln realisiert. Zwei der ADC-Moduln dienen normalerweise als System-ADCFs und sind mit den beiden Ausgängen XSEL bzw. YSEL des neuen zentralen analogen Auslesesystems verbunden (Abb.4). Auf XSEL und YSEL kann softwaregesteuert jeweils einer der 64 Eingänge oder einer der 128 Ausgänge einer der beiden analogen Teilschaltmatrizen angewählt werden. Das früher in Verwendung stehende ASEL-System wurde wegen der zu langsamen Schaltzeiten eliminiert.

Zur besseren Unterstützung der Dokumentation des Zeitverlaufs logischer Variablen ist ein eigenes Auslesesystem für den PBP geplant, das bereits entworfen, aber noch nicht realisiert wurde. Ebenfalls erst in der Designphase befinden sich spezielle, hybride Funktionseinheiten (HYFUs) zur tabellengesteuerten, nichtlinearen Funktionserzeugung von Funktionen einer oder zweier Variablen. Diese sollen die veralteten DCFGs ersetzen, aber auch leicht zur

parallelen Hardwarerealisierung von Totzeiten, Zeitverzögerungen etc. herangezogen werden können. Es ist auch geplant, die Online-Skalierung durch eigene Hardwareelemente zur automatischen Bestimmung des Maximums, Minimums und des Überlaufzeitpunktes zu unterstützen.

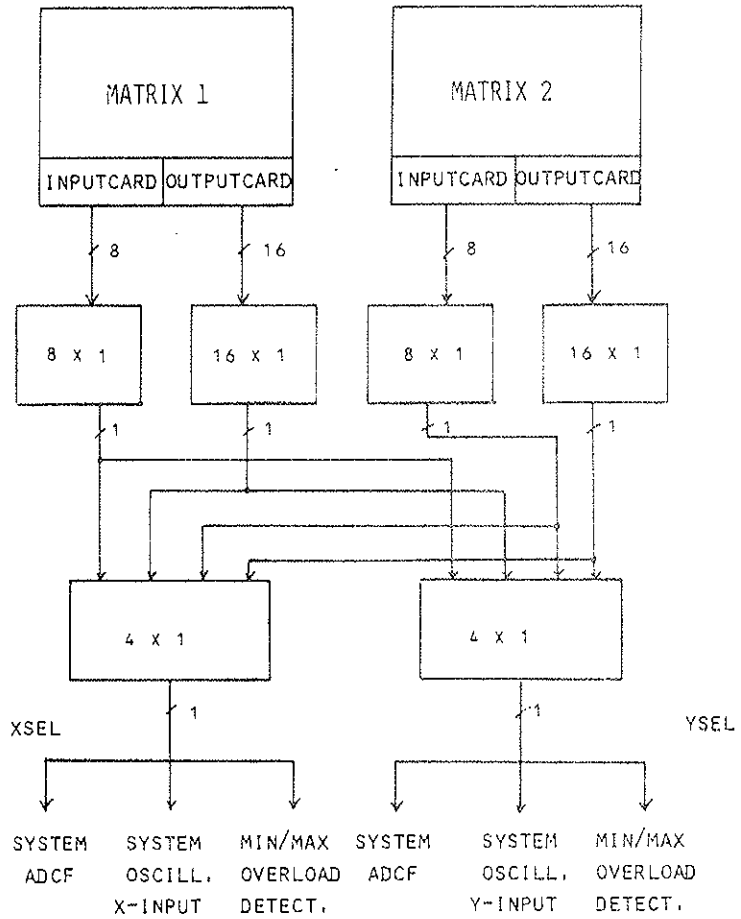


Abbildung 4
Zentrales analoges Auslesesystem

Aus Abb.2 läßt sich ersehen, daß für die interaktive Steuerung des Time-Sharing-Systems und zur graphischen Dokumentation von Simulationsergebnissen neben dem Tektronix 4006 auch zwei HP 2623A Graphik-Terminals zur Verfügung stehen. Im Laufe des nächsten Jahres soll der Zugang durch zwei Wählleitungsmodems auch außerhalb der Operatorbetriebszeiten möglich gemacht werden. Als Ersatz des früher verwendeten Oszillographen wurden im Rahmen des Projekts MACHYS mikroprozessorgesteuerte Simulationskonsolen entworfen, die auf einem 512x512 Punktraster die Darstellung dynamischer Veränderungen skalierter Variabler in Echtzeit ermöglichen. Mit ihrem Einsatz ist allerdings erst Mitte 1983 zu rechnen.

5. Die Time-Sharing-Implementierung des interaktiven Sprachprozessors HYBSYS 5 TS

Mit dem an der Hybridrechenanlage von D. Solar entwickelten interaktiven Sprachprozessor HYBSYS steht eine einfache, problemorientierte Simulationssprache mit blockstrukturierter Modelldeklaration zur Verfügung. Alle hardwareabhängigen Aufgaben der Modellaufbereitung am Hybridrechner, wie das Erstellen der Schaltverbindungen, das Setzen der Koeffizienten und die automatische Online-Skalierung werden vom Prozessor selbsttätig unterhalb der Benutzerebene ausgeführt. Befehle mit selbsterklärender Syntax ermöglichen eine einfache Untersuchung des Modells mittels Parametervariation, Lösung von Teilproblemen wie Randwertaufgaben und Parameteroptimierung bis hin zur Modelländerung selbst. Numerische und graphische Darstellung der Lösungen und Verfahrensabläufe bieten jederzeit einen ausreichenden Einblick in das Modellverhalten und ermöglichen dadurch sowohl einen Eingriff in laufende Verfahren als auch eine ausführliche Dokumentation. Das Simulationssystem besteht aus einem hardwareunabhängigen Teil, der die Modelldatenbasis und alle problembezogenen Moduln zur Modelldeklaration und Modelluntersuchung enthält, und einem hardwareabhängigen Teil, der die Hardwaredatenbasis und alle darauf zugreifenden Moduln wie Hardwaretests, Matrix-Set-Up, Koeffizienten-Set-Up, etc. umfaßt. Die beiden Datenbasen haben eine isomorphe Struktur, die es dem Zuordnungsmodul ermöglicht, eine bijektive Abbildung der Modelldatenbasis in die Hardwaredatenbasis herzustellen (Abb.5).

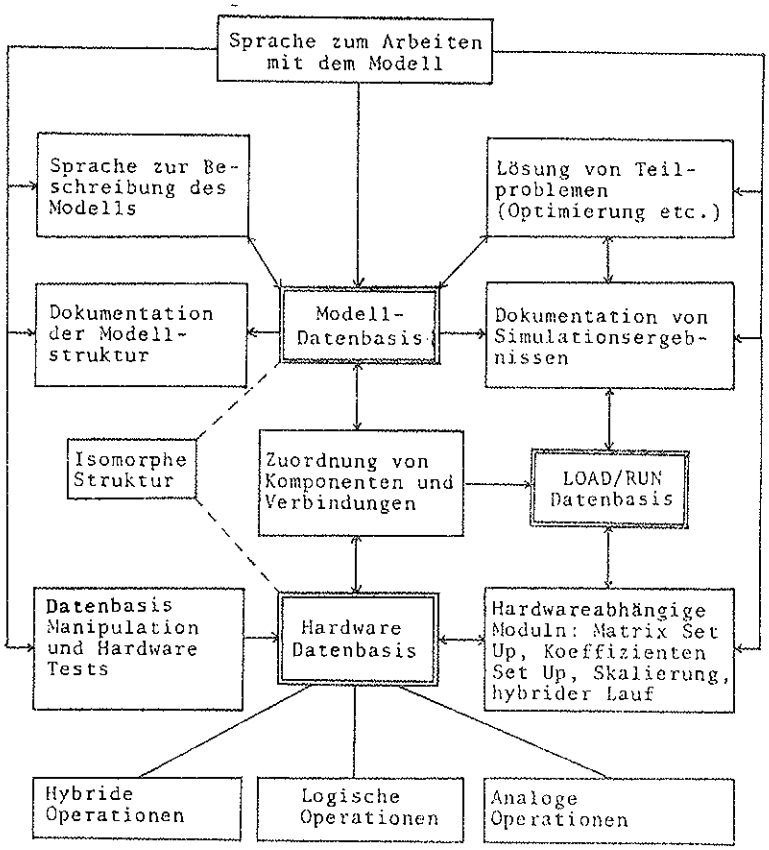


Abbildung 5
HYBSYS Systemarchitektur

Den verschiedenen Benutzern werden nicht feste Zeitscheiben, sondern die gesamte geforderte hybride Rechenzeit (d.h. die OP-Zeit für die Integrierer) zur Verfügung gestellt. Die Benutzeranforderungen nach einem hybriden Simulationslauf (die implizit entweder von einem Dokumentationsbefehl oder einem anderen Befehl, der Resultate der Simulation benötigt, erstellt werden) werden in der Reihenfolge ihres zeitlichen Eintreffens in Warteschlangen gestellt und je nach der Dauer gerade laufender Aktivitäten abgearbeitet, wobei darauf geachtet wird, daß mehrere Anforderungen eines Benutzers nur dann hintereinander bedient werden, wenn er gerade der einzige ist, der solche Anforderungen stellt (keine Prioritäten!). Die Zeit, die für einen Benutzerwechsel benötigt wird, hängt von der Größe und dem Teil der LOAD/RUN-Datenbasis ab, die bei einem Wechsel neu geladen werden muß. Um diese Zeit möglichst kurz zu halten, ist die LOAD/RUN-Datenbasis in drei Informationsgruppen unterteilt (siehe auch Tab.1):

PREPARE-Daten: Sie beinhalten die Daten für das Set-Up der analogen Schaltmatrix und des Logikprozessors und müssen bei jeder qualitativen Änderung des aufgebauten Modells neu abgesetzt werden.

SET-Daten: Sie enthalten im wesentlichen die analogen und logischen Koeffizienten der Schaltung und müssen bei jeder quantitativen Änderung am Modell neu abgesetzt werden.

RUN-Daten: Sie beinhalten Informationen über die aus dem Hybridlauf benötigten Daten. Diese Datei wird bei jedem Lauf abgesetzt und am Ende des Laufs mit den gewünschten Daten aus dem Hybridlauf beschrieben.

Aufgrund dieser Struktur der LOAD/RUN-Datenbasis ergeben sich unterschiedliche Zeiten für einen Benutzer- bzw. Problemwechsel, die aber unabhängig von der Modellgröße sind. Folgende Zeiten wurden experimentell ermittelt:

Modellwechsel (=Userwechsel)	25 ms	ohne Logik, HYBMEM
	150 ms	mit Logik
	250 ms + n*5ms	mit Logik und HYBMEM
Parameteränderung (kein Userwechsel)	5 ms	
Laufwiederholung (kein Userwechsel)	1 ms	

Bei den angeführten Zeiten ist zu bedenken, daß aus Speicherplatzgründen sowohl die Informationen für die PBP-Wahrheitstabellen als auch die Werte von ADCF- bzw. DACF-Variablen in virtuellen Feldern auf der Platte stehen und bei Bedarf jedesmal Massenspeichertransfers durchgeführt werden müssen.

Die Version 5.1 TS von HYBSYS enthält nicht nur die eben beschriebenen Änderungen für den Time-Sharing-Betrieb, es wurden auch entsprechende Neuerungen und Verbesserungen eingebaut, die die Modellbeschreibung, das rasche Darstellen von skalierten Variablen am graphischen Terminal (Ersatz für Speicheroszillographen), die verkürzte Behandlung indizierter Variabler, das rekursive Aufrufen von Overlays bis zu einer Tiefe von 10 und anderes betreffen. Alle Änderungen zur letzten Version 4 M/A sind im neuesten HYBSYS-Bulletin beschrieben, der aktuelle Stand ist im neuen HYBSYS User Manual dokumentiert [7].

Schon mit der AutoPATCH-Erweiterung, die 1980 als HYBSYS 4 M/A installiert wurde, verschwand die Hardware aus der Benutzerebene, und das Hybridsystem konnte als Black-Box mit den Eigenschaften eines Parallelprozessors mit lokaler Festkommaarithmetik zur Lösung von auf Differentialgleichungen beruhenden Simulationsmodellen angeboten werden. Die Bedienung erfolgt nur mehr von graphischen Terminals als Simulationskonsolen aus.

Um einen Mehrbenutzerbetrieb im Time-Sharing-Verfahren zu ermöglichen, mußten alle Hardwarezugriffe auf Betriebssystemebene verlegt werden. Anstelle der bisher erfolgten direkten Hardwarezugriffe wird nun für jeden einzelnen Benutzer eine LOAD/RUN-Datenbasis beschrieben.

In der derzeit implementierten Version 5.1 übernehmen verschiedene, für jeden Benutzer parallel laufende Kopien von HYBSYS sowohl die Mehrbenutzerverwaltung mit Hilfe von betriebssystemunterstützten Semaphoren als auch das Absetzen der jeweiligen LOAD/RUN-Datenbasis nach Beendigung einer eventuell laufenden hybriden Aktivität an die AutoPATCH-Hardware und das Abspeichern gewünschter Daten am Ende des Simulationslaufes.

Der derzeit implementierte einfache First-Fit-Algorithmus für das Time-Sharing-Verfahren ist in Abb.6 als mögliches Zeitablaufdiagramm dargestellt.

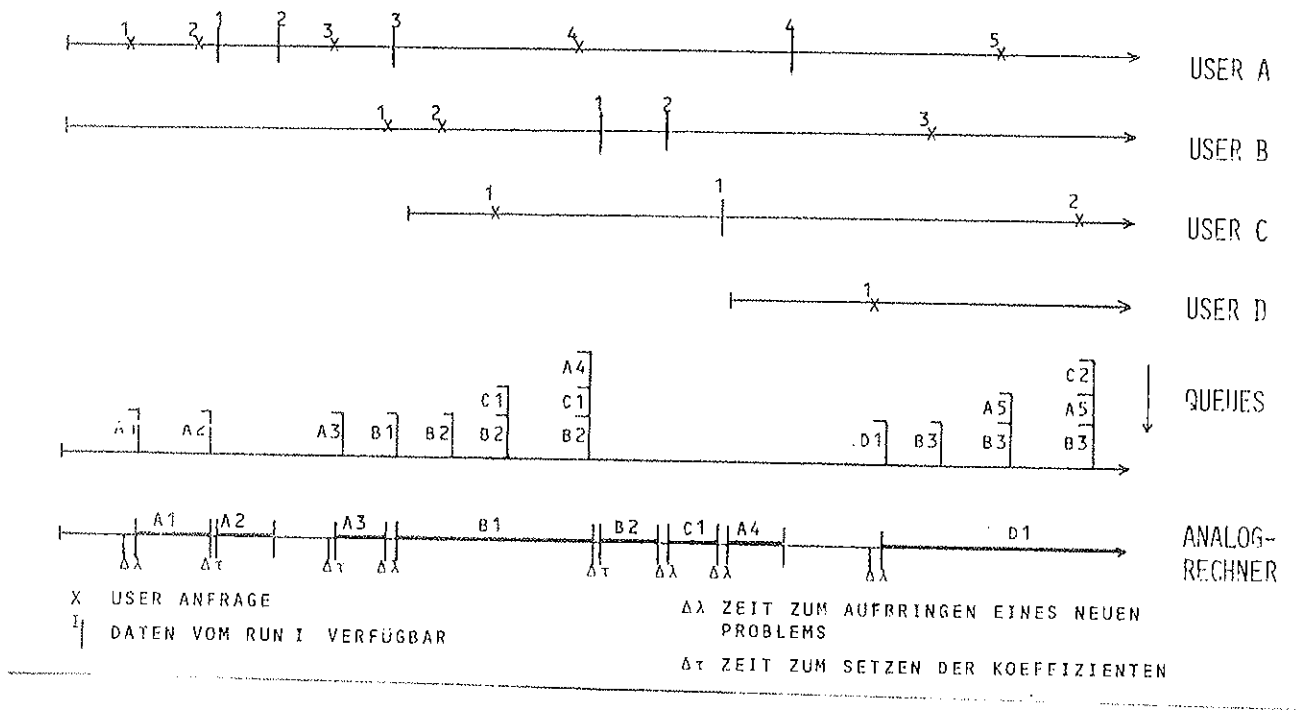


Abbildung 6
Zeitablaufdiagramm

LOAD/RUN DATENBASIS

	WORTE à 16 BIT
PREPARE DATEN	4928
Matrix Setup	
Matrix 1 Route Information	320
Matrix 2 Route Information	320
PBP Setup	
PBP Route Information	192
PBP Truth Tables	4096
SET DATEN	427 (ohne Hybrid Memories)
Coefficient Setup	
DCA's	152
CDAC's	64
DAC's	20
Logic Control Setup	
Controllines	13
PBP Input Inversion	8
PBP Counter	34
FF-IC	2
Function Setup	
DCFG	128
Hybrid Memories	n*1024 max.
Timer Setup	6
RUN DATEN	2039
Display Selection	3
Readout Information	
Display Variable	2024 max.
Terminal Values	7
Scaling Information	5

Tabelle 1

6. AUSBLICK

In der derzeit implementierten ersten Version des hybriden Time-Sharing-Systems MACHYS steht unseren Benutzern eines der weltweit fortgeschrittensten Hybridrechensysteme zur Verfügung. Daß zur Zeit nicht mehr als 3 Benutzer gleichzeitig mit HYBSYS simulieren können, liegt am beschränkten Hauptspeicherplatz. Mit der für Anfang 1983 geplanten Erweiterung können dann sofort 8 Time-Sharing-Benutzer gleichzeitig rechnen. Ferner sollen alle Hardwareabhängigkeiten direkt ins Betriebssystem verlagert werden. Das wird zusammen mit einer geplanten neuen virtuellen Arraytechnik einen schnelleren, interruptgetriebenen Problemwechsel ermöglichen. Im nächsten Jahr sollen auch alle oben erwähnten, bisher erst konzipierten Hardware-Teilsysteme (Skalierungshardware, HYFUs, PBP-Auslesesystem und Simulationskonsolen) realisiert, getestet und eingebaut werden. Auf dem Gebiet des Time-Sharing selbst sollen andere Mechanismen

(z.B. Round Robin Algorithmus und feste Zeitscheiben) getestet werden. Als nächste Stufe von HYBSYS ist eine weitgehend ACSL syntaxkompatible Deklarationssprache geplant, die endgültig alle hardwarebezogenen Teile der Sprache eliminieren soll. Weiters soll eine Methodenbank zur Integration angeschlossen werden, durch die neben dem analogen auch mehrere digitale Integrationsverfahren zur Verfügung stehen, die einen unmittelbaren Vergleich mit der analogen Integration ermöglichen.

LITERATUR

1. Kleinert W., Berger F., Stallbaumer H. und Wittek E.: Das hybride Time-Sharing-System MACHYS an der Technischen Universität Wien. Simulationstechnik. Proceedings des 1. Symposiums Simulationstechnik in Erlangen, April 1982. Herausgeber: M. Goller. Informatik Fachberichte Nr. 56, Springer Verlag, 1982.

2. Solar D. und Berger F.: Interaktive Simulationssoftware für ein virtuelles hybrides Mehrbenutzersystem. Simulationstechnik. Informatik Fachberichte Nr. 56, Springer Verlag, 1982.

3. Kleinert W. und Wittek E.: Der programmierbare Logikprozessor für das hybride AutoPATCH System an der Technischen Universität Wien. Simulationstechnik. Informatik Fachberichte Nr. 56, Springer Verlag, 1982.

4. Kleinert W.: Die neue AutoPATCH-Hardware. Interface 15/16, Juni 1980.

5. Kleinert W. und Wittek E.: Der Pipelined Boolean Processor als Ersatz für die parallel Logik des EAI 680 Analogrechners. Interface 18, Jänner 1982.

6. Blauensteiner A.: Neues Betriebssystem JCS/VS 8. Interface 17, April 1981.

7. Solar D.: HYBSYS User Manual, Version 5 TS. Hybridrechenanlage, Technische Universität Wien, November 1982.

EAI COMPUTER USERS' GROUP MEETING 1982

An der Universität von Ljubljana trafen am 21. September 1982 Teilnehmer aus zehn verschiedenen Ländern zum EAI Computer Users' Group Meeting 1982 ein. Zunächst wurde die Möglichkeit geboten, das Rechenzentrum der Universität zu besichtigen. Das Meeting wurde durch Prof. Bremšak eröffnet. Am Abend fuhren alle Teilnehmer zum eigentlichen Veranstaltungsort, nach Portorož.

Während der nächsten zwei Tage wurden 17 technische Papers vorgetragen. Wie beim letzten Meeting wurde an beiden Tagen die Möglichkeit geboten, an Working Groups teilzunehmen. Es wurden Themen wie ECSSL, EAI-PDP-Connections, Hard- und Software behandelt, wobei auch die Antworten, die von EAI auf die im letzten Jahr ausgearbeiteten Fragen gegeben wurden, diskutiert wurden. In zwei EAI Sessions berichteten Mitarbeiter von EAI über neue Produkte und es gab Diskussionen zwischen EAI und den Mitgliedern der Users' Group.

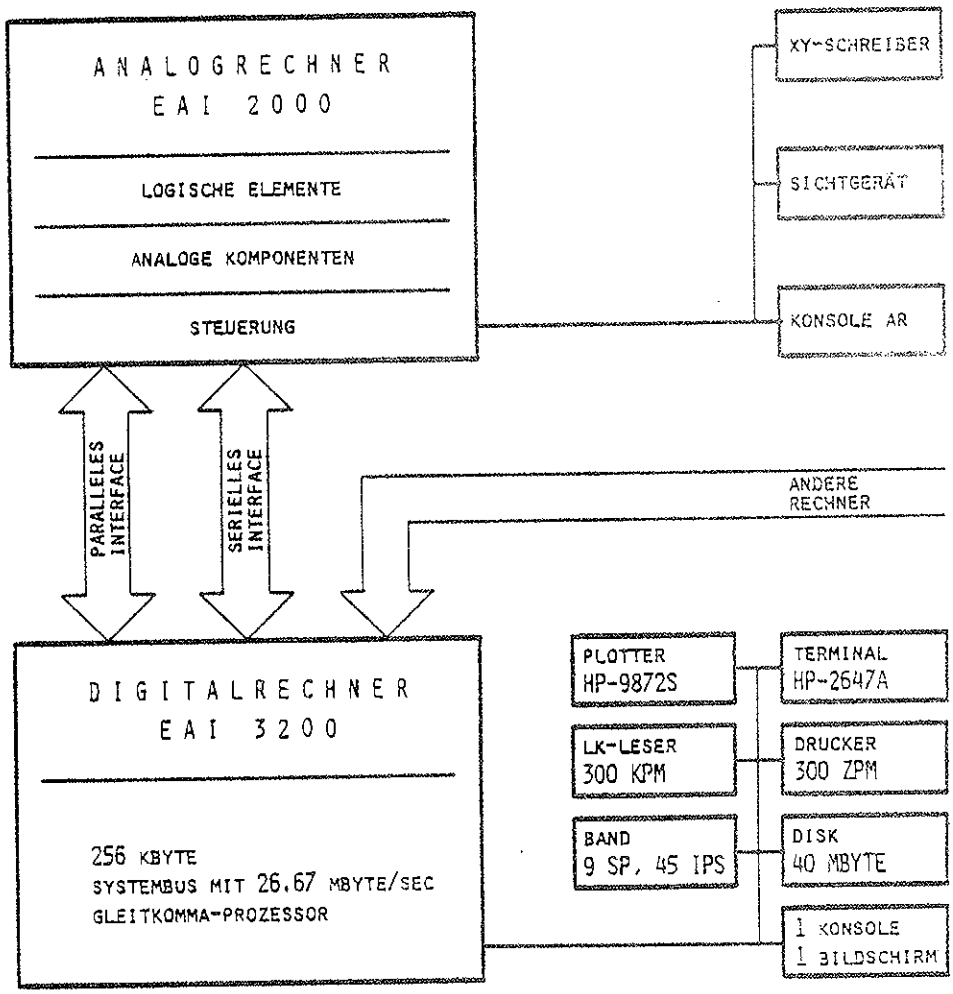
Von Mitarbeitern der Hybridrechenanlage der TU Wien wurden Vorträge über die Entwicklung des FORTRAN 77 Compilers FTU (A. Blauensteiner) und über das hybride Time-Sharing-System MACHYS sowie über die Hardware-Entwicklung an der Hybridrechenanlage (W. Kleinert) gehalten.

EDV-Zentrum der Technischen Universität Graz

Hybridrechenanlage

F. Haselbacher

Das EDV-Zentrum der Technischen Universität Graz hat zur Verbesserung der EDV-Einrichtungen für spezielle technisch-wissenschaftliche Anwendungen einen modernen Hybridrechner installiert und in Betrieb genommen.



Dieser Rechner wird in Zusammenarbeit mit dem Institut für Elektromagnetische Energieumwandlung betrieben und kann von allen Instituten und Einrichtungen der Technischen Universität benützt werden.

Aufstellungsort: Institut für Elektromagnetische Energieumwandlung, Kopernikusgasse 24, 1.Stock, Seite Stremayrgasse, Rechnerraum

Benützung: In allen Fragen der Benützung, der Organisation und des Betriebes wenden Sie sich bitte an einen der folgenden Herren:

Univ.-Prof. Dr. M. Rentmeister	(7244)
Dipl.-Ing. G. Walzel	(7247/7248)
Dr. J. Theurl	(6390)
Dipl.-Ing. F. Haselbacher	(6393)

KONFIGURATION DES HYBRID-RECHENSYSTEMS EAI-2000/3200 (Okt.82)

PROZESSRECHNER EAI-3200-77

1 Grundgerät EAI-3200-77/FP mit 256 KByte MOS-Speicher und Gleitkomma-Prozessor
1 Magnetplatteneinheit, Modell 9521-2, 40 MByte
1 Magnetbandeinheit, Modell 9563-2
1 Bildschirm-Sichtgerät, Modell 9203 - CONSOLE
1 Kartenleser, Modell 9210-2
1 Zeilendrucker, Modell 9225-2
1 Asynchrones Dateninterface, Modell 9122
1 Standard-Digitalsoftware (Real Time Monitor, Assembler, Fortran IV, ...)
1 Grafik-Terminal, Modell HP-2647A
1 Vierfarben-Plotter, Modell HP-9872S
1 Display-Terminal, Modell Hazeltine 1600

HYBRID INTERFACE

1 Serielles Interface, Modell 2.2082
1 Daten-Interface, Modell 2.2297
1 Analog/Digitalwandler mit 16 Multiplexer-Kanälen, 13 bit + Vorzeichen: 85 KHz
16 Digital/Analog-Multiplizierer
8 Sense Lines
8 Control Lines
8 Interrupt Lines
1 Standard-Hybrid-Software (Fortran)

ANALOGTEIL

1 Grundgerät EAI-2000
11 Integrierer
5 Integrierer mit Limiter
3 Summierer mit Track-Store und Limiter
5 Summierer mit Limiter
24 Summierer
10 Multiplizierer/Dividierer
29 Elektronische Schalter
5 Digital einstellbare Funktionsgeneratoren (DSFG)
16 Potentiometer, manuell
21 Potentiometer, digital
12 Komparatoren
6 Flip-Flops
9 Gatter
3 Differentiatoren
3 Binär-Zähler
2 8-bit-Register
1 Matrix-Printer-Terminal ZIP 30/ASR

ANALOGE PERIPHERIE

1 XY-Schreiber
1 Vierkanal-Sichtgerät

MASSENSPEICHER-SAVE

I. Husinsky

Die Massenspeicherkapazität an der Hybridrechenanlage beträgt zur Zeit etwa 13 Millionen Worte auf 8 Drives. Das Betriebssystem JCS/VS 8 unterstützt 50 virtuelle Massenspeicher (Districts), die den physikalischen Massenspeichern zugeordnet sind. Den einzelnen Districts sind im laufenden Betrieb bestimmte Aufgaben zugeteilt (z.B. Line Printer Spool District, Library District, etc.). Bestimmte Districts sind zum Abspeichern von Benützerprogrammen und -datenbeständen vorgesehen. Durch die vorhandenen Wechselplatten können noch weitere Kapazitäten zum Einsatz kommen. Es sind etwa 30 Platten in Verwendung. Alle Magnetbänder und Wechselplatten werden in einem flammsicheren Schrank im Rechenraum aufbewahrt.

Zum Schutz der vorhandenen Massenspeicherinformation werden zwei Programme eingesetzt, die den Inhalt von Platten auf Magnetband kopieren und das Rückkopieren der Information ermöglichen.

Das Programm BACKUP überträgt den Inhalt sämtlicher gerade im Betrieb aktiven Massenspeicher in komprimierter Form auf Magnetband (2 Bänder, keine Verifizierung). BACKUP läuft zweimal täglich, wobei eine mehrstufige Save-Strategie eingehalten wird: das älteste Backup-Band enthält Informationen, die mindestens einen Monat alt sind, die jüngste Information ist höchstens einen halben Tag alt. Die auf Backup-Band befindliche Information steht zu verschiedener Verwendung zur Verfügung: die Software unterstützt das Wiederherstellen eines ganzen Drives oder eines Districts (das Wiederherstellen von einzelnen Files ist in Arbeit).

Das Programm CDRIVE kopiert einen Drive auf einen beliebigen anderen Drive, dieser kann auch die Magnetbandeinheit sein. Hier kann auch ein Verify-Pass durchgeführt werden. Die so erhaltene Kopie kann selbst auf einem Drive eingesetzt werden (Wechselplatte) oder wieder zurückkopiert werden. Diese Kopien werden nur auf Anordnung gemacht. Benützer, die eigene Wechselplatten verwenden und eine Bandkopie ihrer Platte wünschen, können dies mit dem Operator vereinbaren. Da diese Benützerplatten nicht automatisch gerettet werden, empfehlen wir den Benützern, regelmäßig eine Kopie von ihren Platten anfertigen zu lassen.

F T U - FORTRAN 77 COMPILER

A. Blauensteiner

PROZESSOREN AN DER HYBRIDRECHENANLAGE

An der Hybridrechenanlage gibt es nur einen FORTRAN IV Compiler, der für das selbstentwickelte Betriebssystem JCS/VS 8 adaptiert und erweitert wurde. Dieser relativ alte FORTRAN IV Compiler ist der einzige digitale Sprachprozessor an dem PACER 1ØØ Digitalrechner. Dieser Compiler kann aber nicht mehr gewartet und auch nicht mehr verbessert werden.

Immer mehr wurde die Frage nach einem leistungsfähigen FORTRAN 77 Compiler gestellt. Die größten Schwierigkeiten für die Programmentwicklung an der Hybridrechenanlage bereiten die Beschränkung eines Programmes auf 32 K sowie die mangelnde Exekutionsgeschwindigkeit, besonders bei Gleitkommaoperationen.

GRÜNDE FÜR DIE ENTWICKLUNG EINES NEUEN COMPILERS

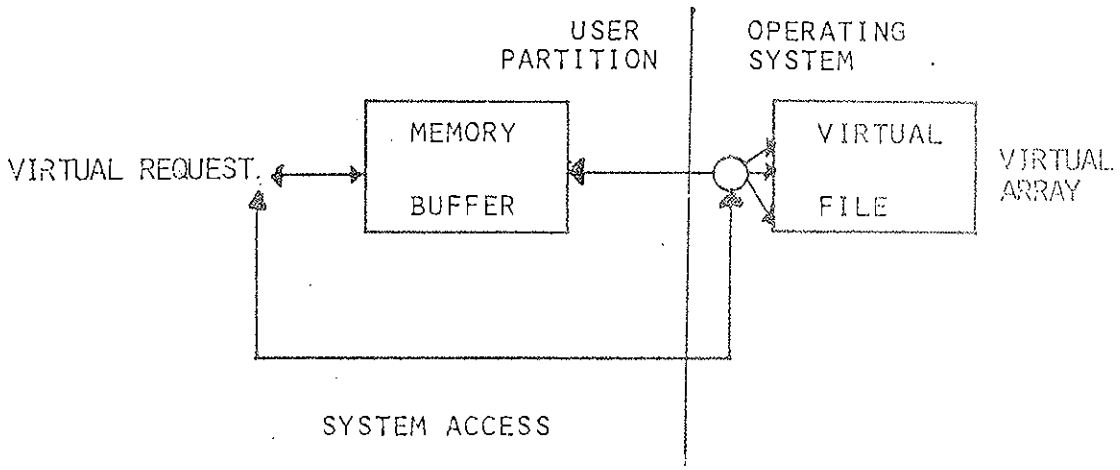
Es gab verschiedene Gründe für die Eigenentwicklung eines neuen FORTRAN 77 Compilers. Einerseits wird FORTRAN 77 weltweit als neuer Standard anerkannt und daher von den Benützern verlangt. Ausschlaggebend für die Eigenentwicklung waren jedoch eine Reihe von notwendigen Erweiterungen, welche auf jeden Fall für den momentanen FORTRAN Compiler geplant waren. Diese Erweiterungen hätten vermutlich ebensoviel Arbeit in Anspruch genommen wie die Neuentwicklung selbst. Hauptgesichtspunkt für den Compiler FTU waren die Generierung eines optimalen Codes bezüglich der Exekutionsgeschwindigkeit des Programmes und die Ausnützung der neuen Gleitkommahardware, des Coprozessors (COP).

Der COP hat eine Reihe von PACER-Instruktionen, welche es ermöglichen, Floating-Point, Complex, Double Precision und einige Integer-Operationen direkt im Maschinencode abzusetzen. Der Code ist typparametrisiert, das heißt, die gleiche Instruktion kann in Abhängigkeit vom Datentyp variabel kodiert werden. Diese Spezifikationen der Gleitkommahardware wurden mit dem FTU-Projekt abgestimmt und erlauben daher einen optimalen Compilecode für die Zuweisungen. Außerdem hat der COP einige Instruktionen, die Character Strings direkt manipulieren. Der Compiler FTU wurde so entwickelt, daß er in zwei Durchläufen des Sourceprogrammes einen symbolischen Output erzeugt, welcher vom Assembler weiter prozessiert wird. Im Zusammenhang mit der Möglichkeit, über einen Metaassembler rechnerunabhängige Programme zu entwickeln, gewinnt der Compiler FTU dadurch an Bedeutung, daß Mikroprozessoren direkt vom PACER mit FORTRAN-Programmen programmiert werden können. Dadurch erspart man sich neben einem Entwicklungssystem auch diverse Peripherie für die Mikroprozessoren.

EIGENSCHAFTEN DES COMPILERS FTU

Neben einer Reihe von Verbesserungen bringt FORTRAN 77 vor allem die Möglichkeit der Verzweigung nach IF mit THEN und ELSE, sowie den CHARACTER-Typ für String-Management. Außerdem können für Arrays untere und obere Grenzen angegeben werden, die Datentypen in arithmetischen Ausdrücken können frei gemischt werden, und die Schrittgröße in DO-Schleifen ist variabel.

Daneben gibt es eine Reihe von speziellen Eigenschaften. Die wichtigste dieser Erweiterungen zu FORTRAN 77 ist der virtuelle Variablentyp VIRTUAL. Jedes Variablenarray kann mit VIRTUAL deklariert werden. Dies bedeutet, daß der Zugriff zu Elementen dieses virtuell deklarierten Feldes nur über einen Zwischenbuffer im Memory erfolgt. Der eigentliche Datenbestand wird auf einen File gespeichert. Den virtuellen Zugriff zu einem solchen Arrayelement bewerkstelligt das Betriebssystem.



Virtueller Zugriff

Dadurch wird es möglich, mehrere große Datenfelder in einem Programm zu verwenden. Die Beschränkung auf 32 K pro Programm ist dadurch nicht mehr so gravierend. Es können beliebig viele Variablen mit VIRTUAL deklariert werden. Die VIRTUAL-Option kann Variablen beliebigen Typs gegeben werden. Außerdem können virtuelle Variablen in COMMON-Blöcken und als Unterprogrammargumente weitergegeben werden.

Zwei weitere spezielle Erweiterungen im Compiler FTU sind die Befehle LABEL und REWIND, die das Markieren von bestimmten Positionen auf Systemfiles (Inputfile bzw. Spoolfile) ermöglichen und das spätere Positionieren dieser Marken ermöglichen. Statement-Functions haben keinerlei Beschränkungen; so ist es zum Beispiel möglich, auch Felder als Argumente von Statement-Functions zu verwenden.

Der Compiler FTU ermöglicht dem Benützer eine optimale Dokumentation seines Programmes: jede Zeile wird durchnumeriert und zwar so, daß die Zeilennummern der Liste mit der des Sourcefiles korrespondieren. Alle Fehlermeldungen werden benützerfreundlich im Klartext unmittelbar nach Auftreten des Fehlers ausgegeben. Alle Seiten erhalten einen Seitenkopf mit Datum, Uhrzeit und den wichtigsten Informationen über das übersetzte Programm. Nach jedem Programmteil erhält der Benützer eine strukturierte Liste mit detaillierten Informationen über Argumente, Variablen, Konstanten, externe Referenzen (Unterprogramme), Be-

fehlsnummern und COMMON-Blöcke (Map-Liste). Außerdem kann auf Wunsch der vom Compiler generierte symbolische Code ausgegeben werden.

```
FORTRAN COMPILER  FTU.0/8.04  FTU08   39   82/09/17 12.19       PLAY      PAGE 024

SUBROUTINE  PLAY

ARGUMENTS:

IWZ      N1        A1        N2        A2        .VBD

VARIABLES:

IWZ      I              10                        N1       I       S       1
A1       R      TS       1                        N2       I       S       1
A2       R      TS       1                        DOU      0       T       0
P        R C    T        0        8        8     10 ST       I C     T       0      27       2      16
KZ       I C    T        0       16                IV       I C     T       0      16
JV       I C    T        1       16                ZUG      I       TS     33     100       5       4
FV       R      TS       1        7                FEX      L       T       5
ENDE     L   F  T        2                         OPEN     L   F   T       1
KX       I C             0                         IP       I C             5
ISCH     I C             0                         IFIN     I C             0
LWZ      I C             0                         NZUG     I C             0
KZUG     I C             0                         NKILL    I C             0
NFORCE   I C             0                         KFV      I C             0
KILVAL   I C             0                         IROCH    I C             0
NSTEPS   I C             0                         LIMES    I C             4

CONSTANTS:

.CAA    0                0 23.430-11               .CAB     I               0       1
.CAC    I                0  32767                  .CAD     I               0       3
.CAE    I                0      0                  .CAF     I               0       2
.CAG    I                0      5                  .CAH     I               0     601
.CAI    I                0    100                  .CAJ     I               0       4
.CAK    I                0     11                  .CAL     I               0       7
.CAM    I                0     10                  .CAN     I               0       9

EXTERNALS:

ENDE     OPEN     ORDNE    BEST     TRACE    LEVEL

LABELS:

   99       5          63       1          64       1          65       2
   67       1          66       1          32       1          22       3
    2       2           1       1         439       1          68       1
   90       1          91       2          92       1          62       0
    4       1          10       2
```

Beispiel für eine Map-Liste

Weiters gibt es ein komfortables Trace-System im Compiler FTU: durch Angabe spezieller Options kann ein Trace-Punkt im Programm kodiert werden, der übersichtlich die wichtigsten Informationen an dieser Stelle am Terminal ausgibt und interaktiv die Fortsetzung des Programmes, den Abbruch oder das Umgehen der nächsten Traces ermöglicht. Diese Trace-Punkte wirken zur

Exekutionszeit des Programmes und können an folgenden Punkten wahlweise gemischt vorgesehen werden: innerhalb einer FORTRAN-Zeile an bestimmten Punkten arithmetischer Ausdrücke, vor Beginn jeder FORTRAN-Zeile, vor Beginn jeder mit einer Befehlsnummer versehenen FORTRAN-Zeile, vor dem Einsprung in ein Unterprogramm und bei jedem Durchlaufen einer DO-Schleife. Die Informationen am Trace-Punkt richten sich nach der Art des Trace-Punktes.

Außerdem gibt es eine Reihe weiterer nützlicher Optionen: es können auf Wunsch speziell markierte Kommentarzeilen zur Compilierung gebracht werden. Es können die Indizes von Variablen-Arrays überprüft werden und zwar entweder zum Compilezeitpunkt oder zum Zeitpunkt der Exekution des Programmes. Auf Wunsch kann eine Cross-Reference-Map der Variablen ausgegeben werden. Ebenso können arithmetische Ausdrücke mathematisch reduziert werden und Redundanzen von arithmetischen Ausdrücken berücksichtigt werden. Variablenfelder können vor Beginn des Programmes auf Null gesetzt werden und es kann überprüft werden, ob eine Variable vor ihrer ersten Verwendung einen zugewiesenen Wert besitzt.

DIE DURCHFÜHRUNG DES PROJEKTS

Das Projekt FTU wurde für drei Monate geplant und konnte auch in diesem Zeitraum durchgeführt werden. Während dieser Zeit arbeiteten acht Personen an diesem Projekt. Der Arbeitsaufwand wurde in mehrere Teilprojekte unterteilt. Über diesen Teilprojekten stand nur die allgemeine Analyse und Koordination dieser Teilprojekte, die in ihrer Detailanalyse und Durchführung voneinander unabhängig waren. Sie richteten sich nach wohldefinierten Schnittstellen im Aufruf und in der Funktion und nach den allgemein gültigen Strukturen und Formaten, welche generell definiert wurden. Die Prinzipien des Compileprozesses und der Aufbau des Compilers wurden global festgelegt und überwacht.

In der ersten Phase des Projekts wurde ein Organisationsrahmen des Compilers entwickelt, welcher alle Hilfsfunktionen für Input/Output, File-Management sowie für den Compileprozeß zur Verfügung stellte. Auf diesen Organisationsrahmen aufbauend wurde der zweite Teil des Projekts begonnen: die eigentliche Implementierung der verschiedenen Compilefunktionen. Alle Teilprojekte wurden sofort dokumentiert und die Informationen darüber den einzelnen Mitarbeitern des Projekts weitergegeben. Bevor die Implementierung der eigentlichen Compilefunktionen begann,wurde der Rahmen aller Hilfs- und Organisationsroutinen absolut getestet, um die Arbeit der einzelnen Mitarbeiter untereinander nicht zu behindern. So konnte jeder Mitarbeiter mit einer wohldefinierten Zwischenrelease einen weiteren Modul entwickeln. Zu diesem Zeitpunkt arbeiteten die anderen Mitarbeiter an ihren Moduln, ohne daß sie von der Entwicklung der anderen gestört werden konnten. Nach der Fertigstellung und dem Testen eines solchen Moduls wurde dieser in einer neuen Zwischenrelease freigegeben und konnte von allen Mitarbeitern für die Weiterarbeit bereits verwendet werden. Die Entwicklung stieß auf keine besonderen Schwierigkeiten und es ist zu erwarten, daß der FORTRAN 77 Compiler FTU zusammen mit der COP-Hardware 1983 für die Benützer der Hybridrechenanlage zur Verfügung stehen wird.

BEISPIEL FÜR DIE CODIERUNG EINES FORTRAN-BEFEHLS

```
84     439 IF(.NOT.ENDE(IWZ,JL+1,IKHE)) FV(LKL-2)=3*JV(LKL-2)

      1059  L00439 EQU     *
      1060  .TAU   CALL    .TRAL   }
      1061         DEC     00084   }  LABEL TRACE
      1062         IMV     00001   }
      1063         ADR     JL      }
      1064         ADR     .VAC    }  .VAC=JL+1
      1065         CALL    ENDE    }
      1066         ADR,I   IWZ     }
      1067         ADR     .VAC    }  CALL ENDE
      1068         ADR     IKHE    }
      1069         ADR     0       }
      1070         OCA             }
      1071         SKN             }  IF
      1072         J       .TAV    }
      1073         DMV     00002   }
      1074         ADR     LKL     }  .VAU=LKL-2
      1075         ADR     .VAU    }
      1076         DMV     00001   }
      1077         ADR     .VAU    }  FV  }
      1078         ADR     .VAY    }      }  ARRAY BASEN
      1079         DMV     00001   }      }
      1080         ADR     .VAU    }  JV  }
      1081         ADR     .VBB    }
      1082         XLD     0       }
      1083         ADR     JV      }
      1084         ADR     .VBB    }  JV(LKL-2)
      1085         STX     .VBD    }
      1086         XLD     1       }
      1087         ADR     FV      }
      1088         ADR     .VAY    }  FV(LKL-2)
      1089         STX     .VBF    }
      1090         RMUL    0       }
      1091         ADR     .CAD    }  3.*
      1092         ADR,I   .VBD    }
      1093         FLOAT   0       }
      1094         STR     1       }  FV=
      1095         ADR,I   .VBF    }
```

METAASSEMBLER

A. Blauensteiner

Im letzten INTERFACE wurde der Metaassembler METASM, welcher an der Hybridrechenanlage implementiert ist, vorgestellt. Mittlerweile wurden dessen Eigenschaften beträchtlich erweitert, sodaß nun von der Syntax her ein leistungsfähiger, maschinenunabhängiger, instruktionsorientierter Compiler mit variabler Wortlänge bis 128 Bits zur Verfügung steht.

Durch eine allgemeine symbolische Beschreibung von Bitketten, sogenannten Frames, kann ein bestimmter Instruktionssatz symbolisch definiert werden. Durch geeignete Wahl von Bitgruppen kann so in Zusammenarbeit mit dem Source Prozessor ein einfaches Arbeiten mit dem Metaassembler erreicht werden. Neben einem übersichtlichen Dokumentationsteil ist nun auch die Eingabe erleichtert: es sind Fortsetzungszeilen möglich, und die Liste der assemblierten Instruktionen wird in ihrem Gesamtbild je nach maximal verwendeter Bitlänge gestaltet. Neben der Möglichkeit, die definierten Frames nunmehr auch selbst wieder durch Frames zu definieren, und der Möglichkeit, mehrere voneinander unabhängige Programmteile zu übersetzen und ein entsprechendes Objekt zu erzeugen, seien hier zwei besondere Eigenschaften herausgestrichen: die Macros und das Affixing.

MACROS

Es können neben Frames, welche beliebige Bitketten repräsentieren und an sich noch keine Instruktionen bilden, und neben Metas, welche die eigentlichen Instruktionen definieren, auch Macros definiert werden. Jeder Macro besteht aus einer beliebigen Anzahl von Metas, also abgeschlossenen Instruktionen, welche wiederum unterschiedliche Wortlänge haben dürfen. Der Macro kann mit einem Parameter versehen sein, welcher dann sinngemäß in den Instruktionen bei deren Assemblierung eingesetzt wird.

Die Dokumentation eines Macros besteht aus einer Liste aller definierten Instruktionen pro Macro. Die eigentliche Assemblerliste besteht bereits aus den durch den Macro definierten Instruktionen.

AFFIXING

Jede Instruktion kann definitiv einen Parameter enthalten, das heißt, im allgemeinen eine referenzierte Adresse (Tag) oder einen instruktionsspezifischen konstanten Wert. Daneben kann jede Instruktion auch ein Affix enthalten, welches zum Unterschied von einem Parameter eine variable Interpretation dieser Instruktion erst zum Zeitpunkt der Erstellung des Assemblerprogrammes ermöglicht. Jede Instruktion kann höchstens ein Affix enthalten, welches, durch Beistrich getrennt, an den symbolischen Instruktionsnamen angehängt wird. Dieses Affix ist syntaktisch ein definiertes Frame und wird bei der Erstellung eines Assemblerprogrammes codiert. In diesem Fall wirken in der definierten Instruktion die Bits des Frames in der Zone der definierten Affix-Bits. So kann in einer bestimmten Instruktion in dieser Affix-Bit-Zone je nach Verwendung eines bestimmten Affixes (Frame) ein anderer Bitaufbau gesetzt werden. Dabei kann auch auf einen Parameter der Instruktion Bezug genommen werden, genauso können Macros mit Affix versehen werden. Dadurch ergeben sich sehr dynamische Programmierungsmöglichkeiten, weil z.B. in einem

mit Affix versehenen Macro dieses Affix eine ganze Instruktion darstellen könnte, die dann sozusagen parametrisiert dem Macro übergeben wird und daher je nach Verwendung variabel wäre:

```
UP              FRAME       16    LLHL LHLL LLLL LLLH
J+3             L      1     4    LHLL LLHL LLLL LLHH
J-4             L      1     4    LHLL LLHH HHHH HHLL
DN              FRAME       16    LLHL LHLH HHHH HHHH
COD             L      1     4    FFFF FFFF FFFF FFFF
LINK     V      L      1     4    LHHL LLHD DDDD DDDD
DO       V          MACRO   5    COD     SKU     J+3     LINK    J-4
```

Auszug aus den Instruktionsdefinitionen:
L reserviert ein "LOW"-Bit, H ein "HIGH"-Bit, D ein Bit eines Displacements zu der angesprochenen Adresse der Instruktion "LINK", F reserviert Bits für ein Affix, d.h. diese Bits werden erst zum Assemblierungszeitpunkt mit den Definitionsbits des Frames besetzt, welches die Instruktion affixiert. So ist COD eine Pseudoinstruktion, die nur aus Affix-Bits besteht, und bewirkt im Macro DO, welcher aus den Instruktionen COD, SKU, ... gebildet wird, das Einsetzen des Frames UP oder DN anstelle von COD, je nachdem mit welchem DO affixiert ist.

```
        META:   PACER
*
*       DO FOR CNT1 ( DO FOR CNT2 ( * ) TIMES ) TIMES
*
        LX      CNT1    OUTER LOOP COUNT
        DO,UP   FUN/A
        JCS             END
*
*       CNT2 LOOP
*
FUN/A   ADR     @       OUTER ROUTINE
        STX     HELPX   SAVE X
        LX      CNT2    INNER LOOP COUNT
        DO,DN   FUN/B
        LX      HELPX   RESTORE X
        J,I     FUN/A   RETURN
*
*       FUNCTIONAL PROCEDURE
*
FUN/B   ADR     @       INNER ROUTINE
*                       SOME CALCULATIONS
        J,I     FUN/B   RETURN
*
*       DATA + LOOP COUNTER
*
HELPX   NOP             TEMPORARY X
CNT1    DEC     32599   OUTER COUNT
CNT2    DEC     1324    INNER COUNT
        END
```

Metaassemblerprogramm:
Liste eines Programmes, wie es codiert wird. Mit META: wird ein bestimmter Meta-Instruktionssatz geladen (PACER). Dieser bestimmt die folgende Assemblierung. Der Macro DO wird zweimal aufgerufen, einmal mit dem Affix UP, um zu inkrementieren, einmal mit dem Affix DN, um zu dekrementieren. Definitiv arbeitet der Macro DO ein gegebenes X-Register in die Richtung des Affixes ab und exekutiert dabei jedesmal die als Parameter codierte Routine (FUN/A und FUN/B), anschließend setzt er bei der nächsten Instruktion fort.

```
METAASSEMBLER  METASM 8.08   JCS/VS 8 * PACER 100      82/11/04 09.47

                *
                *       DO FOR CNT1 ( DO FOR CNT2 ( * ) TIMES ) TIMES
                *
0000  5614        0014          LX      CNT1      OUTER LOOP COUNT
0001  2401                      COD,UP
0002  2F0F                      SKU
0003  4203                      J+3
0004  6203        0007          LINK    FUN/A
0005  43FC                      J-4
0006  2E12                      JCS                 END
                *
                *       CNT2 LOOP
                *
0007  0000        0000  FUN/A   ADR           0     OUTER ROUTINE
0008  520B        0013          STX     HELPX     SAVE X
0009  560C        0015          LX      CNT2      INNER LOOP COUNT
000A  25FF                      COD,DN
000B  2F0F                      SKU
000C  4203                      J+3
000D  6204        0011          LINK    FUN/B
000E  43FC                      J-4
000F  5604        0013          LX      HELPX     RESTORE X
0010  4BF7        0007          J,I     FUN/A     RETURN
                *
                *       FUNCTIONAL PROCEDURE
                *
0011  0000        0000  FUN/B   ADR           0     INNER ROUTINE
                *                     SOME CALCULATIONS
0012  4BFF        0011          J,I     FUN/B     RETURN
                *
                *       DATA - LOOP COUNTER
                *
0013  2F00                HELPX   NOP               TEMPORARY X
0014  7F57        7F57    CNT1    DEC       32599   OUTER COUNT
0015  052C        052C    CNT2    DEC        1324   INNER COUNT
                                  END
```

Assembler-Liste des Programmes

ANWENDUNG

Derzeit können die vom Metaassembler erzeugten Objektfiles auf zwei Arten bearbeitet werden:

Es gibt eine Bibliotheksroutine META, welche einen bestimmten Meta-Objektfile lädt und exekutiert. Diese Routine ermöglicht es, Programmteile, die im Metaassembler erstellt wurden, zu verwenden.

Außerdem gibt es einem Loader METCIG, welcher ein gegebenes Objekt lädt und exekutiert. Mit diesem Loader können komplett im Metaassembler erstellte Programme am PACER 100 zur Exekution gebracht werden.

Der Metaassembler wurde unter anderem zur Codierung des Coprozessors (COP) herangezogen. Hier ein Beispiel für eine Instruktion des COP:

```
METAASSEMBLER  METASM 8.08   JCS/VS 8 * PACER 100     82/11/23 11.31    PAGE     6

  236: ********************************************************************
  237: *
  238: *       SUBROUTINE ALIGN
  239: *       Performs Alignment of two Floating Point Numbers in R1 & R2
  240: *
  241: ********************************************************************
  242: 008A   A48B E32C 40FF FFFC 0      0003    ALIGN   MOF,R1 R3       Save Exp R1 to Exp R3
  243: 008B   A65C 9144 65F1 FFFC 0      0002            AND,R2 R24      Check if R1 or R2 equals Zero
  244: 008C   648C 91EC C5F8 98E6 0      0098            IF,RAZ RALIGN   If one of them is Zero, Return
  245: 008D   A489 4B44 40EF FFFC 0      0002            ESUB    R2      Subtract Exp., (R1 R2) to R1, Mantissa unchanged
  246: 008E   648C 91EC C5F8 98E6 0      0098            IF,RAZ RALIGN   If Exponents equal, Return
  247: 008F   648C 91EC E3F8 98E6 0      009A            IF,A<B ALIGN1   If Exp R1 less than Exp R2, adjust R1 Mantisse
  248: *
  249: *       Exp R2 less than Exp R1, adjust Mant. R2, Exp R1 holds positiv Diff.
  250: *
  251: 0090   648C 91EC E7F8 99E6 0      0099            IF,MOF ALCLR2   If Overflow, set R2 to Zero
  252: 0091   A48A 693C 40F8 17FC 0      0017            ESUB1   R23     Perform (23-Exp R1), no update, Load MSR
  253: 0092   648C 91EC E3F8 99E6 0      0099            IF,A<B ALCLR2   If Difference was greater than 23, set R2 to Zero
  254: 0093   9ECB E129 F533 FBFC 0      0000            TER,R2 POL0     Transfer Exp R1 to BSL, POL=0,
  255: *                        Clock Mantisse R2 to Multiplier X-Reg.
  256: 0094   9FCC 91EC E510 15FC 0      0015            TM,BSL CLRY     Transfer BSL to Multiplier Y-Reg.
  257: 0095   9FDC 91E8 65F1 E6FC 0      0002            MP,RNR R2       Multiply, Result to R2, no Postnorm., no Rounding
  258: 0096   A48B E364 40FF FFFC 0      0001            MOF,R3 R1
  259: 0097   A48B E368 40FF FFFC 0      0002            MOF,R3 R2       Update proper Exponents
  260: 0098   648C 91EC E5FF FFF5 0              RALIGN RETURN
  261: 0099   A49C 95E8 F5F8 98E7 0      0098    ALCLR2 F70,R2 RALIGN    Zero to R2, Return
  262: *
  263: *       Exp R3 less than Exp R2, adjust Mant. R1, Exp R1 holds negativ Diff.
  264: *
  265: 009A   648C 91EC E7F8 A3E6 0      00A3    ALIGN1 JE,MOF ALCLR1    If Overflow, set R1 to Zero
  266: 009B   A48A 6324 40FF FFFC 0              END,01          Absolute Diff. in Exp R1
  267: 009C   A48A 693C 40F8 17FC 0      0017            ESUB1   R23     Perform (23-Exp R1), no update, Load MSR
  268: 009D   648C 91EC E3F8 A3E6 0      00A3            IF,A<B ALCLR1   If abs. Diff. was greater than 23, set R1 to Zero
  269: 009E   9ECB E129 E533 FBFC 0      0000            TER,R1 POL0     Transfer Exp. R1 to BSL, POL=0,
  270: *                        Clock Mantisse R1 to Multiplier X-Reg.
  271: 009F   9FCC 91EC E510 15FC 0      0015            TM,BSL CLRY     Transfer BSL to Multiplier Y-Reg.
  272: 00A0   9FDC 91E4 65F1 E6FC 0      0001            MP,RNR R1       Multiply, Result to R1, no Postnorm., no Rounding
  273: 00A1   A48B E344 40FF FFFC 0      0001            MOF,R2 R1       Update Exponent R1
  274: 00A2   648C 91EC E5FF FFF3 0                      RETURN
  275: 00A3   A49C 9504 F5F8 98E7 0      0098    ALCLR1 F70,R1 RALIGN    Zero to R1, Return
```

A C S L

ADVANCED CONTINUOUS SIMULATION LANGUAGE

· H. Hummer

Diese digitale Simulationssprache mit der weltweit größten Zahl an Installationen (>400) konnte von der Abteilung Hybridrechenanlage des EDV-Zentrums der Technischen Universität Wien von Mitchell & Gauthier, Concordia, Mass. erworben werden und steht nach der Installation an der CDC CYBER 720 den Benutzern sowohl im Batch- als auch im Time-Sharing-Modus zur Verfügung. Die Betreuung der Benutzer und die System-Wartung (Installation einer neuen Release, etc.) erfolgen durch das EDV-Zentrum der Technischen Universität, Abteilung Hybridrechenanlage.

ACSL entspricht den Spezifikationen des Technical Committee on Continuous System Simulation Languages und damit dem sogenannten CSSL-Standard. Die Herstellerfirma betreibt die Weiterentwicklung und laufende Betreuung dieser Software schnell und mit besonders hervorzuhebender Benutzerfreundlichkeit.

ACSL dient zum Modellieren zeitabhängiger, nicht linearer Differentialgleichungen (auch mit diversen Übergangsfunktionen). Typische Anwendungen liegen in den Gebieten der Kontrolltheorie, der Simulation chemischer Prozesse, der Aerodynamik und der Hydrodynamik. In ACSL werden z.B. derzeit auf dem Gebiet der Reaktorsicherheit von damit befaßten Firmen Systeme der Ordnung >500 Gleichungen "gefahren".

ACSL gestattet in der Beschreibung der Modelle weitestgehende Freiheit der Syntax, wie Mischung von blockorientierter Eingabe mit FORTRAN-Statements. Außerdem stehen simulationsorientierte Operatoren in großer Zahl zur Verfügung (variable Zeitverzögerung, tote Zone, Funktionsgeneratoren, Regler, etc.). Die Makromöglichkeiten erlauben dem Benutzer, sein Problem in selbst definierbaren Einheiten zu formulieren, und nachfolgende Benutzer, die ähnliche Probleme behandeln, können mit diesen einmal definierten problemorientierten Bausteinen weiterarbeiten.

ACSL bietet als besondere Vorzüge eine große Zahl von wählbaren Integrationsalgorithmen und Einflußnahme auf den Lauf der Integration, womit der Komfortabilität der Modellbeschreibung eine in gleicher Weise sorgfältige Konstruktion der Simulationssprache vom Gesichtspunkt der numerischen Mathematik aus entspricht.

Die in ACSL formulierten Modelle werden vom Übersetzer in FORTRAN-Quelltext übersetzt, und aus diesem wird von FORTRAN-Übersetzer und LOADER unter Beifügen verschiedener ACSL-System- und Benutzer-Bibliotheken ein exekutables Programm erzeugt. Dieses Programm läuft zum Zeitpunkt seiner Exekution unter Kontrolle eines Laufzeitinterpreters, sodaß zu diesem Zeitpunkt zwar nicht mehr das Modell, jedoch alle seine Parameter sowie eine große Zahl von Systemparametern geändert werden können.

Die Ausgabemöglichkeiten von ACSL reichen von Line-Printer-Plots (und den damit an interaktiven Terminals ohne Graphikfähigkeit möglichen Printer-

Plots) bis zur interaktiven graphischen Ausgabe auf Tektronix-kompatiblen Terminals und zum Erzeugen off-line verarbeitbarer Plotter-Files. Diese hier genannten graphischen Ausgabemöglichkeiten stehen unseren Kunden bereits zur Verfügung, ACSL besitzt jedoch Treiber für weitere Graphik-Pakete wie DISSPLA und verschiedene Plotter, für die On- und Off-Line-Plots leicht realisierbar sind.

Die Hybridrechenanlage und das Institut für Analysis, Technische Mathematik und Versicherungsmathematik, Arbeitskreis Regelungstheorie und Hybridrechentechnik, stellen einführende Beschreibungen von ACSL und der notwendigen Steuerkarten zur Verfügung. Am Donnerstag, den 13.1.1983, findet von 10 bis 12 Uhr eine On-line-Demonstrationsveranstaltung im kleinen Seminarraum des Instituts für Analysis, Technische Mathematik und Versicherungsmathematik, Gußhausstraße 27-29, 1040 Wien (4. Stock, Zimmer 1415) statt.

An einem einfachen Beispiel sollen einige Möglichkeiten von ACSL demonstriert werden:

Die Gleichung

$$\dot{x}=y+kx(1-x^2-y^2)/\sqrt{x^2+y^2}$$

$$\dot{y}=-x+ky(1-x^2-y^2)/\sqrt{x^2+y^2}$$

mit $x(0)=xz;y(0)=yz$

beschreibt eine Kurve in der x-y Ebene, für die für jedes Paar Startwerte xz,yz (außer für xz=yz=Ø) gilt:

$$x^2+y^2 \to 1 \text{ für } t \to \infty$$

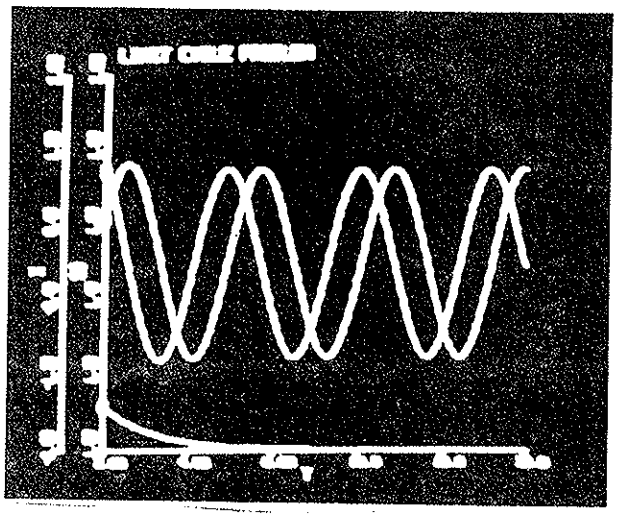
Die Form der Kurve wird also immer mehr dem Einheitskreis gleichen.

Die Modellbeschreibung in ACSL-Statements:

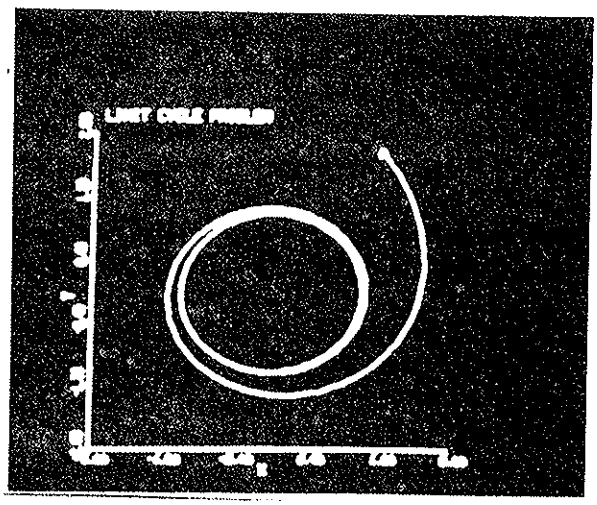
```
CALL,RUNACSL(MODEL=M2,ACSLSYS=E020204,E=ET)
1          **********ADVANCED  CONTINUOUS  SIMULATION  LANGUAGE*********
      ACSL TRANSLATOR VERSION 1 LEVEL 7C1 82/11/22. 16.25.49.      PAGE  1


 PROGRAM LIMIT CYCLE
 "---DEFINE ALL PRESET VARIABLES"
 CONSTANT  XZ=0.5   , YZ=1.0
 CONSTANT  K=0.2    , TF=9.99
 CINTERVAL CINT=0.2
 "---GIVE NAME TO RMS VALUE"
 SQ = SQRT(X**2 + Y**2)
 "---ISOLATE COMMON FACTOR"
 KK = (1.0 - SQ**2)/SQ
 "---LIMIT CYCLE EQUATIONS"
 X = INTEG( Y + K*X*KK, XZ)
 Y = INTEG(-X + K*Y*KK, YZ)
 "---DEFINE STOPPING CONDITION"
 TERMT (T .GE. TF)
 END $ "OF PROGRAM"

    2.173 CP SECONDS      2062 TABLE SPACE USED        3 TABLE MOVES
```

Der Run-Time-Output:

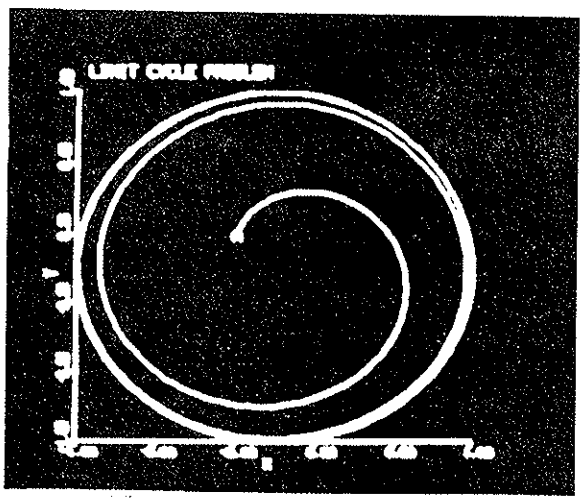
```
START
 T 0.                    X 0.50000000          Y 1.00000000
SQ 1.11803399

 T 1.00000000            X 1.07143048          Y 0.11524377
SQ 1.07761050

 T 2.00000000            X 0.65941072          Y-0.81887087
SQ 1.05136473

 T 3.00000000            X-0.32732337          Y-0.98097465
SQ 1.03414305

 T 4.00000000            X-0.99128128          Y-0.25178710
SQ 1.02275868

 T 5.00000000            X-0.74193795          Y 0.69293307
SQ 1.01519858

 T 6.00000000            X 0.18130838          Y 0.99375825
SQ 1.01016246

 T 7.00000000            X 0.93107118          Y 0.38308500
SQ 1.00680071

 T 8.00000000            X 0.82357251          Y-0.57520097
SQ 1.00455355

 T 9.00000000            X-0.03897837          Y-1.00229241
SQ 1.00305005

 T 10.0000000            X-0.86359260          Y-0.50823121
SQ 1.00204348
```



Verlauf von X und Y und die Differenzfunktion über der Zeit



Beliebiger Startwert außerhalb des Einheitskreises



Beliebiger Startwert innerhalb

Die folgende Originalarbeit stellte uns freundlicherweise Dr. D.J. Murray-Smith, Senior Lecturer an der Universität Glasgow, zur Verfügung. Dr. Murray-Smith hielt zu diesem Thema im Dezember 1981 an der Technischen Universität Wien einen Vortrag.

A SIMULATION OF WATER TURBINE GOVERNOR SYSTEMS: AN ILLUSTRATIVE EXAMPLE, USED IN THE TEACHING OF CONTROL ENGINEERING

D.J. Murray-Smith
Dept. of Electronics and Electrical Engineering,
University of Glasgow
Glasgow, G12 8QQ
Scotland

1. INTRODUCTION

The increasing size of electrical generators installed in new thermal power stations in the United Kingdom during the last twenty years has caused the electricity supply and distribution network to become more vulnerable to frequency transients caused by the sudden failure of a single item of generating plant. Some thermal plant must therefore be available to provide spare capacity at all times. This 'spinning spare', as it is known, is provided by generators which are synchronised to the electricity supply network but are not fully loaded and are thus available to pick up a load quickly following the loss of a plant elsewhere in the system. The use of thermal plant as spinning spare is, however, both costly and inefficient. The North of Scotland Hydro Electric Board (NSHEB), which is responsible for electricity supply in the northern part of Scotland, has been interested, for some time, in the use of water turbine generators as spinning spare instead of thermal plant.

Conventional hydro-turbine plant has, in theory, the potential for rapid response to disturbances and would be excellent for the provision of spare generating capacity of this kind were it not for one major difficulty. The turbine governors in many of the Scottish installations are set up to give stable operation on the infrequent occasions when the generator is supplying a load in an isolated part of the distribution network which has no thermal plant. Under more normal operating conditions when the hydro plant is connected to a large stable grid system, dominated by thermal plant, the governor has little influence on overall stability. Under these conditions a governor with settings which give stable operation for an isolated load has a sluggish response to frequency disturbances and thus lacks the characteristics required for satisfactory operation as spinning spare. Alternatively, if a governor is set up to give a very fast response to frequency disturbances when connected to the grid, the generating unit may be unstable when supplying an isolated load.

Since about 1972 a considerable amount of research work has been carried out at the University of Glasgow to investigate the effects of different forms of linear and nonlinear governors on the overall characteristics of a turbine and electrical generator. This research, which has been carried out in very close collaboration with NSHEB engineers, has involved a number of members of the teaching staff of the University and several research students. Extensive use has been made of analogue, digital and hybrid simulation techniques in the development of improved forms of electronic control equipment to replace, on an experimental basis, existing forms of hydraulic governor equipment. The simulation work [1,2] has been concerned mainly with a 32.5 MW turbine generator at Sloy Power Station where much of the experimental work has been performed.

The research has resulted in the development of a stable electronic governor giving a dominant time constant, in response to frequency disturbances, which is much smaller than the time constant associated with the original hydraulic governor [2,3,4]. Although this electronic governor gave a substantial improvement it was recognised that the full potential of the turbine was not yet being realised and a microprocessor-based adaptive governor has been developed subsequently and tested on site [5,6].

The research carried out on governor systems has had a number of beneficial effects in terms of control systems teaching activities within the Department. Although the hybrid simulation developed for research purposes was not itself suitable for use as a teaching aid, simplified representations of the turbine and governor system have been developed which are of considerable value in illustrating the effects of different forms of controller on the overall characteristics of a closed-loop system.

A simulation involving a linearised turbine model, of a size which students can understand reasonably quickly and use without difficulty, has been

developed for a facility based on an EAI 2000 analogue processor coupled by serial and parallel links to a DEC PDP 11-45 digital computer.The simulation can be used as a test bed for controller transfer functions obtained by students using classical frequency domain or other control system design techniques. It provides, in the context of a practical system, a very simple but powerful means of illustrating factors such as the effects of backlash on limit cycle behaviour and the influence of measurement noise on the overall performance of the system with different controllers.

Downstream from the surge-shaft the penstock divides into two and finally four separate pipes to supply four Francis turbines of 32.5 MW capacity. At the turbines water passes into a spiral casing which is concentric with and surrounds a multi-orifice cylindrical structure consisting of a number of guide vanes. This structure serves as the inlet control valve for the turbine and is driven by a hydraulic servomotor through mechanical linkages. The guide vanes are shaped so that the water passing through them is accelerated and the pressure energy is thus

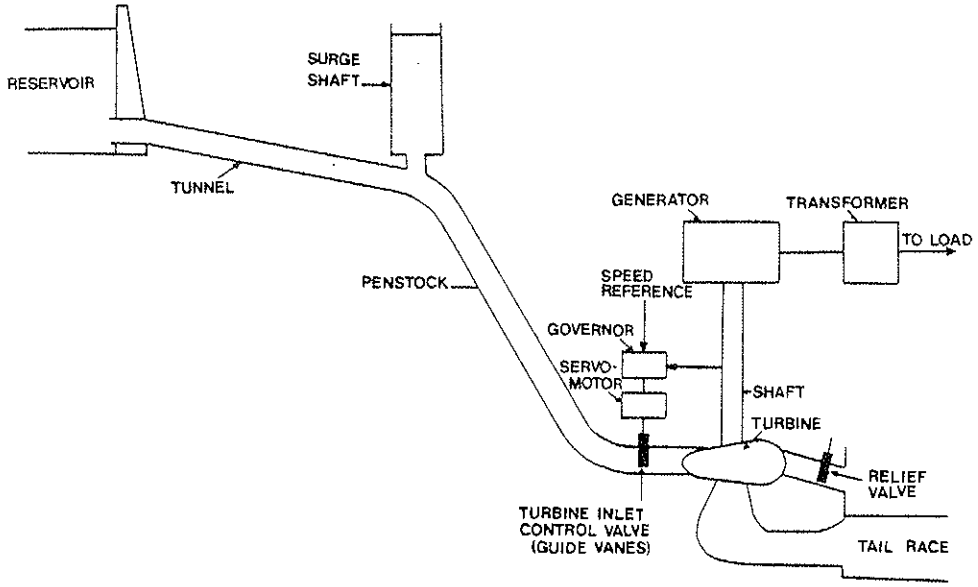


Fig. 1
Schematic diagram of a hydro-electric power plant

converted to kinetic energy. Francis turbines are reaction machines in which the runner (the rotating part) is designed so that water passing between the blades is accelerated still further producing an opposite reaction head on the runner. The difference between the torque developed by the turbine and the load torque gives rise to an accelerating torque on the turbine and the generator to which it is mechanically coupled. Speed governing of the system is provided through an appropriate external feedback path to the guide vanes through a controller (the "governor") and the servo motor. Fig.2 is a block diagram which incorporates the essential features of the system.

2. SYSTEM DESCRIPTION

Fig.1 is a schematic diagram showing the main features of a hydro-electric generating installation of the type at Sloy. Water from an upper reservoir flows to the turbine through a tunnel, penstock and surge-shaft system. The surge-shaft forms an escape route for water from the tunnel in the event of a reduction in turbine demand and also provides a temporary supply of water should this demand increase suddenly.

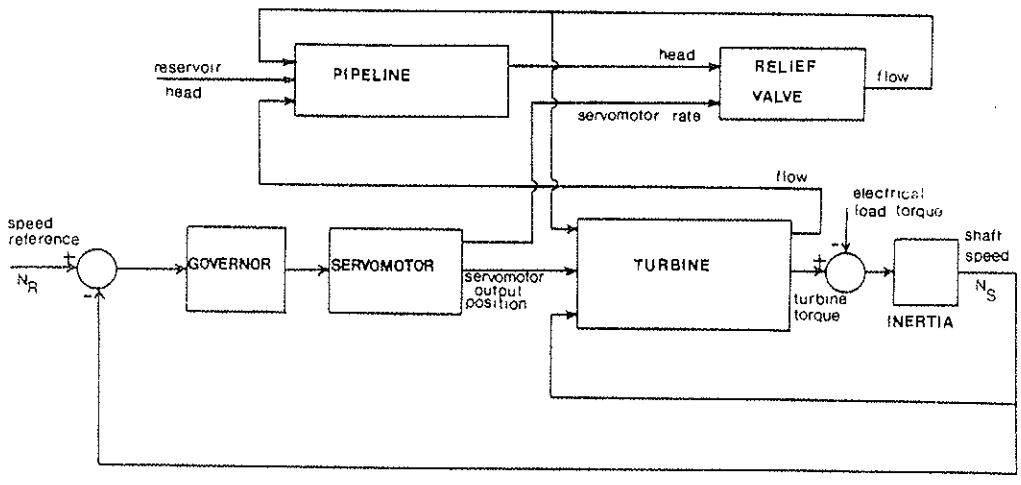


Fig. 2
Block diagram for speed governing system

A relief valve is provided to allow water from the penstock to be diverted from the turbine in the event of a rapid turbine shut-down so that the dangers of large pressure surges are avoided. The valve reacts to the rate of change of servomotor position and opens only when the servomotor closes the guide vanes at a rate greater than a preset threshold value.

3. SYSTEM SIMULATION

Detailed modelling of the system, including the pipeline, surge-shaft, turbine, relief valve and the electrical load has led to the development of a relatively complex hybrid computer simulation [2]. Although very useful for the purposes of research this simulation has been considered to be of limited value for the teaching of control system concepts. It is too complex to be understood and used easily by students and includes many features which are unnecessary for teaching applications.

By linearising the model of the turbine and by incorporating effects associated with the pipeline into the turbine model, it is possible to produce a much simplified representation of the system which can be applied easily to study the influence which the governor transfer function has on the overall characteristics of the controlled system. Fig.3 is a detailed block diagram for this simplified model. Here an impulse representation has been used for the turbine and pipeline. Although this is not truly representative of a Francis turbine, it was regarded as adequate for teaching applications in which interest is centred upon the overall control systems aspects. The action of the relief valve is neglected in the simplified model but realistic rate limits are incorporated at the servomotor and backlash can be included between the servomotor and the guide vanes to represent the effects of the mechanical linkages.

A number of different forms of governor transfer function may be incorporated easily into this model structure. The governors incorporated in many of the hydro-electric power stations in Scotland are of the so-called "temporary droop" type and have transfer functions of the form

$$\frac{Y(s)}{U(s)} = \frac{1 + sT_x}{\sigma + (\sigma + \mu)T_x s + T_x T_s s^2} \qquad \ldots(1)$$

A second form of governor which has been studied extensively as a means of giving governor action is the "double derivative" type [7]. The transfer function in this case may be written as

$$\frac{Y(s)}{U(s)} = \frac{1}{\sigma\left(1 + s\frac{T_3}{\sigma}\right)}\left[1 + \frac{K_1 s}{1 + sT_1} + \frac{K_2 s^2}{1 + sT_2}\right] \qquad \ldots(2)$$

A third form of governor transfer function which has received attention is a form of lead-lag compensator with transfer function.

$$\frac{Y(s)}{U(s)} = \frac{K(1 + sT_d)}{(1 + sT_b)(1 + sT_c)} \qquad \ldots(3)$$

In eqn. (1) parameters σ and μ may be regarded as quantities which are readily adjustable in terms of conventional governor hardware. In eqn. (2) σ, K_1 and K_2 are the principal parameters which may be selected to satisfy the overall control system performance specifications while in eqn. (3) it is assumed that the gain constant K and the three time constants T_d, T_b and T_c may all be chosen independently.

The system of Fig. 3 has been simulated using the EAI 2000 analogue computer at Glasgow University and the simulation has been used for teaching purposes on a regular basis. Students make use of the simulation in a pre-patched form since the emphasis, in this application, is on the control systems aspects rather than on the simulation techniques being used.

Parameter values for the turbine and other fixed elements of the model, such as the rate limits, are, in most cases, not altered in any way during the study. The parameters normally adjusted in the course of the work include the variable quantities within the governor transfer functions, the backlash and the size and form of load disturbances or speed reference changes. Extensive use is made of repetitive operation of the EAI 2000 to allow investigation of the sensitivity of the system responses to the adjustable parameters.

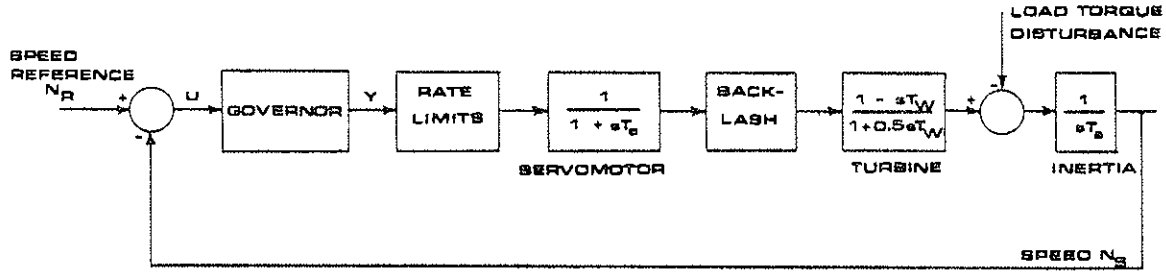


Fig. 3
Simplified block diagram for speed governing system

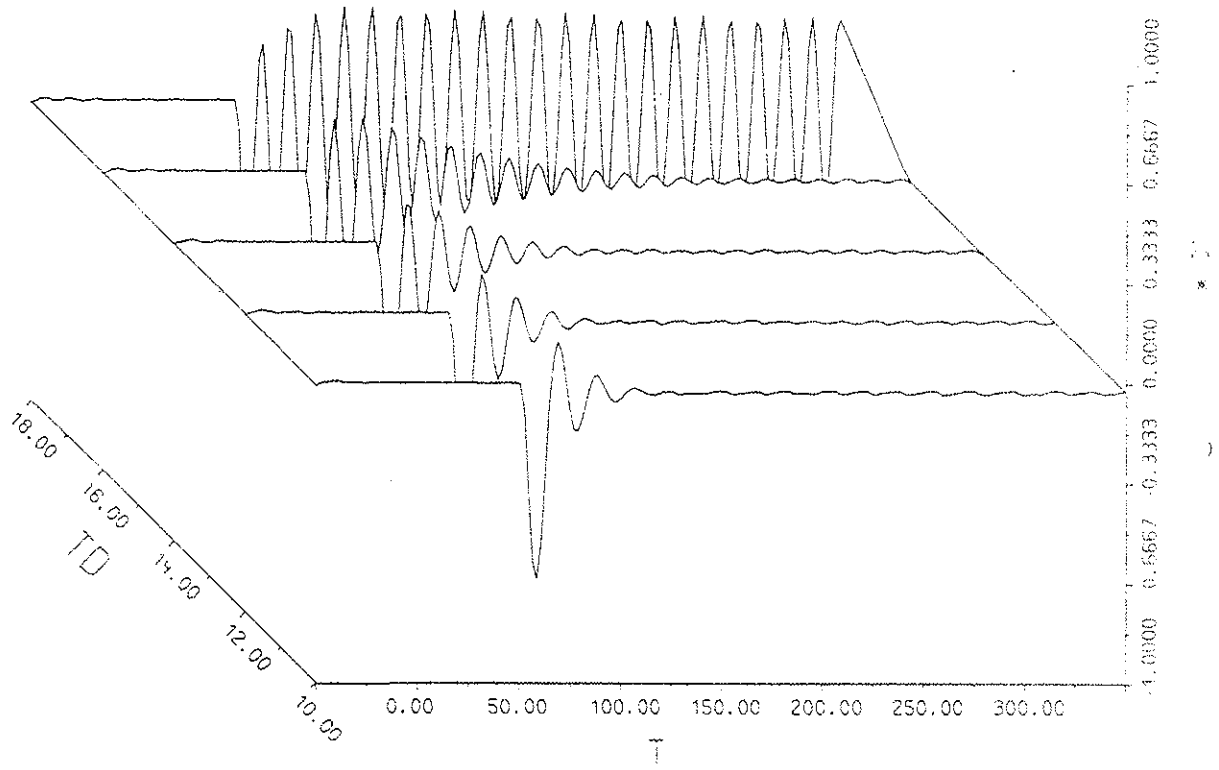


Fig. 4
Responses of simplified model of Fig. 3 to electrical load disturbance steps applied at time t = 50 secs for five different values of parameter T_d in the governor transfer function of eqn. (3). Each plot represents the speed error, u (normalised), versus time t (seconds).

Fig.4 shows some results using the lead-lag form of governor described by eqn. (3). These results which show the sensitivity of the system to variation of the parameter T_d were obtained using HYBSYS at the Technical University of Vienna.

Further details of this teaching simulation, including a full set of parameter values, scaled equations, static test data and further examples of results, may be obtained from the author.

ACKNOWLEDGEMENTS

The author wishes to acknowledge with thanks the contribution which the work of other members of staff and research students at the University of Glasgow has made to the development of the turbine governor system model now used for teaching purposes. Discussions with members of staff and students at the Technical University of Vienna when the model was demonstrated on the hybrid computer system there in December 1981 also proved very helpful in the further development of the simulation as a teaching aid.

REFERENCES

1. AGNEW, P.; The governing of Francis turbines. Water Power, 26, 333-341, 1974

2. BRYCE, G.W., FOORD, T.R., MURRAY-SMITH, D.J. and AGNEW, P.W.: Hybrid simulation of water-turbine governors. Simulation Councils Proceedings Series, 6, Part 1, pp. 35-44, 1976

3. WINNING, D.J., BRYCE, G.W., AGNEW, P.W., FOORD, T.R. and MARSHALL, A.G.: On-site investigation of electrohydraulic governors for water turbines. Proc. I.E.E., 124, 147-153, 1977

4. AGNEW, P.W. and BRYCE, G.W.: Optimising turbine operation by electronic governing. Int. Water Power and Dam Construction, 29, 1, 36-38, 1977

5. WINNING, D.J., MARSHALL, A.G., FINDLAY, D.G.E., AITKEN, K.H. and GRANT, N.F.: Controller testing facility on a 32.5 MW water turbine. Proc. I.E.E., Part C, 127, 357-359, 1980

6. FINDLAY, D.G.E., DAVIE, H., FOORD, T.R., MARSHALL, A.G. and WINNING, D.J.: Microprocessor-based adaptive water turbine governor. Proc. I.E.E., Part C, 127, 360-369, 1980

7. SCHLIEF, F.R. and BATES, C.G.: Governing characteristics for 820000 horsepower units for Grand Coulee third powerplant. I.E.E.E. Transactions, PAS-90, 882-888, 1971

Die folgende Originalarbeit stellte uns freundlicherweise Dr. D.J. Murray-Smith, Senior Lecturer an der Universität Glasgow, zur Verfügung. Dr. Murray-Smith hielt zu diesem Thema im Dezember 1981 an der Technischen Universität Wien einen Vortrag.

MATHEMATICAL MODELLING OF PHYSIOLOGICAL SYSTEMS WITH APPLICATIONS IN RESPIRATORY MEDICINE

D.J. Murray-Smith
Dept. of Electronics and Electrical Engineering,
University of Glasgow
Glasgow, G12 8QQ
Scotland

1. INTRODUCTION

Computer simulation techniques and the associated mathematical approaches used for the dynamic analysis of complex engineering systems provide a range of potentially powerful methods for the investigation of physiological systems [1,2]. Although it may appear that physiological and engineering systems are very different in form, they have much in common in terms of the difficulties encountered in their quantitative study. Nonlinearities, complex interactions and problems associated with measurement noise or inaccessibility to measurement are features commonly encountered in the investigation of both man-made and physiological systems. In terms of modelling, however, physiological systems are more difficult to handle because their structure is inexactly known and they are generally more complex and more highly interactive. Despite these difficulties the application of the techniques of dynamic systems analysis to supplement more conventional methods of physiological investigation can give substantial practical benefits. In the study of respiratory gas exchange this has led both to a better understanding of physiological mechanisms and to the development of new methods of clinical measurement.

2. THE MODELLING PROCESS FOR PHYSIOLOGICAL SYSTEMS

Mathematical modelling is essentially an interactive process but it may be divided conveniently into the stages of model formulation, model analysis and simulation, model validation and application. Fig.1 shows one possible flow-chart description for a form of modelling process which is appropriate to the study of physiological systems. This illustrates very clearly the interactions which are usually necessary between the experimental and simulation activities for the successful modelling of physiological systems.

2.1 Model formulation

Different mathematical descriptions may be obtained for a given system at different levels of complexity. The form of model used in any particular study is determined mainly by the intended use of the model.

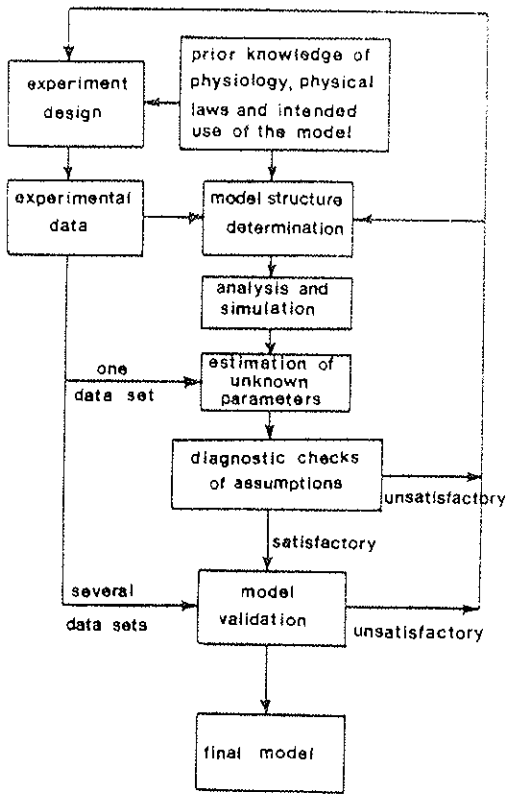


Fig. 1
Flow chart of the modelling process.

In most biomedical applications involving biological system modelling, prior knowledge of the anatomy and physiology of the system must be used extensively at the formulation stage. Sometimes it may be possible to obtain equations defining the model from this prior knowledge, together with simplifying assumptions which reduce the problem to one that can be handled mathematically. Since the acceptability of any assumption depends not only on physiological considerations but also on the intended use of the model, it is essential that anyone using a model should be aware of all the assumptions made in its development.

In many physiological applications models are used as aids in hypothesis testing. In such applications it is usually necessary to be able to relate variables and parameters of the model to specific physiological and anatomical quantities. Clearly a differential equation type of formulation derived from the known physiology is likely to be appropriate in such cases. On the other hand, for a model which represents a component within a larger system a transfer function or other similar description, obtained entirely from input and output measurements may suffice, provided that the internal variables of the subsystem are not of importance.

For both types of model mentioned above the values of all of the coefficients appearing in the model equations are unlikely to be known at the formulation stage and they may not be accessible to direct measurement. Techniques of system identification and parameter estimation are therefore of considerable importance in the development of physiological models and also for the routine non-invasive estimation of physiological quantities in a clinical environment.

2.2 Model analysis and simulation

Physiological models are frequently highly non-linear in form and mathematical analysis can be extremely difficult unless use is made of computer simulation techniques. The value of simulation in the study of physiological systems is being increasingly recognised as interest grows in developments associated with the use of other control systems concepts and approaches in clinical medicine. Interactive simulation models have, of course, a useful role not only in research and development but also as teaching and training aids. The 'MacPuf' model [3] provides an interesting example of this in the respiratory field.

2.3 Model validation

Model validation should be regarded as an integral part of the modelling process. There are clearly many different criteria by which model validity may be assessed. Some of these are associated with the internal consistency of the model and the associated computer simulation, while other criteria must relate to a comparison of the model performance with that of the physiological system being modelled. Direct validation of this kind may present difficulties in cases where important variables of the model are inaccessible to measurement. Failure at the evaluation stage requires that the model be re-examined to determine whether the problem is associated with the model structure or with inaccurate estimates of some of the unknown parameters due perhaps to poor experimental design.

3. MODELS OF RESPIRATORY GAS EXCHANGE PROCESSES

The lung is the central organ of respiration in humans and provides the interface between the air which ventilates the alveoli and the blood which perfuses the pulmonary capillaries. Gas transport between atmospheric air and blood is a physiological process which is particularly well suited to a modelling and simulation approach. This suitability for modelling is due mainly to the fact that the principal input to the system is the gas flow at the mouth, which is accessible for measurement. Controlled disturbances may be applied easily through changes of inspired gas concentration, by imposing a particular pattern of breathing, or through the application of a stimulus involving exercise.

In attempting to model the complex structure of airways and capillaries which form the real lung it is essential to seek a balance between extremely detailed models with a very large number of equations which would be difficult, or perhaps impossible, to solve, and highly simplified representations which may be inadequate for the intended application. In the course of research at Glasgow University involving the use of modelling and system identification techniques for problems in respiratory medicine a number of gas exchange models have been developed for different applications [2,4-8].

3.1 A simple homogeneous lumped-parameter lung model

Fig.2 shows the structure upon which the model is based. There is a single, uniform, compliant, alveolar compartment (volume V_A) in which gas exchange takes place. This compartment is coupled

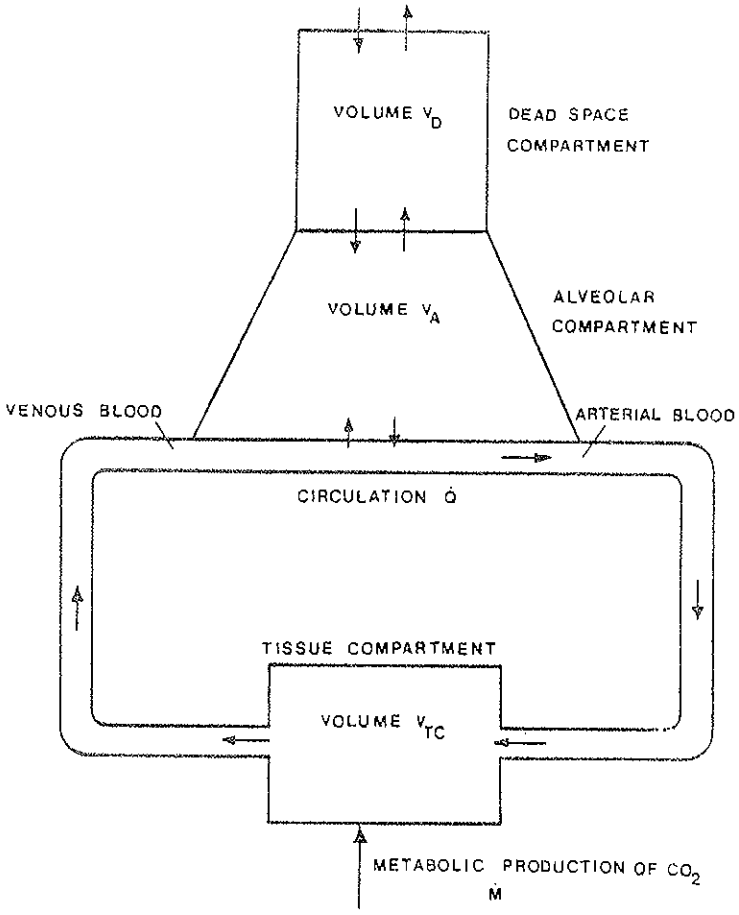


Fig. 2
Structure of homogeneous lumped-parameter lung model.

to a rigid and non-reacting "dead space" compartment (volume V_D) which corresponds to the conducting airways. The third compartment (volume V_{TC}) represents the tissues and this is coupled to the alveolar compartment by the circulation (blood flow $\dot{Q}$). Although this model structure is of general form and provides a basis for the study of many different gases within the lungs most attention has been given to gases which react chemically with blood, as in the case of oxygen or carbon dioxide.

Using this model structure the respiratory cycle may be split into three separate stages in terms of the ventilatory gas flow rate $\dot{V}(t)$:

(i) inspiration of dead space gas to alveoli

$$\dot{V}(t) \geq 0 \text{ and } \int_{t_1}^{t} \dot{V}(t)dt < V_D$$

where t_1 defines the time of start of inspiration

(ii) inspiration of atmospheric gas or test gas to alveoli

$$\dot{V}(t) \geq 0 \text{ and } \int_{t_1}^{t} \dot{V}(t)dt \geq V_D$$

(iii) expiration, $\dot{V}(t) < 0$

The principle of conservation of mass can be applied to each stage of the cycle to establish a set of differential equations describing the dynamics of gas exchange in the alveolar compartment. An important assumption which leads to a simplification of the model equations is that an equilibrium condition exists between the partial pressures of gas in the alveolar compartment and in arterial blood. It is also assumed that changes in volume of the alveolar compartment are due entirely to ventilation with negligible differences in gas flow to and from the blood. The circulatory delay in the venous path (duration T) is assumed to be very much greater than the circulatory delay in the arterial path and the latter is therefore neglected.

For the special case of carbon dioxide, in which a linear relationship between concentration and partial pressure in blood is assumed, the equation for the alveolar compartment is

$$V_A \beta \frac{d P_A}{dt} = S\dot{V}(t)(P_I(t) - P_A(t)) + \dot{Q}[b(P_{TC}(t-T) - P_A(t)) + a_d] \quad \ldots(1)$$

Here P_A, P_{TC} and P_I are the partial pressure values for carbon dioxide in the alveolar compartment, in the tissue compartment and in the inspired gas mixture respectively. The constants b and a_d are associated, respectively, with the slope and intercepts of linearised representations of the relationship between concentration and partial pressure of carbon dioxide in blood [7]. The quantity β is a capacitance coefficient, as used by Piiper et al. [9]. In this equation the left hand side relates to the rate of change of gas in the alveolar compartment while terms on the right hand side involve the transfer from the dead space and from the tissues. For the tissue compartment the corresponding equation is

$$b V_{TC} \frac{d P_{TC}}{dt} = \dot{M} - \dot{Q}[b(P_A(t) - P_A(t)) + a_d] \quad \ldots(2)$$

The right hand side of this equation involves the difference between the metabolic production of carbon dioxide and the rate of transfer to the alveolar compartment.

It should be noted that in eqn. (1) the volume of the alveolar compartment will vary throughout the breath cycle. The change in volume from a given initial value $V_A(0)$ may be determined by integrating the ventilatory flow $\dot{V}(t)$ with respect to time.

The transfer of gas between the dead space compartment and the alveolar compartment varies throughout the different stages of the breath cycle. The term in eqn. (1) associated with transfer of gas through the dead space must therefore change at each stage. This is achieved using the factor S which appears in the equation. S has value zero in stages (i) and (iii) and unity in stage (ii).

This very simple model of pulmonary gas exchange processes has been tested extensively and has been found to be an adequate representation, in experiments of the order of a few minutes duration, for human subjects having no known respiratory abnormalities. More detailed consideration of the physiological assumptions implicit in this model may be found elsewhere [4,5,7].

The model has been evalued under a variety of different experimental conditions and has formed a basis for the development of a method for the indirect and non-invasive estimation of the pulmonary blood flow (parameter $\dot{Q}$ of the model) using system identification techniques [6,7]. In this application the variables $P_I(t)$ and $\dot{V}(t)$, in a computer simulation of the model of eqns. (1) and (2), are obtained from continuous flow and concentration measurements at the subject's mouth. The model is thus subjected to an input identical to that applied to the human subject under test. Estimation of model parameters such as $V_A(0)$, V_{TC} and $\dot{Q}$ is carried out from measurements of the concentration of expired gas using a particular form of the maximum likelihood method [7]. Uncorrelated residuals associated with this parameter estimation process have provided useful additional evidence supporting the validity of this simple model structure.

The model has also been used to represent gas exchange processes within a much larger and more complex model of the overall control system associated with respiration [10]. The gas exchange

model is well suited to this application in that the different stages of the breath cycle, which are believed to be of importance in the control of breathing, are explicitly represented.

3.2 Inhomogeneous lumped-parameter lung models

The model of Fig. 2, having a single homogeneous alveolar compartment, is incapable of describing abnormalities of pulmonary gas exchange processes associated with the maldistribution of ventilation or perfusion. Studies have therefore been carried out, using both analogue and digital simulation techniques, to investigate models with two alveolar compartments, for which the associated ventilation and blood flow rates could be adjusted separately. Fig. 3 shows a typical structure for a model of this type which has formed the basis of a teaching simulator for use with medical students [8].

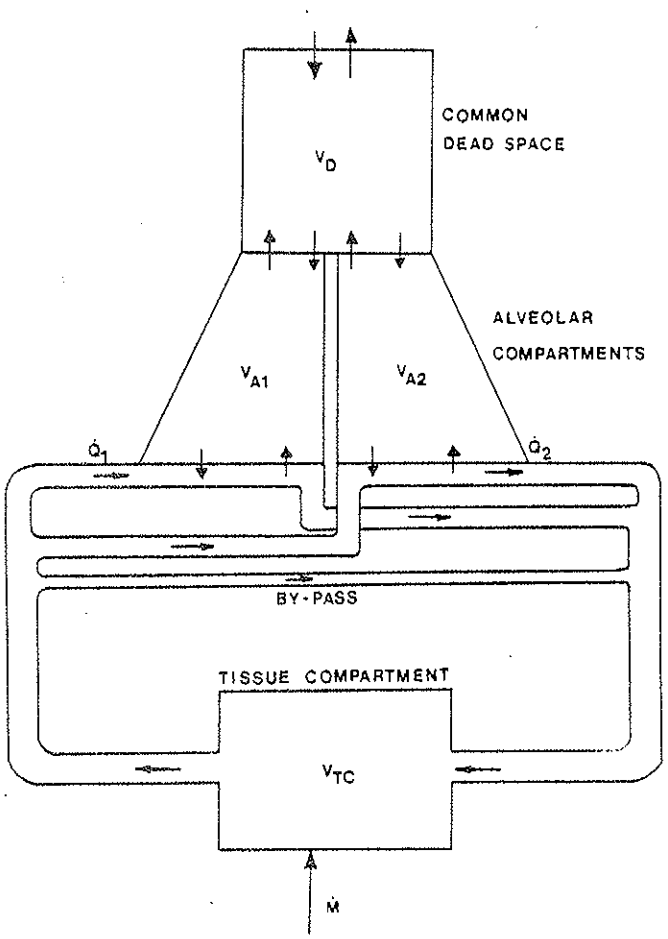


Fig. 3
Structure of inhomogeneous lumped-parameter lung model.

By analogy with eqn.(1) the differential equation for alveolar compartment 1, which has a proportion k of the ventilation, is

$$V_{A1}\ \beta \frac{d P_{A1}}{dt} = k\, S_1 \dot{V}(t)\ (P_D(t) - P_{A1}(t)) + k\, S_2 \dot{V}(t)\ (P_I(t) - P_{A1}(t)) + \dot{Q}_1 [b(P_{TC}(t) - P_{A1}(t)) + a_d] \quad \ldots(3)$$

In this equation the circulatory delays have been neglected. The partial pressure of carbon dioxide in the dead space, $P_D(t)$, is related to the partial pressures in the alveolar compartments by the equation

$$P_D(t) = k\, P_{A1}(t) + (1 - k)\, P_{A2}(t) \quad \ldots(4)$$

Similarly for alveolar compartment 2

$$V_{A2}\ \beta \frac{d P_{A2}}{dt} = (1 - k)\, S_1\, \dot{V}(t)\ (P_D(t) - P_{A2}(t)) + (1 - k)\, S_2\, \dot{V}(t)\ (P_I(t) - P_{A2}(t)) + \dot{Q}_2 [b(P_{TC}(t) - P_{A2}(t)) + a_d] \quad \ldots(5)$$

The equation for the tissue compartment is

$$b\, V_{TC} \frac{d P_{TC}}{dt} = \dot{M} - \dot{Q}_1 [b(P_{TC}(t) - P_{A1}(t)) + a_d] - \dot{Q}_2 [b(P_{TC}(t) - P_{A2}(t)) + a_d] \quad \ldots(6)$$

Fig. 4 shows results obtained from a simulation based upon this model using appropriate parameter values. The two sets of results, which were obtained using HYBSYS on the hybrid computer system at the Technical University of Vienna, show the partial pressure in one of the alveolar compartments for sinusoidal ventilation with periods of breath-holding and hyperventilation. As well as providing the time course of changes of alveolar or tissue (mixed venous) partial pressure following an alteration of ventilation (as in Fig. 4) the model can be used to illustrate the effects of alveolar ventilation and perfusion on alveolar gas partial pressure and on the arterial-alveolar pressure difference. The role of the dead space in determining the level of alveolar ventilation and the resulting alveolar gas partial pressure can also be investigated easily.

4. CONCLUSIONS

Relatively simple lumped parameter gas exchange models which incorporate a cyclic ventilatory pattern are now available. These models may be applied to individual human subjects by using the measured gas flow at the mouth as a variable of the model. System identification techniques allow dynamic models of this type to form a basis for the non-invasive estimation of physiological quantities. This is particularly important in a clinical environment where the use of a model-based approach for indirect measurement with a test stimulus which simple, convenient and safe has major attractions.

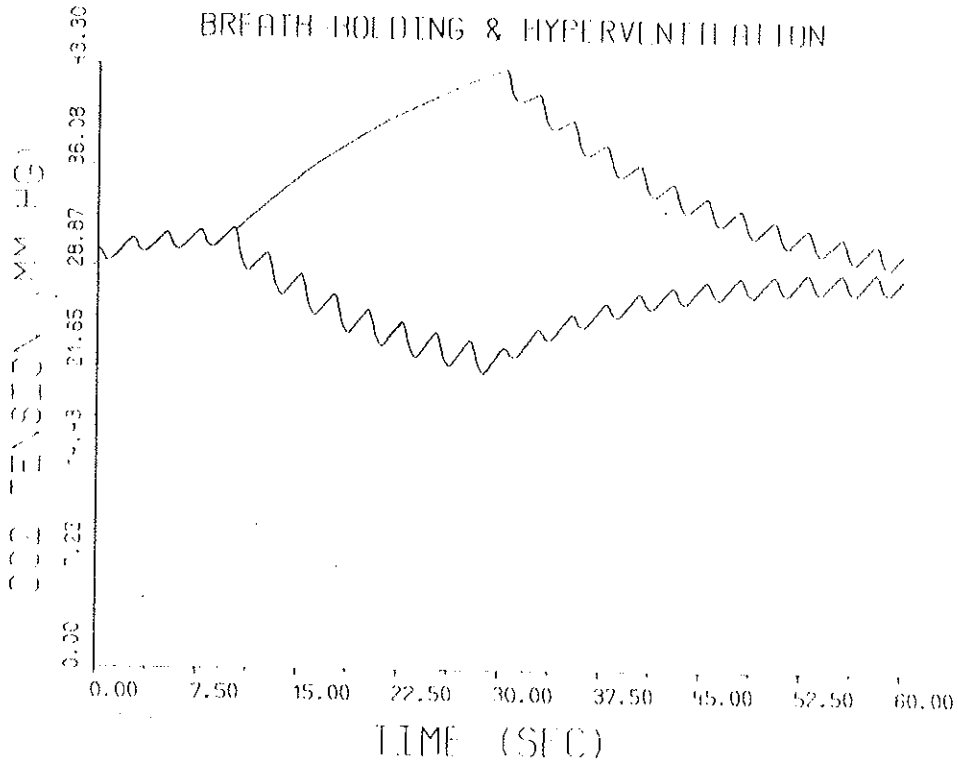


Fig. 4
Responses of inhomogeneous model showing changes of partial pressure in one of the two alveolar compartments caused by changes in ventilation. The upper curve shows the effect of a period of breath-holding of 20 sec. duration. The lower curve shows the effect of a similar period of hyperventilation.

ACKNOWLEDGEMENTS

Close collaboration with the Centre for Respiratory Investigation at Glasgow Royal Infirmary has been essential for the success of the modelling and identification work reported here and in the references. The contributions of Dr. F. Moran (Physician in Charge of the Centre) and of Dr. A.I. Pack are particularly acknowledged, together with those of Dr. R.A. Bache and other former research students at the University of Glasgow.

REFERENCES

1. PACK, A.I. and MURRAY-SMITH, D.J.: Mathematical models and their applications in medicine. Scottish Medical Journal, 17, 401-409, 1972

2. PACK, A.I., MURRAY-SMITH,D.J., MILLS, R.J., HOOPER,M. and TAYLOR, J.: Applications of mathematical models in respiratory medicine. Bull. Inst. of Math. and its Application, 10, 20-26, 1974

3. DICKINSON, C.J.:A computer model for human respiration. MTP Press, 1977

4. EMERY, B., MORAN, F. and PACK, A.I.: A ventilation driven simulation of pulmonary gas exchange in man. J. Physiol., 222, 38p, 1972

5. MURRAY-SMITH, D.J. and PACK, A.I.: Techniques of computer simulation applied to respiratory gas exchange. In Taylor, D.E.M. and Whamond, J.S. (eds): Non-Invasive Clinical Measurement. Pitman Medical, 186-202, 1977

6. FERGUSON, D.R., MILLS, R.J., MORAN, F., MURRAY-SMITH, D.J. and PACK, A.I.: Estimation of the parameters of a lung model with clinical applications. In Eykhoff, P. (ed): Identification and System Parameter Estimation. North Holland, 213-219, 1973

7. BACHE, R.A., GRAY, W.M. and MURRAY-SMITH, D.J.: Time-domain system identification applied to non-invasive estimation of cardiopulmonary quantities. I.E.E. Proc., 128, Pt.D., pp56-64, 1981

8. MILLS, R.J., MIDDLETON, S., MORAN, F., MURRAY-SMITH, D.J. and PACK, A.I.: Simulation in the teaching of concepts of respiratory gas exchange. Int. J. Math. Educ. Sci. Technol., 5, 381-387, 1974

9. PIIPER, J., DEJOURS, P., HAAB, P. and RAHN, H.: Concepts and basic quantities in gas exchange physiology. Resp. Physiol., 13, 212-304, 1971

10. GREER, W., JORDAN, M.M. and MURRAY-SMITH, D.J.: A structural approach to respiratory control simulation. In "Computing in Medicine" (Proc. of Fourth Series of Strathclyde Bioengineering Seminars), edited by J.P. Paul, MacMillan Publishers (in press).

UNTERSUCHUNG EINES DIGITALEN REGELALGORITHMUS AM HYBRIDRECHNER

P. Kopacek und St. Wagner
Inst. für Wasserkraftmaschinen und Pumpen
Fachbereich "Maschinen- und Prozeßautomatisierung"
Technische Universität Wien

1. EINLEITUNG

Digitale Algorithmen werden nicht zuletzt durch das Vordringen der Mikrorechner in zunehmenden Maße zur Regelung von maschinenbaulichen und verfahrenstechnischen Prozessen eingesetzt. Derzeit fehlen aber noch Erfahrungswerte zur Wahl geeigneter Regelalgorithmen in Abhängigkeit vom Prozeß, zur Festlegung der Reglerparameter sowie Anhaltswerte über die "Robustheit" derartiger Algorithmen gegenüber Änderungen der Prozeßparameter.

Anhand der Drehzahlregelung einer Francisturbine soll auf den letztgenannten Themenkomplex - den Einfluß von veränderlichen Streckenparametern auf die Regelgüte - näher eingegangen werden. Ein vereinfachtes Modell einer Francisturbine wurde am Analogteil des Hybridrechners nachgebildet und der digitale Regelalgorithmus am Digitalteil des Rechners programmiert. Bei variierenden Parametern des Turbinenmodells wurden zunächst die Reglerparameter konstant gehalten (Robustheit) und dann mit Hilfe eines Syntheseverfahrens versucht, für jede Parameterkombination der Turbine optimale Reglerparameter zu bestimmen.

2. MODELL DER FRANCISTURBINE

Das dynamische Verhalten einer Francisturbine mit kurzer Rohrleitung kann einschließlich des Servomotors zur Leitschaufelverstellung näherungsweise durch die Übertragungsfunktion

$$F(s) = \frac{X(s)}{U(s)} = \frac{1-u_1 T_{WN} s}{(1+T_s s)(1+0{,}5u_1 T_{WN} s)(u_1 e_s + T_{AN} s)} \qquad \ldots \quad (1)$$

beschrieben werden. Darin sind

- $X(s)$..... die Laplacetransformierte der Regelgröße (Turbinendrehzahl)
- $U(s)$..... die Laplacetransformierte der Stellgröße (Stellung des Leitradservomotors)
- u_1..... die bezogene Nennleistung
- T_{WN}, T_{AN}, T_s die Zeitkonstanten der Druckleitung, der rotierenden Massen und des Servomotors
- e_s..... der Selbstregelfaktor

Faßt man diese Auslegungsparameter zu Verstärkung und Zeitkonstanten zusammen, ergibt sich eine Übergangsfunktion der Form

$$F(s) = \frac{K \cdot Ts}{(1+T_1 s)(1+T_2 s)(1+T_3 s)} \qquad (2)$$

In Übereinstimmung mit ausgeführten Anlagen wurden die in (2) enthaltenen Parameter folgendermaßen festgelegt:

$K = 1$, $T = 1\text{sec}$, $T_1 = 0{,}05\text{sec}$, $T_2 = 0{,}5\text{sec}$, $T_3 = 5{,}0\text{sec}$.

Mit diesen Werten ergibt sich die (2) entsprechende Differentialgleichung

$$0{,}125\dddot{x}(t) + 2{,}775\ddot{x}(t) + 5{,}55\dot{x}(t) + x(t) = u(t) + \dot{d}(t) \qquad \ldots \quad (3)$$

3. REGELALGORITHMUS

Nachdem in [1] verschiedene digitale Algorithmen hinsichtlich ihrer Eignung zur Regelung dieser Strecke untersucht und miteinander verglichen wurden, lieferte ein Regler, dessen Struktur durch die z- Übertragungsfunktion

$$R(z) = \frac{U(z)}{X_d(z)} = \frac{q_0 + q_1 z^{-1} + \ldots + q_\nu z^{-\nu}}{1 - p_1 z^{-1} - \ldots - p_\mu z^{-\mu}} \qquad (4)$$

gegeben ist, die besten Ergebnisse. Als tragbarer Kompromiß zwischen einer möglichst kleinen Anzahl von zu bestimmenden Reglerparametern p_μ und q_ν und einer hohen Regelgüte ergab sich aus einem Syntheseverfahren $\mu=\nu=7$. Allerdings treten bei diesem Algorithmus für die Praxis zu hohe Stellamplituden, die zu Druckstössen in der Anlage führen könnten, auf. Um dies zu vermeiden, wurden die ersten vier Werte der Stellgröße $u(kT_0)$ (T_0..Abtastzeit, $k=0,1,2,3$) geeignet vorgegeben. Der optimale digitale Regler dieser Struktur kann daher durch die z- Übertragungsfunktion

$$R(z) = \frac{q_0 + q_1 z^{-1} + q_2 z^{-2} + \ldots + q_7 z^{-7}}{1 - p_1 z^{-1} - p_2 z^{-2} - \ldots - p_7 z^{-7}} \qquad (5)$$

welcher im Zeitbereich die Differenzengleichung

$$u(k) = 1 + p_1 u(k-1) + p_2 u(k-2) + \ldots + p_7 u(k-7) + q_0 x_d(k) + q_1 x_d(k-1) + \ldots + q_7 x_d(k-7) \qquad \ldots \quad (6)$$

mit $T_0=1$ entspricht, beschrieben werden.

4. HYBRIDRECHNERSIMULATION UND ERGEBNISSE [2]

Die Regelstrecke wurde anhand der Differentialgleichung (3) mit Hilfe des Simulationssystems HYBSYS am Analogteil des Rechners simuliert. Die Programmierung des Regelalgorithmus erfolgte nach Gleichung (6) am Digitalteil. Dieser diente auch zur Berechnung der optimalen Reglerparameter. Die Bestimmungsgleichungen für q_0 bis q_7 und p_1 bis p_7 wurden als Overlay programmiert und mit dem Befehl PARABE aufgerufen.

Um den Einfluß veränderlicher Streckenparameter zu untersuchen, wurden die Zeitkonstanten T_1, T_2 und T_3 nacheinander in den Grenzen T_1=0,01 - 0,1sec, T_2=0,36 - 0,90sec und T_3=2,0 - 20,0sec variiert. Um beispielsweise den Einfluß der Zeitkonstanten T_3 auf die Übergangsfunktion der ungeregelten Strecke zu verdeutlichen, ist in Abb.1 über der Zeit die normierte Regeldifferenz (Drehzahlsollwert - Drehzahlistwert) der Turbine für eine sprungförmige Belastungsänderung des Maschinensatzes um 10% aufgetragen. Die dritte Achse dieses räumlichen Diagrammes ist nach der veränderlichen Zeitkonstanten T_3 geteilt. Es ist unmittelbar ersichtlich, daß mit wachsenden T_3 die Allpaßeigenschaften der Strecke, erkenntlich an dem anfänglichen Knick der Übergangsfunktion, abnehmen.

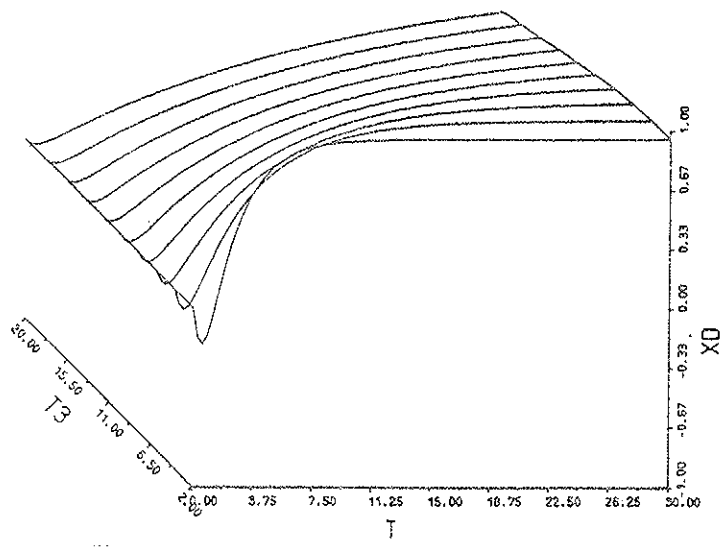


Abbildung 1

Um erste Anhaltspunkte über die Güte der Regelung zu gewinnen, wurde für den Regelalgorithmus (6) für die in (3) verwendeten Werte der Streckenparameter die optimalen Reglerkoeffizienten bestimmt und mit diesen für verschiedene Zeitkonstantenkombinationen der Strecke Übergangsfunktionen aufgenommen. Eine davon zeigt Abb.2. Die obere Kurve stellt die auftretende Regeldifferenz und die untere die Werte der Stellgröße dar. Als Gütemaße für die Regelung dienten unter anderen die Betragsregelfläche $|I_X|$ und die quadratische Regelfläche I_X. Im anderen Fall erfolgte für jede Streckenparameterkombination die Berechnung optimaler Reglerparameter.

Einige der in [2] gewonnenen Versuchsergebnisse sind in Abb.3 zusammengestellt. Über der veränderlichen Zeitkonstante T_3 sind die Werte der Betragsregelfläche $|I_X|$ (strichlierte Linien) und der quadratischen Regelfläche I_X (durchgezogene Linien) aufgetragen. Das linke Bild gilt für die Regelung mit konstanten Reglerparametern und das rechte Bild für jene mit optimalen Reglerparametern. Infolge der Allpaßeigenschaften der

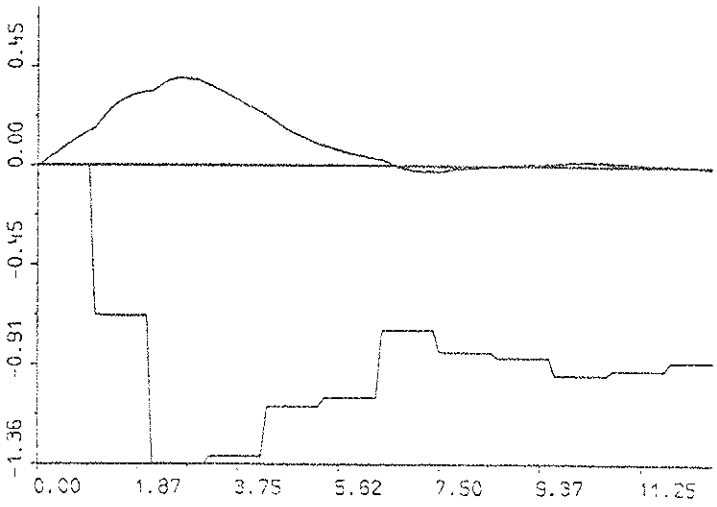


Abbildung 2

Regelstrecke bei kleinem T_3 treten bei T_3=2 relativ große Werte der Regelfläche auf - das Minimum der quadratischen Regelfläche im linken Bild liegt bei dem Wert T_3=5, dem Auslegungsparameter des Reglers. Der starke Anstieg der quadratischen Regelfläche bei gleichzeitig konstanter Betragsregelfläche deutet auf ein starkes Überschwingen der Übergangsfunktion für größere Werte von T_3 hin. Wesentlich bessere Ergebnisse ergeben sich bei der Verwendung von optimalen Reglerparametern. Unter Berücksichtigung der verschiedenen Maßstäbe beider Diagramme ergeben sich bis zu T_3=5 annähernd gleiche Werte beider Regelflächen. Für größere Werte von T_3 kann die Regelung jedoch wesentlich verbessert werden, da die Strecke immer träger wird.

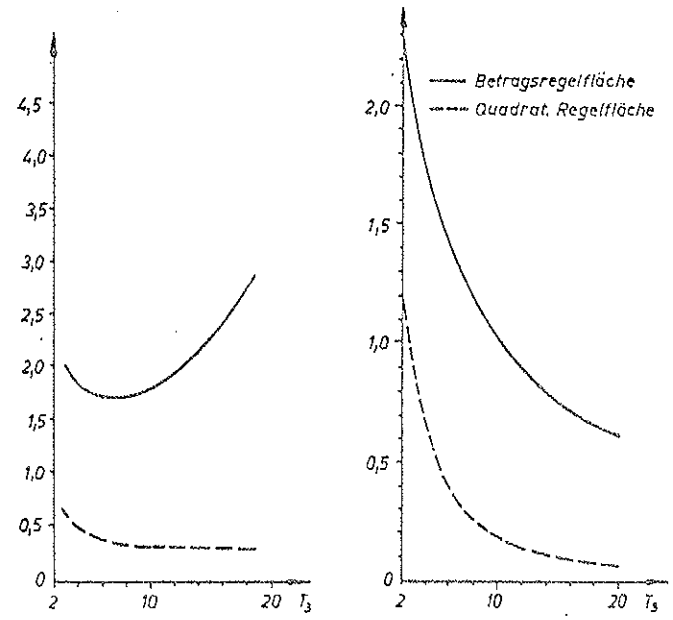


Abbildung 3

LITERATUR

1. KOPACEK, P. und ZAUNER, E.: Governing turbines by microcomputers. Water Power and Dam Construction, Sept. 1982, S.26-30.

2. WAGNER, St.: Robustheit von digitalen Regelalgorithmen am Beispiel einer maschinenbaulichen Regelstrecke. Diplomarbeit, TU-Wien, 1981.

SIMULATION DES TEMPERATURVERLAUFES BEIM WÄRMETRANSFORMATOR

Razenberger G.
Physikalisches Institut
Universität Wien

1. EINLEITUNG

Die mit der Zeit fortschreitende Erschöpfung der Energiequellen führt zu der Aufgabe, neue Technologien für eine bessere Energienutzung zu entwickeln. Ein erheblicher Teil der Primärenergie wird für die Raumheizung und industrielle Zwecke verwendet. Bei vielen industriellen Prozessen wird der größte Teil der eingesetzten Energie als Abfallwärme abgeführt. Dabei fällt die Abfallwärme meistens bei Temperaturen oberhalb der Umgebungstemperatur an. Diese Temperaturen reichen jedoch für eine direkte Nutzung der Abfallwärme im Prozeß selbst oder für Heizzwecke nicht aus. Daraus ergibt sich die Aufgabe, die Temperatur der Abfallwärme soweit anzuheben, daß sie für eine weitere Verwendung nutzbar gemacht werden kann.

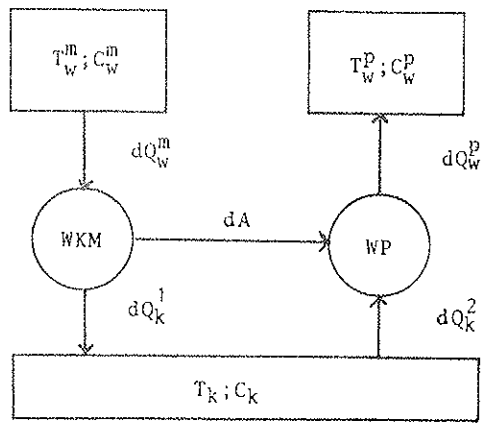
Eine Wärmepumpe mit mechanischem Verdichter ist zwar in der Lage, die Temperatur eines Wärmestromes anzuheben, benötigt jedoch für ihren Antrieb wertvolle Energie (Exergie). Da jedoch die Temperatur der Abfallwärme oberhalb der Umgebungstemperatur liegt, kann das bestehende "Potential" (Unterschied zwischen der Abfallwärmetemperatur und Umgebungstemperatur) zur Anhebung der Temperatur von einem Teil der Abwärme herangezogen werden. Dieses Prinzip läßt sich durch einen "Wärmetransformator" verwirklichen.

2. AUFBAU UND WIRKUNGSWEISE DES WÄRMETRAFOS

Es wird eine Anlage zum Erzeugen von Wärme vorausgesetzt, die aus einer Wärmekraftmaschine (WKM) besteht, an die eine Wärmepumpe (WP) gekuppelt ist. Beide Maschinen sollen vollkommen arbeiten und Carnot-Prozesse durchlaufen. Die Anlage (Wärmetrafo) dient dazu, bei der Temperatur T_w^p nutzbare Wärme zu erzeugen. Die Temperatur der Umgebung sei gleich T_k; die der Antriebsenergie entsprechende Temperatur werde mit T_w^m bezeichnet. Bei der Simulation des Temperaturverlaufes T_w^p in Abhängigkeit von T_w^m bzw. T_k in Abhängigkeit von T_w^m wurden folgende zwei Modelle verwendet.

3. PHYSIKALISCHE MODELLE

Variante A:



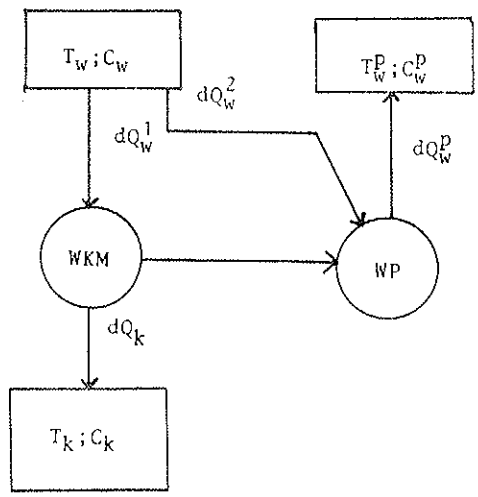
Die Temperaturen T_w^m bzw. T_w^p sollen zu Beginn des Prozesses denselben Wert (313 K ≙ 40°C) haben. Die Umgebungstemperatur sei 283 K ≙ 10°C. Bei der Variante A wird die Wärmepumpe zwischen den Wärmeniveaus T_k und T_w^p betrieben.

C_w^m, C_w^p und C_k sind die Wärmekapazitäten der Abfallwärme (C_w^m+C_w^p) bzw. der Umgebungswärme (C_k).

dA bedeutet die Arbeit, die aus der Wärmekraftmaschine gewonnen wird und in die Wärmepumpe fließt.

dQ_w^m, dQ_k^1, dQ_w^p und dQ_k^2 bezeichnen die zu- bzw. abfließenden Wärmemengen.

Variante B



Temperaturen zu Beginn des Prozesses wie bei Variante A; die Wärmepumpe wird aber nun zwischen den Wärmeniveaus T_w und T_w^p betrieben.

4. MATHEMATISCHES MODELL

Zur Erstellung von Systemgleichungen betrachten wir die Energie- und Entropiebilanzen der reversiblen Kreisprozesse.

	Variante A	Variante B
Energiebilanzen	$dA = -dQ_w^m - dQ_k^1$	$dA = -dQ_w^1 - dQ_k$
	$dA = dQ_w^p + dQ_k^2$	$dA = dQ_w^p + dQ_w^2$
Entropiebilanzen	$\frac{dQ_w^m}{T_w^m} + \frac{dQ_k^1}{T_k} = 0$	$\frac{dQ_w^1}{T_w} + \frac{dQ_k}{T_k} = 0$
	$\frac{dQ_w^p}{T_w^p} + \frac{dQ_k^2}{T_k} = 0$	$\frac{dQ_w^p}{T_w^p} + \frac{dQ_w^2}{T_w} = 0$

Aus diesen vier Gleichungen erhält man nach Umformung die gesuchten Differentialgleichungen.

Variante A	Variante B
$\frac{dy}{dx} = \frac{m.y.(T_{k-x})}{p.x.(y-T_k)}$	$\frac{dy}{dx} = \frac{m.y.(T_{k-x})}{p.x.(y-T_k)}$
$\frac{dT_k}{dx} = -T_k.(\frac{m}{x} + \frac{p}{y} \cdot \frac{dy}{dx})$	$\frac{dT_k}{dx} = -m.\frac{T_k.(y-x)}{x.(y-T_k)}$
$T_w^p = y;\quad T_w^m = x$	$T_w^p = y;\quad T_w = x$
$\frac{C_w^m}{C_k} = m;\quad \frac{C_m^p}{C_k} = p$	$\frac{C_w}{C_k} = m;\quad \frac{C_w^p}{C_k} = p$

5. LÖSUNG DER PROBLEME AM HYBRIDRECHNER

Die angegebenen Probleme wurden mithilfe des Simulationssystems HYBSYS am Hybridrechner simuliert. Für m und p wurden sämtliche Kombinationen der Zehnerpotenzen von 10^{-3} bis 10^{1} berechnet und geplottet. Dabei stellte sich heraus, daß man zwei Gruppen von Ergebnissen unterscheiden konnte. Bei den Varianten A und B blieben die Umgebungstemperaturen entweder konstant (Rundungsfehler des Rechners da sehr kleine Änderungen $\frac{dy}{dx}$ bzw. $\frac{dT_k}{dx}$) oder sie stiegen an (Abbruchkriterium: aus bei $T_{W,End}^m = T_{k,End}$).

Im folgenden sind jeweils vier Fälle der Varianten A und B abgedruckt und die Endtemperaturen angegeben.

Beim Plotten der Kurven wurde die Variable x durch $x1 = T_W^m - T_{W,End}^m$ ersetzt, was zur Folge hatte, daß die erhaltenen Kurven in x-Richtung gedehnt erscheinen. Die obere Kurve zeigt somit y in Abhängigkeit von x1; die untere Kurve T_K in Abhängigkeit von x1.

Die Temperaturen T_w^m (Variante A) bzw. T_w (Variante B) sanken klarerweise ab.

Variante A:

Parameter: m=0,001;p=0,001

$T_{w,End}^m = 283,6K;\ T_{k,End} = 283,2K;\ T_{w,End}^p = 326,2K$

Parameter:m=0,1;p=0,001

$T_{w,End}^m = 286,1K;\ T_{k,End} = 285,6K;\ T_{w,End}^p = 664,4K$

Parameter:m=1;p=0,01

$T_{w,End}^m = 297K;\ T_{k,End} = 297,1K;\ T_{w,End}^p = 565,3$

Parameter:m=10;p=0,1

$T_{w,End}^m = 310,4K;\ T_{k,End} = 309,7K;\ T_{w,End}^p = 412,9K$

Variante B:

Parameter:m=0,001;p=0,001

$T_{w,End}^m = 283,5K;\ T_{k,End} = 283,5K;\ T_{w,End}^p = 325,8K$

Parameter:m=0,1;p=0,001

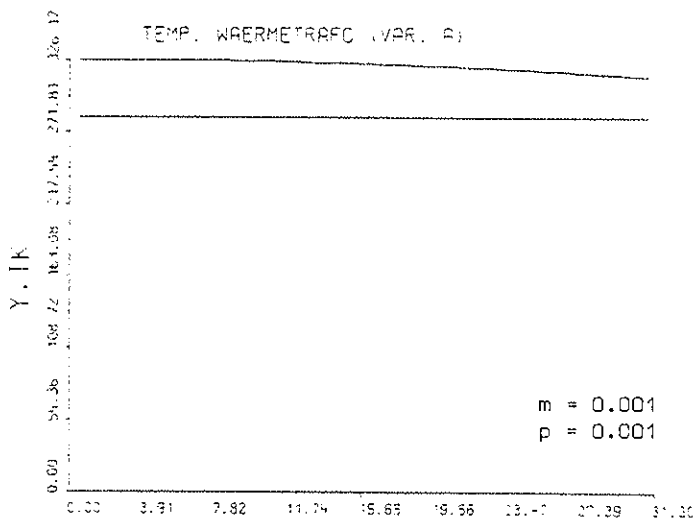
$T_{w,End}^m = 285,7;\ T_{k,End} = 285,8K;\ T_{w,End}^p = 650,4K$

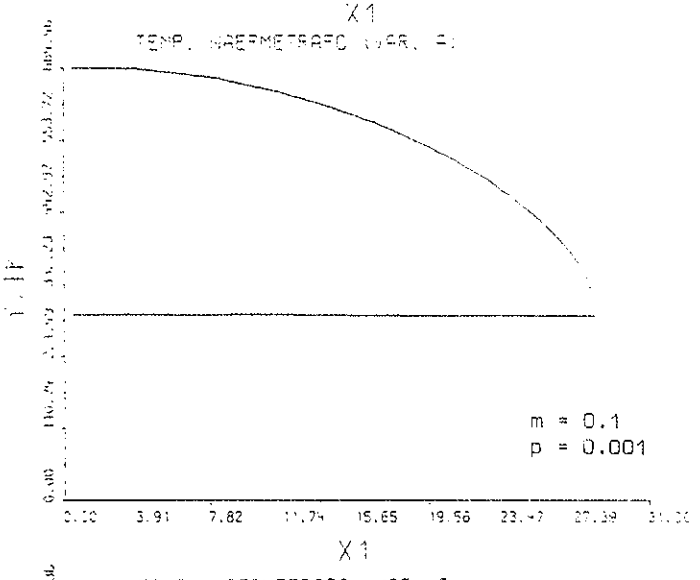
Parameter:m=1;p=0,01

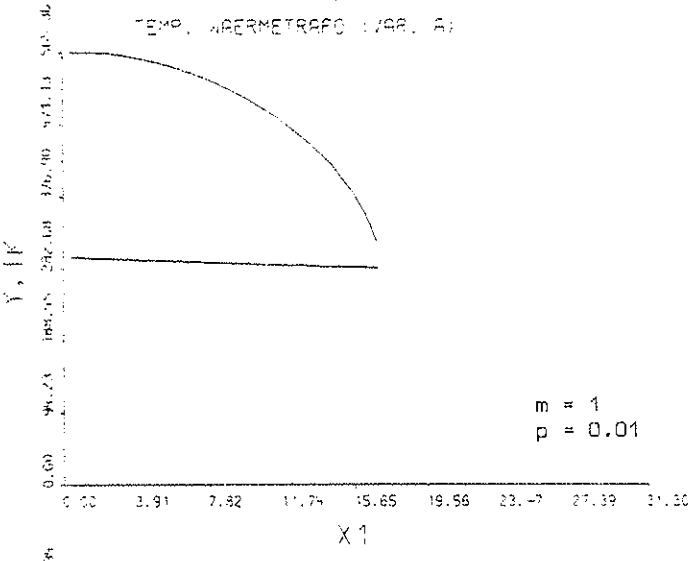
$T_{w,End}^m = 297K;\ T_{k,End} = 297,2K;\ T_{w,End}^p = 559K$

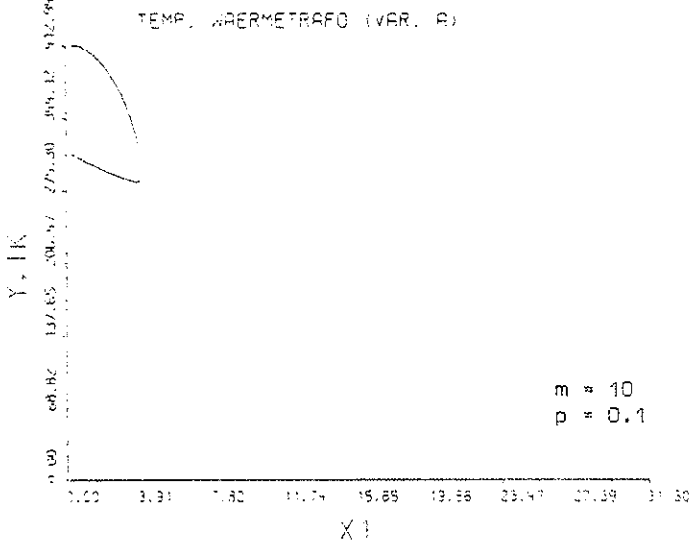
Parameter:m=10;p=0,1

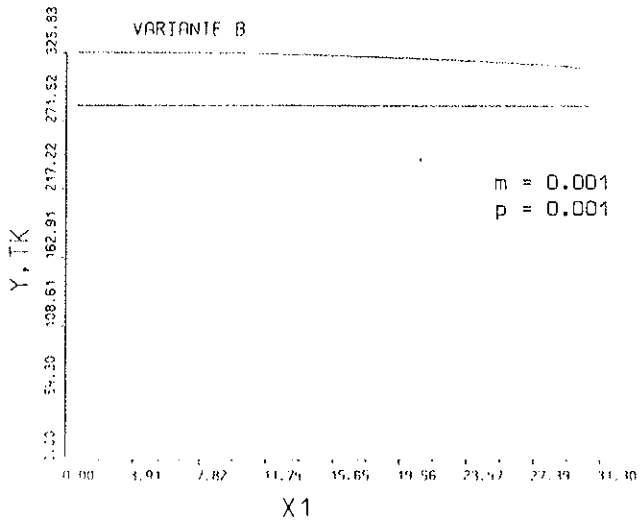
$T_{w,End}^m = 309,8K;\ T_{k,End} = 309,5K;\ T_{w,End}^p = 409,5K$

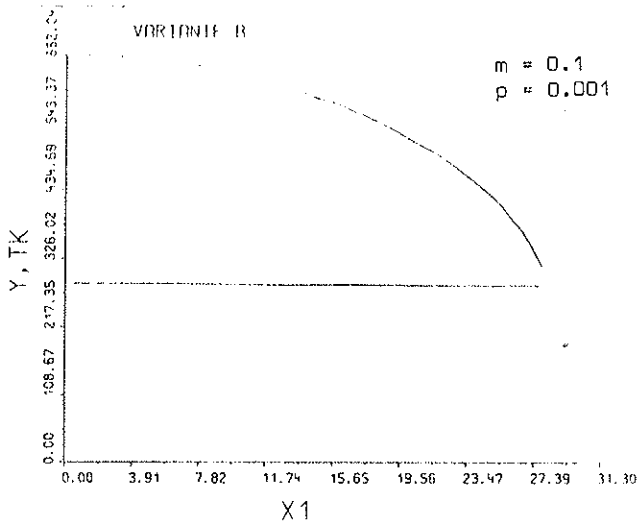


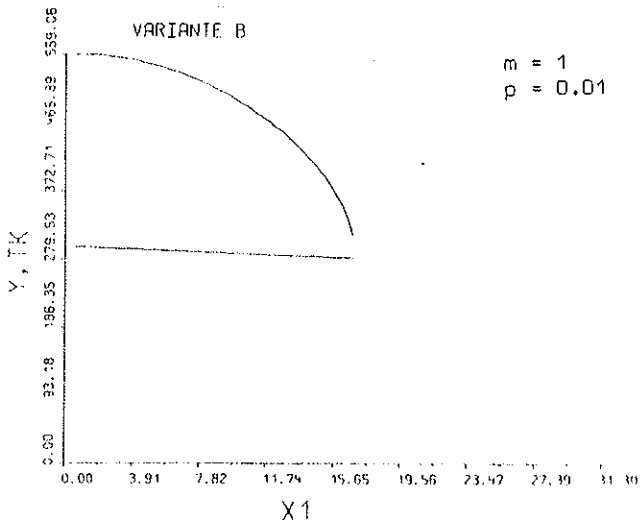


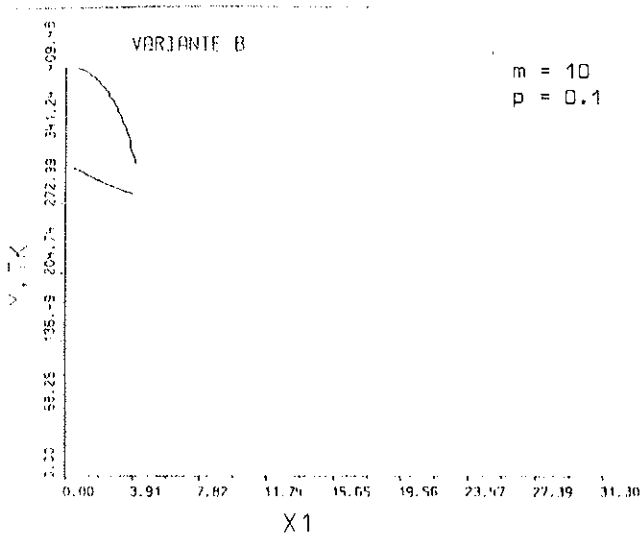












6. SCHLUSZFOLGERUNGEN

Die Varianten A und B mit stark steigender Umgebungstemperatur sind praktisch nicht vertretbar, da sie eine große Gefahr für die Ökologie der betreffenden Umwelt bedeuten. Das heißt, sinnvoll sind also nur jene Fälle, wo T_k=konstant oder nur sehr schwach steigt. Dabei zeigt nun die Untersuchung der einzelnen Ergebnisse bei verschiedener Wahl von m und p, daß die Variante A eine höhere Endtemperatur $T^p_{w,End}$ liefert. Eine spätere Aufgabe wird es dann sein, die Parameter m und p der Variante B so zu optimieren, daß die Nutzwärmetemperatur und Nutzwärmemenge maximal werden.

7. ANWENDUNGSGEBIETE DES WÄRMETRAFOS

Der Wärmetransformator läßt sich überall dort einsetzen, wo Wärmeströme bei niedrigen Temperaturen anfallen. Dieses ist vor allem bei Eindampfanlagen der Fall. Hierbei können der Austreiber und der Verdampfer mit dem Brüdendampf der letzten Stufe beheizt werden. Die anfallende Nutzwärme wird in der ersten Stufe abgegeben oder für andere Zwecke verwendet. Auch bei Destillations- und Trocknungsanlagen liegt ein ähnlicher Fall vor. Der Wärmetransformator wäre zwischen dem heißen und dem kalten Ende der Anlage zu schalten. Weitere Anwendungsfälle liegen bei der Nutzung der Abwärme von Kühl- und Reinigungswässern in Molkereien und Brauereien vor.

Mit heute verfügbaren Anlagen kann ein Wärmeverhältnis (=abgegebene Energie zur aufgenommenen Antriebsenergie) mit 40% bis 70% des höchstmöglichen Wertes erreicht werden. Deshalb dürfte es beispielsweise schon nicht mehr zweckmäßig sein, in großem Maßstab Kohle zum Erzeugen von Wärme bei einer Temperatur unter 150°C sowie Öl oder Gas zum Erzeugen von Wärme bei einer Temperatur unter 250°C unmittelbar zu verbrennen.

SUMMARY

In order to recover the waste heat quantity produced in various processes, which under normal circumstances is dissipated to the ambient medium, it is generally necessary to raise its temperature. This principle can economically be realized by a "heat transformer". Two methods are dealt with, which show the way how to raise the temperature of part of the waste heat quantity.

The ratio analysis for "energy" and "entropy" produces the essential differential equations which have been processed by the hybrid simulation system (HYBSYS). The diagrams referred to show the temperature variation of the waste heat and the ambient heat in relation to the reduced part of the waste heat quantity. Alternative method B produces a higher ultimate temperature as proven by variation of the parameters "m" and "p".

It is left to a further investigation to optimize the parameters to achieve both the most economic effective temperature and heat quantity.

SIMULATION EINES TRAMPOLINSPRINGERS

Aus dem Fortgeschrittenpraktikum am Hybridrechner
F. Rattay
Institut für Analysis, Technische Mathematik und Versicherungsmathematik
Arbeitsbereich Regelungstheorie und Hybridrechentechnik (Prof. I. Troch)
Technische Universität Wien

Als Beispiel für einen Bewegungsvorgang, der zu verschiedenen Zeiten durch verschiedene Differentialgleichungen beschrieben wird, diente die Simulation von Sprungbrett und Turner beim Trampolinspringen.

In dem einfachen Modell wurde dem Verhalten des Endes des Trampolins das Hooke'sche Gesetz zugrunde gelegt.

Die Bewegungen des Springers und des Trampolins wurden zunächst über der Zeit dargestellt (Abb. 2), dann wurde jedoch auch der dynamische Vorgang durch einen zweidimensionalen örtlichen Display veranschaulicht, wobei die Biegelinie des Trampolins durch ein Polynom dritten Grades und der Springer durch einen vertikal bewegten Punkt am Oszillographen aufgebracht wurden.

Dem Bewegungsvorgang mit variabler Dämpfung d wurden folgende Parameter zugrunde gelegt:

Erdbeschleunigung	$g = 10 msec^{-2}$
Masse des Springers	$m_1 = 80kg$
reduzierte Trampolinmasse	$m_2 = 20kg$
Federkonstante des Trampolins	$c = 10.000N/m$
Streckungslänge des Springers	$l = 0,4m$
Sprungkraft	$s = 2gm_1$

Damit wurde folgendes Bewegungsmodell errichtet.

I. Phase: Springer oberhalb des Trampolins

Der Springer springt zur Zeit t=0 aus gegebener Höhe auf das Trampolin.

$$y(0) = y_0, \quad \dot{y}(0) = 0$$

Das Trampolin ist in Ruhelage

$$\dot{z}(0) = \ddot{z}(0) = 0$$

(aus (1.4) wird damit $z(0) = 0,02$)

Aus dem Kräftegleichgewicht ergibt sich

$$m\ddot{y} = -mg \qquad (1.1)$$

$$\ddot{y} = -10 \qquad (1.2)$$

$$m_2\ddot{z} = -d\dot{z} - cz - g.m_2 \qquad (1.3)$$

$$\ddot{z} = -\frac{d}{20}\dot{z} - 500z - 10 \qquad (1.4)$$

Die I. Phase endet mit dem Aufkommen des Springers auf dem Trampolin, also wenn y=z wird ($t=t_1$).
Zur Modellvereinfachung wird die Masse des Trampolins beim Aufsprung (Stoßvorgang) vernachläßigt, d.h. die Geschwindigkeit des Springers wird nicht abrupt abgebremst ($\dot{z}(t_1) := \dot{y}(t_1)$).

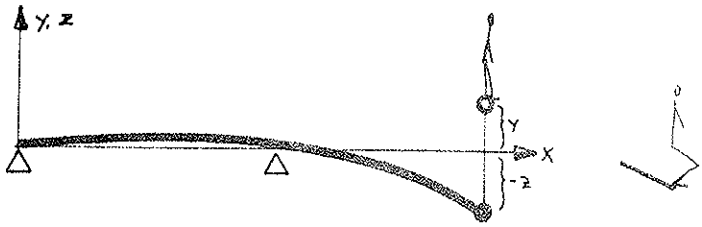


Abbildung 1

Situation des Springers im freien Fall (links) und beim Absprung, wobei das Strecken des Sportlers ein längeres Wirken der Sprungkraft ermöglicht. Die stark gezeichneten Teile wurden am Oszillographen dynamisch dargestellt.

II. Phase: Springer am Trampolin

Bis zum Ende der Phase (Absprung) bleibt

$$y \equiv z$$

aber aus (1.3) wird

$$(m_1 + m_2)\ddot{z} = -d\dot{z} - cz - (m_1 + m_2)g \qquad (1.5)$$

$$\ddot{z} = -\frac{d}{100}\dot{z} - 100z - 10 \qquad (1.6)$$

III. Phase: Absprung

In diesem Modell ist der Absprung an folgende, gleichzeitig wirkende Bedingungen gebunden:

Aufwärtsbewegung des Trampolins,
Springer am Trampolin,
Absichern, daß vorher Phase II aktiv war,
aufwärtswirkende Beschleunigung ist kleiner als die Sprungkraft (sonst wäre der Sportler zu schwach, um abzuspringen),
logische Größe L=1 (zur Unterdrückung des Absprungs und Simulation des Ausschwingvorganges),

also

$$\dot{z} > 0 \wedge y \leq z \wedge (\dot{y} - \dot{z})^2 < \varepsilon \wedge \ddot{z} < 10 \wedge L$$

Die Bewegungsgleichungen werden nun

$$m_1\ddot{y} = -m_1g + 2m_1g$$

$$\ddot{y} = 10 \qquad (1.7)$$

$$m_2\ddot{z} = -d\dot{z} - cz - (m_1 + 2m_2)g \qquad (1.8)$$

$$\ddot{z} = -\frac{d}{20}\dot{z} - 500z - 90 \qquad (1.9)$$

Die III. Phase endet mit

$$y - z > l$$

d.h., sobald der Sportler sich beim Sprung um die Länge l gestreckt hat, hat er keinen Kontakt mehr mit dem Sprungbrett. Er befindet sich nun im freien Fall und die Simulation setzt wieder mit der I. Phase fort (jedoch mit den neuen Anfangsbedingungen).

Durch den gleichzeitigen Aufbau der Gleichungen (1.6) und (1.9) und den aus dem Schaltplan ersichtlichen Maßnahmen der Übergabe der Anfangswerte für die nächste Bewegungsphase wird es möglich, den Phasenwechsel ohne Zeitverlust zu simulieren.

Damit ist es auch auf den Kleinrechnern EAI 1000 möglich, mit 100facher Echtzeitgeschwindigkeit zu fahren, sodaß man Parametereinflüsse aus der Simulation im Repop-Betrieb direkt studieren kann (y(t),z(t)).

Für die reale Echtzeitdarstellung wurden dann die Integrierer in die langsame Rechengeschwindigkeit (ZE = 1 Sekunde) versetzt, gleichzeitig aber durch eine Hilfsschaltung die jeweilige Lage des Trampolins erzeugt, sodaß der Bewegungsvorgang wie in Abbildung 1, jedoch dynamisch, einen ansprechenden optischen Eindruck des Springens und des Schwingvorganges vermittelte.

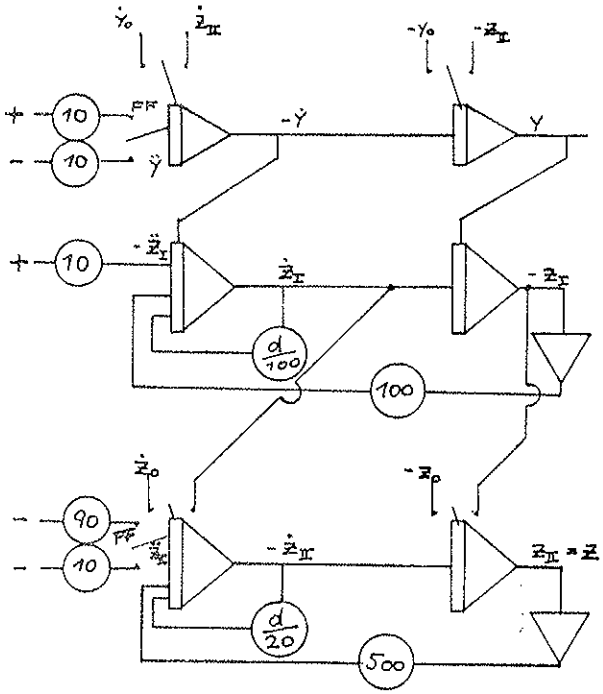


Abbildung 3

Unskalierter Schaltplan zur Erzeugung von y und z

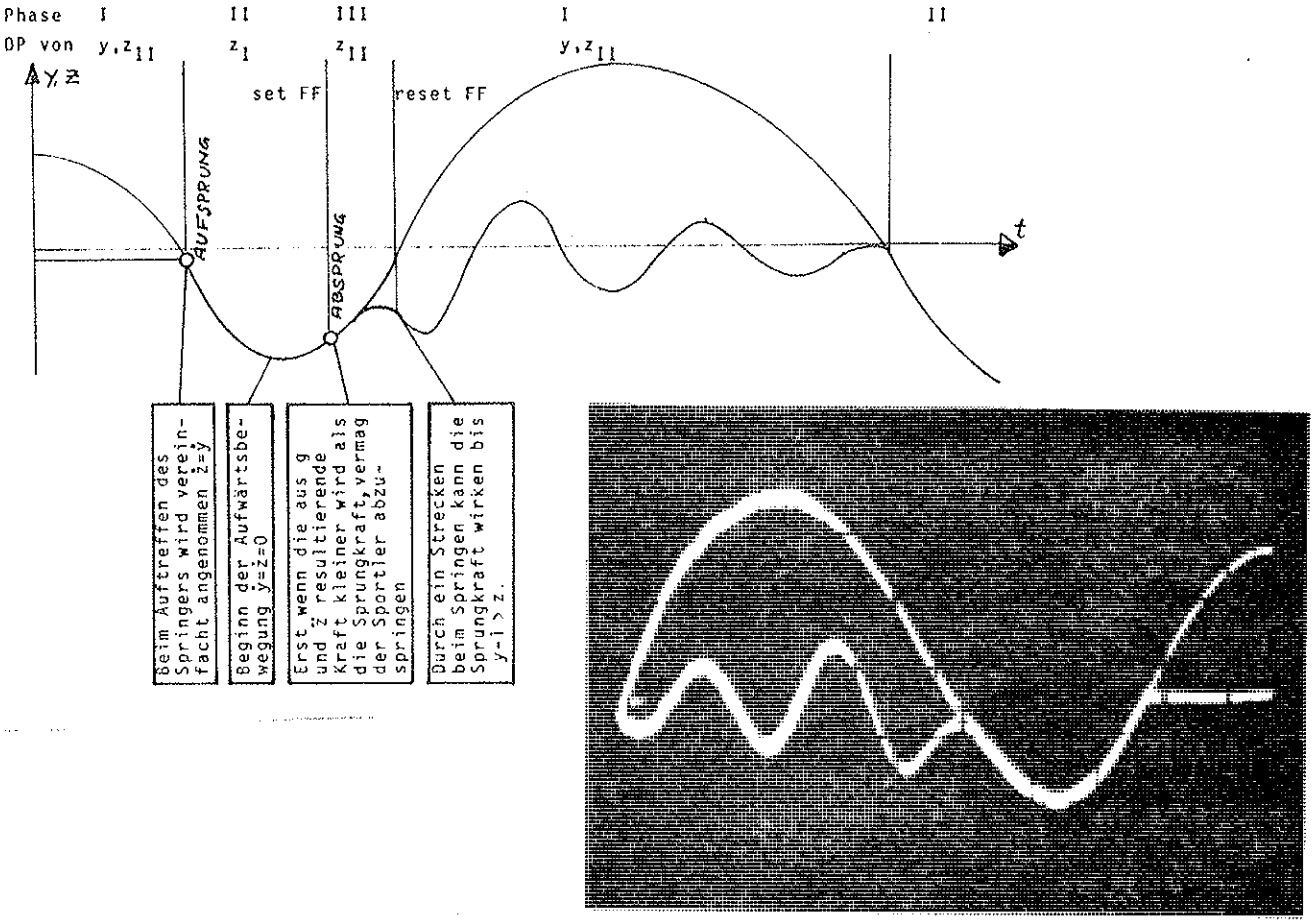


Abbildung 2

Schematische Darstellung des Bewegungsablaufes