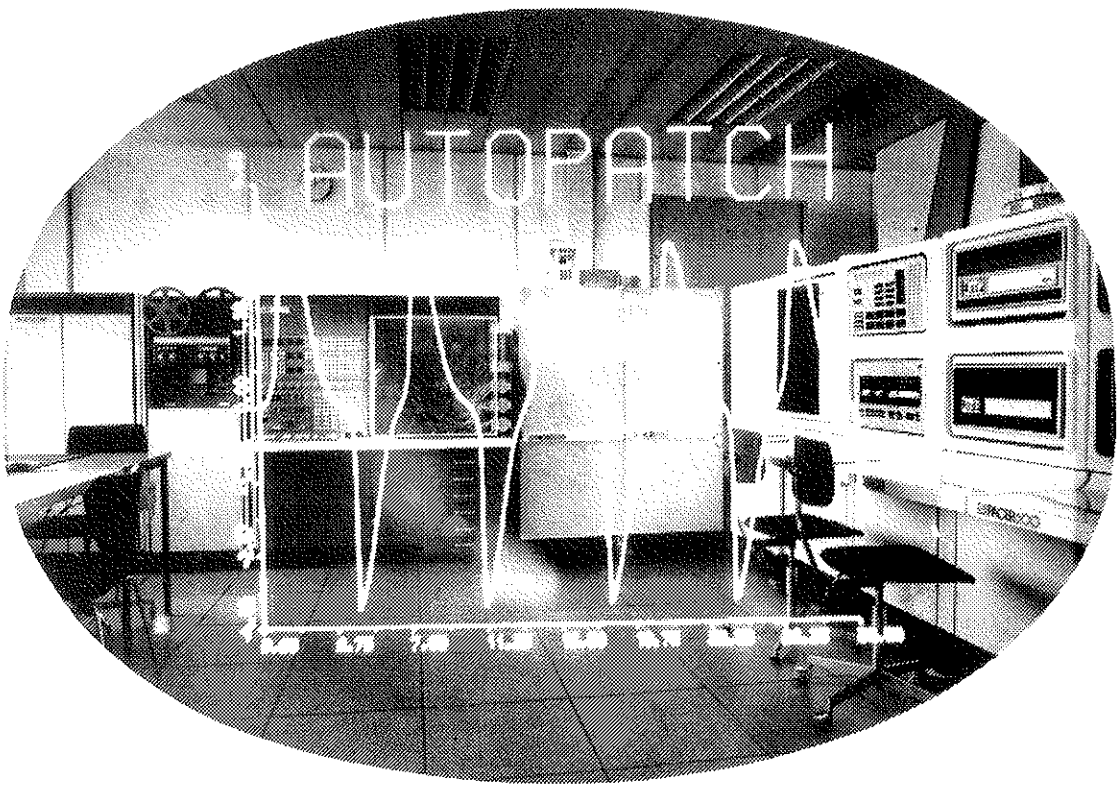
Interface

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

Nummer 15/16
Juni 1980



Das AutoPATCH-Zeitalter hat begonnen

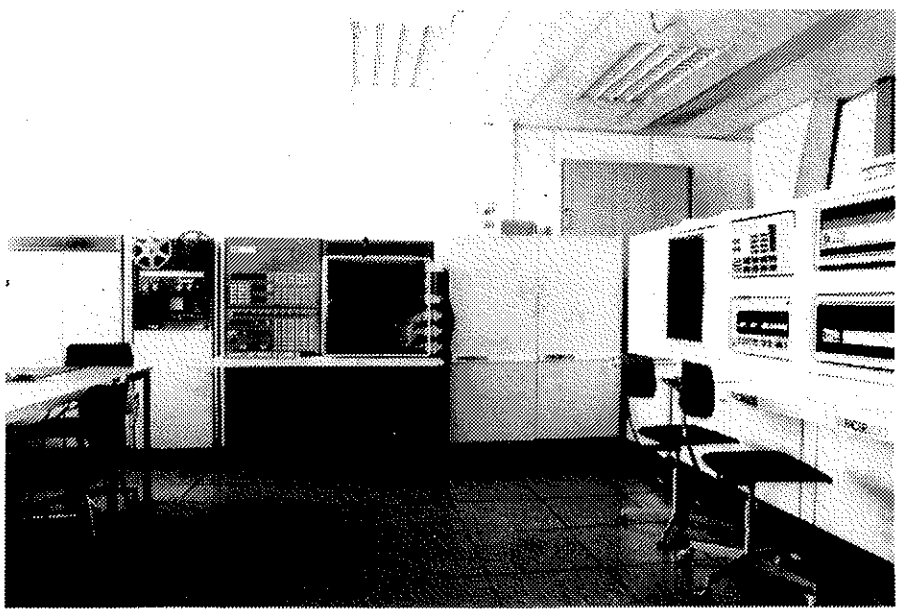
INHALTSVERZEICHNIS

DAS AUTOPATCH-ZEITALTER HAT BEGONNEN !

Wie im letzten INTERFACE berichtet, wurde die Lieferung und Installation der Hardware für die AutoPATCH-Erweiterung der Hybridrechenanlage am 19.11.1979 von der Firma EAI erfolgreich abgeschlossen. Ende Jänner wurden auch die Leistungstests erfolgreich beendet. In einer auch für uns selbst überraschend kurzen Zeit konnte die geplante Erweiterung des hybriden Sprach-Prozessors HYBSYS,zur automatischen Erstellung von Schaltverbindungen in einer vollkommen maschinenunabhängigen Syntax von Terminals aus,realisiert werden. Die Implementierung erfolgte Ende Februar 1980 (Beschreibung dieser Software siehe Seite 23).

Dank der Unterstützung von 19 Universitätsprofessoren der Technischen Universität Wien und der Finanzierung durch das Bundesministerium für Wissenschaft und Forschung ist also seit März 1980 an der Technischen Universität Wien ein neues Zeitalter der hybriden Rechentechnik angebrochen. Die Zeiten des händischen Steckens sind endgültig vorbei, der Übergang zu einem hybriden Time-Sharing-Zugriff zum analogen Parallel-Prozessor kann zügig in Angriff genommen werden.

Ein Schwerpunkt dieser Nummer des INTERFACE liegt daher in der Beschreibung der neuen Hardwarekonfiguration und der neuen hybriden Software.



Seit Anfang dieses Jahres kann also das AutoPATCH-System in einer vollkommen maschinenunabhängigen Syntax vom Terminal aus bedient werden. Wie erste praktische Erfahrungen zeigen, wird dadurch der Zugang zu unserem spezialisierten Rechnersystem auf eine komplett neue Stufe gestellt. Alle Vorteile des Hybridrechnens, wie parallele Arbeitsweise, enge Mensch-Modell-Beziehung, hochgradige Interaktivität, hohe Simulationsgeschwindigkeit, Anschaulichkeit der Lösung, hohe Rechengeschwindigkeit (wegen der parallelen Verarbeitung unabhängig von der Problemgröße) und rasche Modellfindung bzw.-bewertung können nun einem nicht mehr speziell und lange auszubildenden Benutzerkreis zur Verfügung gestellt werden. Das derzeit installierte System vermeidet praktisch alle bisher in der hybriden Rechentechnik aufgetretenen maschinenabhängigen Probleme, die eine breite Nutzung dieses für bestimmte Problemklassen so gut geeigneten Hilfsmittels verwehrten:

- Der Benutzer muß nun nicht mehr von einem Koppelplan und Potentiometerlisten ausgehen, sondern kann sich ausschließlich auf seine Modellgleichungen konzentrieren.
- Das Programmieren durch händisches Stecken fällt weg, das Erstellen der Schaltverbindungen geschieht in wenigen Sekunden - früher Stunden oder Tage - und ist zusätzlich fehlerfrei.
- Die komplizierte Aufgabe der Skalierung wird von der Software automatisch durchgeführt.
- Probleme können auf Massenspeicher abgelegt und jederzeit in Sekundenschnelle abgerufen und aufgebracht werden.
- Für ein weites Feld von hybriden Anwendungsbeispielen werden alle Echtzeitprobleme der Synchronisation dem Benutzer von der Software abgenommen.
- Das System erstellt alle nur erdenklichen graphischen und alphanumerischen Dokumentationen.

Derzeit sind Projekte beim Fonds zur Förderung der wissenschaftlichen Forschung und beim Bundesministerium für Wissenschaft und Forschung eingereicht, deren Ziel es ist, daß, aufbauend auf der vorhandenen Hard- und Software ein hybrides Time-Sharing-System konzipiert und realisiert wird, das einen interaktiven Terminalzugriff von bis zu acht Echtzeitbenutzern, die gleichzeitig an verschiedenen graphischen Terminals arbeiten, ermöglichen soll. Da ein Wechsel zwischen fertigen Problemen nunmehr in wenigen Millisekunden durchgeführt werden kann,sollen in einem neuartigen sowohl parallelen als

auch seriellen Verfahren den verschiedenen Benutzern die Resourcen des kompletten Hybridsystems so rasch zur Verfügung gestellt werden, daß der Eindruck des alleinigen Zugriffs entsteht. Diese Terminals sind nicht mehr an den Aufstellungsort des Hybridrechners gebunden.

Darüber hinaus soll ein Softwaresystem für die Entwicklung dynamischer Modelle an einem hybriden Time-Sharing-System speziell für den Einsatz in der Lehre entwickelt werden. Ein oder mehrere Studenten sollten gleichzeitig an Simulationskonsolen (graphische Rasterdisplays Hewlett & Packard HP 2648A) ihre Differentialgleichungen in der gewohnten mathematischen Notation eingeben können, ohne sich im geringsten um die Besonderheiten digitaler bzw. hybrider Simulationssprachen oder um die noch weit schwierigeren Details der analogen und hybriden Datenverarbeitung kümmern zu müssen.

Bei der Realisierung dieses Projekts werden der dynamischen Modellbildung und Simulation im Ausbildungsprogramm zahlreicher Studienrichtungen endlich ein fester Platz eingeräumt werden können. Bisher waren die verschiedenen Ausbildungsgänge oft schon mit dem Erlernen der schwierigen Simulationstechniken ausgefüllt. Das hat dazu geführt, daß die Vorteile dynamischer Simulationen komplexer technischer, biologischer sowie ökonomischer Systeme den meisten Studenten während ihres gesamten Studiums nicht anschaulich nähergebracht werden konnten. Das vorgeschlagene System wird es erlauben, daß in relativ kurzer Zeit wirklichkeitsnahe Modelle - die auch in einem direkten Zusammenhang mit experimenteller Hardware stehen - entwickelt werden können. Für die Bedienung dieses Systems wird nicht einmal die Kenntnis einer Programmiersprache erforderlich sein. Der Benutzer soll sich voll auf den schwierigsten Teil seiner Problemstellung, die Modellfindung, konzentrieren können.

W. Kleinert

ÖFFNUNGSZEITEN

Die Hybridrechenanlage ist von

Montag bis Freitag
von
8 Uhr bis 20 Uhr

geöffnet.

In den Ferialzeiten (30.6.80 - 26.9.80) ist nur ein eingeschränkter Betrieb von

Montag bis Freitag
von
8 Uhr bis 16 Uhr

möglich.

SOFTWARE

Virtuelle Felder

Um mit sehr großen Datenmengen arbeiten zu können, ohne dabei auf den internen Aufbau von Datenfiles (Anzahl der Worte pro Record) Rücksicht nehmen zu müssen, ist es möglich, große Felder, die den vorhandenen Kernspeicherplatz überschreiten, auf Platte zu errichten und zu dimensionieren. Die Felder, dreidimensionale Matrizen, werden durch den Aufruf der Routine DEFINE dimensioniert und können auch von später laufenden Programmen unter den definierten Namen angesprochen werden. Zum Schreiben und Lesen von diesen Feldern stehen die Routinen TOCORE und OFCORE zur Verfügung, die bei Verwendung von mehreren Feldern immer dasjenige Datenfeld ansprechen, dessen Name zuletzt im DEFINE Statement initialisiert wurde, d.h. sobald sich die Befehle auf ein neues Datenfeid beziehen sollen, muß vorher durch Aufruf von DEFINE softwaremäßig umgeschaltet werden.

Es ist auch möglich, ein bereits definiertes Feld durch DEFINE mit veränderten Indexgrenzen aufzurufen, sodaß die abgespeicherten Daten umstrukturiert ausgelesen werden können. Das Produkt der drei Indexgrenzen muß aber dabei kleiner gleich dem Produkt der drei Indexgrenzen beim ersten aufgetretenen DEFINE-Statement für dieses Feld sein.
Bei den Routinen TOCORE und OFCORE muß mit drei Indizes dasjenige Feldelement angegeben werden, ab dem die Daten abgespeichert werden, sowie die Anzahl der Worte, die geschrieben oder gelesen werden. Dabei ist zu beachten, daß ein REAL-Wert zwei 16-Bit-Worte und DOUBLE PRECISION oder COMPLEX Werte vier 16-Bit Worte benötigen.
Um die Datenbehandlung rasch zu ermöglichen, wird im Kernspeicher ein Buffer von 264 Worten, der jeweils die zuletzt angesprochene Page enthält, angelegt. Damit die Plattenzugriffe (Austausch der Page) möglichst selten sind, ist es am günstigsten, den ersten Feldindex am schnellsten zu variieren. Eine Page wird erst dann wieder auf Platte geschrieben, wenn ein Feldelement, das außerhalb des im Kernspeicher befindlichen Bereichs liegt, angesprochen wird. Dadurch muß sowohl vor dem Ausstieg aus dem Programm als auch beim Ansprechen eines anderen Feldes der Update der im Kernspeicher befindlichen Page vom Benutzer selbst (z.B. durch Lesen eines Feldelementes, das außerhalb der Page liegt) gesichert werden.

Datenübertragung vom EAI PACER 600 zur CDC CYBER 74

Die Routinen zum Datentransport von der Hybridrechenanlage PACER 600 zur Digitalrechenanlage CYBER 74 wurden in analoger Weise wie die Routinen zum Datentransport vom Digitalrechner zum Hybridrechner (siehe INTERFACE 15) fertiggestellt. Beschreibungen der Routinen DIOPEN, DIPUTI, DIPUTR, DIPUTC, DICLOS am Hybridrechner und der Routinen HYPOSI, HYGETI, HYGETR, HYGETC am Digitalrechner sind an der Abteilung Hybridrechenanlage erhältlich.

GDP1

Seit kurzem steht an der Hybridrechenanlage ein umfassendes Programmpaket zur numerischen Berechnung und graphischen Darstellung der Lösungen von Anfangswertproblemen gewöhnlicher Differentialgleichungen zur Verfügung. Ein ausführlicher Artikel ist auf Seite 44 zu finden.

HARDWARE

Die wichtigste Neuigkeit ist wohl die Installation der AutoPATCH-Erweiterung, deren Beschreibung den größten Teil dieser Nummer des INTERFACE ausmacht (Artikel auf Seite 11).

HYBSYS

Für die Bedienung des AutoPATCH-Systems wurde eine neue Version des hybriden Prozessors HYBSYS (siehe Seite 23) entwickelt. Diese Version erlaubt sowohl den konventionellen Steckbrett-Betrieb als auch das automatische Erstellen der Schaltung unter Verwendung der Schaltmatrix. Die nächste Version wird allerdings nur mehr das hybride Rechnen mit dem AutoPATCH-System unterstützen, sodaß wahrscheinlich ab dem kommenden Wintersemester bei Verwendung des hybriden Prozessors HYBSYS ein gewöhnlicher Steckbrett-Betrieb nicht mehr möglich sein wird.

kurse

RH3 HINWEISE FÜR FORTRAN-PROGRAMMIERER AN DER HYBRIDRECHENANLAGE

Termin: 1980/10/14, 1 Stunde
Vortragender: Dipl. Ing. F. Blöser

RH7 SOFTWAREUNTERSTÜTZUNG FÜR DIE BENÜTZUNG DES PACER 600 ALS PLOTTER SYSTEM

Termin: 1980/10/14 und 1980/10/15, je 3 Stunden
Vortragender: Dipl. Ing. F. Blöser

Die im folgenden angeführten Kurse werden nach Vereinbarung abgehalten:

RH1 GERÄTETECHNIK EAI PACER 600

RH2 BENÜTZUNG DES BETRIEBSSYSTEMS JCS/VS 8

RH6 EAI ASSEMBLER

RH9 EINFÜHRUNG IN DAS HYBRIDE AUTOPATCH-SYSTEM

RH10 DATENVERBINDUNG MIT CYBER 74

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.: 65-37-85/803 Durchwahl).

EAI COMPUTER USERS' GROUP MEETING IN WIEN

VOM 8. - 11. SEPTEMBER

Das diesjährige EAI Computer Users' Group Meeting wird vom 8. - 11. September 1980 in Österreich stattfinden.

Nach der Registrierung der Teilnehmer am Montag vormittag beginnt das Meeting mit der Präsentation unseres neuen PACER 600 Auto-PATCH-Systems und den ersten Vorträgen an der Technischen Universität Wien. Im Gegensatz zu früheren Meetings findet die Vorführung der Rechenanlage des Gastgebers diesmal am ersten Tag statt, da der eigentliche Veranstaltungsort das Alpenhotel Gösing ist, das ungefähr 120 km von Wien entfernt ist. In Gösing wird es jeden Tag möglich sein, on-line über graphische Terminals mit dem PACER 600 AutoPATCH-System zu rechnen. Donnerstag vormittag endet das Meeting mit einer gemeinsamen Busfahrt zurück nach Wien.
Beim Meeting werden nicht nur rein technische Berichte zu hören sein, sondern auch kurze Präsentationen über die Möglichkeiten des Einsatzes von EAI-Geräten, das Arbeiten mit diesen Geräten und die dabei auftretenden Probleme.

Ziel des Meetings ist es,

- Erfahrungen auszutauschen
- den Kontakt zwischen den Benutzern von EAI-Geräten zu fördern
- Fragen und Probleme zu behandeln
- stichhaltige Informationen von EAI zu erhalten.

Den Teilnehmern wird auch die Möglichkeit geboten, an Arbeitsgruppen teilzunehmen, bei denen Themen wie ECSSL, EAI 2000, PDP/EAI Systeme, Hardware, spezielle Software etc. behandelt werden.

DIE NEUE AUTOPATCH-HARDWARE

Abb. 1 stellt ein Blockschaltbild der AutoPATCH-Erweiterung unseres Hybridrechners dar. Kernstück ist dabei die analoge Schaltmatrix mit 5120 integrierten analogen MOSFET-Schaltern.

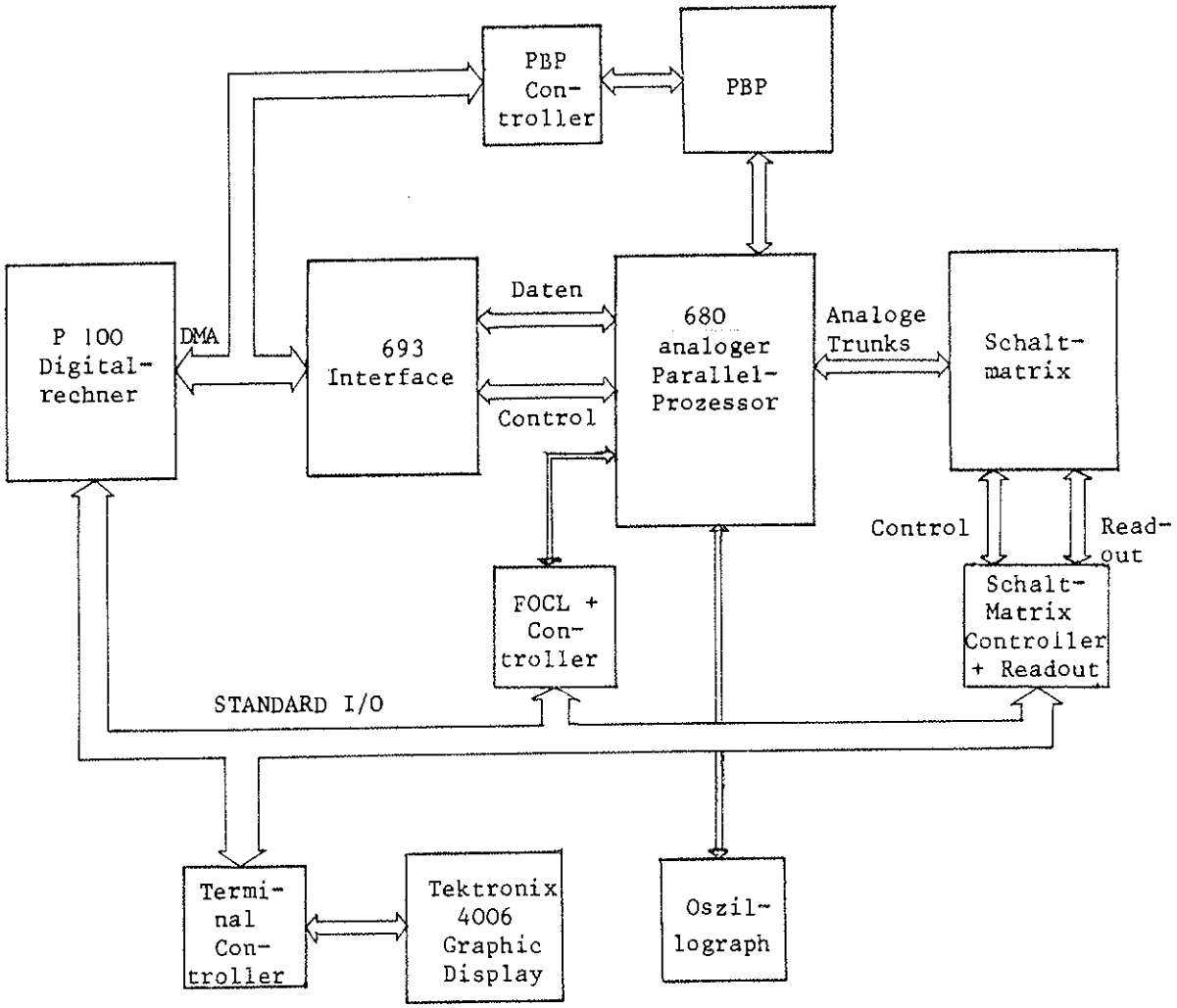


Abb. 1

Die aus zwei Submoduln mit je 64 Eingängen und 128 Ausgängen bestehende analoge Schaltmatrix ist über Trunk-Leitungen mit fest vorprogrammierten analogen Rechenelementen (Makros) über ein spezielles AutoPATCH-Steckbrett mit dem analogen Parallel-Prozessor EAI 680 verbunden. Weiterhin ist der Analogrechner über Control- und Datenleitungen mit dem Standardinterface 693 gekoppelt. Über einen speziellen Controller am Standard-I/O-Kanal kann die Schaltmatrix programmgesteuert konfiguriert und mit Hilfe eines eigenen Read-Out-Systems können die Matrix-ein- und -ausgänge ausgelesen werden. Ebenfalls über den Standardkanal werden die 64 zusätzlichen fest verdrahteten Operation-Controllines (FOCL) angesprochen, die

in der AutoPATCH-Konfiguration u.a. zur Programmierung der individuellen Integrierer-Zeitkonstanten verwendet werden (siehe Abb. 9). Ein eigener, an der Hybridrechenanlage selbst entwickelter logischer Prozessor (Pipelined Boolean Processor), ist mit vorprogrammierten Makros in der parallelen Logik des Analogrechners durch Spezialkabel verbunden. Er wird von einem eigenen Controller, der am hybriden direkten Speicherkanal angeschlossen ist, gesteuert. Zum gegenwärtigen Zeitpunkt ist der PBP allerdings noch nicht softwaremäßig in den hybriden Prozessor HYBSYS integriert.

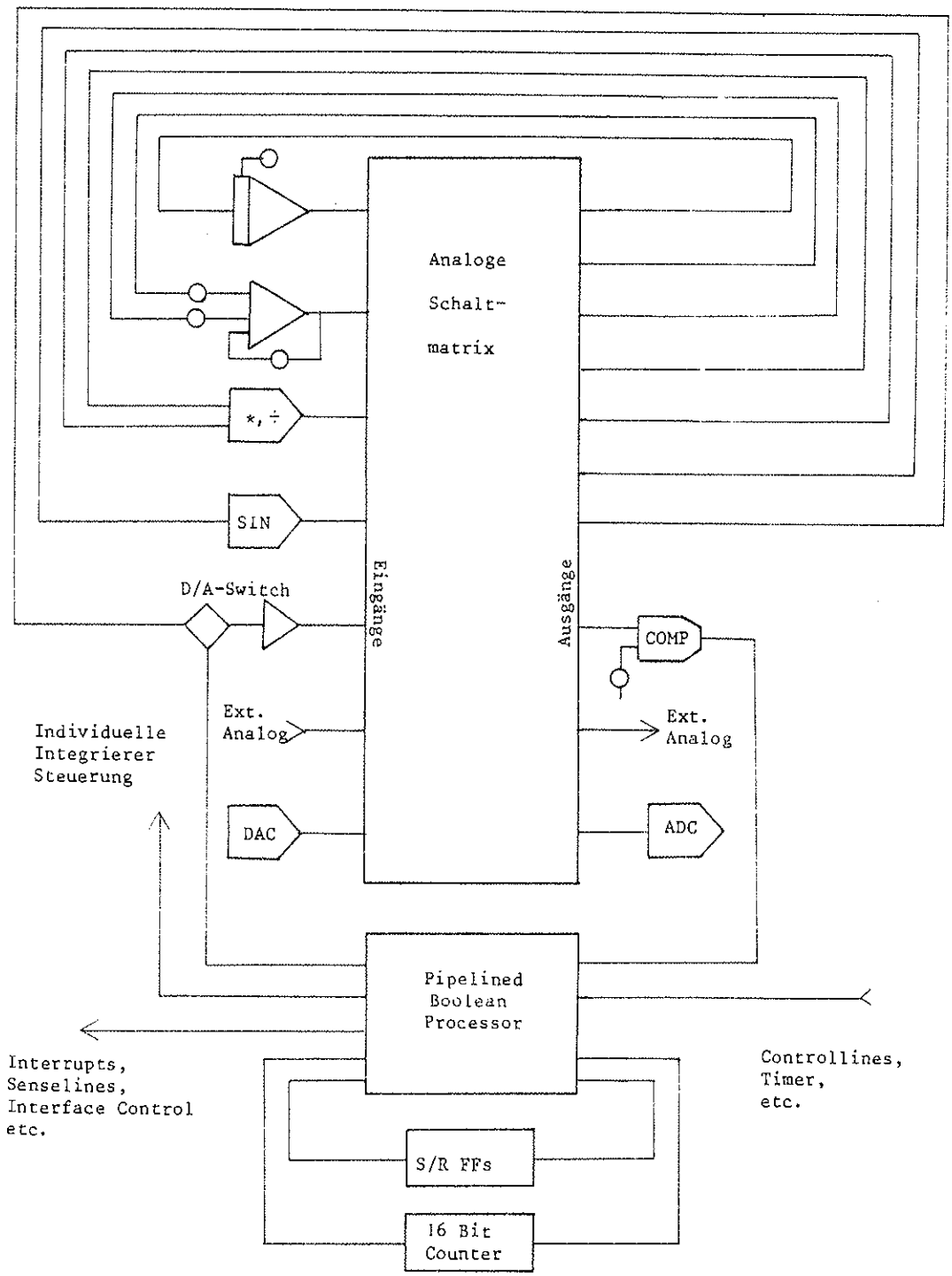


Abb. 2

Abb. 2 zeigt ein Prinzipschaltbild der Verbindungen von den Ein- und Ausgängen der analogen Schaltmatrix bzw. des logischen Prozessors zu den analogen bzw. logischen Rechenelementen des analogen Parallel-Prozessors, die zu bestimmten Funktionsmoduln (Makros) zusammengefaßt sind. Die dargestellten Makros haben nur beispielhaften Charakter.

Beschreibung der Schaltmatrix

Die bei uns installierte Schaltmatrix besteht aus zwei analogen Teilmatrizen mit je 64 Eingängen und 128 Ausgängen; die Matrizen sind durch je 8 Verbindungsleitungen miteinander gekoppelt (Interconnections). Das gesamte System ist daher in der Lage, 112 Eingänge auf 240 Ausgänge zu schalten.

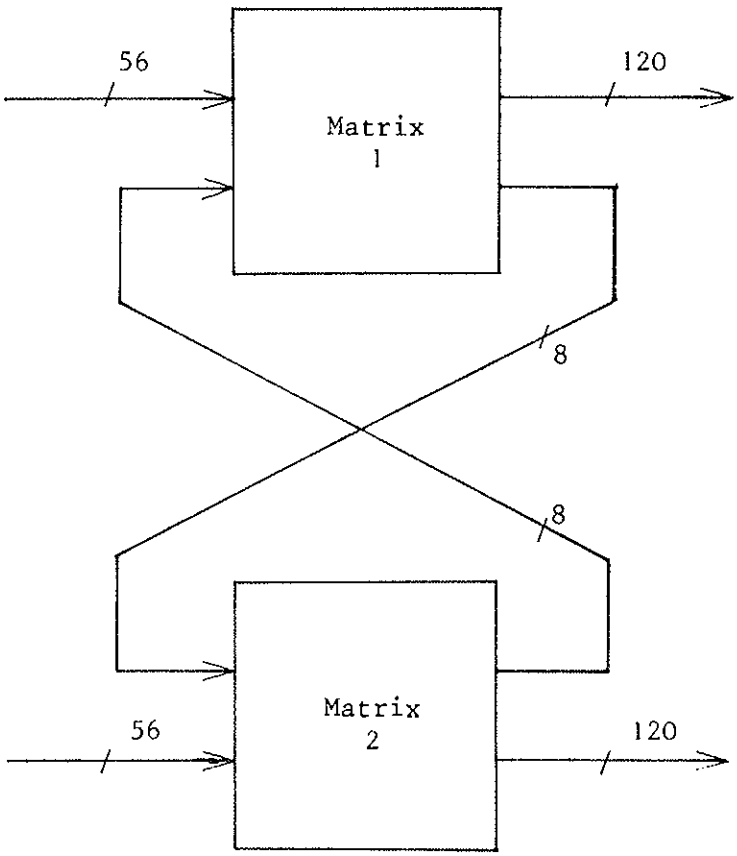
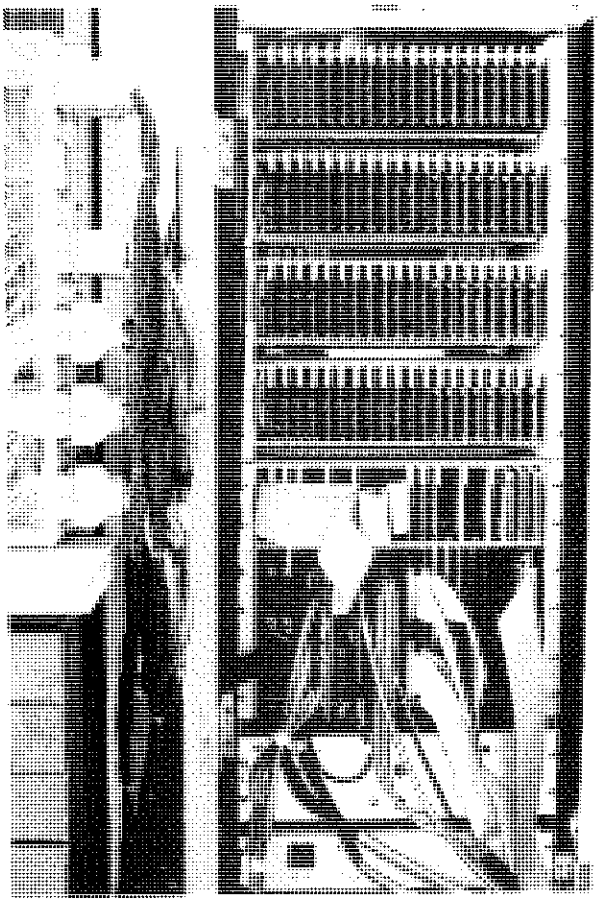


Abb. 3

Da die beiden Matrizen im Prinzip voneinander unabhängig sind, kann die Zahl der Interconnections beliebig definiert werden. Die gegenwärtige Konfiguration (8 Interconnections) wurde aufgrund einer Analyse der durchschnittlichen Komplexität der bei uns durchgeführten hybriden Simulationen festgelegt. Diese Konfiguration kann bei Bedarf in ähnlich einfacher Weise wie die verwendeten analogen Makros geändert werden.



Analoge Schaltmatrizen

Jede analoge Schaltmatrix ist ein in CMOS-Technologie aufgebautes dreistufiges Netzwerk (Hannauer'sches Schema), das 64 Eingänge mit 128 Ausgängen verbindet. Abb. 4 zeigt den Aufbau einer solchen dreistufigen Schaltmatrix.

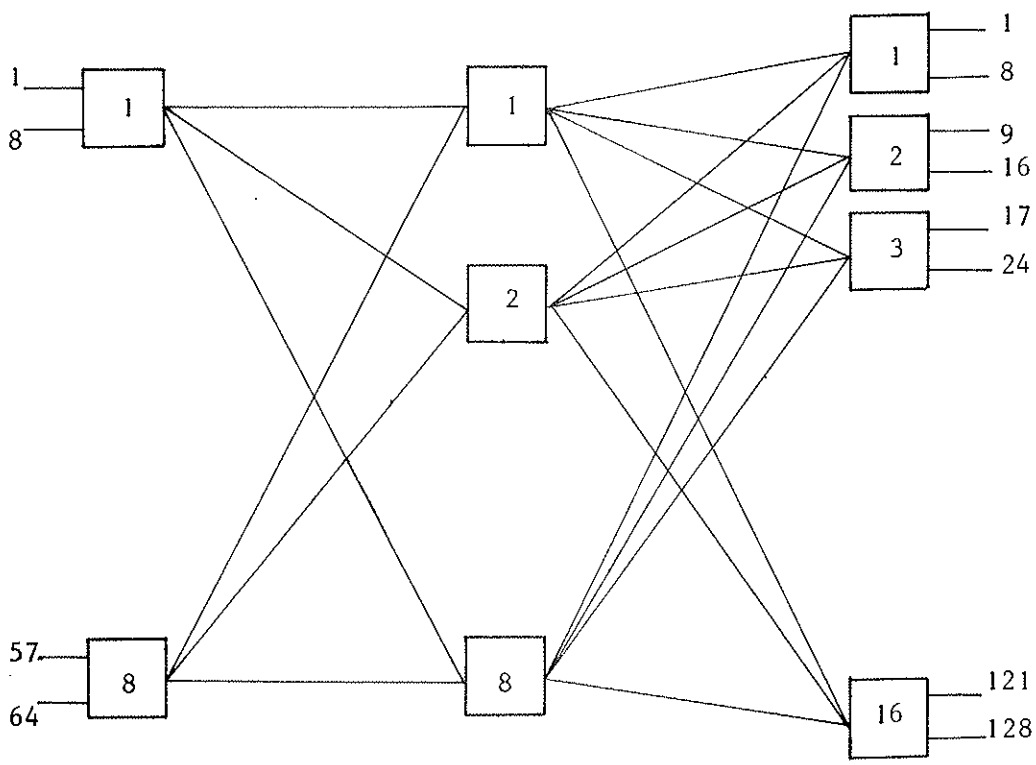


Abb. 4

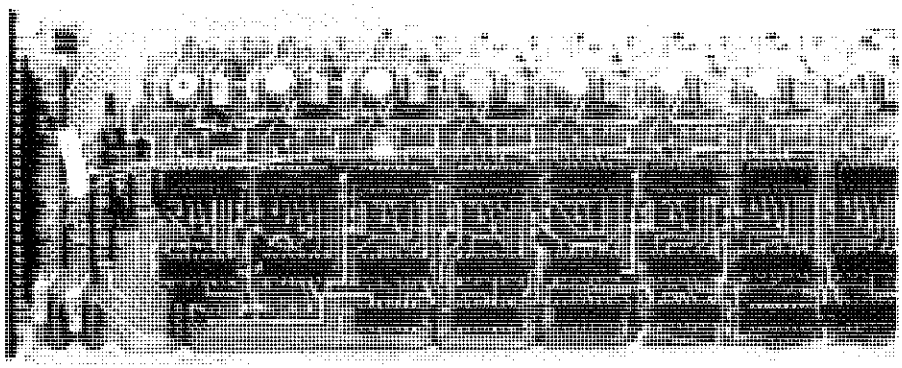
Die Ein- und Ausgänge sind zu quadratischen Blöcken der Größe 8 x 8 zusammengefaßt. In einem solchen Block kann jeder Eingang mit jedem Ausgang verbunden werden. Es gibt daher 8 Eingangs- und 16 Ausgangsblöcke. Die Mittelblöcke sind rechteckig von der Größe 8 x 16 und wurden aus Gründen der Modularität durch 8 x 8-Blöcke mit gemeinsamem Eingang realisiert.

Es gibt nur eine Verbindung zwischen einem Eingangs- und einem Mittelblock. Ebenso gibt es nur eine Verbindung zwischen einem Mittel- und einem Ausgangsblock. Daher ist der Verbindungsweg zwischen einem bestimmten Paar von Ein- und Ausgängen eindeutig definiert, sobald der dazugehörige Mittelblock spezifiziert ist.

Eine *Verbindungszuordnung* ist eine Menge, deren Elemente aus den benötigten Paaren von Ein- und Ausgängen bestehen. Die von EAI gelieferte Grundsoftware löst das sogenannte "Routing Problem", das darin besteht, bei einer gegebenen Verbindungszuordnung eine entsprechende Auswahl der Mittelblöcke vorzunehmen. Eine *Blockierung* der Schaltmatrix ist eine solche Schaltstellung, die die i-te Verbindung in einer Verbindungszuordnung sicherstellt, bei der aber für die i+1-te Verbindung keine Schaltmöglichkeit mehr existiert. Die vorhandene Software versucht nun, die Mittelblöcke so umzuordnen, daß die Blockierung aufgehoben ist. Ein Verbindungsnetzwerk heißt blockungsfrei, wenn durch Umordnung jede Verbindungszuordnung unabhängig von der Größe des benötigten Netzwerkes, dem Grad der erhaltenen Ausgangsverteilung (Fan-Out) und ihrer Komplexität realisiert werden kann. Obwohl theoretisch gezeigt wurde, daß die vorhandene Schaltmatrix wegen der verwendeten Matrix-Parameter nicht blockungsfrei ist, zeigt die praktische Erfahrung während der ersten zweieinhalb Monate AutoPATCH-Betrieb, daß alle Verbindungszuordnungen gelöst werden konnten.

Struktur der Schaltmatrix

Die Matrix basiert auf der Verwendung eines integrierten analogen 8 x 1 Multiplexer-Chips (DG 508). Auf einer Karte ist ein 8 x 8-Block, bestehend aus acht Multiplexern und dazugehöriger Steuerlogik (ebenfalls CMOS), realisiert.
(Siehe Abb. 5).



Ausgangsblock-Karte

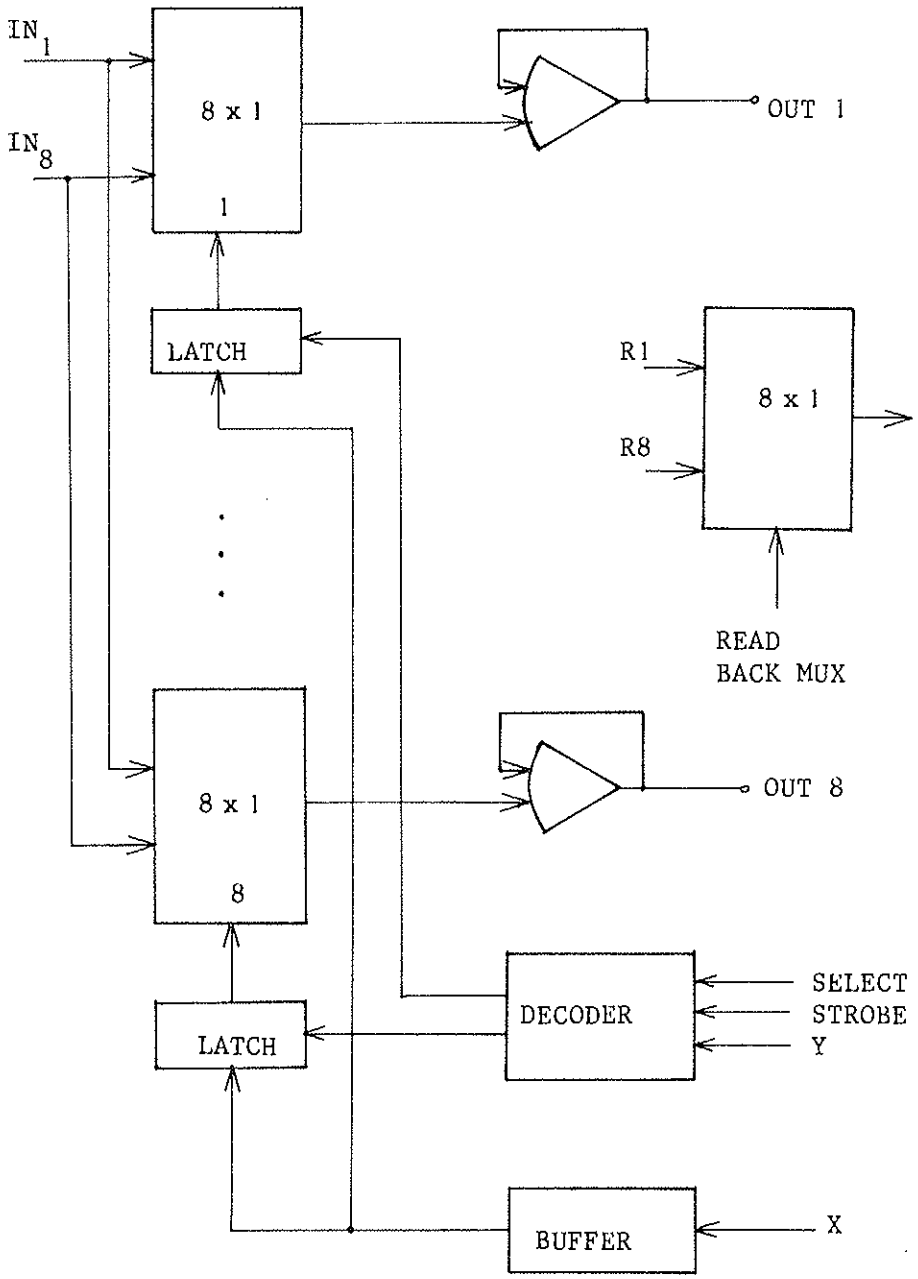


Abb. 5

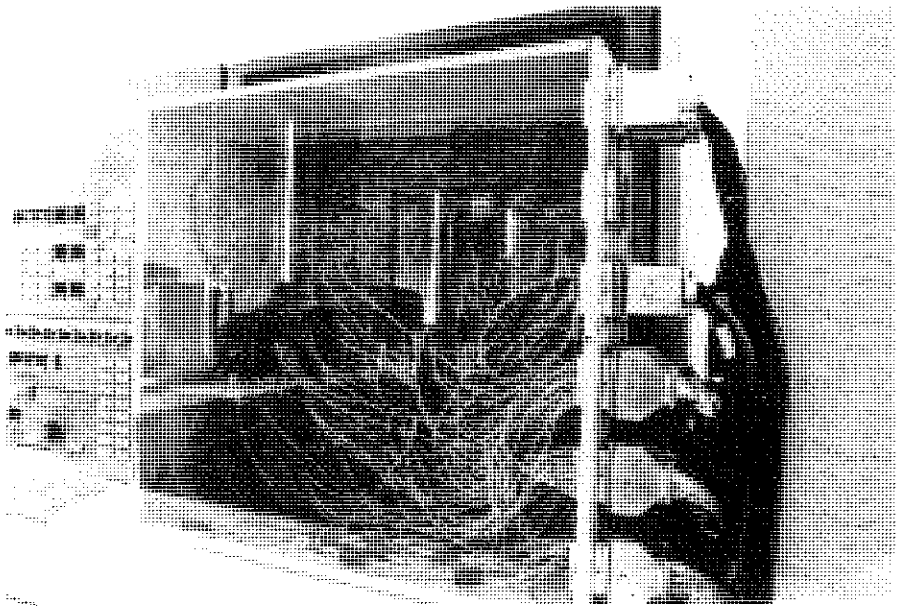
Die Ausgangsblöcke besitzen als Ausgangsstufe nicht-invertierende Operationsverstärker (LF356H) mit hoher Eingangs- und niedriger Ausgangsimpedanz, die als Spannungsfolger beschaltet sind. Eingangs- und Ausgangsblöcke besitzen einen zusätzlichen Multiplexer zum Auslesen der Ein- bzw. Ausgänge durch das Read-Back-System. Ein Mittelblock wird durch zwei 8 x 8-Karten mit gleichen Eingängen realisiert.

Der Verstärker LF356H ist in der Lage, Eingangssignale mit voller Amplitude (20 V p/p) bis annähernd 10 kHz verzerrungsfrei wiederzugeben. Für kleine Eingangssignale (2 V p/p) beträgt die Bandbreite zwischen 200 und 300 kHz. Die Offset-Spannung ist auf Werte kleiner als 100 µV einstellbar. Diese elektrischen Charakteristika sind für die üblichen Zeitkonstanten im Millisekundenbereich durchaus zufriedenstellend. Allerdings haben von uns durchgeführte Experimente gezeigt, daß bei Verwendung eines plug-in-kompatiblen Operationsverstärkers (Burr-Brown 3551) eine Bandbreite von 500 kHz für volle Amplitude und 2 MHz für kleine Signale erzielt

werden kann. Wir haben die Absicht, die Ausgangsstufen im Laufe des nächsten Jahres entsprechend umzubauen.

Das Read-Out-System verwendet wiederum eigene Multiplexer-Chips auf jeder Karte, die über weitere Multiplexer einem eigenen, nur der Switch-Matrix zur Verfügung stehenden ADC zugeführt werden. Dieser arbeitet mit einer Genauigkeit von 12 Bit, besitzt eine Konversionszeit von 40 µsec und wird von der System Software über den Standard-I/O-Kanal angesprochen.

Die Switch-Matrix ist über Subminiatur-Koaxkabel und -stecker mit einem speziellen AutoPATCH-Steckbrett am 68Ø verbunden.



AutoPATCH-Steckbrett

Für jede Matrix sind ein Eingangs- und zwei Ausgangskabel vorhanden. Diese Methode erlaubt sowohl einen AutoPATCH- als auch einen konventionellen Steckbrettbetrieb. Darüber hinaus gibt es ein mit Spannungsteilern versehenes Testkabel, das mit dem jeweiligen Eingangskabel der Switch-Matrizen verbunden werden kann und den vollautomatischen Ablauf einer äußerst ausführlichen Diagnostik-Software von EAI erlaubt, mit deren Hilfe sehr rasch eventuell fehlerhafte Komponenten in der Matrix lokalisiert werden können. Diese Diagnostik läuft aus Gründen der präventiven Wartung täglich. Das AutoPATCH-System ermöglicht aber nicht nur eine effektive diagnostische Software für die Switch Matrix selbst, sondern es werden an der Hybridrechenanlage auch sehr effiziente - in der HYBSYS-Syntax formulierte - Testprogramme für die analogen Rechenelemente entwickelt, die rasch über defekte Elemente Aufschluß geben (siehe Seite 31).

Struktur des Pipelined Boolean Processors

Bei dem an der Hybridrechenanlage selbst entwickelten Pipelined Boolean Processor (PBP) handelt es sich um ein dreistufiges digitales Netzwerk, dessen interne Verbindungsstruktur prinzipiell der einer analogen Schaltmatrix (siehe Abb. 4) entspricht. Der Hauptunterschied liegt darin, daß die ebenfalls 8 x 8 dimensionierten Ausgangsblöcke nicht bloß aus 8 Multiplexern bestehen, sondern durch die Verwendung eines 256 Byte umfassenden RAM-Speichers jedes Output-Pin bei entsprechender Programmierung des Speichers eine beliebige Boolesche Funktion aller acht Eingänge des betreffenden Ausgangsblocks repräsentieren kann. Durch Erstellen einer geeigneten Schaltverbindung kann somit jeder Ausgang des PBP eine beliebige Boolesche Funktion

von acht beliebigen Eingängen darstellen.

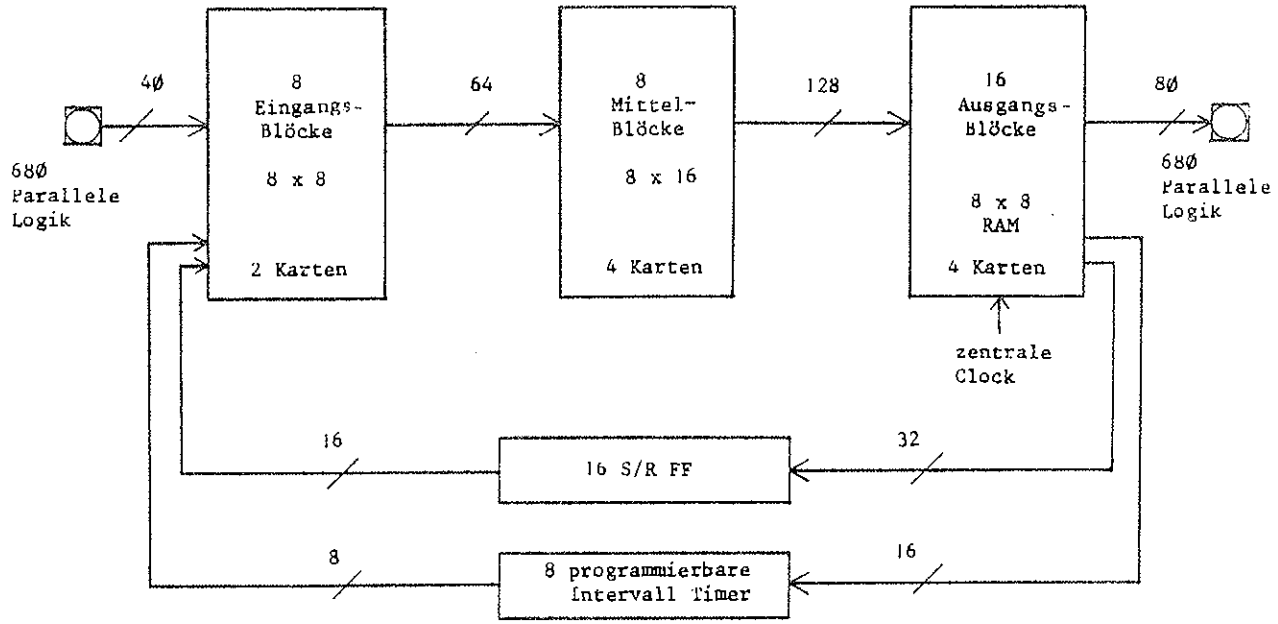


Abb. 6

Abb. 6 zeigt ein allgemeines Blockdiagramm des PBP. Die Eingangs- und Mittelblöcke sind entsprechend der analogen Schaltmatrix aufgebaut und bestehen aus je acht 8 x 1-Multiplexern mit gemeinsamem Eingang. Die Verbindung zwischen den Eingangs-, Mittel- und Ausgangsblöcken erfolgt wie in Abb. 4. Ein Teil der Aus- und Eingänge ist durch feste Rückkopplungen belegt. Es können 16 Set/Reset-Flip-Flops (74LS107) und acht programmierbare 16 Bit Intervall-Timer (INTEL 8253) für Aufgaben der sequentiellen Logik programmiert werden. Die Intervall-Timer können als einfache Zähler, programmierbare Monostabile, Rechtecksgeneratoren oder Differenziatoren mit variabler Zeitverzögerung eingesetzt werden. Die Ausgangsblöcke sind zusätzlich noch mit zentral getakteten Registern versehen und können synchron mit dem zentralen 1MHz Takt der parallelen Logik des 68Ø laufen.

Bei dem PBP handelt es sich also um einen getakteten logischen Parallel-Prozessor, der in der Lage ist, komplexe Aufgaben der kombinatorischen und sequentiellen Logik äußerst rasch zu bewältigen. Für sich genommen, könnte der PBP bei geeigneter Rückkopplung auch für die Emulation von 16 Bit Mini- bzw. Mikrocomputerzentraleinheiten verwendet werden.

In unserer AutoPATCH-Konfiguration sind die Ein- und Ausgänge des PBP mit festen Funktionen der parallelen Logik des 68Ø verbunden. Als Eingänge dienen Komparatorausgänge, Controllines, die Ausgänge des digital programmierbaren ABCD-Timers (siehe INTERFACE 12), die zentralen Frequenzen sowie Interface Kontrollsignale. Mit den Ausgängen können D/A-Switches, die OP-Eingänge der einzelnen Integrierer, T/S-Einheiten, Interfacekomponenten und die Real-Time-Clock gesteuert werden. Darüber hinaus werden auch alle acht General Purpose Interrupts und alle acht Senselines bedient.

Der PBP ist mit speziellen Twisted Pair Kabeln ebenfalls mit dem AutoPATCH-Steckbrett verbunden. Seine Logik ist direkt kompatibel mit der Logik des 68Ø. Jeder logische Ausgang im 68Ø besteht aus TTL-kompatiblen Line Drivern, die auch bei größerer Kabellänge sehr gutes Übertragungsverhalten in Verbindung mit den Low Power Schottky Eingängen auf Seiten des PBP besitzen. Andererseits sind auch die verwendeten Ausgangsstufen des PBP (74LS374, Oktalregister mit bustreibenden Tristate Ausgängen) sehr gut in der Lage, Leitungen mit hoher Kapazität gegen die niederohmigen Eingänge der 68Ø-Logik zu treiben.

Ein auf einer dreistufigen Schaltmatrix basierender paralleler Logik-Prozessor mit Speichern in den Ausgangsblöcken wurde bereits von A. Asthana und J. Shotliff (A Parallel Logic Processor for Automatically Patched Hybrid Computers, 1977) vorgeschlagen. Der an der Hybridrechenanlage aufgebaute PBP arbeitet aber intern mit einem anderen Verfahren. Um die physischen Ausmaße möglichst gering zu halten und doch eine genügend große Schnelligkeit zu erzielen, wird im PBP intern ein Time-Sharing für die einzelnen 8 x 8-Blöcke durchgeführt, wobei im Pipelining-Verfahren das Time-Sharing der Eingangs-, Mittel- und Ausgangsblöcke simultan erfolgt. So werden je vier 8 x 8-Blöcke auf einer Karte im Europa-Karten-Format zusammengefaßt.

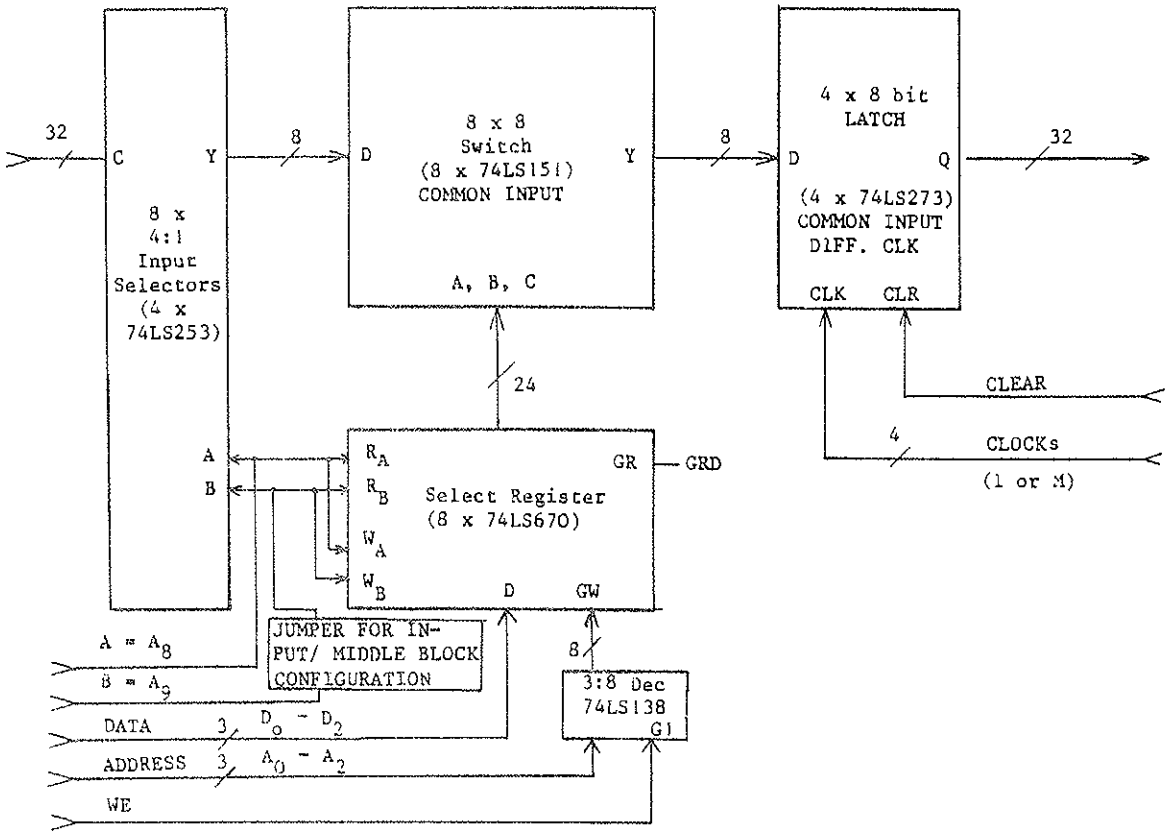


Abb. 7

Abb. 7 zeigt das Blockschaltbild einer Eingangsblock - Karte, auf der vier 8 x 8-Eingangsblöcke realisiert sind. Die 32 Eingänge liegen an 4:1 Input Selektoren, deren Ausgänge auf den nur einfach vorhandenen 8 x 8-Switch Block gelegt werden. Das zentrale Timing steuert nun die Adressbits A und B sowie vier indi-

viduelle Clocks derart, daß mit jedem Subzyklus die Eingangsselektoren auf einen der vier Eingänge geschaltet und die entsprechenden Informationen für den 8 x 8-Switch Block aus einem 4 x 4-Register-File geladen werden. Am Ende des Subzyklus werden die Ausgänge des Switch Blocks auf eines der vier 8-bit Ausgangslatches gelegt. Jede solche Karte kann durch einen einfachen Jumper , der das Weiterschalten der Input-Selektoren unterbindet, ohne parallele Verdrahtung zu einem zweifachen 8 x 16-Mittelblock umkonfiguriert werden.

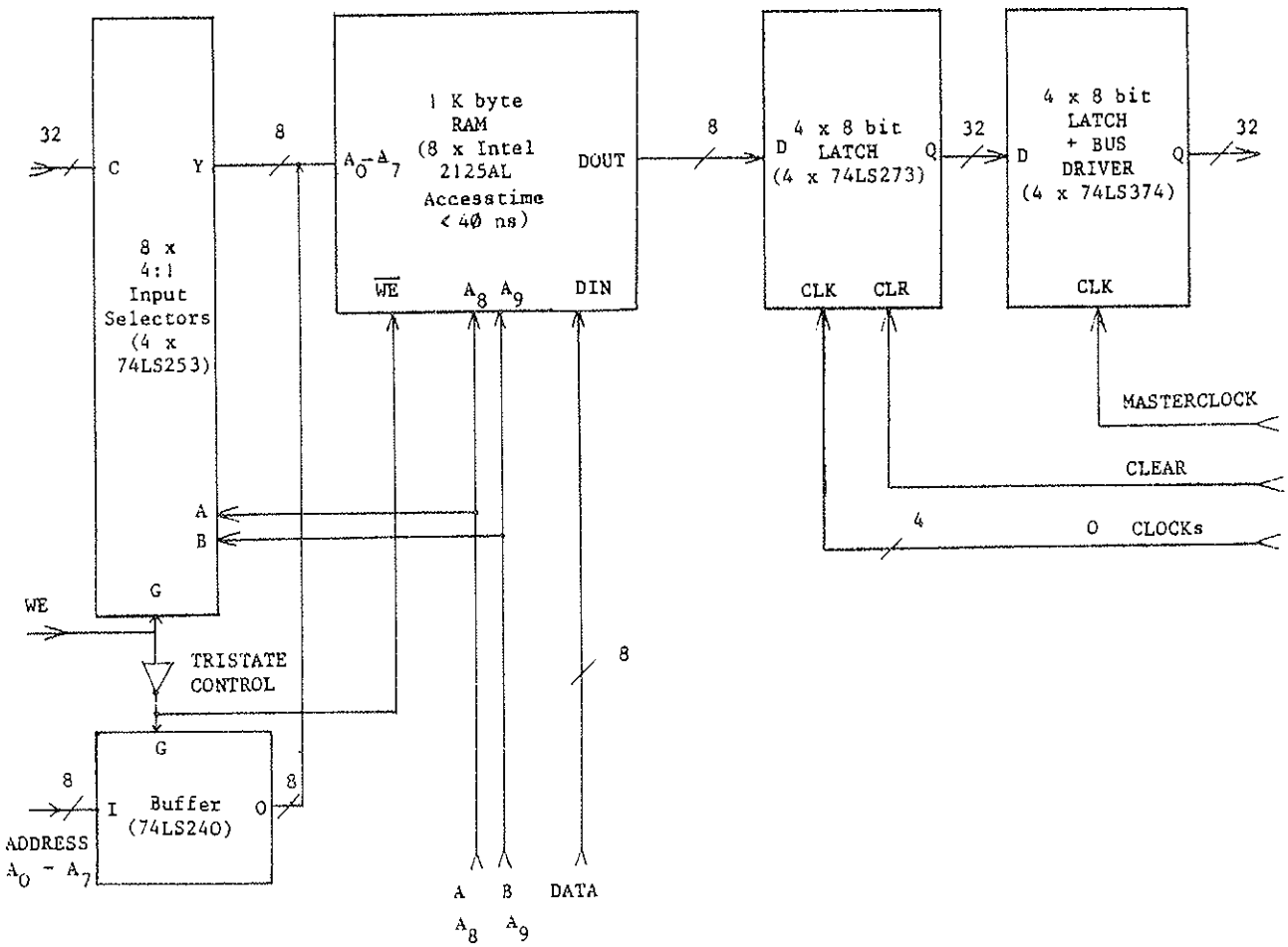
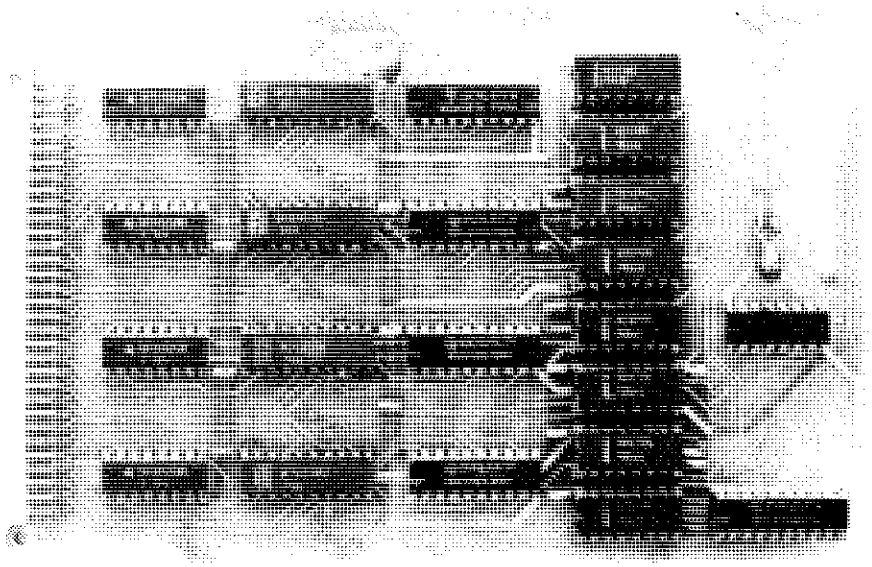


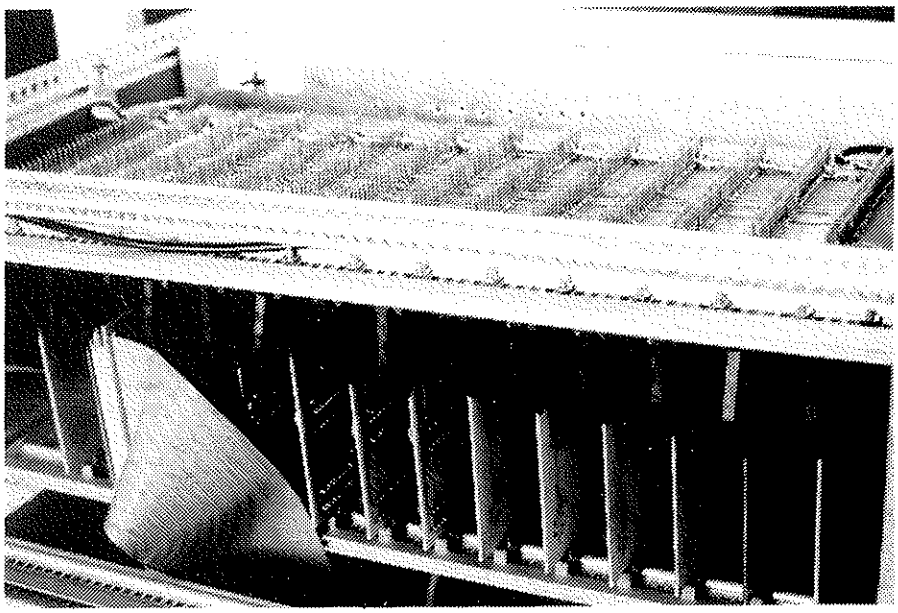
Abb. 8

Abb. 8 zeigt das ganz ähnlich aufgebaute Blockschaltbild für eine vierfach 8 x 8-Ausgangsblock-Karte. Die vier 256-RAM-Blöcke sind durch 1K x 1 statische RAM mit einer extrem schnellen Zugriffszeit von weniger als 40 nsec realisiert. Die Ausgangslatches sind doppelt ausgeführt, wobei das zweite zentral getaktete Register gleichzeitig als Bus Driver dient.



RAM-Karte

Die Steuerinformation für die Schalter der Eingangs- und Mittelblöcke sowie die Wahrheitstafeln für die Ausgangsblöcke können über einen speziellen Controller am direkten Speicherkanal sowie eine Interface-Karte im PBP geladen werden. Der gesamte PBP ist auf 12 Eurokarten in einem 19 Zoll-Einschub untergebracht. Wegen der vielen Ein- und Ausgänge mußte ein Doppel-Bus-System entworfen werden, wobei der eine Bus nur zur Verdrahtung der Verbindungen zwischen den Eingangs-, Mittel- und Ausgangsblöcken sowie der Aufnahme der steckbaren Ein- und Ausgangskabel dient. Die Spannungsversorgung sowie alle Steuersignale zum Laden des PBP erfolgen über einen zweiten Bus, der, unter 90^{o} angebracht, über AMP Zero-Insert-Force (ZIF) Stecker mit den Karten verbunden ist.



Pipelined Boolean Processor

Die Logik zur Steuerung der Time-Sharing-Subzyklen ist mikroprogrammiert und verwendet einen internen 20 MHz Takt (Motorola LOCO II).

Der DMA-Controller für den PBP erlaubt zufällige und sequentielle Adressierung der PBP-Speicher. Damit besteht auch bei einem zukünftigen Time-Sharing-Betrieb die Möglichkeit, Teile des PBP mit neuer Information zu laden, während andere Teile ihre normalen Operationen aufrecht erhalten.

Abb. 9 zeigt das Blockschaltbild für die 64 zusätzlichen Operation Controllines (FOCL), die auf einer Play-Pack-Karte aufgebaut sind. Der jeweilige Status der 64 Controllines wird wegen der optimalen Ansteuerung nicht nur in einem mit 74LS173 aufgebauten 64-Bit-Register, sondern auch in einem 16 x 4-Speicher mit Read/Modify/Write-Logik abgespeichert. Die FOCLs sind über Trunkkabel mit dem 68Ø Steckbrett verbunden. Sie können mit FORTRAN-Subroutines (SETCL(I), RSTCL(I)) gesetzt bzw. zurückgesetzt werden. Im AutoPATCH-System sind sie fest mit den individuellen Steuerungen der Integrierer-Zeitkonstanten verbunden und werden von HYBSYS zur automatischen Anwahl der Zeitkonstanten verwendet.

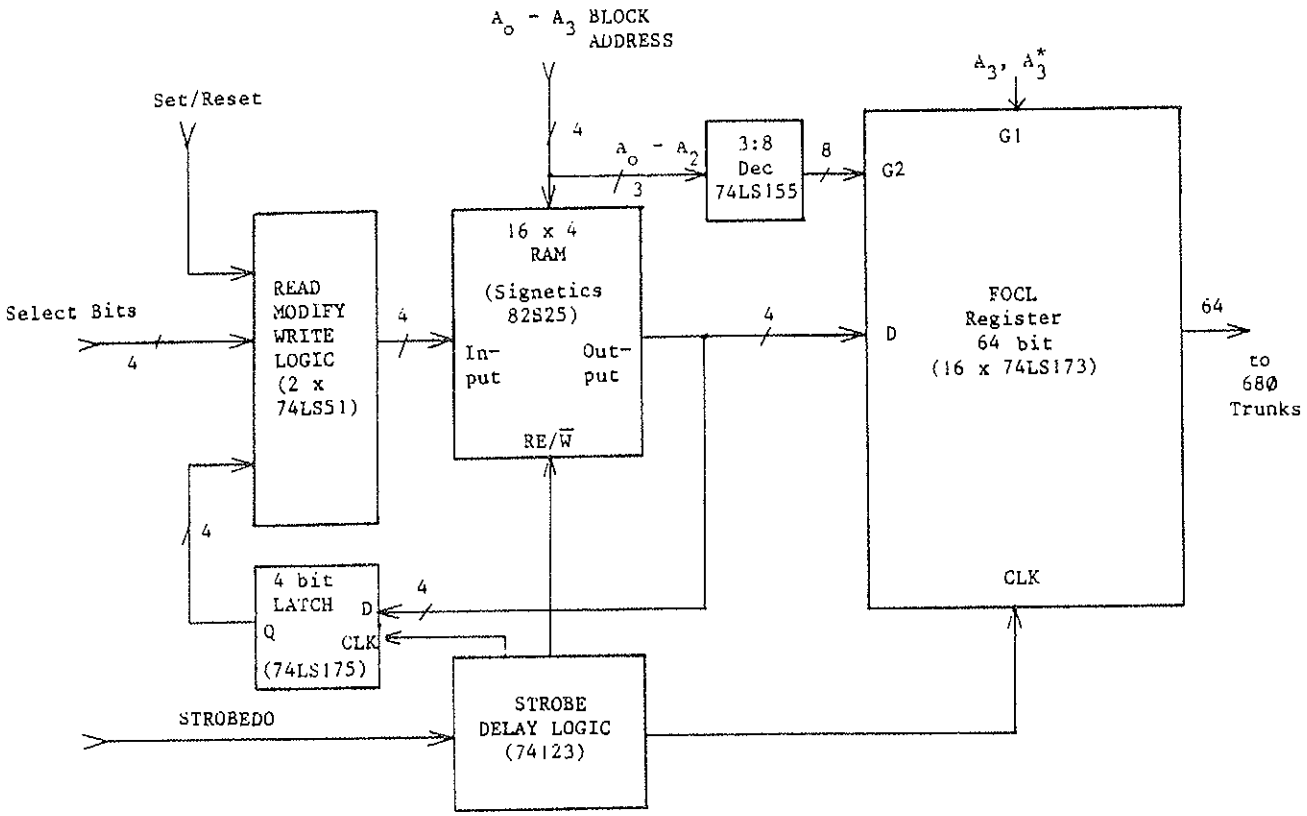


Abb. 9

W. Kleinert

Sämtliche Photos in dieser Ausgabe wurden an der Hybridrechenanlage von den Mitarbeitern des EDV-Zentrums der TU Wien, Abt. Hybridrechenanlage, aufgenommen.

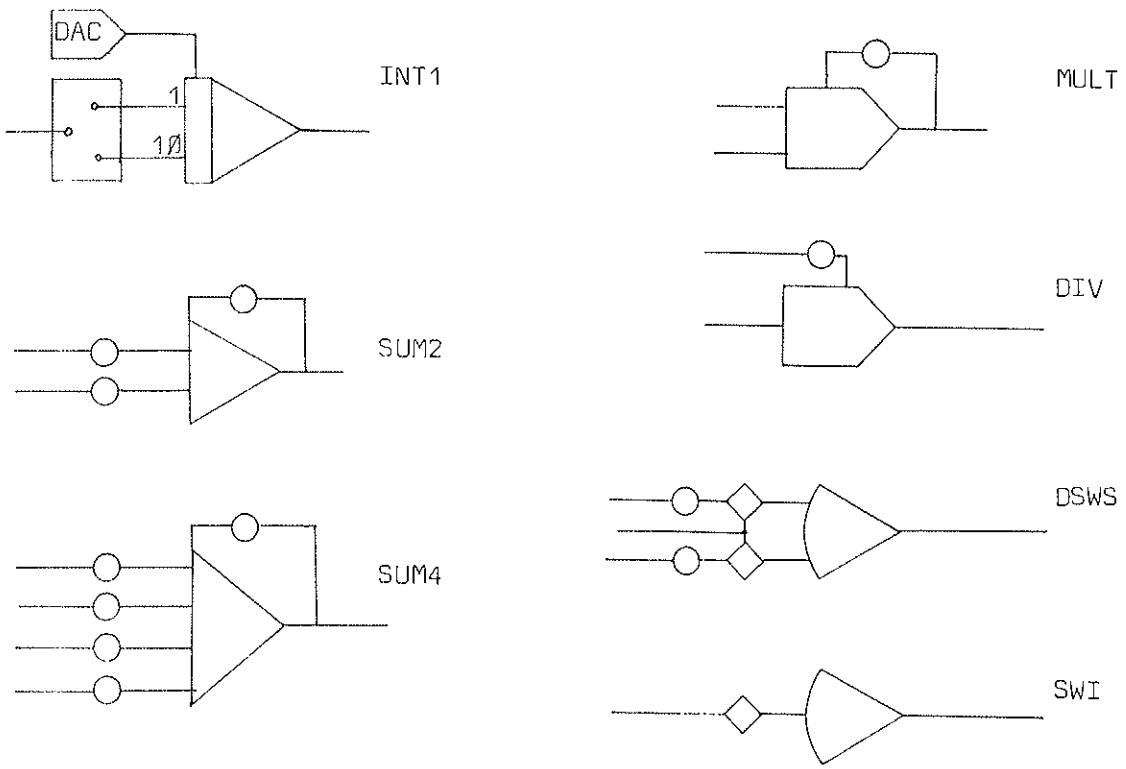
AutoPATCH-Erweiterung des Prozessors HYBSYS

Mit der Installation der beiden EAI Schaltmatrizen konnten erstmals die Voraussetzungen für eine gleichungsorientierte, maschinenunabhängige Eingabe von Simulationsmodellen und für automatisches, fehlerfreies Erstellen der Schaltdaten geschaffen werden. Zur Integration dieser AutoPATCH-Erweiterung in das bestehende Software-System mußte ein geeignetes Makrokonzept für diese beiden Schaltmatrizen entworfen werden, das eine eindeutige Zuordnung der Problemvariablen zu den entsprechenden analogen Komponenten (Makros) ermöglicht. Aufbauend auf diesem Makrokonzept als Datenbasis für die beiden Schaltmatrizen konnten die benötigten Softwaremoduln zur Eingabe der Simulationsgleichungen, Zuordnung der Hardware-Komponenten und Erstellung der Schaltverbindungen entwickelt und an das bestehende Simulationssystem angeschlossen werden.

MAKROKONZEPT

Die HYBSYS-Syntax stellt eine Systematisierung aller möglichen Analogschaltungen dar, mit der jedem Benutzerproblem die zur Realisierung benötigte Hardware eindeutig zugeordnet werden kann. Ausgehend vom derzeitigen Entwicklungsstand dieser Syntax müssen die zu definierenden analogen Makros ein abgeschlossenes System von analogen Operationen darstellen, wobei sich die Abgeschlossenheit auf die zugrunde gelegte Syntax und auf die hybriden Vorbereitungsfunktionen (Koeffizientenberechnung und Skalierbarkeit) bezieht.

Die genaue Definition der Makros (Anzahl der Eingänge, Stückzahl, etc.) hängt jedoch von den Anforderungen der Benutzer ab, die sich auf Grund der globalen Verwendbarkeit des gesamten Systems noch nicht genau abschätzen lassen. Daraus ergibt sich die Forderung einer Invarianz der Software gegenüber Änderungen einzelner Makros. Die Anzahl und Art der Makros ist durch die Anzahl der verschiedenen analogen Komponenten und die Dimension der Schaltmatrizen begrenzt. Die folgende Abbildung zeigt einige der verwendeten Makros.



Aus diesen Überlegungen ergab sich eine Datenbasis mit einer der HYBSYS-Syntax ähnlichen Struktur, die aus den drei folgenden Tabellen besteht:

Makro-Operatoren-Tabelle

Diese Tabelle enthält die definierten Makrotypen, wobei für jeden Typ die Bezeichnung (Makro-Operator), der bei der Exekution zugeordnete HYBSYS-Operator, die Anzahl der vorhandenen Makros dieses Typs und die Anzahl der Eingänge je Makro angegeben sind. Diese Tabelle kann mit jedem beliebigen Makrotyp erweitert werden, für den ein HYBSYS-Operator existiert.

MOP	OPN	MCT	ICT	MEC	IV1	IC1
1	INT1	1	0	INT	0	0
2	INT2	12	3	INT	1	1
3	SUM2	4	0	SUM	0	0
4	SUM3	6	4	SUM	13	37
5	DIV	3	2	DIV	19	61
6	MULT	10	3	MULT	22	67
7	SQR	2	1	SQR	32	97
8	SQRT	2	1	SQRT	34	99
9	ADCF	2	1	ADCF	36	101
10	DACF	2	6	DACF	38	0
11	ADC	1	1	ADC	40	103
12	DCFG	4	1	DCFG	41	104
13	COMP	4	2	COMP	45	108
14	SWI	5	1	SWI	49	116
15	AKKU	2	1	AKKU	54	121
16	TRST	2	1	TRST	56	123
17	SIN	4	1	SIN	58	125
18	COS	3	1	COS	62	129
19	ASIN	1	2	ASIN	65	132
20	ABS	2	1	ABS	66	134
21	TRI	1	0	EXT	68	0
22	MI	8	0	INV	69	0

Makro-Operatoren-Tabelle (Stand von Mai 1980)

Makro-Variablen-Tabelle

Diese Tabelle besteht aus allen an der Schaltmatrix auftretenden Variablen und enthält für jede Variable die Makrooperation, Verstärkeradresse, Matrixinputadresse sowie spezielle Hardwareinformationen (z.B. PBP-Output bei logischer Steuerung), deren Bedeutung von der jeweiligen Makrooperation abhängt. Die Tabelle entspricht also im wesentlichen einer Zuordnung der Variablen zu den Matrixeingängen. Es treten jedoch auch Variable auf, die keinem Matrixeingang zugeordnet werden können (z.B. ADCs).

MV	OPN	MAD	MI	MC0	MC1	LVU
1	INT2	30	1	0	0	0
2	INT2	5	9	0	0	0
3	INT2	10	17	0	0	0
4	INT2	15	25	0	0	0
5	INT2	20	34	0	0	0
6	INT2	25	41	0	0	0
7	INT2	55	49	0	0	0
8	INT2	50	57	0	0	0
9	INT2	35	8	0	0	0
10	INT2	41	15	0	0	0
11	INT2	45	23	0	0	0

62	COS	102	30	0	0	0
63	COS	62	39	0	0	0
64	COS	92	90	0	0	0
65	ASIN	67	98	0	0	0
66	ABS	36	40	0	0	0
67	ABS	46	46	0	0	0
68	TRI	37	48	0	0	0
69	MI	120	62	0	0	0
70	MI	120	65	0	0	0
71	MI	91	89	0	0	0
72	MI	8	81	0	0	0
73	MI	22	73	0	0	0
74	MI	137	24	0	0	0
75	MI	145	38	0	0	0
76	MI	153	56	0	0	0

Ausschnitt aus der derzeitigen Makro-Variablen-Tabelle

Makro-Verbindungs-Tabelle

Diese Tabelle enthält alle Schaltverbindungen zu den definierten Variablen (Makros) und entspricht im wesentlichen einer Zuordnung der Matrixausgänge zu den Makroeingängen. Jede Schaltverbindung enthält an Information die Matrixoutputnummer, Potentiometer- bzw. DCA-Adresse, Relay- oder Switch-Adresse falls vorhanden, PBP-Output-Adresse der logischen Steuerung für die Schaltverbindung (bei definierten Switches), Eingangsbezeichnung der Verbindungen beim Makro (z.B. IC, 1,10, HG, SF, etc.) und die Bezeichnungen der dazugehörigen Makros (Operation und Hardwareadresse).

MC	MO	MPT	MRL	LMO	MRS	MV	OPN	MAD	LCU
1	1	0	4	0	1	1	INT2	30	0
2	2	2	34	0	1	1	INT2	30	0
3	0	-13	0	0	IC	1	INT2	30	0
4	9	5	64	0	1	2	INT2	5	0
5	10	7	94	0	1	2	INT2	5	0
6	0	-6	0	0	IC	2	INT2	5	0
7	17	10	9	0	1	3	INT2	10	0
8	18	12	39	0	1	3	INT2	10	0
9	0	-18	0	0	IC	3	INT2	10	0
10	25	15	69	0	1	4	INT2	15	0
11	26	17	99	0	1	4	INT2	15	0
12	0	-16	0	0	IC	4	INT2	15	0
13	36	20	14	0	1	5	INT2	20	0
14	37	22	44	0	1	5	INT2	20	0
15	0	-19	0	0	IC	5	INT2	20	0
16	44	25	74	0	1	6	INT2	25	0
17	45	27	104	0	1	6	INT2	25	0
18	0	-21	0	0	IC	6	INT2	25	0
19	52	55	10	0	1	7	INT2	55	0
20	53	57	40	0	1	7	INT2	55	0
21	0	-8	0	0	IC	7	INT2	55	0
22	59	50	79	0	1	8	INT2	50	0
23	60	52	109	0	1	8	INT2	50	0
24	0	-7	0	0	IC	8	INT2	50	0
25	70	35	0	0	1	9	INT2	35	0
26	71	37	0	0	1	9	INT2	35	0
27	0	-14	0	0	IC	9	INT2	35	0

Ausschnitt aus der derzeitigen Makro-Verbindungs-Tabelle

Jede Variable und jede Verbindung enthalten zusätzlich einen Benutzerlevelkenncode, der angibt, ob bzw. von welchem Benutzer die entsprechende Variable oder Verbindung gerade belegt ist.

ZUORDNUNG VON MAKROS ZU PROBLEMVARIABLEN

Für jede Problemvariable mit gegebenem analogen HYBSYS-Operator wird in der Makro-Operatoren-Tabelle jener Makro gesucht, dem dieser HYBSYS-Operator zugeordnet ist. Existieren mehrere Makrotypen mit demselben HYBSYS-Operator, so wird jener Typ ausgewählt, dessen Eingangsanzahl am besten mit der Eingangsanzahl der gerade behandelten Problemvariablen übereinstimmt. Anschließend wird in der Makro-Variablen-Tabelle der nächste freie Makro dieses Typs gesucht, der Problemvariablen zugeordnet und mit dem Benutzerkenncode gekennzeichnet. Nach Zuordnung aller Problemvariablen zu den entsprechenden Makros werden die benötigten Eingänge aus der Makro-Verbindungs-Tabelle entnommen und mit dem Benutzerkenncode versehen. Treten bei Integrierern oder Summierern mehr Eingänge auf, als bei den zugeordneten Makros vorhanden sind, so werden Hilfssummierer erzeugt, die betroffene Schaltverbindung entsprechend umdefiniert und neu zugeordnet.

ERSTELLEN DER SCHALTVERBINDUNGEN

Nach fehlerfreier Zuordnung der Variablen und Verbindungen werden die Eingabedaten für die von EAI gelieferte Basissoftware zum Herstellen der benötigten Schaltverbindungen erzeugt. Dabei werden auch alle nicht belegten Eingänge verwendeter Makros mit Ground verbunden und weiters alle nicht benötigten Potentiometer an Eingängen auf Ø bzw. in der Rückführung auf 1 gesetzt, um ein Übersteuern nicht verwendeter Verstärker zu vermeiden.

EINGABE DER PROBLEMGLEICHUNGEN

Auf Grund der Automatisierung der Komponentenzuordnung und der Schaltungserstellung bestand erstmals die Möglichkeit, sämtliche Hardware-Informationen von der Eingabe abzutrennen und die Eingabe rein problemorientiert zu gestalten. Dazu bot sich als momentane, leicht realisierbare Zwischenlösung eine blockorientierte Eingabe an, die später zu einer gleichungsorientierten Eingabe erweitert werden wird.

Die Eingabe besteht aus einer Deklaration aller auftretenden Variablen nach ihrer Operation bzw. ihrem Typ und der Angabe der Gleichungen in Form von Listen von Operanden, wobei jeder Operand aus einem Vorzeichen und einem Produkt aus einer Variablen mit einem Parameter bestehen kann.

HYBSYS-IMPLEMENTATION

Die angeführten Erweiterungen wurden an HYBSYS in Form eines Systemoverlays DECLAR, der die eingegebenen Problemdaten in die HYBSYS-Syntax übersetzt und in die Problemdatenbasis speichert, und eines Systemoverlays PREPAR, der aus den Benutzerdaten entsprechend der momentanen Makrodatenbasis die Komponentenzuordnung durchführt und die Analogschaltung erzeugt, angeschlossen.
Außerdem gibt es einen für die Benutzer nicht zugänglichen Overlay, der es erlaubt, die Makrodatenbasis beliebig zu erweitern oder zu ändern.

Mit dieser Erweiterung wurde das fehleranfällige und zeitaufwendige Stecken der Schaltverbindungen durch das Eintippen der sechs Zeichen P R E P A R ersetzt.

D. Solar

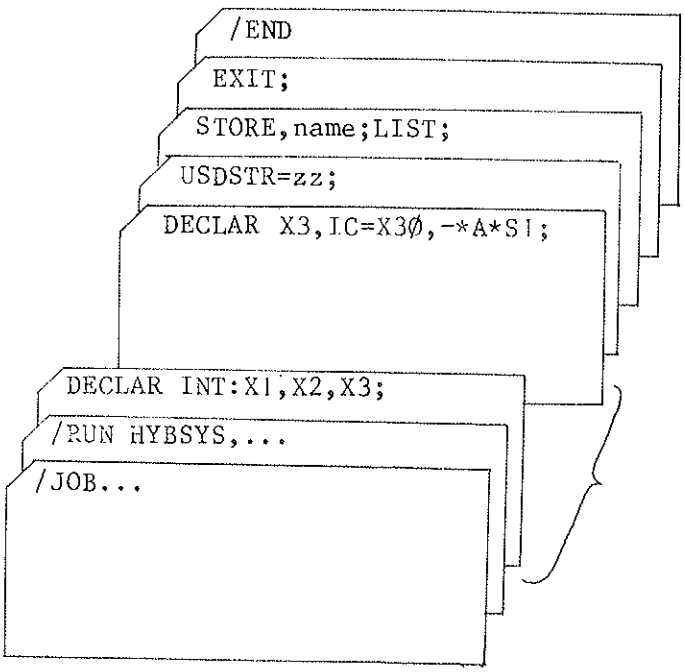
TIPS FÜR HYBSYS-BENUTZER

Mit der Installation der AutoPATCH-Hardware im November vergangenen Jahres und der Erweiterung des hybriden Prozessors HYBSYS wurde den Benutzern das Rechnen mit dem Hybridrechner sehr erleichtert. Da das händische Stecken wegfällt und außerdem detaillierte Hardwarekenntnisse nicht mehr in dem Maße erforderlich sind wie früher, kann der Hybridrechner einem größeren Benutzerkreis zugänglich gemacht werden. Durch die steigende Zahl von Benutzern und den entstandenen Übergang von kleinen akademischen Beispielen zu komplexen wissenschaftlichen Problemen wird die Programmberatung immer aufwendiger und schwieriger, da der Personalstand an der Hybridrechenanlage seit Jahren gleich geblieben ist und auch die Wartung der steigenden Resourcen und die notwendigen Neuentwicklungen immer mehr Zeit in Anspruch nehmen. In einem Forschungsprojekt, das von der Hybridrechenanlage der Technischen Universität Wien beim Fonds zur Förderung der wissenschaftlichen Forschung beantragt wurde, soll ein hybrides Time-Sharing verwirklicht werden, wodurch der momentan entstandene Engpaß bei der Rechenzeit behoben werden könnte.

Vorbereitungen im Closed-Shop-Betrieb

Solange es nicht möglich ist, daß zwei oder mehr Benutzer "gleichzeitig" hybrid rechnen, sollte die reservierte Rechenzeit von den Benutzern optimal ausgenutzt werden. Ein großer Teil der Vorbereitung kann bereits im Closed-Shop-Betrieb erfolgen:

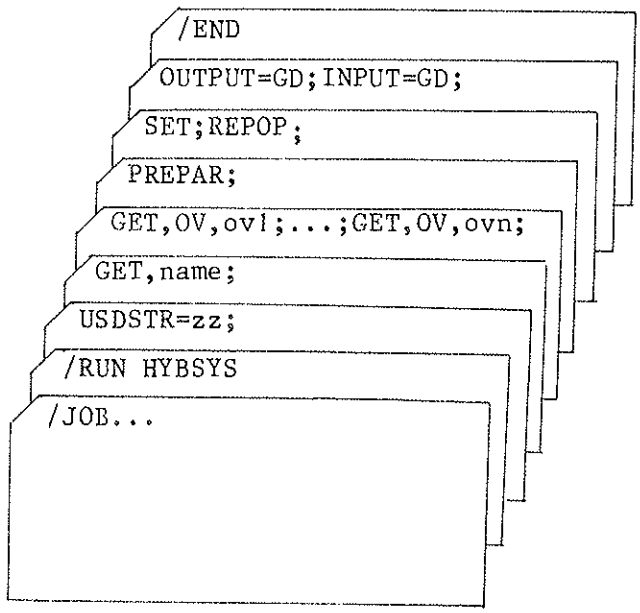
1) Das Erstellen der Schaltstruktur nimmt momentan noch viel Zeit in Anspruch und kann in Form eines BATCH-Jobs gemacht werden.



Definition der Schaltstruktur

Die Dokumentation der Fehler erfolgt am Line Printer. Falls bei der Definition der Schaltstruktur Fehler auftreten, ist es nicht notwendig, den unter name abgespeicherten Schaltdatenfile zu löschen, da bei jedem der STORE-Befehle der File überschrieben wird.
Als User Districts können die Districts 19-24 verwendet werden; auf jedem anderen District werden die abgespeicherten Files gelöscht. Standardmäßig wird District 19 angesprochen.

2) Nachdem die fehlerfreie Schaltstruktur abgespeichert worden ist und sämtliche benötigten User-Overlays gebunden worden sind, kann mit dem interaktiven Rechnen im Open-Shop-Betrieb begonnen werden. HYBSYS-Befehle, die bei jedem RUN benötigt werden, sollen ebenfalls mittels Lochkarten eingegeben werden:



Standardmäßig werden als Ein- bzw. Ausgabeeinheit der Card Reader und der Line Printer angesprochen. Deshalb muß für den Open-Shop-Betrieb die Ein- und Ausgabe auf das Graphic Display verlegt werden. Dabei ist zu beachten, daß zuerst der Output und als letzte Anweisung der Input verlegt wird, da nach der Input-Option nichts mehr vom Card Reader eingelesen wird. (Falls der Output nicht vorher verlegt wurde, meldet sich der HYBSYS-Prozessor nicht, obwohl er die Eingabe vom Terminal aus erwartet).

3) Änderungen bzw. Erweiterungen der Schaltdaten können sowohl im Open-Shop- als auch im Closed-Shop-Betrieb erfolgen. Vor dem ersten DECLAR oder DEFINE-Befehl müssen die Befehle USDSTR=zz und GET,name exekutiert werden. (Wird vor dem Deklarationsteil der Befehl GET,name nicht exekutiert, so wird beim nächsten STORE-Befehl der alte Schaltdatenfile überschrieben und enthält nur mehr die Änderungen bzw. Erweiterungen.)

Bei Auftreten von Fehlern der momentan laufenden Version von HYBSYS sollen diese in das aufliegende Fehlerheft eingetragen werden. Außerdem können darin etwaige Wünsche für Erweiterungen festgehalten werden. Behobene Fehler und neue HYBSYS-Releases werden von den Mitarbeitern der Hybridrechenanlage ebenfalls in diesem Heft vermerkt. Falls eine neue Release gemacht wurde - dies wird auch im HYINFO bekanntgegeben - müssen von den Benutzern sämtliche abgespeicherten User-Overlays gelöscht und wieder neu gebunden werden.

Nach dem Abschluß der Arbeiten am Hybridrechner sollen die nicht weiter benötigten Files (Schaltstrukturen, Datenfiles, Overlays, etc.) den Operatoren bekanntgegeben werden oder vom Benutzer selbst gelöscht werden, damit der Speicherplatz möglichst bald für andere Benutzer zur Verfügung steht.

Dokumentation mit HYBSYS

PLOT

Zur graphischen Dokumentation der Kurven steht der HYBSYS-Overlay PLOT zur Verfügung, wobei drei verschiedene Modes möglich sind:

GRAFIC = 1
Die Ausgabe erfolgt am Data Plotter (Standardmäßig im Closed-Shop-Betrieb)

GRAFIC = 2
Die Ausgabe erfolgt am Tektronix (Standardmäßig im Open-Shop-Betrieb)

GRAFIC = 3
Die Ausgabe erfolgt am Tektronix, wobei mit der Copy-Taste eine Hardcopy am Data Plotter gezeichnet werden kann.

Treten bei einem PLOT-Befehl die Fehlermeldungen

- A/D DMA Transfer stopped at point p
- D/A DMA Transfer stopped at point p
- D/A ST I/O Transfer stopped at point p

auf, so ist der Diskretisierungsabstand zwischen zwei Punkten zu klein

A/D DMA	<	30 µsec	pro Variable
D/A DMA	<	10 µsec	
D/A ST I/O	<	100 µsec	

Der Fehler kann auf zwei Arten behoben werden:

1) Verringerung der Anzahl der Diskretisierungspunkte (POINTS , die Anzahl der Diskretisierungspunkte sollte immer um 1 höher sein als die Anzahl der gewünschten Intervalle)

2) Zeittransformation durch Vergrößerung von BETA

Nach jeder Änderung von POINTS oder BETA müssen anschließend die Befehle SET und REPOP exekutiert werden.

PRINT

Für die numerische Ausgabe von Ergebnissen steht der HYBSYS-Overlay PRINT zur Verfügung. Der Aufruf erfolgt mit

PRINT,dv,f,var_1,...,var_n $(1 \leq n \leq 6)$

wobei dv die Device, f das Format und var_i die Namen der zu dokumentierenden Variablen angeben.

Wird dv weggelassen, so erfolgt die Ausgabe auf der momentan angewählten Ausgabeeinheit. Es stehen zwei verschiedene Formate,S und E,zur Auswahl:

S	Time	var_1	. . .	var_n	
	*E±zz	*E±zz	. . .	*E±zz	
	±.ssss	±.ssss		±.ssss	Anzahl der Punkte (POINTS)
	⋮	⋮		⋮	
	±.ssss	±.ssss	. . .	±.ssss	

E	Time	var_1	. . .	var_n	
	±.ssssE±zz	±.ssssE±zz	. . .	±.ssssE±zz	Anzahl der Punkte (POINTS)
	⋮	⋮		⋮	
	±.ssssE±zz	±.ssssE±zz	. . .	±.ssssE±zz	

Wird f weggelassen, so erfolgt die Ausgabe standardmäßig im S-format. Die Fehlermeldungen, die auftreten können, sind die gleichen wie beim PLOT-Befehl und werden auch auf die gleiche Art behandelt.

Ein neues Softwarepaket zur Verwendung von DCFGs in HYBSYS ist derzeit in Arbeit und wird im nächsten INTERFACE beschrieben werden.

F. Berger

68Ø-Elemente Test mit AutoPATCH-HYBSYS

Die neue AutoPATCH/manual-Version des hybriden Prozessors HYBSYS ermöglicht es, auf einfache Weise im Closed-Shop-Betrieb die Elemente des 68Ø-Analogrechners zu testen. Für die verschiedenen Elementtypen werden Schaltdatenfiles erzeugt und auf einem speziellen District abgespeichert. Die Schaltdaten sind so aufgebaut, daß die Funktionsweise der kompletten Makros (siehe Seite 23) getestet wird und Makros gleichen Typs das gleiche Ergebnis liefern müssen. Mittels eines einfachen Programms, das im Closed-Shop-Betrieb jederzeit exekutiert werden kann, werden von allen Makro-Ausgängen beschriftete Plotter-Zeichnungen angefertigt. Fehlerhafte Makros können leicht durch Abweichungen von der richtigen Lösung erkannt werden. In den Abbildungen 1 bis 3 werden einige Ergebnisse des Makrotests dargestellt.

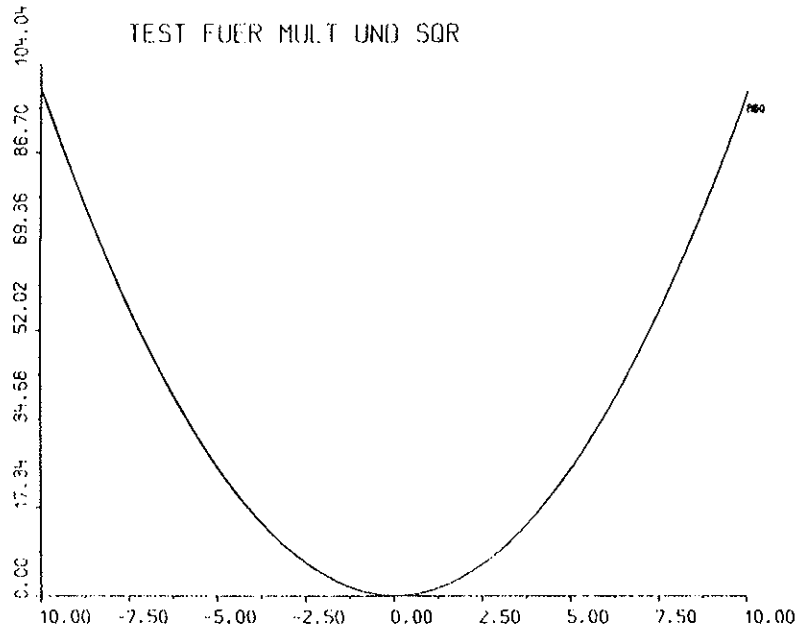


Abb. 1: Test für vier Multiplizierer- und zwei Quadrierermakros

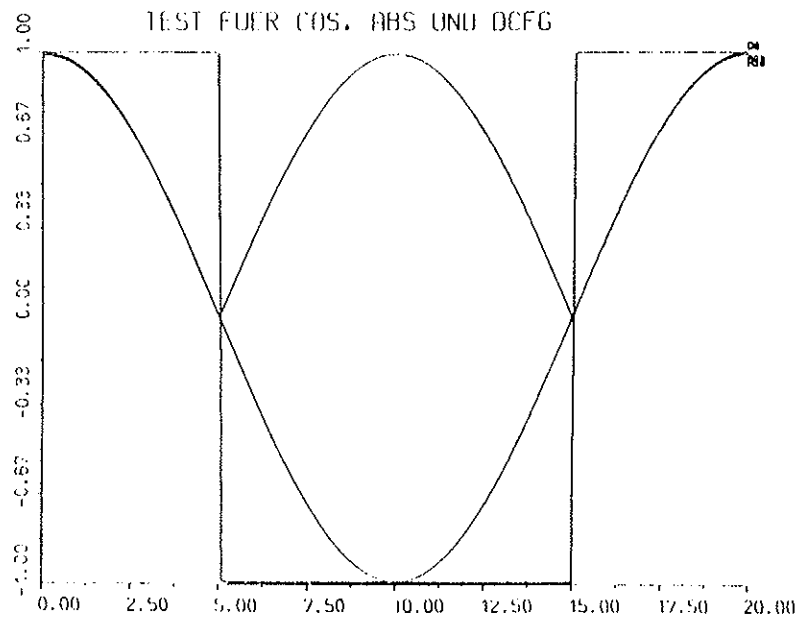


Abb. 2: Test für drei Cosinus-, zwei Absolutbetrags- und vier DCFG-Makros

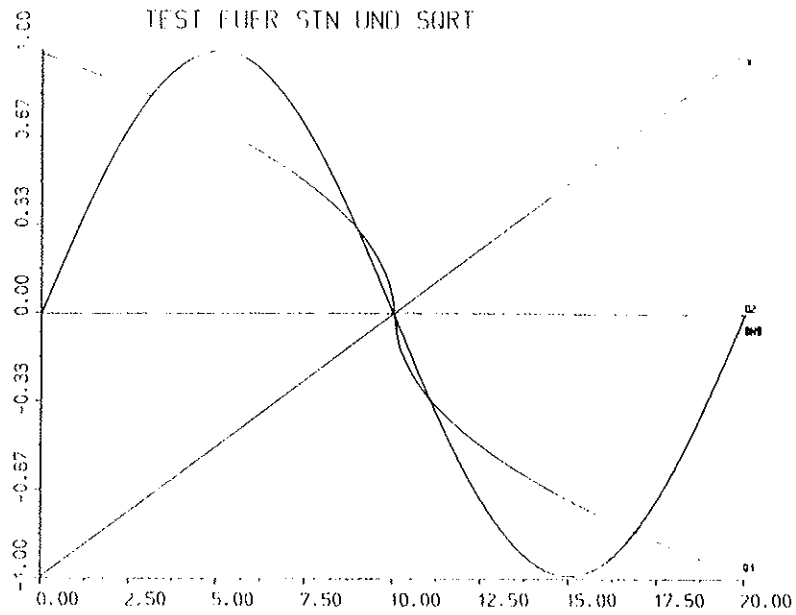


Abb. 3: Test für vier Sinus-, zwei Wurzel- und einen Integrierermakro

Aus Abbildung 3 erkennt man, daß eine Wurzelschaltung (Q2) nicht das erwartete Resultat liefert.

Die als fehlerhaft erkannten Makros werden nun on-line mit Hilfe der abgespeicherten Schaltstruktur und des Prozessors HYBSYS genau untersucht, um zu erkennen, welcher Teil (Verstärker, Potentiometer, Relay, logische Steuerung,etc.) des Makros defekt ist.

Derzeit wird an der Hybridrechenanlage am Ausbau dieses 68Ø-Elemente-Tests und an der Erstellung eines globalen Fehlerfiles gearbeitet. Mit Hilfe dieses Fehlerfiles wird es dann möglich sein, defekte Elemente in HYBSYS zu berücksichtigen, wodurch den AutoPATCH-Benutzern garantiert nur fehlerfreie Elemente zur Verfügung gestellt werden.

F. Berger

AUTOMATISCHE ON-LINE-SKALIERUNG

Einleitung

Das Problem der Skalierung spielt in der hybriden Simulationstechnik eine große Rolle, da die Skalierung einerseits Voraussetzung für die Realisierbarkeit einer Simulation am Analogrechner ist und andererseits die Genauigkeit der Lösung wesentlich beeinflußt. Die Notwendigkeit der Skalierung besteht daher sowohl in der Vorbereitungsphase als auch während der Simulation selbst, wenn Parameteränderungen vorgenommen werden oder das Modell verändert wird.

Unter der Skalierung eines am Analogrechner zu lösenden Modellsystems, das im allgemeinen in Form von Differentialgleichungssystemen (auch Ø-ter Ordnung) beschrieben wird, versteht man die Transformation des gegebenen, durch Problemvariablen beschriebenen Systems in ein äquivalentes, jedoch durch Maschinenvariablen beschriebenes System, das den vom Analogrechner im Hinblick auf Rechengenauigkeit und Realisierbarkeit geforderten Spezifikationen genügt.

Als wesentliche Forderungen treten dabei auf:

- die Einhaltung des gegebenen Operationsbereiches analoger Komponenten wegen der Realisierbarkeit
- die optimale Ausnützung des Operationsbereiches aus Gründen der Rechengenauigkeit
- die Einstellbarkeit der konstanten Koeffizienten.

Der Kern des Problems besteht daher in der Berechnung der Extremwerte aller zeitabhängigen Größen des ursprünglichen Systems und Normierung derselben nach den Extremwerten.

Zur automatisierten Lösung dieses Problems bestehen drei grundsätzlich verschiedene Möglichkeiten:

- Berechnung einer optimalen Skalierung mit Hilfe von Abschätzungen und linearer Optimierung
- Berechnung einer optimalen Skalierung durch einen digitalen Simulationslauf
- Iterative Berechnung einer optimalen Skalierung durch wiederholte analoge Lösung des gegebenen Systems.

Die beiden ersten Methoden werden rein digital durchgeführt, wobei ihre Anwendungen durch das jeweilige Verfahren (Abschätzung und Optimierungsverfahren bzw. Iterationsverfahren) begrenzt sind. Die Menge der auf diese Weise skalierbaren Differentialgleichungssysteme ist daher nicht unbedingt identisch mit den am Analogrechner lösbaren Systemen.

Es ist daher naheliegend, nicht nur die Simulation selbst, sondern auch die Berechnung der Skalierung mit Hilfe des Analogrechners durchzuführen. Dadurch wird die Identität von skalierbaren und lösbaren Systemen hergestellt. Diese Tatsache ist besonders bei Simulationen mit Unstetigkeiten von Bedeutung, deren digitale Behandlung oft mit großen Schwierigkeiten und mit großem Rechenzeitaufwand verbunden ist. Der Vorteil einer nach erfolgter Skalierung sofort voll einsetzbaren Analogschaltung ist vor allem bei einer Nachskalierung während der Simulation nicht zu unterschätzen.

Die hybride Lösung des Skalierungsproblemes setzt jedoch nicht nur, wie im digitalen Fall, die Kenntnis der Problemgleichungen voraus, sondern sie benötigt auch genaue Informationen über die verwendete Analogschaltung und, unabhängig von speziellen Problemgleichungen, über die Skalierbarkeit der einzelnen analogen Operationen. Darüber hinaus treten bei der Realisierung des iterativen Skalierungs-

verfahrens spezielle Probleme wie das Erkennen von Über- und Untersteuerung auf, deren Lösung wesentlich von der vorhandenen Hardware abhängt.

Skalierung als lineare Transformation

Bei der Realisierung gegebener Problemgleichungen am Analogrechner müssen sämtliche am Ausgang eines Verstärkers anliegenden Problemvariablen so transformiert werden, daß sie im zu untersuchenden Zeitintervall einerseits den durch die normierte Referenzspannung gegebenen zulässigen Wertebereich von (-1,+1) nicht überschreiten und andererseits diesen Bereich wegen der Rechengenauigkeit möglichst voll ausnützen. Eine Variable ist dann als optimal skaliert anzusehen, wenn sie diesen Bereich voll durchläuft, aber nicht überschreitet. Das Problem der Skalierung ist also rein rechentechnisch bedingt und muß daher unabhängig von der Struktur gegebener Problemgleichungen, d.h. elementweise, betrachtet werden.

Voraussetzung für die Automatisierung der Skalierung ist weiters die Existenz eines allgemeinen Problemgleichungssystems sowie eines allgemeinen Maschinengleichungssystems, die durch die Skalierung als Transformation auseinander hervorgehen. Wegen der notwendigen Unabhängigkeit von speziellen Problemgleichungen muß diese Transformation eine bijektive Abbildung der Problemvariablen auf die Maschinenvariablen und insbesondere eine weitgehend identische Abbildung aller auftretenden Operationen und Funktionen sein.

Eine automatische Skalierung kann daher nur mit linearen Transformationen durchgeführt werden. Andere Formen der Skalierung, z.B. eine zeitabhängige Skalierung, sind problemspezifisch. Eine allgemeine Anwendung ist daher weder sinnvoll noch möglich, zumal auch diese Transformationen letzten Endes die lineare Transformation zur quantitativen Realisierung benötigen. Die lineare Transformation ist daher die einzige Skalierungsform, die für eine Automatisierung geeignet ist.

Durch diese Transformation wird jeder Problemvariablen $x(t)$ eine Maschinenvariable $\bar{x}(t)$, die den Ausgang des zugehörigen Verstärkers darstellt, und ein Skalierungsfaktor $s(x)$ zugeordnet, die folgender Beziehung genügen:

$$x(t) = s(x) \cdot \bar{x}(t)$$

Die Maschinenvariable $\bar{x}(t)$ ist dann optimal skaliert, wenn für ihren Skalierungsfaktor

$$s(x) = \max_{t \in T} |x(t)|$$

gilt.

Die Skalierung wird also nicht als Transformation von den Problemvariablen auf die Maschinenvariablen, sondern als Transformation von Maschinenvariablen auf Problemvariablen definiert. Diese Vorgangsweise hat folgende Gründe:

- die Multiplikation einer zeitabhängigen Variablen $x(t)$ mit einem konstanten Parameter kann ohne Verwendung eines analogen Elements durch digitale Multiplikation mit dem Skalierungsfaktor dieser Variablen durchgeführt werden:

$$s(\alpha.x) = \alpha.s(x)$$

- die angeführte Transformation gilt auch für Variable mit $x(t) \equiv \emptyset$.

Skalierbarkeit analoger Operationen

Für eine Automatisierung der Skalierung ist es zunächst notwendig, die Skalierbarkeit analoger Operationen einzeln zu untersuchen, d.h. die Bedingungen für die analoge Realisierung einer "Maschinengleichung"

$$\bar{z} = \bar{f}(\bar{X})$$

bei einer gegebenen Problemgleichung

$$z = f(X)$$

aufzustellen, wobei

f	eine elementare Funktion oder Operation
$\overline{f}$	das analoge Äquivalent von f
X	den Vektor der Eingangsvariablen
$\overline{X}$	den skalierten Vektor
z	die Ausgangsvariable
$\overline{z}$	die skalierte Ausgangsvariable

darstellen. Im Hinblick auf ein automatisches Skalierungsverfahren sind dabei folgende Faktoren von Bedeutung:

- Der zulässige quantitative Bereich der Skalierungsfaktoren von X und z, d.h. die Realisierbarkeit der Skalierung in einer hinreichend großen Umgebung um die optimalen Skalierungsfaktoren.
- Zusammenhänge zwischen den Skalierungsfaktoren der Eingangsvariablen $x_i \varepsilon X$ und der Ausgangsvariablen z:

  Unabhängigkeit

  Notwendige Abhängigkeit: Die Skalierung der Ausgangsvariablen ist durch die Skalierung der Eingangsvariablen festgelegt.

  Mögliche Abhängigkeit: Die Skalierung des Ausganges kann aus der Skalierung der Eingänge derart bestimmt werden, daß bei einer Konvergenz der Eingangsskalierungsfaktoren zum Vektor der optimalen Skalierungsfaktoren die Skalierung des Ausgangs ebenfalls optimal oder hinreichend optimal wird.
- Verhältnis des Elementaufwandes und der Rechengenauigkeit von optimaler und überlauffreier Skalierung.

Aufgrund dieser Faktoren müssen die analogen Komponenten hinsichtlich ihrer Skalierbarkeit in 3 Gruppen eingeteilt werden:

1) Integrierer:
   Die Skalierung eines Integriererausganges ist unabhängig von der Skalierung der Eingangsvariablen. Die benötigten Potentiometerwerte können durch entsprechende individuelle oder globale Zeittransformation problemlos realisiert werden.

2) Summierer:
   Die Skalierung eines Summiererausganges ist unabhängig von der Skalierung seiner Eingangsvariablen. Die benötigten Potentiometerwerte können mit Hilfe eines Feedback-Potentiometers uneingeschränkt realisiert werden. Wird kein Feedback-Potentiometer verwendet, so sind die Werte nach oben mit 100 beschränkt.

3) Nichtlineare und andere Elemente
   Die Skalierung der Ausgänge dieser Elementgruppe ist je nach Art der Realisierung entweder teilweise oder vollständig von der Skalierung der Eingangsvariablen abhängig. Diese Abhängigkeit hat im wesentlichen folgende Ursachen bzw. Gründe:

   Eine unabhängige Skalierung ist nicht sinnvoll, da sie keine Erhöhung der Rechengenauigkeit bringt, sondern sogar den Elementaufwand vergrößert. Zusätzlich müßte die Ausgangsvariable auch in den Skalierungsalgorithmus aufgenommen werden (z.B. x^2, $|x|$).

   Die Operation kann mit der vorhandenen Hardware nicht optimal skaliert durchgeführt werden (z.B. $\sqrt{x}$), oder die Skalierung ist von der Hardware festgelegt (z.B. sin x).

Die Möglichkeit einer optimalen Skalierung hängt vom Verlauf der Eingangsvariablen ab (z.B. x/y).

Mit den folgenden Mindestanforderungen an die Schaltkonfiguration analoger Komponenten

- jedem Integriereingang einschließlich IC muß ein Potentiometer vorgeschaltet sein
- jedem Summiereingang muß ein Potentiometer vorgeschaltet sein (ein Feedback-Potentiometer kann verwendet werden, ist jedoch nur bei extremen Potentiometerwerten notwendig)
- bei einer Division muß dem Eingang des Zählers ein Potentiometer vorgeschaltet sein

kann jede Analogschaltung, die keine Sinus-, Logarithmus- oder Exponentialfunktion enthält, mit der einfachen linearen Skalierung automatisch zumindest überlauffrei skaliert werden.

Potentiometer (DCAs) dienen in der Regel zur Multiplikation einer zeitabhängigen analogen Variablen mit einem konstanten Parameter. Wie auf Seite 34 gezeigt wurde, kann diese Multiplikation mit den Skalierungsfaktoren durchgeführt werden, d.h. sie ist nicht an die Existenz eines Potentiometers gebunden. Daher werden Potentiometer nicht als eigene analoge Elemente, sondern als Teile der Eingangs- bzw. Rückführnetzwerke analoger Verstärker (=analoge Variable) aufgefaßt. Sie dienen somit nur der quantitativen Realisierung der Skalierung. Die erwähnten Parameter treten natürlich indirekt, nämlich in den Skalierungsfaktoren in Potentiometerwerten auf. Da Parameter beide Vorzeichen annehmen können und annehmen können sollen, erfordert eine allgemeine Analogrechnerkonzeption die Verwendung von 4-Quadranten-DCAs. Aus diesem Grund wurde auch auf die vorzeichenumkehrende Wirkung der Verstärker keine Rücksicht genommen, da diese für eine allgemeine Behandlung entweder durch das Vorzeichen des Skalierungsfaktors oder durch entsprechende Modifikationen der Hardware oder der hybriden Standardsoftware berücksichtigt werden muß. Die Referenzspannung diente in der bisherigen Analogrechnerpraxis in Verbindung mit einem Potentiometer zur Erzeugung eines konstanten analogen Parameters. Wenn die Referenzspannung als optimal skalierter, konstanter Parameter, der natürlich gleich 1 ist, betrachtet wird, fällt die Notwendigkeit der Existenz eines Potentiometers im Zusammenhang mit der Referenzspannung weg, sodaß konstanten analogen Parametern die Referenzspannung als analoges Element zugeordnet werden kann.

Theorie des Skalierungsalgorithmus

Zunächst ist diejenige Transformation zu betrachten, die ein mit einem beliebigen Vektor $S^{(0)}(X)$ skaliertes Gleichungssystem in ein mit einem Vektor $S^{(1)}(X)$ skaliertes Gleichungssystem derart überführt, daß alle skalierten Variablen $\bar{X}^{(1)}$ analog erzeugt werden können und eine Schranke S genau erreichen, aber nicht überschreiten, also

$$\max_{t \in T} |\bar{X}^{(1)}(t)| = S$$

erfüllt ist. Die Schranke S sei dabei in einem Bereich

$$0 < S_{UFL} \leq S < S_{OLD}$$

frei wählbar, wobei S_{UFL} jene Grenze angibt, ab der man sinnvolle Verstärkerausgänge erhält und S_{OLD} jene Grenze, bei der die Verstärkerausgänge gerade noch übersteuerungsfrei sind.
Wegen

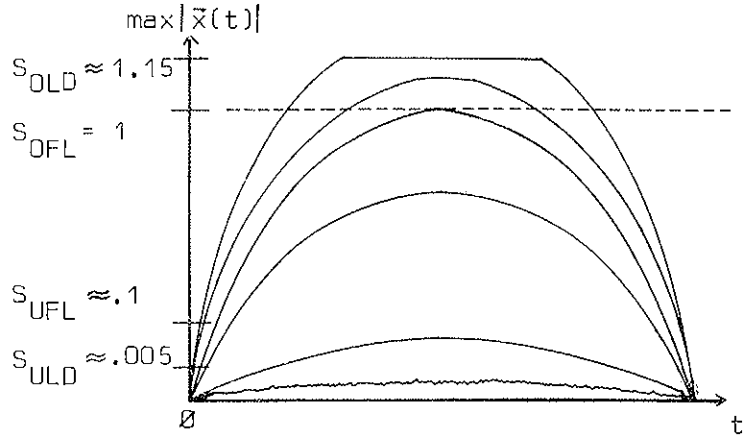
$$\max |X(t)| = S^{(0)}(X) \cdot \max |\bar{X}^{(0)}(t)|$$
$$\max |X(t)| = S^{(1)}(x) \cdot \max |\bar{X}^{(1)}(t)|$$

hat diese Transformation die Gestalt

$$s^{(1)}(x) = \frac{\max|\bar{x}^{(0)}(t)|}{S} \cdot s^{0}(x)$$

Werden die Maximalwerte $\max|\bar{x}^{(0)}(t)|$ durch analoge Lösung des Gleichungssystems bestimmt, so führt diese Transformation nur dann zum Ziel, wenn
- kein Verstärkerausgang übersteuert
- die Maximalwerte aller Verstärkerausgänge hinreichend genau ausgelesen werden können, also größer S_{UFL} sind.

Dabei sind folgende Wertebereiche der Verstärkerausgänge zu unterscheiden:



S_{OLD} Overload-Limit
S_{OFL} Overflow-Limit
S_{UFL} Underflow-Limit
S_{ULO} Underload-Limit

$S_{OLD} < \max\|\bar{x}(t)\|$	Übersteuerung
$S_{OFL} < \max\|\bar{x}(t)\| \leq S_{DLD}$	Überlauf. Der Verstärker ist zwar optimal skaliert, bei einer hybriden Weiterverarbeitung über einen ADC wird jedoch abgeschnitten.
$S_{OFL} = \max\|\bar{x}(t)\|$	Optimale Skalierung
$S_{UFL} \leq \max\|\bar{x}(t)\| < S_{OFL}$	Überlauffreie Skalierung (bei Nachskalierung von Bedeutung)
$S_{ULD} \leq \max\|\bar{x}(t)\| < S_{UFL}$	Untersteuerung
$\max\|\bar{x}(t)\| < S_{ULD}$	Starke Untersteuerung, oder die Variable ist identisch Ø (der Verstärker liefert praktisch nur ein Rauschen)

Übersteuerung

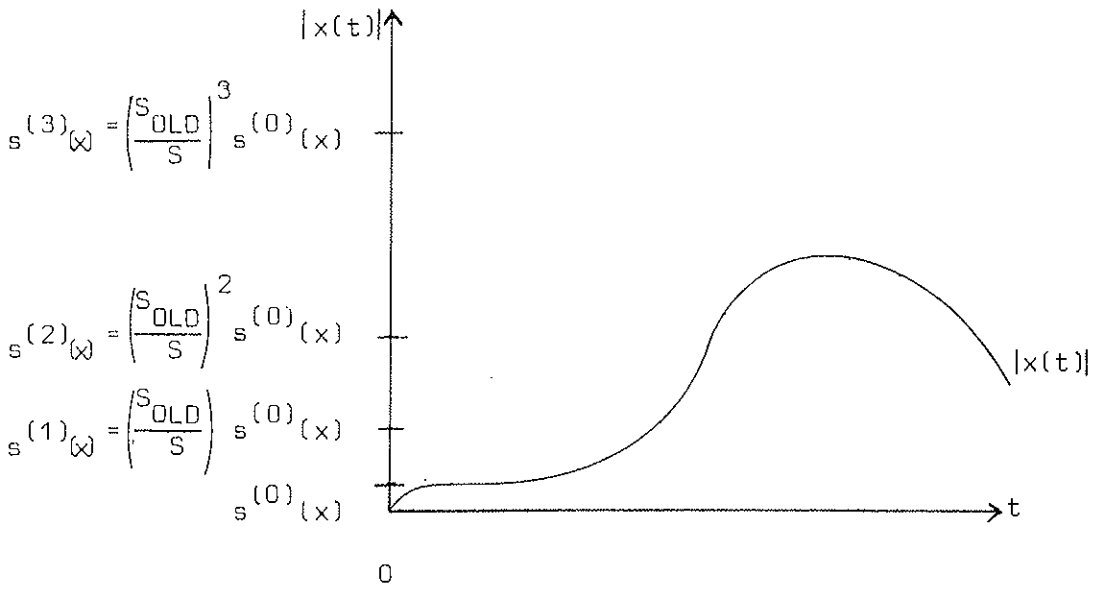
Im Falle einer Übersteuerung einer Variablen $\bar{x}^{(0)} \varepsilon \bar{x}^{(0)}$ gilt für den vom Analogrechner erhaltenen Maximalwert

$$S_{OLD} = \max|\bar{\bar{x}}^{(0)}(t)| \leq \max|\bar{x}^{(0)}(t)| = \max|x(t)|/s^{(0)}(x)$$

$\max\|x(t)\|$	Maximum der unskalierten Variablen x(t)
$\max\|\bar{x}^{(0)}(t)\|$	Maximum der mit $s^{(0)}(x)$ skalierten Variablen
$\max\|\bar{\bar{x}}^{(0)}(t)\|$	Am Analogrechner erhaltenes Maximum der mit $s^{(0)}(x)$ skalierten Variablen x(t).

Der neue Skalierungsfaktor $s^{(1)}(x)$ ist wegen $0 < S_{UFL} \leq S \leq S_{OLD}$ zwar größer als der vorige Skalierungsfaktor $s^{(0)}(x)$, der dazugehörige Verstärkerausgang wird aber größer als die Schranke S. Im günstigsten Fall ist der entsprechende Verstärker mit diesem neuen Skalierungsfaktor überlauffrei skaliert, sodaß eine nochmalige Anwendung der oben angeführten Transformation das gewünschte Resultat bringt.

Bei einer neuerlichen Übersteuerung muß die Transformation so oft wiederholt werden, bis Überlauffreiheit eintritt. Ein Unterlauf kann während dieser Iterationen nicht auftreten, da das ausgelesene Maximum nicht kleiner als S werden kann ($S \leq \max|\bar{x}^{(i)}(t)| \leq S_{OLD}$ für jede Iteration i bis zum Eintreten der Überlauffreiheit).



Untersteuerung

Eine Untersteuerung tritt dann auf, wenn der absolute Maximalwert einer mit $s^{(0)}(x)$ skalierten Variablen x kleiner als die Schranke S_{UFL} ist

$$\max|\bar{x}^{(0)}| = \frac{\max|x|}{s^{(0)}(x)} < S_{UFL}$$

sodaß dieser nicht mehr hinreichend genau ausgelesen werden kann oder identisch mit dem Rauschpegel des dazugehörigen Verstärkers ist. Im letzteren Fall ist dabei der ausgelesene Wert $\max|\bar{x}^{(0)}|$ (unter Vernachlässigung der Rechenfehler analoger Elemente) größer als das tatsächliche Maximum $\max|\bar{x}^{(0)}|$ der mit $s^{(0)}(x)$ skalierten Variablen. Um eine Skalierung zu erhalten, die die Bedingung

$$\max_{t\in T}|\bar{x}^{(1)}(t)| = S$$

erfüllt, muß die angegebene Transformation sooft durchgeführt werden, bis das Maximum des Verstärkers hinreichend genau ausgelesen werden kann, also

$$S_{UFL} \leq \max|\bar{x}^{(i)}| \leq S_{OLD}$$

gilt, sodaß eine weitere Anwendung der Transformation die gewünschte Skalierung liefert.

Konvergenzgeschwindigkeit und Konvergenzbedingungen

Die Konvergenzgeschwindigkeit des Verfahrens ist direkt proportional zur Größe

$$KG = \left|\frac{\lg \max|\bar{x}(t)|}{S}\right|$$

d.h. es gilt

$$KG = \lg\frac{S_{OLD}}{S} \quad \text{bzw.} \quad KG \geq \left|\lg\frac{S_{ULD}}{S}\right|$$

bei Übersteuerung bzw. Untersteuerung.
Da dieser Faktor auch direkt die Größenordnung von Potentiometeränderungen während des Verfahrens bestimmt, ist es sinnvoll, diesen nicht allzu groß zu wählen. In der Praxis hat sich für den Fall einer Übersteuerung der Wert KG = lg 2, also eine Verdopplung der Skalierung, bewährt, sodaß das Verfahren eine Skalierung mit $\max|\bar{x}(t)| = S = S_{OLD}/2 = 0.575$ liefert, für den Fall einer Untersteuerung ergab sich S = 1 als sinnvoll, sodaß die Iteration direkt zur optimalen Skalierung führt. Bei Untersteuerung konvergiert also das Verfahren wegen $KG \geq |\lg S_{OLD}| > \lg 2$ wesentlich schneller als bei Übersteuerung.

Bei der Behandlung des Skalierungsalgorithmus wurde bis hierher angenommen, daß sich die einzelnen Variablen nicht gegenseitig beeinflussen. Dies ist natürlich nicht der Fall, da eine Über- oder Untersteuerung einer Variablen auf alle von ihr abhängigen Variablen einwirkt. Bei übersteuernden Variablen gilt die Abschätzung daher nur für das Zeitintervall bis zur ersten Übersteuerung, bei untersteuernden Variablen bis zu dem Zeitpunkt, zu welchem ein Verstärkerausgang infolge des Driftens größer als der Rauschpegel wird.

Bei dem (den) als erster (ersten) übersteuernden Verstärker(n) ist das Zeitintervall nach der Übersteuerung ohne Bedeutung, bei allen anderen Verstärkern tritt jedoch eine Verfälschung der Extremwerte auf, die die Konvergenzgeschwindigkeit ungünstig beeinflussen kann, aber keinen direkten Einfluß auf die Konvergenz selbst hat.

Die Konvergenz des Verfahrens ist bei folgenden Voraussetzungen gegeben:

- Das Gleichungssystem hat eine eindeutige, stabile Lösung.
- Das skalierte Gleichungssystem ist mit jedem beliebigen Skalierungsvektor aus einer durch den Startskalierungsvektor begrenzten Umgebung um den optimalen Skalierungsvektor stabil.
- Die dazugehörige Analogschaltung ist mit jeder Skalierung aus dieser Umgebung stabil.
- Rechenungenauigkeit und Fehlerfortpflanzung dürfen nicht zu einem stabilen Schaltkreis führen, der nicht Lösung des gegebenen Gleichungssystems ist.
- Alle Verstärkerausgänge müssen der Bedingung

$$\max|\bar{\bar{x}}(t)| \geq S_{OLD} \Rightarrow \max|\bar{x}(t)| \geq S_{DLD}$$

genügen, d.h. ein Verstärker darf nur dann übersteuern, wenn das theoretisch skalierte Maximum $\max|\bar{x}(t)|$ übersteuern würde. Die Bedingung ist für eine Konvergenz jedoch nicht notwendig, sondern inklusive Stabilität nur hinreichend. (Die Division ist die einzige Operation, bei der diese Bedingung bei Untersteuerung nicht immer erfüllt ist.)

Die Skalierungseinheit

Ein besonderes Problem bilden diejenigen Variablen, die bei bestimmten Parameterkonstellationen identisch Null sind, sodaß die Skalierungstransformation zu einer "Null"-Skalierung führt. Dieses Problem kann durch die Definition einer Skalierungseinheit SCU und Aufrunden aller Skalierungsfaktoren auf das nächstgrößere Vielfache dieser Einheit behoben werden. Auf diese Weise wird den Skalierungsfaktoren von Null-Variablen eine untere Schranke gesetzt.

Die Wahl der Skalierungseinheit stellt eine problemspezifische Aufgabe dar. Um eine optimale Skalierung zu ermöglichen, muß sie kleiner sein als das kleinste auftretende absolute Maximum aller Variablen. Bei Systemen mit sehr unterschiedlichen Amplituden kann sie auch dazu benutzt werden, extreme Potentiometereinstellungen zu vermeiden, indem sie entsprechend größer als das kleinste auftretende Maximum gewählt wird.

Nachskalierung

Die Aufgabe der Nachskalierung besteht in der Berechnung einer überlauffreien Skalierung derart, daß die Maxima aller Verstärkerausgänge eine untere Schranke S_{UFL} nicht unterschreiten mit dem Ziel, bei ausreichender Rechengenauigkeit einen möglichst breiten Spielraum für Parameteränderungen zur Verfügung zu haben. Dabei wird im allgemeinen eine bereits skalierte Schaltung zugrunde gelegt, bei welcher nun infolge von Parameteränderungen eine Über- oder Untersteuerung auftritt.

Unter der Annahme, daß die neu skalierte Schaltung auch für die vor Eintreten der Über- oder Untersteuerung verwendeten Parametereinstellungen brauchbare Resultate liefern soll, ist folgende Vorgangsweise angebracht:

- Die Skalierung nicht über- oder untersteuernder Variablen bleibt unverändert ($S_{UFL} \leq \max|\bar{x}(t)| \leq S_{OFL}$).
- Übersteuernde Variable werden optimal skaliert ($\max|\bar{x}(t)| = S_{OFL}$)
- Untersteuernde Variable werden so skaliert, daß ihre skalierten, absoluten Maxima genau die Unterlauf-Schranke S_{UFL} erreichen ($\max|\bar{x}(t)| = S_{UFL}$)

Unter der Voraussetzung, daß die ursprüngliche Skalierung für alle Parameterwerte aus einem vorgegebenen Bereich eine über- und unterlauffreie Schaltung liefert, führt diese neue Skalierung zu einer Schaltung mit minimaler Über- und Untersteuerung in dem erweiterten Parameterbereich.

Um die geforderte Skalierung zu erhalten, wird der beschriebene Algorithmus bei der letzten Iteration mit $S = S_{OFL}$ bei übersteuernden und $S = S_{UFL}$ bei untersteuernden Variablen angewandt.

Optimale Skalierung

Die Aufgabe der optimalen Skalierung besteht in der optimalen Aussteuerung aller Verstärkerausgänge bei einem festen Parametersatz mit der Nebenbedingung einer möglichst genauen Einstellbarkeit aller Potentiometerwerte und dem Ziel einer größtmöglichen Rechengenauigkeit.

Wenn man zunächst von der Nebenbedingung absieht, können die optimalen Skalierungsfaktoren auf zwei Arten erhalten werden:

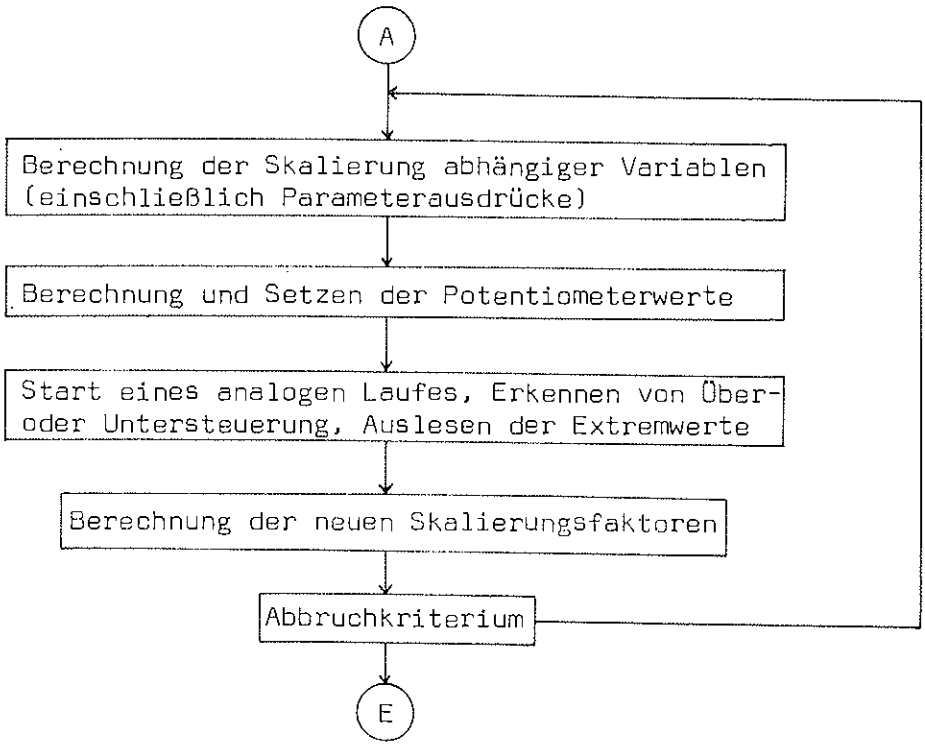
- Durch Nachskalierung, die zu einer über- und unterlauffreien Schaltung führt und anschließender Berechnung der optimalen Skalierungsfaktoren aus den Maxima der überlauffreien Schaltung.
- Durch sofortige optimale Skalierung bei jeder Iteration. Das entspricht einer Nachskalierung, bei welcher die Unterlaufschranke S_{UFL} gleich 1 gesetzt wird.

Diese beiden Verfahren unterscheiden sich hauptsächlich durch die Art der Konvergenz und somit durch den Bereich der während des Verfahrens einzustellenden Potentiometerwerte. Das zweite Verfahren besitzt zwar wegen $S_{UFL} = 1$ eine höhere Konvergenzgeschwindigkeit bei untersteuernden Variablen, im allgemeinen wird es jedoch von den speziellen Problemgleichungen abhängen, welches der beiden Verfahren günstiger ist.

Die Nebenbedingung von optimalen Potentiometereinstellungen ist vom Iterationsverfahren selbst unabhängig und wird bei der Berechnung der Potentiometerwerte für die einzelnen analogen Rechenkomponenten berücksichtigt.

Das hybride On-Line-Skalierungsverfahren

Ausgehend von einer vorgegebenen Anfangsskalierung zeigt das folgende Blockdiagramm die für die Durchführung einer automatischen On-Line-Skalierung wesentlichen Funktionsblöcke.



Die Berechnung der Skalierung abhängiger Variablen sowie der Potentiometerwerte ist vom Skalierungsverfahren selbst unabhängig und wird im nächsten INTERFACE behandelt. Die Berechnung der neuen Skalierungsfaktoren erfolgt nach dem beschriebenen Algorithmus.

Erkennen von Über- oder Untersteuerung und Auslesen der Extremwerte

Zum Erkennen einer Über- oder Untersteuerung und zum Auslesen der Maxima stehen grundsätzlich zwei Möglichkeiten zur Verfügung:

a) Mit dem zentralen Overload-Signal wird der Analogrechner in den Hold-Mode versetzt. Anschließend werden alle Verstärker über ein Select-System angewählt, ausgelesen und so die übersteuernden Verstärker bestimmt.
Der Vorteil dieser Methode liegt darin, daß unabhängig von der Anzahl der verwendeten Rechenelemente nur ein analoger Lauf pro Iteration benötigt wird, es ist jedoch keine Aussage über eine etwaige Untersteuerung möglich, sodaß im günstigsten Fall eine überlauffreie Skalierung erreicht werden kann.

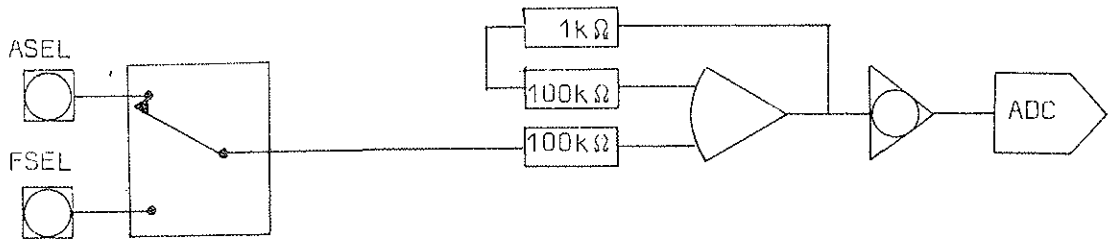
b) Für jeden Verstärker mit unabhängiger Skalierung wird ein analoger Lauf mit gleichzeitiger analoger oder digitaler Berechnung der Extremwerte durchgeführt. Diese Methode ermöglicht eine optimale Skalierung jedes Verstärkers. Da bei jedem Lauf sowohl Über- als auch Untersteuerung erkannt werden, konvergiert das Iterationsverfahren schneller, wobei auch eine wesentlich günstigere Gewichtung der einzustellenden Potentiometerwerte erreicht wird. Der Nachteil dieses Verfahrens liegt jedoch darin, daß die Anzahl der Analogläufe pro Itera-

tion von der Anzahl der unabhängig skalierbaren Verstärker abhängt und daher bei Problemen mit kleiner Zeitkonstanten (lange Rechenzeit pro Lauf) zu einer nicht mehr akzeptablen Rechenzeit führt.

Die ideale Lösung des Problems der hybriden On-line-Skalierung ist eine Kombination der beiden Verfahren, wobei das erste Verfahren für eine überlauffreie Skalierung und anschließend das zweite Verfahren zur Nachskalierung oder optimalen Skalierung verwendet wird.

Auf Grund ungünstiger Spezifikationen und Eigenschaften des zentralen Overload-Signals und der Hold-Steuerung (Relay) am vorhandenen Analogrechner EAI 680 wurde bisher nur das unter b) beschriebene Verfahren realisiert.

Die Verstärker werden über das Amplifier Select System angewählt, dessen analoger Ausgang über einen Korrekturverstärker und einen 4-Quadranten DCA mit einem schnellen ADC verbunden ist. Über diesen ADC wird der jeweils angewählte Verstärker während des OP-Intervalls im Abstand von 40 µsec bei gleichzeitiger Berechnung der Extremwerte ausgelesen. Der DCA wird dabei auf den Wert $1/S_{OLD}$ gesetzt, um den vollen Operationsbereich der Verstärker auslesen zu können.



Abbruchkriterium und Fehlerkontrolle

Das Verfahren wird im Falle einer Nachskalierung dann beendet, wenn eine über- und unterlauffreie Schaltung erreicht ist. Bei einer optimalen Skalierung wird das Verfahren mit der auf eine über- und unterlauffreie Skalierung folgenden Iteration abgeschlossen.

Da das Verfahren nur bei Übereinstimmung von analoger Schaltung und digital gespeicherter Schaltstruktur, einwandfreiem Funktionieren der analogen Komponenten und Stabilität der Schaltung konvergiert, muß laufend eine Überprüfung auf Fehler durchgeführt werden.

Als Kriterium eignet sich dafür besonders der erste Übersteuerungszeitpunkt:

- Eine bis zu einem Zeitpunkt t_1 übersteuerungsfreie Analogschaltung kann bei der nächsten Iteration nur zum selben oder zu einem späteren Zeitpunkt übersteuern. Tritt eine Übersteuerung zu einem früheren Zeitpunkt auf, so bedeutet dies entweder fehlerhafte Schaltung, fehlerhafte Komponenten oder Instabilität der Schaltung.
- Tritt bei mehreren aufeinanderfolgenden Iterationen immer zum selben Zeitpunkt eine Übersteuerung auf, so bedeutet dies entweder eine nicht definierte Operation (z.B. $\sqrt{-x}$) oder einen sehr steilen Anstieg des Verstärkerausganges (z.B. bei schlecht skalierten Sprungfunktionen), sodaß mehrere Iterationen benötigt werden, bis sich der Übersteuerungszeitpunkt um 40 µsec ändert.

Zur Überprüfung der optimalen Skalierungsfaktoren wird die Skalierungseinheit SCU verwendet. Die Ungenauigkeit des optimalen Skalierungsfaktors darf bei linearen Elementen nicht größer als die Skalierungseinheit sein, d.h. für den analogen Ausgang muß folgende Bedingung erfüllt sein:

$$\frac{s(x) - SCU}{s(x)} \leq \max|\bar{x}(t)| \leq 1$$

Bei nichtlinearen Elementen mit abhängigem Skalierungsfaktor kann eine größere Ungenauigkeit entstehen.

Die Stabilität jedes Verstärkerausganges wird nach Erreichen von Über- und Unterlauffreiheit durch Vergleich der absoluten unskalierten Maxima zweier aufeinanderfolgender Iterationen überprüft. Der Verstärker wird dann als stabil angesehen, wenn der relative Fehler kleiner als 0.01 ist, also

$$\left| 1 - \frac{s^{(i)}(x).\max|\bar{x}^{(i)}(t)|}{s^{(i+1)}(x).\max|\bar{x}^{(i+1)}(t)|} \right| < 0.01$$

Untersteuernde Variable, deren Skalierungsfaktor immer gleich der Skalierungseinheit bleibt, können nicht besser skaliert werden und werden daher beim Abbruchkriterium nicht berücksichtigt.

D. Solar

GDP1 — *Ein Programmpaket zur numerischen Berechnung und graphischen Darstellung der Lösungen von Anfangswertproblemen gewöhnlicher Differentialgleichungen*

1. EINLEITUNG

Will man ein System von n gewöhnlichen Differentialgleichungen

$$\begin{aligned} y_1'(t) &= f_1(t,y_1(t), \ldots , y_n(t)) \\ &\vdots \\ y_n'(t) &= f_n(t,y_1(t), \ldots , y_n(t)) \end{aligned} \tag{1}$$

mit den Anfangsbedingungen

$$\begin{aligned} y_1(t_o) &= y_{1o} \\ &\vdots \\ y_n(t_o) &= y_{no} \end{aligned} \tag{2}$$

praktisch lösen, dann bieten sich hauptsächlich drei Zugänge an:

1) Man sucht einen geschlossenen Ausdruck für die Lösungsmannigfaltigkeit des Systems (1) und bestimmt die freien Parameter so, daß man eine Lösung erhält, die an der Stelle t_o die vorgegebenen Werte $y_{1o}, \ldots, y_{no}$ annimmt ("analytischer" Lösungsweg).

2) Man verwendet einen Analogrechner (bzw. den Analogteil eines Hybridrechners) zur Ermittlung einer Näherungslösung des Problems (1), (2).

3) Man verwendet numerische Näherungsverfahren, um sich auf einem Digitalrechner eine Näherungslösung zu verschaffen.

Um diese drei Lösungswege einander wertend gegenüberstellen zu können, muß die eigentliche Problemstellung einer genaueren Diskussion unterzogen werden. Dazu soll der Einfachheit halber das betrachtete explizite System gewöhnlicher Differentialgleichungen 1. Ordnung kurz

$$y' = f(t,y) \tag{3}$$

geschrieben werden, d.h. y bezeichnet eine Funktion $y: \mathbb{R} \to \mathbb{R}^n$ und $f: \mathbb{R}^{n+1} \to \mathbb{R}^n$. Die Anfangsbedingung lautet dann in vektorieller Schreibweise

$$y(t_o) = y_o. \tag{4}$$

Es werden stets Anfangswertprobleme betrachtet, bei denen die Lösung in einem Intervall $[t_o, t_{end}]$ gesucht ist.

Selbst bei den einfachsten Anfangswertproblemen vom Typ (3), (4)

$$y' = Ay \qquad (5)$$
$$y(t_o) = y_o$$

d.h. bei linearen Differentialgleichungen mit konstanten Koeffizienten (A ist eine *beliebige* n x n Matrix mit konstanten Elementen), wo sich die geschlossene Lösung durch

$$y(t) = \exp[A(t - t_o)]\, y_o \qquad (6)$$

sofort angeben läßt, ist man (außer in Spezialfällen) *nicht* in der Lage, für einen konkreten t-Wert y(t) exakt zu ermitteln: Die Matrixexponentialfunktion in (6) kann i.a. nur näherungsweise (und mit großem Aufwand) berechnet werden. Man wird sich also auch bei der Lösungsvariante 1) i.a. mit Näherungslösungen zufrieden geben müssen. Da somit bei allen drei Lösungswegen fehlerhafte Resultate zu erwarten sind, erhebt sich die Frage, wann man ein derartiges Resultat als Lösung akzeptiert. Im "analytischen" Sinn war diese Frage leicht zu beantworten: Wenn eine Funktion y die Differentialgleichung (3) erfüllt, d.h. wenn für jedes $t \in [t_o, t_{end}]$ die Beziehung

$$y'(t) = f(t, y(t)) \qquad (7)$$

erfüllt ist, und wenn y an der Stelle t_o den Wert y_o annimmt, dann handelt es sich um eine Lösung des gegebenen Anfangswertproblems (3), (4). Welche *Näherungs*lösungen als "Lösung" akzeptabel sind, hängt von dem konkreten Anwendungsfall ab; so wird z.B. die Flugbahnberechnung eines Satelliten eine höhere Genauigkeit erfordern als die grob-qualitative Ermittlung des Einschaltstromstosses eines elektrischen Schaltkreises. Kann man also die "praktische" Lösung des mathematischen Problems (3),(4) nicht von dem zugrundeliegenden konkreten Anwendungsproblem trennen? Da eine derartige Trennung äußerst erstrebenswert ist - sie ermöglicht erst die universelle Anwendbarkeit praktischer Lösungsverfahren - kann man sich z.B. der folgenden *erweiterten Problemdefinitionen* bedienen: Gesucht wird eine Funktion $\tilde{y}$, die von der exakten Lösung y des Problems (3),(4) um nicht mehr als eine - vom Anwender entsprechend vorgeschriebene - Toleranz abweicht.

Was leisten nun die drei oben angeführten Lösungswege bezüglich dieser neuen Problemstellung:

1) Eine genauere Diskussion des "analytischen" Lösungsweges ist nicht möglich. Wenn überhaupt eine geschlossene Lösung für (3), (4) gefunden werden kann (dies ist bei vielen praktisch auftretenden Problemen *nicht* der Fall), so

sind die zu erwartenden Schwierigkeiten doch von Fall zu Fall grundverschieden. Als Beispiel sei nur auf die in (6) benötigte Matrixexponentialfunktion verwiesen: abhängig von Eigenschaften der Matrix A kann es numerisch unmöglich sein, $\exp[A(t - t_o)]$ mit ausreichender Genauigkeit zu berechnen, um die vorgegebene Toleranz zu erfüllen. Da es äußerst schwierig ist, den Fehler bei der Berechnung der Matrixexponentialfunktion abzuschätzen, kann dieser Weg als ungeeignet für die Lösung des erweiterten Problems bezeichnet werden.

2) Der Analogrechner liefert stets (im Rahmen der gewählten Skalierung,...) das genauestmögliche Resultat. Damit ist eine Entkoppelung von Anwendungsproblem und praktischer Lösung gewährleistet. Ob das erhaltene Resultat jedoch die geforderte Genauigkeit aufweist (d.h. ob der Analogrechner überhaupt geeignet ist, das erweiterte Problem zu lösen), läßt sich ohne Zuhilfenahme "systemfremder" Hilfsmittel (z.B. einer "Referenzlösung" am Digitalrechner) nicht entscheiden. Der Analogrechner *allein* ist somit ebenfalls ungeeignet für die Lösung des erweiterten Problems.

3) Numerische Lösungsverfahren weisen ebenfalls einen entscheidenden Nachteil auf: Da bei diesen Verfahren eine Näherungslösung nur auf Grund einer endlichen "Stichprobe" des Richtungsfeldes der Differentialgleichung (3) konstruiert wird, ist die Qualität der Näherungslösung sehr stark davon abhängig, wie gut sich das Richtungsfeld durch diese endliche Anzahl von Werten charakterisieren läßt. Die meisten praktischen Probleme sind so beschaffen, daß die numerischen Lösungsverfahren einwandfrei funktionieren, *garantieren* kann man die Qualität der Lösung i.a. jedoch nicht.

   Die meisten numerischen Algorithmen kommen der Lösung der erweiterten Problemstellung (Anfangswertproblem plus Toleranz) sehr stark entgegen: In den Parameterlisten dieser Programme gibt es "Toleranzparameter", deren Wert den Erfordernissen des konkreten Anwendungsfalls angepaßt werden kann. Man muß sich jedoch klar machen, was die Funktion dieser Parameter ist. Alle Programme zur numerischen Lösung von Anfangswertproblemen (3),(4) beruhen auf sogenannten Vorwärtsschrittverfahren d.h. ausgehend vom Vektor y_o an der Stelle t_o werden schrittweise Näherungswerte $\tilde{y}_1$, $\tilde{y}_2$, ... an den Stellen $t_o < t_1 < t_2 < \ldots \leq t_{end}$ berechnet. Die vorgegebenen Toleranzwerte dienen lediglich zur geeigneten Bestimmung des Punktes t_{i+1}, wenn das Verfahren an der Stelle t_i bereits einen Näherungswert $\tilde{y}_i$ bestimmt hat, d.h. es wird *lokal* das Anfangswertproblem

$$\begin{aligned} y' &= f(t,y) \\ y(t_i) &= \tilde{y}_i \end{aligned} \qquad (8)$$

   so gelöst, daß $\tilde{y}_{i+1}$ an der (vom Algorithmus ermittelten) Stelle t_{i+1} von der

exakten Lösung des Anfangswertproblems (8) nicht stärker abweicht, als die vorgegebene Toleranz vorschreibt. Man beachte: Es wird die Toleranz *nicht* bezüglich der Abweichung von der exakten Lösung des ursprünglichen Problems (3),(4) verwendet. Selbst wenn der *lokale* Fehler tatsächlich die vorgegebene Toleranz erfüllt, ist somit nicht gewährleistet, daß dies für den *globalen* Fehler (an dem der Anwender letztlich interessiert ist) ebenfalls gilt.

Zusammenfassend läßt sich sagen, daß *keiner* der drei Wege in seiner herkömmlichen Form zur Lösung des erweiterten Anfangswertproblems geeignet ist. Bevor konkrete Lösungswege aufgezeigt werden, sollen noch kurz die *Vorteile* der drei Zugänge gegenübergestellt werden:

1) Die geschlossene Lösung einer Differentialgleichung ermöglicht eine genaue Analyse des *qualitativen* Verhaltens (z.B. Stabilitätseigenschaften,...). Da hier in erster Linie die konkrete Lösung eines oder mehrerer Anfangswertprobleme diskutiert werden soll, werden wir im folgenden den "analytischen" Lösungsweg ausklammern.

2) Der Analogrechner ermöglicht die *extrem schnelle* Ermittlung von Lösungen, wie sie z.B. bei vielen Echtzeitanwendungen (etwa bei Flugsimulatoren,...) unerläßlich ist. Die erforderliche Rechenzeit zur Lösung eines Systems gewöhnlicher Differentialgleichungen ist i. a. unabhängig von der Dimension n des Systems (dies ist auf die parallele Verarbeitung der einzelnen Komponenten zurückzuführen), aber auch unabhängig von Eigenschaften wie Linearität, Nichtlinearität usw. Die Möglichkeit der interaktiven Änderung von Parametern des Differentialgleichungssystems in Verbindung mit der sehr anschaulichen Lösungsdarstellung am Oszillographen gestattet es,in einfachster Weise das Verhalten der untersuchten Systeme zu studieren.

3) Programme am Digitalrechner haben den Vorteil der steuerbaren Lösungsgenauigkeit (was die Grundlage jeder Art von Fehlerschätzung ist) sowie den Vorteil der sehr einfachen Problemaufbereitung (der Hauptaufwand besteht im Verfassen eines Unterprogrammes zur Beschreibung der rechten Seite der Differentialgleichung (3)).

Als Lösungswege für das erweiterte Anfangswertproblem kann man auf Grund aller bisher genannten Eigenschaften z.B. folgende Alternativen in Betracht ziehen:

a) *Der rein digitale Weg:* In diesem Fall müssen die Programme mit einem effizienten und zuverlässigen Mechanismus zur Schätzung des *globalen* Fehlers ausgerüstet werden (der eine Entscheidung gestattet, ob die gelieferte Lösung für den Anwender akzeptabel ist oder nicht); die Möglichkeit zur graphischen Lö-

sungsdarstellung (Graphic Display, Plotter,...) muß geschaffen werden.

b) *Der hybride Weg:* In diesem Fall müssen wie bei a) Digitalrechner-Programme bereitgestellt werden, die eine Schätzung des globalen Fehlers ermöglichen. Diese Programme können dann zur Berechnen von "Referenzlösungen" verwendet werden, mit denen eine Abschätzung der Genauigkeit der am Analogrechner erhaltenen Lösungen ermöglicht wird. Wenn man sich in einer ersten Versuchsphase von der ausreichenden Genauigkeit der analogen Lösung überzeugt hat, können die eigentlichen "Produktionsläufe" (die sich z.B. nur durch geringfügige Parameteränderungen voneinander unterscheiden) am Analogrechner durchgeführt werden. Digitale Ermittlungen von Lösungen können immer wieder zur Genauigkeitskontrolle eingeschoben werden. Dieser kombinierte digital/analoge Lösungsweg bietet auch noch den weiteren Vorteil, daß die immer vorhandene Wahrscheinlichkeit eines Versagens der digitalen Programme (auf Grund der nur endlich vielen Informationen über die Differentialgleichung) durch Vergleich mit der analog erhaltenen Lösung noch deutlich verringert werden kann.

2. KONZEPT DES PROGRAMMPAKETS

Um die am Ende des vorigen Abschnitts skizzierten Lösungswege beschreiten zu können, waren, über vorhandene, leicht zugängliche Programme hinausgehend, folgende Maßnahmen erforderlich:

a) die Bereitstellung von Programmen zur digitalen Lösung von Anfangswertproblemen, die Informationen über den *globalen* Fehler jeder einzelnen Lösungskomponente liefern,

b) die Bereitstellung von Programmen zur graphischen Darstellung von (digital erhaltenen) Lösungen der Anfangswertprobleme und zur Darstellung von Fehlerkurven (auf Graphic Display, Plotter,...).

Wie man sieht, waren nur auf dem digitalen Bereich "Lücken" zu schließen. Das gesamte Paket GDP1 ist daher auf Digitalrechner zugeschnitten, was jedoch die Verwendung in der hybriden Lösungsvariante von Abschnitt 1 keineswegs ausschließt.

Um dem Ziel einer möglichst universellen Anwendbarkeit entgegenzukommen, wurde im GDP1 besonderer Wert auf die Bereitstellung einer großen Zahl von kompatiblen "Hilfsprogrammen" gelegt, die der Anwender dann in seinem (genau auf seine konkreten Anforderungen zugeschnittenen) Hauptprogramm aufrufen kann. Eine derartige Konzeption kommt erfahrungsgemäß den Anwendern im technisch-naturwissenschaftlichen Bereich (der von uns als Zielgruppe des GDP1 angesehen wurde) am weitesten entgegen. Ein Demonstrationshauptprogramm zeigt in konstruktiver Weise einige (aber

bei weitem nicht alle) Möglichkeiten der Verwendung der 41 Unterprogramme des GDP1 auf.

Als Programmiersprache wurde eine Teilmenge des FORTRAN 66-Standards verwendet, die größtmögliche Transportabilität der Programme gewährleistet. Die Differentialgleichungsprogramme lieferten ohne die geringste Änderung auf folgenden Anlagen vollkommen zufriedenstellende Resultate: CDC Cyber 74, DEC System 20, EAI Pacer 600, PDP 11/34, Siemens System 300/330 und auf einer Burroughs-Anlage. Für die graphische Ausgabe wurde CalComp-kompatible Basissoftware vorausgesetzt. Alle eventuell erforderlichen speziellen Anpassungen an ein bestimmtes graphisches System können durch Änderung einiger Größen in einem BLOCK DATA Unterprogramm bewerkstelligt werden.

3. DIFFERENTIALGLEICHUNGS-UNTERPROGRAMME

Die Zahl der numerischen Verfahren zur Lösung von Anfangswertproblemen gewöhnlicher Differentialgleichungen ist außerordentlich groß. Eine grobe Unterteilung dieser Methoden kann man nach der Klasse von Anfangswertproblemen vornehmen, für die das jeweilige Verfahren die größte Effizienz aufweist:

a) Verfahren, die *nicht-steife* Differentialgleichungen effizient lösen, bei steifen Differentialgleichungen jedoch einen unwirtschaftlich hohen Aufwand erfordern (der in vielen Fällen so groß sein kann, daß eine praktische Lösung mit diesen Verfahren überhaupt unmöglich ist);

b) Verfahren, die speziell zur praktischen Lösung von *steifen* Differentialgleichungen entwickelt wurden und die bei nicht-steifen Differentialgleichungen i.a. einen erheblich größeren Aufwand erfordern als die Verfahren der Gruppe a).

Bei *steifen* Differentialgleichungen (engl.: *stiff equations*) handelt es sich, grob gesprochen, um einen besonderen Typ von Differentialgleichungen mit stark variierenden Zeitkonstanten, der z.B. bei chemischen Reaktionsgleichungen, bei Regelungsproblemen, etc. häufig auftritt. Eine genaue Unterscheidung zwischen steifen und nicht-steifen Differentialgleichungen ist theoretisch schwierig und dem Anwender eines Programmsystems zur praktischen Lösung von Anfangswertproblemen *nicht* zumutbar. In einem praktisch verwendbaren Programmpaket müssen daher Möglichkeiten zur *automatischen* Unterscheidung der beiden Situationen vorgesehen sein.

In der Konzeption des GDP1 mußte die grundlegende Entscheidung getroffen werden, ob Programme für beide Typen (steif und nicht-steif) oder nur für einen Typ zur Verfügung gestellt werden sollen. Wegen der Zielsetzungen aus Abschnitt 2 (insbesondere im Hinblick auf den Einsatz in Verbindung mit einem Analog- bzw. mit

einem Hybridrechner) wurde im GDP1 auf Programme für steife Differentialgleichungen vorläufig verzichtet. Alle Programme des GDP1, die der Lösung von Anfangswertproblemen dienen, sind jedoch mit Entscheidungsmechanismen ausgerüstet, die es gestatten, steife Differentialgleichungen zu erkennen, und die dem Benutzer (über einen Parameter) eine Meldung über diesen Sachverhalt geben. Wenn ein Benutzer einen derartigen Hinweis erhält, dann weiß er, daß er durch die Verwendung eines Spezialprogramms für steife Differentialgleichungen (das er derzeit außerhalb des GDP1 suchen muß) eine Lösung seines Problemes mit geringerem Rechenaufwand erhalten kann.

Auch auf dem Teilgebiet der nichtsteifen Differentialgleichungen ist eine Vielzahl verschiedener Verfahren und fertiger Computer-Programme vorhanden. Als Grundlage der Methodenwahl für das GDP1 konnten umfangreiche theoretische und experimentelle Vergleichsstudien (z.B. der Gruppe von T.E.Hull in Toronto und der Gruppe von L.F. Shampine im Sandia Laboratory in Albuquerque, New Mexico) herangezogen werden. Dabei stellte sich für jene "mittleren" Genauigkeitsanforderungen (z.B. Toleranz von 10^{-3}, ... , 10^{-5} für den absoluten Fehler von "normal" skalierten Problemen, d.h. von Problemen, deren Lösung in der Größenordnung von 1 liegt), die bei technisch-naturwissenschaftlichen Anforderungen die wichtigste Rolle spielen, heraus, daß spezielle Runge-Kutta-Verfahren die wirtschaftlichste Lösungsmöglichkeit darstellen. Es handelt sich dabei um ein Formelpaar der Ordnung 4 und 5, das auf Fehlberg zurückgeht. Die Differenz der beiden Formeln wird als *lokale* Fehlerschätzung zur Schrittweitensteuerung verwendet.

Die Forderung nach einer Schätzung des *globalen* Fehlers kann auf mehrere Arten erfüllt werden. Die bekannteste Methode ist die zweifache Rechnung, einmal mit der Toleranz τ, dann mit $\tau/10$. Die Differenz der Lösungen dient bei dieser Vorgangsweise als Schätzgröße des globalen Fehlers des *ungenaueren* Resultats. Im GDP1 wurde ein besserer Weg beschritten: das Runge-Kutta Formelpaar wird bei jedem Schritt einmal mit dem Schrittweite h und zweimal mit h/2 angewandt, wobei die Integration mit den "Halbschritten" getrennt von jener mit den "Vollschritten" erfolgt. Die Differenz der so erhaltenen Lösungen (die eine Schätzung für den globalen Fehler der ungenaueren Lösung darstellt) wird durch *globale Extrapolation* verbessert, sodaß sie eine Schätzung für den globalen Fehler der genaueren Lösung darstellt. Diese Vorgangsweise ist sehr effizient (der Mehraufwand für die globale Fehlerschätzung liegt bei 60%) und bietet den Vorteil, daß zu jedem Zeitpunkt der Berechnung eine Schätzung des momentanen globalen Fehlers zur Verfügung steht (der Benutzer kann z.B. die Rechnung abbrechen, wenn diese Fehlerschätzung seine Genauigkeitsvorstellungen überschreitet, und mit einer verkleinerten Toleranz die Rechnung neu starten).

4. PROGRAMME ZUR GRAPHISCHEN AUSGABE

Hier sind zwei Gruppen zu unterscheiden:

a) Programme zur Ausgabe von Lösungen
   a1) als Funktionen der unabhängigen Veränderlichen t
   a2) als Phasenbilder

b) Programme zur Ausgabe von Fehlerkurven

Da die numerischen Lösungsalgorithmen in den meisten Fällen so große Schrittweiten wählen, daß eine Verbindung der erhaltenen Näherungswerte $\tilde{y}_1$, $\tilde{y}_2$,... durch einen Polygonzug zu einer sehr "eckigen" graphischen Darstellung führen würde, sind im GDP! Interpolationsprogramme vorhanden, die eine glatte *Lösungs*darstellung ermöglichen. Bei den *Fehlerkurven* wurde absichtlich die Form des Polygonzuges gewählt, um auf die größere Unsicherheit der Fehlerkurve zwischen den Punkten t_1, t_2,... optisch hinzuweisen. Um auch die Darstellung von kleinen Abschnitten der y-Achse zu ermöglichen, sind Clipping-Algorithmen in GDP!-Unterprogrammen implementiert worden.

5. BEISPIEL

Das restringierte Dreikörperproblem

$$x'' = 2y' + x - \mu'(x + \mu)/r_1^3 - \mu(x - \mu')/r_2^3$$

$$y'' = -2x' + y - \mu' y/r_1^3 - \mu y/r_2^3$$

mit

$$r_1 = [(x + \mu)^2 + y^2]^{1/2}$$

$$r_2 = [(x - \mu')^2 + y^2]^{1/2}$$

und den Konstanten

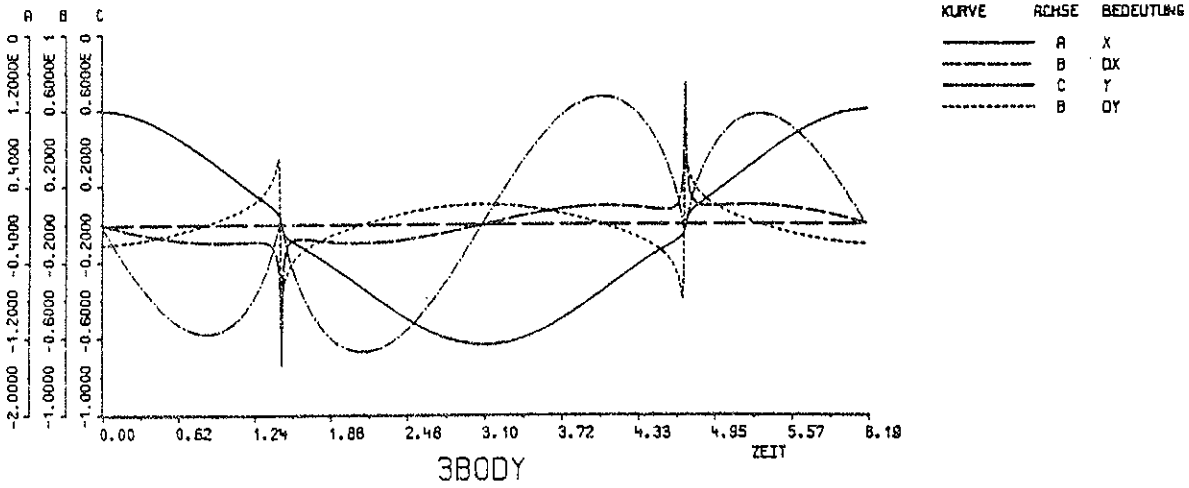
$$\mu := 1/82.45 \quad , \quad \mu' := 1 - \mu$$

beschreibt die Flugbahn eines Satelliten, der sich unter Einfluß der Anziehungskräft von Erde und Mond in einem Koordinatensystem bewegt, welches die Positionen von Erde und Mond fest läßt. Mit den Anfangswerten

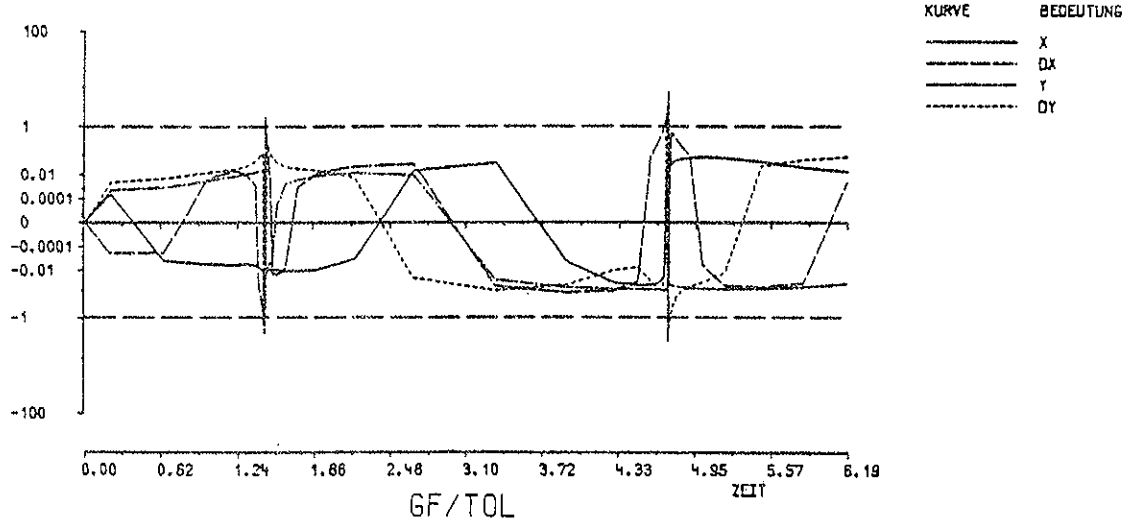
$$x(0) = 1.2 \qquad x'(0) = 0$$

$$y(0) = 0 \qquad y'(0) = -1.049357509...$$

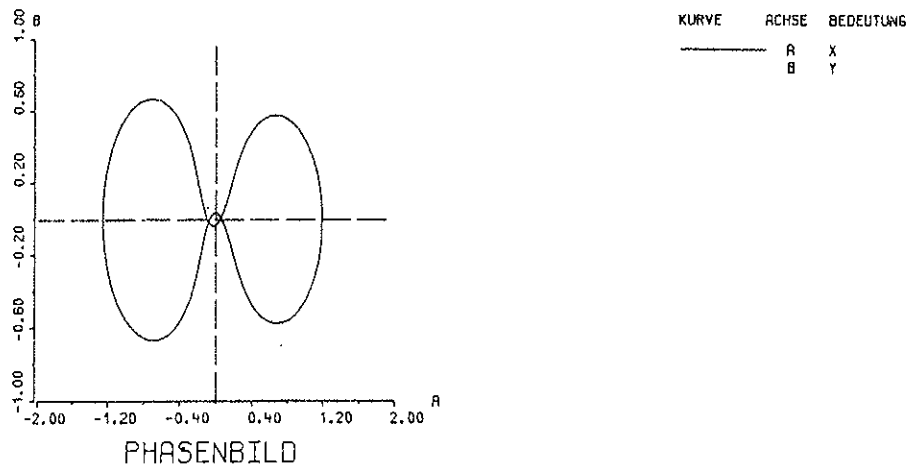
ist die Bewegung periodisch mit der Periode 6.19216... (Arenstorf-Bahn 1). Dieses Anfangswertproblem stellt relativ hohe Anforderungen an die Schrittweitensteuerungsstrategie eines numerischen Lösungsverfahrens. Mit einer Toleranz von 10^{-3} für die absoluten *lokalen* Fehler liefert das Demonstrationshauptprogramm des GDPI die folgenden graphischen Darstellungen:



3BODY



GF/TOL



PHASENBILD

Die Darstellung mit dem Titel GF/TOL bedarf einiger Erläuterungen: Hier werden die Schätzungen des globalen Fehlers in Relation zur Toleranz in einem nichtlinearen Maßstab dargestellt. Ein Überschreiten des Wertes 1 (bzw. Unterschreiten von -1) bedeutet also, daß an diesen Stellen die Schätzung des globalen Fehlers größer ausgefallen ist als die Toleranz, wobei sich aus der Zeichnung grob entnehmen läßt, um welchen Faktor diese Überschreitung stattgefunden hat. Die Toleranz von 10^{-3} (für den lokalen Fehler) konnte in diesem Fall somit nur einen globalen Fehler kleiner als 10^{-2} garantieren, obwohl an manchen Stellen der globale Fehler deutlich kleiner ausgefallen ist.

6. ZUSAMMENFASSUNG

In dem Programmpaket GDPI, das ein Ergebnis der Zusammenarbeit des Hybridrechenzentrums der Technischen Universität Wien und des Instituts für Numerische Mathematik der Technischen Universität Wien ist, wurde dem technisch-naturwissenschaftlichen Anwender ein Softwareprodukt zur Verfügung gestellt, das eine zufriedenstellende Lösung von (nichtsteifen) Anfangswertproblemen gewöhnlicher Differentialgleichungen (mit einer vom konkreten Anwendungsfall abhängigen Genauigkeitsvorstellung) ermöglicht. Es sei noch darauf hingewiesen, daß die Programme des GDPI am Hybridrechner der Technischen Universität Wien auf Platte gespeichert sind und problemlos abgerufen werden können, und daß ausführliche Beschreibungen am Hybridrechenzentrum erhältlich sind.

Friedrich Blöser (Hybridrechenzentrum der TU Wien)
Erwin Mitstorfer
Christoph W. Überhuber (Inst. f. Num. Mathematik, TU Wien)

COMPUTERUNTERSTÜTZTER UNTERRICHT IM HÖRSAAL X

Problemstellung

Zur Erweiterung der didaktischen Möglichkeiten im Rahmen der Lehrveranstaltungen des Instituts für elektrische Regelungstechnik, Vorstand Prof. Dr. A. Weinmann, wurde ein direkter Informationsaustausch zwischen der Hybridrechenanlage und einer Videoanlage im Hörsaal gewünscht. Ziel dieser Arbeit sollte es sein, daß während einer Vorlesung verschiedene graphische Darstellungen mit beliebig veränderbaren Parametern von der Hybridrechenanlage berechnet werden und dann im Hörsaal dargestellt werden.

Um diese Aufgabe zu erfüllen, benötigt man eine Ausgabeeinheit an der Hybridrechenanlage, eine Übertragungsleitung, Graphiksysteme, die die digitalen Signale für handelsübliche Fernsehapparate aufbereiten, und ein Steuergerät für den gesamten Betriebsablauf.

Hardwarekonzept

Im weiteren werden der Bau und die Konstruktion eines Datenverbundes zwischen der mikroprozessorgesteuerten Fernsehanlage (Graphiksystem) im Hörsaal X und dem zur Simulation verwendeten Hybridrechner beschrieben.

Da an diesem Datenaustausch mehrere Prozessoren mit verschiedener Arbeitsgeschwindigkeit beteiligt sind, wird für die Übertragung ein HAND SHAKE Modus verwendet. Zur Erhöhung der Betriebssicherheit erfolgt die Datenübertragung mittels Differenzsignalen. Da die Übertragungsleitung eine Länge von ca. 200m hat, wird eine Potentialtrennung der Datensignale vorgenommen. Diese Potentialtrennung mittels Optokoppler begrenzt die Übertragungsgeschwindigkeit auf 100 kByte/s.

Zur Erleichterung des Betriebs und der Wartung der Anlage ist eine Gegensprechanlage parallel zum Datenweg installiert.

Eine weitere Aufgabe ist die Aufbereitung und die Verteilung der Videosignale für die Fernsehapparate im Hörsaal X.

Bild 1 zeigt den Datenfluß zwischen der Hybridrechenanlage und den peripheren Baugruppen. Wie aus Bild 1 ersichtlich, wurde

auch eine Nebenstelle im Labor 1 des Instituts eingerichtet, um auch für den Laborübungsbetrieb eine Zugriffsmöglichkeit zur Anlage zu besitzen.

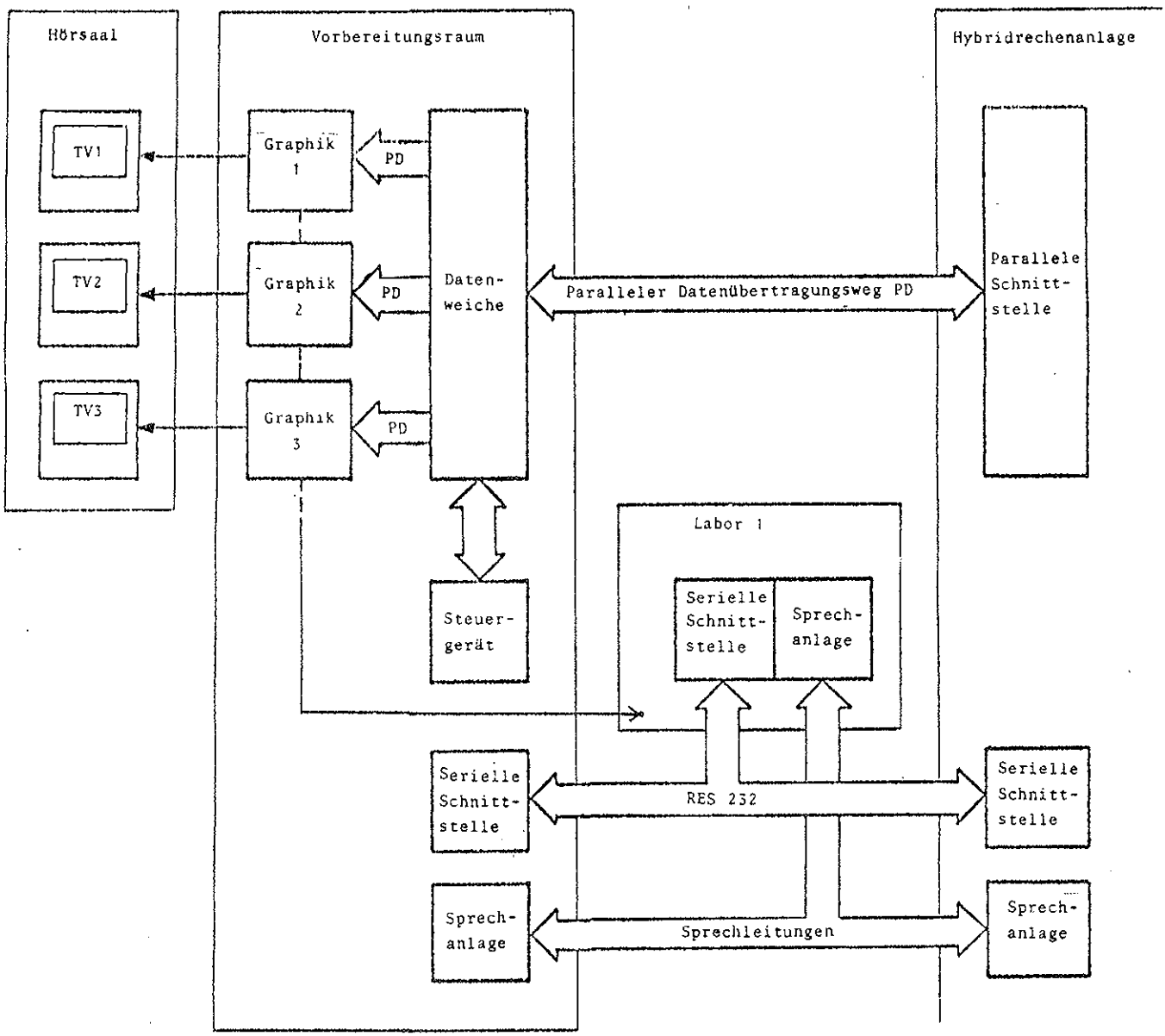


Bild 1: Anlagenübersicht des graphischen Systems

Insgesamt sind 4 Mikroprozessoreinheiten mit dem Leitrechner verbunden und zwar 3 Z-80 Videomikrocomputer (Graphik 1-3) und zur Verwaltung des Datenweges ein Steuergerät mit einer SC/MP Platine.

Die Bildinformation wird Byte-seriell übertragen (8bit/Byte). Daher besteht der Datenübertragungsweg aus 10 bidirektionalen

Kanälen. Für die Signalübertragung werden 8 Datenkanäle und ein Kanal STROBE in Übertragungsrichtung, sowie ein Kanal für das Signal READY in Gegenrichtung benötigt. Zusätzlich existieren zwischen Steuergerät und Hybridrechenanlage noch 5 Hilfskanäle für betriebstechnische Aufgaben. Die Datenweiche ermöglicht sowohl einen bidirektionalen Austausch zwischen Steuergerät und Hybridrechenanlage, als auch einen monodirektionalen Datenfluß von der Hybridrechenanlage oder vom Steuergerät zu den Graphiksystemen. Das Blockschaltbild 2 zeigt den Aufbau des Übertragungsweges mit den einzelnen Signalgruppen.

Je nach Signalfluß wird der Übertragungsweg entsprechend eingeschaltet und bleibt in diesem Zustand, bis er wiederum vom Steuergerät verändert wird.

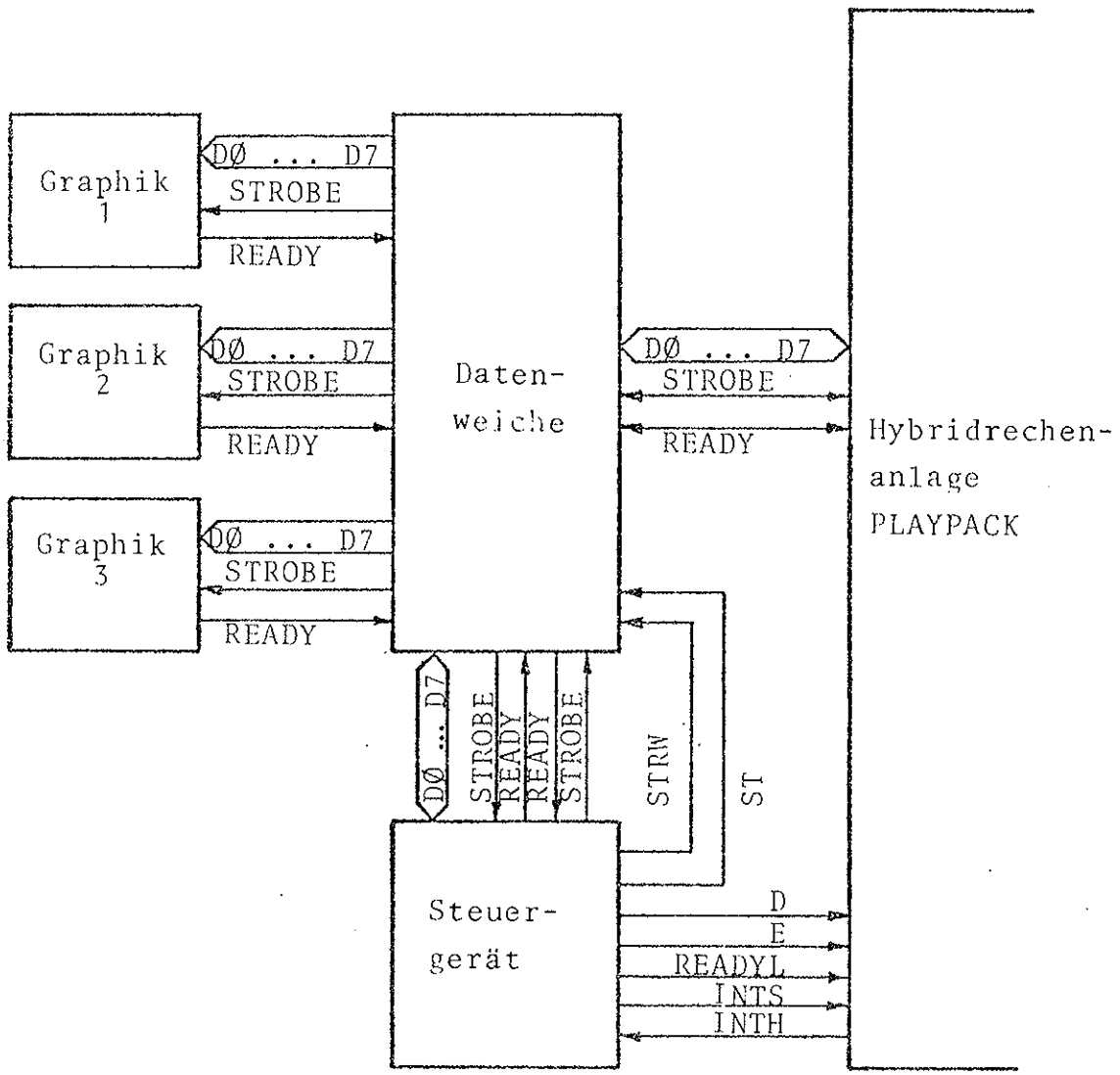


Bild 2: Blockschaltbild des parallelen Datenübertragungsweges

Softwarekonzept:

Das Gesamtkonzept kann in zwei große Blöcke geteilt werden:

a) Problemformulierung im Dialog Vortragender - Hybridrechner und hybride Problemlösung

b) Datenaufbereitung zur Übertragung und Erzeugung eines TV-Bildes.

Die Ergebnisse zu Punkt a) werden zu einem späteren Zeitpunkt präsentiert werden, da zur Zeit hiezu noch zwei Diplomarbeiten ausgeführt werden. In diesem Beitrag soll speziell Punkt b) behandelt werden.

Die Beschreibung der Prozessorsoftware für die Videoanlage erfolgt überblicksmäßig und chronologisch bezüglich des Ablaufes einer Vorführung im Hörsaal X.

Nach dem Systemstart (POWER ON RESET) durchlaufen alle mP - Systeme unabhängige Initialisierungsprogramme, die einerseits einen Systemtest der drei Videogeräte mit abschließendem Testbild generieren, andererseits durch gezielten Eingriff des Steuerprozessors über den Datenkanal das gesamte Subsystem im Hörsaal X in einen definierten Zustand bringen, der einen sicheren Datenaustausch mit dem Hybridrechner gewährleistet.

Bezüglich der Stellung innerhalb der System-Hierarchie sind die Videogeräte reine Datenempfänger. Das Steuergerät hingegen kann über seine diversen Serviceroutinen unbeschränkt arbeiten, sofern nicht der Datenweg durch den Leitrechner besetzt ist.

So können vom Vortragenden via Steuerprozessor auf jedem Bildschirm Leuchtzeiger (Cursor) gesetzt werden oder aber die einzelnen TV-Geräte wahlweise zu- und weggeschaltet werden, wenn der Vortrag es erfordert. Die zugehörige Information wird dann dem entsprechenden Videoprozessor vom Steuerprozessor übermittelt. Ferner können Parameterwerte (auf einer Hilfskonsole einstellbar) über den Steuerprozessor an den Leitrechner gesandt werden. Umgekehrt sorgt der Steuerprozessor für die richtige Schaltung des Datenmultiplexers gemäß den Hybridrechnerbefehlen und übernimmt die Startwerte für die Parameter auf die Hilfskonsole. Die Verarbeitungssoftware der Videoprozessoren erlaubt die Wahl verschiedener Programme durch Setzen von Steuerbits im höherwertigen Byte der Bildpunktinformation, die der Leitrechner sendet.

Folgende Programmanforderungen können gestellt werden:

- Zeichnen von beliebigen Kurven
- Punkt- und blockweise Datenübernahme
- Wahl der Zeichenfarbe (7 Farben möglich)
- Löschen einzelner Kurven aus einem kompletten Bild
- Beschriften des Kurvenbildes
- Ausgabe längerer zusammenhängender Texte
- Herausheben einzelner Bildelemente durch Invertieren
- Blinkender Cursor an jeder beliebigen Stelle des Bildes
- Wahl der Hintergrundfarbe (schwarz oder weiß)
- Löschen des gesamten Bildes
- Abschalten des Fernsehbildes (Information bleibt erhalten) zur Beschleunigung des Datendurchsatzes
- Zu- und Wegschalten der seriellen Schnittstelle jedes einzelnen Prozessors
- Aufruf des Monitorprogrammes
- Aufruf eines Testprogrammes (resident)
- Aufruf eines Sonderprogrammes zum parallelen Datenaustausch zwischen den Videoprozessoren
- Einbeziehung eines Reservespeicherbereichs

} für Servicezwecke

Die Hard- und Software dieses gesamten "Videosubsystemes" wurde im Rahmen mehrerer Diplomarbeiten am Institut für elektrische Regelungstechnik zusammengestellt und entwickelt.

Am Leitrechner (PACER 100) ist eine Schnittstellensoftware implementiert, die die vom Simulationsprogramm errechneten Ergebnisse in "prozessorgerechter" Form anbietet. Diese Software stützt sich auf die Graphiksoftware, die allgemein am Hybridrechner in Verwendung steht, und wurde vom Hybridrechenzentrum erstellt.

Nach Fertigstellung der Kommunikationssoftware und einer Beispiel-Bibliothek, die im Rahmen zweier Diplomarbeiten am Institut für elektrische Regelungstechnik in Zusammenarbeit mit dem Hybridrechenzentrum entwickelt werden, ist ein weiterer Bericht zu diesem Thema geplant.

M. Haider
Institut für elektrische Regelungstechnik

SIMULATION EINES MODELLS FÜR DIE WAHRNEHMUNG VON LICHTSCHWANKUNGEN

Dipl. Ing. Hermann Geyer
Inst. für Allgemeine Elektrotechnik und Elektronik
Technische Universität Wien

1. AUFGABENSTELLUNG

Das menschliche Auge nimmt Lichtschwankungen je nach Kurvenform und Frequenz dieser Schwankungen unterschiedlich wahr. In Bild 1 sind unter anderem Meßergebnisse eingetragen, welche Amplitudenwerte bei verschiedenartigen Lichtschwankungen auftreten, wenn diese Schwankungen gerade an der Grenze der Wahrnehmbarkeit sind [1].
Es wurde vom Verfasser ein Modell entwickelt ([3], [4], [5]), das imstande ist, diese Meßwerte zu beschreiben.
Es konnten einige Parameter des Modells zwar ohne Simulation auf graphischem Weg bestimmt werden [5], zur Bestimmung der restlichen Parameter erwies sich aber die Simulation des Modells am Hybridrechner als sinnvoll (siehe Bild 1).

2. BESCHREIBUNG DES MODELLS

Aus der relativen Lichtschwankung

$$r(t) = \frac{L(t) - L_o}{L_o}$$

$L(t)$ ... zeitl. Verlauf der Leuchtdichte
L_o ... zeitl. Mittelwert der Leuchtdichte

ist entsprechend der folgenden Gleichungen die Größe $b(t)$ zu bilden:

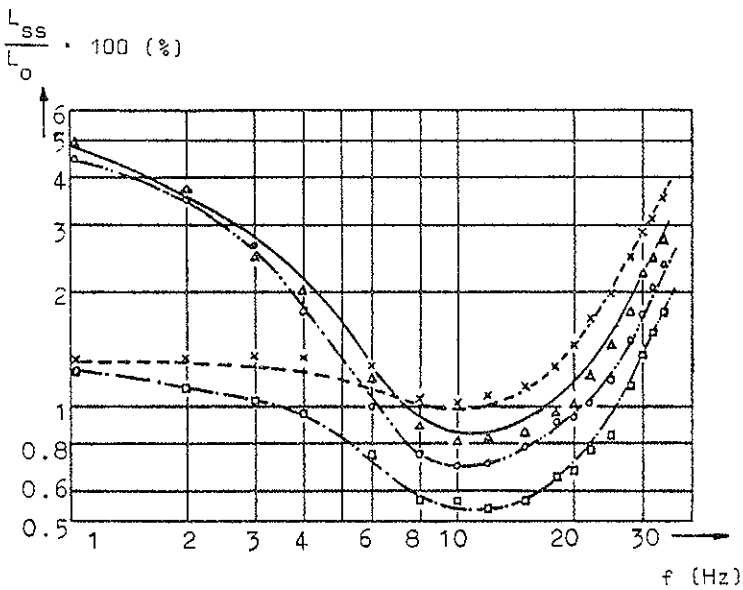
$$\ddot{a}(t) = V\omega_1^2 r(t) - 2d_1\omega_1\dot{a}(t) - \omega_1^2 a(t)$$

$$\ddot{h}(t) = a(t) + (T_A + T_B)\dot{a}(t) + T_A T_B \ddot{a}(t) - \omega_2^2 h(t) - 2d_2\omega_2\dot{h}(t)$$

$$f(t) = h^2(t)$$

$$\dot{g}(t) = \frac{1}{T_Q} f(t) - \frac{1}{T_Q} g(t)$$

$$b(t) = \sqrt{g(t)}$$



Lichtschwankung		Meßwerte	Rechenwerte
dreieckförmig	t	△	———
sägezahnförmig	t	×	- - - - - -
rechteckförmig	t	□	— · — · — · —
sinusförmig	t	○	— · · — · · —

Bild 1: Vergleich der relativen Lichtschwankungsamplituden an der Grenze der Wahrnehmbarkeit. Meßwerte nach [1], Rechenwerte auf Grund der Simulation des neuen Modells.

L_{ss} ... Amplitude der Leuchtdichteschwankung, von Maximum zu Minimum gemessen.
L_o ... Mittelwert der Leuchtdichte
f_o ... Grundfrequenz der Leuchtdichteschwankung

Nur wenn b(t) größer als der Schwellwert b_s=1 ist, ist die vorangegangene Lichtschwankung wahrnehmbar.

Parameterwerte:
bereits graphisch grob vorbestimmt:

ω_1 = 120 sec^{-1} $\quad d_1$ = 0.55
ω_2 = 50 sec^{-1} $\quad d_2$ = 0.55
V = 1.5

durch die Simulation zu bestimmen:

T_A, T_B, T_Q

3. BESTIMMUNG DER PARAMETER

Die richtigen Parameter sind gefunden, wenn das Modell die gleichen Werte für die gerade noch erkennbaren Schwankungen der Amplitude liefert, wie nach Bild 1 gemessen wurden. Es müßten also zeitliche Verläufe der Variablen r(t) erzeugt werden, die in Kurvenform und Frequenz jenen Lichtschwankungen entsprechen, mit denen die in Bild 1 eingetragenen Meßwerte gewonnen wurden. Dann wäre jener Wert der Amplitude von r(t) einzustellen, sodaß der Maximalwert von b(t) gerade b_s=1 wird (Grenze der Wahrnehmbarkeit, ermittelt mit Hilfe des Modells). Da aber die Amplituden von r(t) und b(t) bei einer bestimmten Kurvenform und Frequenz stets proportional zueinander sind (Amplitudenlinearität, [5]), genügt es, r(t) mit konstanter Amplitude $\hat{r}$=1 zu generieren und dann aus dem sich jeweils ergebenden Maximalwert b_{max} von b(t) jene Amplitude von r(t), bei der sich $b_{max} = b_s = 1$ ergeben hätte, zu berechnen:

$$\hat{r}_{bemerkbar} = \frac{1}{b_{max}}$$

4. SIMULATION

Analoger Teil der Simulation

Bemerkungen:
- Es handelt sich um qualitative Koppelpläne.
- Potentiometer sind auf positive und negative Werte setzbar (DCAs).
- Es sind nur jene Potentiometer eingezeichnet, die zur Darstellung eines Parameterwertes nötig waren; z.T. waren zur Skalierung der Schaltung (mit Unterstützung durch HYBSYS [6]) weitere Potentiometer nötig.
- Die Invertierung durch Verstärker ist nicht berücksichtigt, nur speziell als INV bezeichnete Verstärker invertieren.
- Switches sind für die Zeichnung ohne Innenwiderstand angenommen.

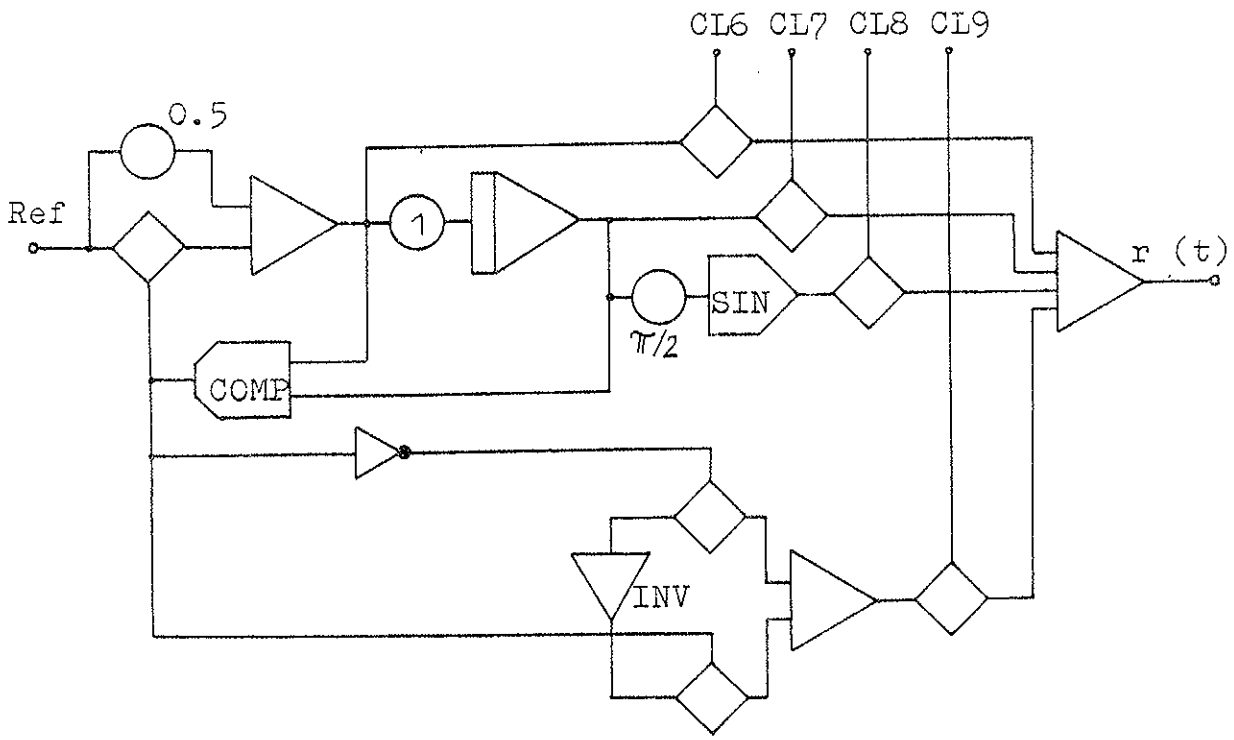


Bild 2: Generierung der verschiedenen Lichtschwankungen r(t)

Potentiometer 1 ... Beeinflussung der Frequenz von r(t) (1 bis 36 Hz)
CL6 bis CL9 ..... Anwahl der Kurvenform von r(t)
Bei Anwahl von CL9 muß die Frequenz um einen Faktor 2 herabgesetzt werden.

rechteckförmig ... CL6
dreieckförmig ... CL7
sinusförmig ... CL8
sägezahnförmig ... CL9

Amplitude stets gleich 1

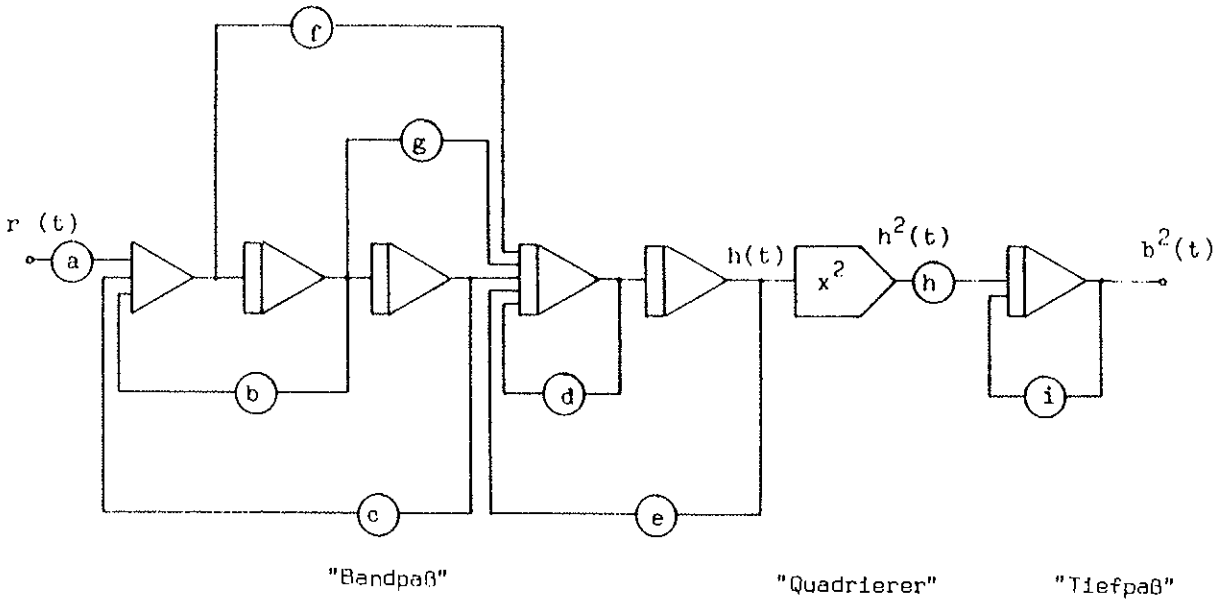


Bild 3: Signalverarbeitung im Modell

Parameter des Modells:

T_A, T_B, ω_1, ω_2, d_1, d_2, V; T_Q.

Die Potentiometer in Bild 3 müssen folgende Werte darstellen:

$a = V\omega_1^2/T_A/T_B$

$b = -2d_1\omega_1$

$c = -\omega_1^2$

$d = -2d_2\omega_2$

$e = -\omega_2^2$

$f = T_A T_B$

$g = -T_A - T_B$

$h = 1/T_Q$

$i = -1/T_Q$

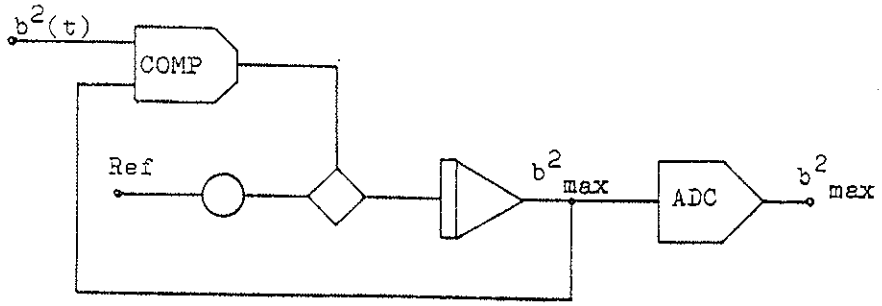


Bild 4: Maximumbildung der Ausgangsgröße des Modells (nach [2]).

Digitaler Teil der Simulation

Das Ziehen der Quadratwurzel aus der Ausgangsgröße b^2_{max} der Analogschaltung erfolgte nach Auslesen über den ADC im Digitalrechner. Weiters wurden vom Digitalrechner die Steuerung des Analogrechners für eine benutzerfreundliche Bedienung (z.B. mehrere Analogläufe mit verschieden wählbarer Kurvenform und Frequenz von r(t)) sowie die Aufbereitung der ausgelesenen Daten (z. B. Errechnung und Ausgabe der Abweichung zwischen Meßwerten und Simulationswerten für die gerade wahrnehmbare Schwankungsamplitude) übernommen. Diese Aufgabe konnte ebenfalls sinnvoll mit Hilfe des Hybrid Simulation Package [6] gelöst werden.

5. ERGEBNISSE

Die endgültig ermittelten Parameterwerte sind:

$\omega_1 = 124\ sec^{-1}$ $\quad d_1 = 0.58$

$\omega_2 = 49\ sec^{-1}$ $\quad d_2 = 0.55$

$V = 1.59$

$T_A = -0.043$ sec $\quad T_B = -0.08$ sec

$T_Q = 0.138$ sec

Die Simulationsergebnisse mit diesen Parametern sind - im Vergleich zu den Meßwerten - in Bild 1 eingetragen.

SUMMARY

This article deals with the simulation of a model describing the perception of changes in brightness (flicker).

A model, which describes the flicker sensitivity of men, was developed by the author. The model's parameters were determined partly by simulating the model on the hybrid computer and comparing the simulation results with reality.
The model is partly nonlinear. Some considerations,which made the simulation easier, are specified. The comparison with reality only at the limit of perceptibility is possible. The model's statements about this case coincide with reality with an error of not more than 15%.

6. LITERATUR

[1] CARJELL, U.: Ein Beitrag zur Beurteilung des Lichtflimmerns bei Netzspannungsschwankungen.
Dissertation, Rhein. Westfäl. Technische Hochschule, Aachen 1972.

[2] FILIPSKY, H.; AIBEL, H.; RATTAY, F.: Die Optimierung der elastischen Lagerung eines Verbrennungsmotors mit Hilfe der Hybridrechenanlage an der TU Wien.
INTERFACE 9, Juni 1977.

[3] GEYER, H.: Modell für die Wahrnehmung von Flicker.
Erscheint demnächst im etz-Archiv.

[4] GEYER, H.: Modelling the perception of changes of brightness.
Simulation '80, 25.-27. Juni 1980, Interlaken.

[5] GEYER, H.: Modellentwurf für die Störempfindung durch Lichtflimmern mit Berücksichtigung von Phaseninformation und begrenztem Erinnerungsvermögen.
Dissertation, Technische Universität Wien, in Fertigstellung.

[6] SOLAR, O.: HYBSYS - interaktiver hybrider Prozessor.
Hybridrechenanlage, Technische Universität Wien, 1979.

aus dem praktikum

SCHACHAUSGABE-PROGRAMM

G. Zemanek
Praktikumsarbeit, ausgeführt am Institut für Numerische Mathematik.
Betreuer: C. W. Überhuber

1. EINLEITUNG

An der Hybridrechenanlage der TU Wien wurde ein Programm zur figuralen Ausgabe des Schachbretts entwickelt. Ein Dialogprogramm ermöglicht damit das Schachspielen an (nicht mit!) der Anlage.

2. AUFGABENSTELLUNG

Erstens sollte ein AUSGABEPROGRAMM zur Bildschirmausgabe eines Schachbretts mit beliebiger Figurenposition entwickelt werden, wobei eine symbolische Darstellung ähnlich jener in den Schachpublikationen erreicht werden sollte. Diese Aufgabe sollte unter Verwendung des an der Hybridrechenanlage vorhandenen Gerätes TEKTRONIX 4006 GRAPHIC DISPLAY gelöst werden.

Zusätzlich sollte zweitens ein Dialogprogramm entwickelt werden, das durch Eingabe der Züge beider Spieler ermöglicht, sowohl an der Anlage zu spielen, als auch eine Partie nach einem Protokoll nachzuspielen. Um das Studium von Varianten zu erleichtern, sollte es möglich sein, Züge zurückzunehmen.

Zur Darstellung des Schachbretts mußten verschiedene physiologische und psychologische Anforderungen mit den Einschränkungen der Geräte-Hardware in Einklang gebracht werden. Bedingt durch den dabei betriebenen Aufwand wurde das Dialogprogramm im Rahmen der Praktikumsarbeit auf ein wohlabgerundetes Minimum beschränkt.

3. DAS AUSGABEPROGRAMM

Das Programm wurde als Unterprogramm mit zwei Parametern, BRETT und MODE, konzipiert. Es konstruiert aus der 8x8-Matrix BRETT die Darstellung des Schachbretts auf dem Bildschirm. Die Matrix enthält verschlüsselt die Figuren an ihren Positionen (i,j), wobei der erste Index aus der üblichen Notation mit A bis H in 1 bis 8 zu übersetzen ist. Als zusätzliche Annehmlichkeit für den Benützer wurde der Parameter MODE eingeführt, mit dem gewählt werden kann, ob das Schachbrett aus der Sicht des weißen oder aus der Sicht des schwarzen Spielers gezeigt werden soll.
Zur Ausgabe wurden Unterprogramme der an der Hybridrechenanlage verfügbaren Plotter Software verwendet, die auch für das Graphic Display geeignet sind.

3.1 Realzeitüberlegungen

Es war die Grundeigenschaft des Gerätes zu beachten, mit weißen, nur geraden Strecken auf dunklem Hintergrund zu arbeiten. Diese sind die sichtbaren Teilzüge eines durch dunkle unsichtbare Teilstrecken zusammenhängend ergänzten Polygonzuges, den das Gerät mit einer Geschwindigkeit von 3.6 msec pro Teilstrecke ausgibt. Die Länge der Teilstrecke spielt dabei keine Rolle. Die Ausgabezeit war eine wesentliche Kenngröße für die Lösung, da das Gerät diesen Polygonzug nur ergänzen, nicht aber teilweise löschen kann. Nach jedem Schachzug ist deshalb das gesamte Bild wieder auszugeben, weil wenigstens eine Figur von ihrem Standfeld fortzieht. Die so entstehende Wartezeit sollte 5 sec nicht überschreiten, und daher war der Umfang des Polygonzuges auf ca. 1388 Teilstrecken begrenzt. Es mußten dabei auch die dunklen Verbindungslinien mitgezählt werden.

3.2 Darstellung der Figuren

Der Darstellung der Figuren wurde besondere Sorgfalt zugewendet. Denn von der raschen und klaren Verständlichkeit der Figurensymbole hängt die Qualität der Gesamtlösung in starkem Maße ab. Die Suche nach einschlägiger Literatur wurde nach entsprechendem Zeitaufwand aufgegeben. In den leicht zugänglichen Quellen sind Programme der gesuchten Art nicht beschrieben (obwohl es in einigen Forschungszentren solche Programme geben muß).

Um die eigene Entwicklung kritisch kontrollieren zu können, wurden 3 Kriterien für die Güte des Bildes aufgestellt:

1) Die verschiedenen Schachfiguren müssen eindeutig und auf einen Blick klar erkennbar und unterscheidbar sein.
2) Figuren gleichen Typs, aber verschiedener Farbe müssen einerseits als vom gleichen Typ erkennbar sein, andererseits muß auf einen Blick klar eine weiße von einer schwarzen Figur unterscheidbar sein.
3) Die Darstellung darf nicht in Konflikt geraten mit der Kennzeichnung der weißen Felder des Brettes.

Dem Kriterium 1) stand die besondere Schwierigkeit entgegen, daß die Software runde Formen in Polygonzüge auflöst und daher bei vielen Krümmungen eine große Anzahl von Einzelstrecken erzeugt, wodurch sich die Ausgabezeit unmäßig erhöht. Um das Kriterium zu erfüllen, können

zwei Eigenschaften der Symbole verwendet werden: Form und Größe. Die Formen der Figuren konnten trotz weitgehender Linearisierung den in Schachpublikationen üblichen Formen angenähert werden. Der Größe der Figuren wurde nicht nur durch verschiedene vertikale Abmessungen entsprochen, sondern auch die Breite der Standlinien der Figuren wurde differenziert, sodaß "schwere" oder "wichtige" Figuren wie König und Dame das Feld mehr ausfüllen als leichte Offiziere, die wiederum das Feld mehr ausfüllen als die kleinsten Figuren, die Bauern.

Kriterium 2) verlangt, daß Figuren verschiedener Farbe gleichen Typs kongruente Umrisse besitzen müssen. Die Unterscheidung muß dann durch die Auffüllung der Innenfläche der weißen Figuren erzielt werden. Der naheliegende Gedanke, die weißen Figuren zu schraffieren, erwies sich als nachteilhaft. Erstens wird durch die Schraffierung die Zahl der Einzelstrecken erhöht. Eine überschlagsmäßige Berechnung zeigt, daß die Schraffierung aller weißen Figuren fast 2 sec Ausgabezeit gekostet hätte, die Hälfte davon nur für unsichtbare Verbindungslinien. Außerdem stellte sich heraus, daß der Erkennungseffekt des Schraffierens fraglich gewesen wäre, denn es ist eine Eigenschaft der vorgegebenen Geräte-Hardware, daß die Helligkeit eines Striches von seiner Länge abhängt (kürzere Striche werden langsamer durchlaufen und dadurch fetter), und damit hätten die Schraffen zu nicht einheitlicher Helligkeit geführt. Da dies auch für die weißen Felder zutrifft, wären weiße Figuren in weißen Feldern zu einem undeutlichen Schatten ihrer selbst geworden. Da das Schraffieren der weißen Felder, wie im nächsten Abschnitt gezeigt wird, nicht möglich war, wurde auch für die Figuren nach einer anderen Lösung gesucht. Als entscheidende Idee mit überraschend guter Wirkung erwies sich hier die Anwendung des Moiré-Effektes. Die Auffüllung wurde durch dachartig aufeinandertreffende Strahlenbüschel gelöst. Diese Büschel gehen von den beiden unteren Ecken der weißen Figuren aus und erzeugen einen Moiré-Effekt, der optisch sehr gut wirkt, sichere Erkennung gewährleistet und dabei durch nur acht zusammenhängende helle Strecken gebildet wird. Damit war auch das Kriterium 2) erfüllt. Das Kriterium 3) hängt eng mit der Darstellung des Schachbretts zusammen und wird daher im nächsten Abschnitt mitbehandelt.

3.3 Darstellung des Schachbretts

Für die Güte der Brettdarstellung wurden folgende Kriterien aufgestellt:

1) Das Brett soll in der Sicht sowohl des weißen als auch des schwarzen Spielers darstellbar sein. Man muß dabei erkennen können, um welche Sicht es sich handelt.

2) Die typische Schachbrettstruktur mit deutlich erkennbaren Schrägen (Diagonalen und Nebendiagonalen) muß gegeben sein.

Das Kriterium 1) konnte leicht erfüllt werden. Die Verteilung der weißen und schwarzen Felder ist für beide Ansichten gleich, sodaß nur die Brettbeschriftung variabel zu gestalten war. Die Brettbeschriftung befindet sich immer am linken und am unteren Rand. Für weiße Ansicht steigen die Buchstaben und Ziffern nach oben und rechts, für schwarze Ansicht fallen sie nach oben und rechts.

Größere Schwierigkeiten bereitete das Kriterium 2), da es auch eng mit der Kennzeichnung der weißen Figuren zusammenhängt. Die Bedeutung des Kriteriums liegt in der Tatsache, daß das Spielfeld erst durch die Einfärbung in weiße und schwarze Felder asymmetrisch wird und daß durch diese charakteristische Färbung erst die Schrägen verdeutlicht werden. Printer-Ausgaben müssen auf eine Unterscheidung weißer und schwarzer Felder verzichten. Dort sieht man deutlich, daß dem Auge die rasche Verfolgung einer Diagonale ohne die optische Hilfe der Färbung sehr erschwert wird. Außerdem sind die Läufer farbgebunden, sodaß man ohne die Färbung nachrechnen muß, welche Felder ein Läufer betreten darf. Diese Eigenschaft der Läufer ist auch dafür verantwortlich, daß manche Schachprobleme sozusagen davon leben, daß die Figuren in einer bestimmten Position und einer bestimmten Farbverteilung stehen. Die optische Hilfe gekennzeichneter weißer Felder zu verweigern, hätte also den Wert der symbolischen Figurendarstellung zumindest stark verringert. Der erste Gedanke zur Lösung ist wieder das Schraffieren weißer Felder. Die Lösung zeigt aber Nachteile: Erstens wird durch die Schraffierung die Zahl der Einzelstrecken erhöht, zweitens wären die Schraffen verschieden hell geworden, je nach ihrer Länge. Schließlich hätte die Deutlichkeit der Figuren gelitten, besonders wenn Figurenränder parallel zur Schraffierung liegen. Es wurde daher nach einer anderen Lösung gesucht. Es wurde von einem karierten Brettraster ausgegangen, und in diesem wurden Experimente mit Umrahmungen der weißen Felder im Inneren vorgenommen. Die einfache Umrahmung der weißen Felder im karierten Raster des Bretts erwies sich letztlich als der beste Kompromiß zwischen den nicht realisierbaren Schraffen und dem leeren Raster. Diese Lösung bietet dem Auge die erforderlichen optischen Anhaltspunkte. Die Erfahrungen des Autors zeigen, daß nach kurzer Zeit das Brett und die Figuren dem Auge so vertraut sind, daß man wie auf dem echten Brett Schrägen leicht entlang laufen kann. Daß man sich an diese Darstellung erst gewöhnen muß, ist kein starkes Gegenargument, da eine Gewöhnung auch bei realen Spielsteinen (z.B. Taschenschach, Reise-Magnetschach u.ä.) manchmal erforderlich ist.

4. DIALOGPROGRAMM

Das Dialogprogramm schien zunächst der einfachere Teil der Aufgabe zu sein: mit einfacher Eingabe muß die dem Zug entsprechende Änderung der Matrix BRETT hervorgerufen werden. Die Überprüfung der Gangarten der Figuren ist erstens eine umfangreiche logische Aufgabe, zweitens gehört sie naturgemäß zu einem Schachspielprogramm und nicht zu einem Ausgabeprogramm und drittens kann die Fehlererkennung gerade bei der optischen Anzeige den Spielern überlassen bleiben. Trotzdem wurde ein geringes Maß an Plausibilitätsprüfungen der Züge eingeführt, und zwar: ob der Zug mit einer

eigenen Figur erfolgt, ob entweder eine gegnerische Figur geschlagen wird oder auf ein leeres Feld gezogen wird.
Während der optischen Experimente wurde festgestellt, daß Züge wie Rochade, Bauernverwandlungen und Schlagen en passant im ersten Konzept gar nicht berücksichtigt waren. Um bei diesen Zügen leicht passierende Irrtümer zu erkennen, mußte für sie ein Teil der Schachregeln geprüft werden.

5. ARCHITEKTUR: BESCHREIBUNG FÜR DEN BENÜTZER

In diesem Abschnitt werden Dialogprogramm und Ausgabeprogramm als Einheit betrachtet.

5.1 Ausgabe am Bildschirm

Der Bildschirm des Tektronix ist ein liegendes Rechteck der Abmessungen 18cm x 13cm. Das Schachbrett wird links oben in dieses Rechteck gesetzt in den Abmessungen 10.5cm x 10.5cm, 0.5 cm davon nimmt die Beschriftung ein am linken und am unteren Rand. Unten sind die Spalten mit Buchstaben A bis H beschriftet, links sind die Zeilen mit 1 bis 8 beschriftet. Entsprechende Befehle bewirken, daß das Brett aus der Sicht des weißen oder des schwarzen Spielers gezeigt wird. Dies ist an der Beschriftung zu erkennen: Für weiße Ansicht steigen die Buchstaben und Ziffern nach rechts und oben, für schwarze Ansicht fallen sie nach rechts und oben.

Die weißen Felder des Schachbretts sind nicht wie üblich als helle Flächen gekennzeichnet, sondern jedes weiße Feld ist zur Kennzeichnung im Inneren durch ein weißes Rechteck als Umrahmung dargestellt.
Das rechte untere Eckfeld ist entsprechend den Schachregeln ein weißes Feld, die übrigen ergeben sich daraus von selbst. Außer dem Schachbrett selbst gibt es einen Dialogbereich auf dem Bildschirm. Dieser entspricht im Aufbau dem üblichen Terminalgebrauch, der durch den Gerätezustand Alphamode definiert wird. In diesem Mode ist der Bildschirm in zwei Textspalten geteilt zu denken. Zeilensprung und Spaltensprung werden dabei von der Hardware gesteuert. Der Dialogbereich beginnt in dieser speziellen Anwendung in der ersten Zeile unter dem Schachbrett, in der zunächst die Eingabeaufforderung "WEISS AM ZUG" oder "SCHWARZ AM ZUG" erscheint. Sobald durch Dialog mit dem Programm die Zeilen unter dem Schachbrett gefüllt sind, erfolgt der Spaltenwechsel und der Dialog wird in der rechten Hälfte des Bildschirms oben fortgesetzt. Da das Schachbrett selbst in diese Hälfte hineinreicht, wird es dadurch teilweise überschrieben. Dann kann durch den Befehl REPEAT der Dialogbereich geleert werden.

5.2 Dialogablauf

Nach jedem Bildwechsel (und nach dem Laden des Programmes) meldet sich das Programm mit der Eingabeaufforderung: "WEISS AM ZUG" oder "SCHWARZ AM ZUG". Das Gerät ist nun zur Eingabe eines Zuges oder eines Befehls bereit. Auf fehlerhafte Eingabe reagiert das Programm mit einer Fehlermeldung, und die Zeile kann in korrigierter Form sofort wieder eingegeben werden.

Das Programm akzeptiert sieben Befehle:
(Nur die unterstrichenen Buchstaben werden interpretiert):

BACK	Zurücknehmen von Zügen
MW	Zeige Spielbrett in der Ansicht für Weiß
MS	Zeige Spielbrett in der Ansicht für Schwarz
MB	Zeige Spielbrett in der Ansicht für den Ziehenden
NEWGAME	Beginne ein neues Spiel
ENDE	Ende der Session
REPEAT	Wiederhole die Ausgabe der Stellung

Schachzüge können in einer der folgenden, der algebraischen Notation angepaßten Formen eingegeben werden:

	Zugart	Ziehend	Schlagend
1.	Figurenzug	fax-by	fax:by
2.	Bauernzug	ax-by	ax:by
3.	Verwandlungszug	Bax-by(v)	Bax:by(v)
4.	Verwandlungszug	ax-by(v)	ax:by(v)
5.	En passant schlagen		Bax:byEP
6.	En passant schlagen		ax:byEP
7.	Rochade kurz	0-0	
8.	Rochade lang	0-0-0	

Dabei bedeuten:

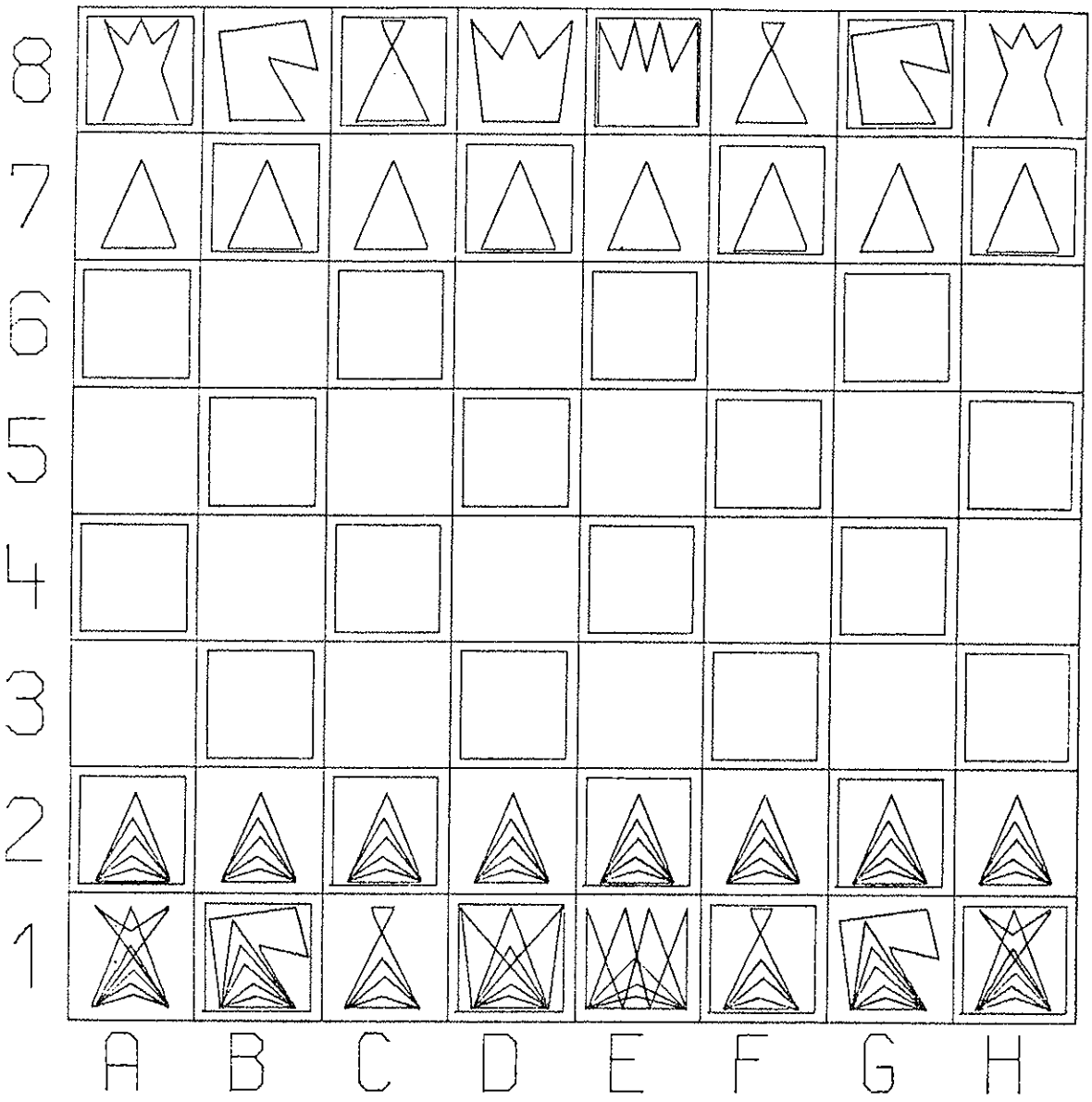
f	ein Figurensymbol B, S, L, T, D oder K
ax	Position des Startfeldes
by	Position des Zielfeldes
-	Figur zieht
:	Figur schlägt
(v)	Bauer wird in Figur v verwandelt
EP	Bauer schlägt en passant

Wie die Zeilen 2,4 und 6 zeigen, ist es erlaubt, Bauernzüge ohne Figurensymbol "B" einzugeben. Die in Schachpublikationen manchmal verwendeten weiteren Kurznotationen sind nicht erlaubt.
Die Gangarten der Figuren werden (mit wenigen Ausnahmen) nicht geprüft, d.h. man kann mit jeder Figur überall hinziehen. Da weder das Ausgabeprogramm noch das Dialogprogramm Schach spielen sollten, wurde die Überprüfung der Gangart der Figuren nicht gefordert. Weiters wurde darauf verzichtet, mit einer beliebigen Stellung (nach dem n-ten Zug) anfangen zu können. Die Eingabe beliebiger Stellungen hätte umfangreiche Prüfungen und sogar ein gewisses Maß an retrospektiver Analyse erfordert und damit den Rahmen einer Praktikumsarbeit gesprengt.

Die Betonung der Aufgabenstellung lag vor allem auf den optisch-physiologischen Problemen der Brettdarstellung und weniger auf den dialogtechnischen Fragen. Der Dialog könnte also noch mehr Erweiterungen erfahren, während die Brettdarstellung kaum Wünsche offen läßt.

Summary

Two programs have been written to enable symbolic output of a chessboard on the TEKTRONIX GRAPHIC DISPLAY 4006. A subroutine enables output of the playingboard in a symbolic form, similar to those in chesspublications. Several physiological and psychological aspects had to be observed. The representation also had to agree with the technical requirements of the terminal station.
The second program enables a little dialogue to play on (not with) the machine.



Grundstellung des Schachbretts

Dieses Bild wurde durch Ausgabe des Bildschirminhaltes auf dem Plotter erzeugt. Es handelt sich deshalb um ein Negativ, das nur einen ungefähren Eindruck vermitteln kann, da außerdem die Strichbreite am Plotter viel geringer ist.

Redaktion: Elisabeth Wittek

Eigentümer, Herausgeber, Verleger: EDV-Zentrum der Technischen Universität Wien, Abteilung Hybridrechenanlage, Vervielfältigung: Österreichische Hochschülerschaft Technik, für den Inhalt verantwortlich: Dipl.Ing. Dr.techn. W. Kleinert, alle: Gußhausstraße 27-29, 1040 Wien. Telex: 76875 rzthw a

INTERFACE Juni 1980