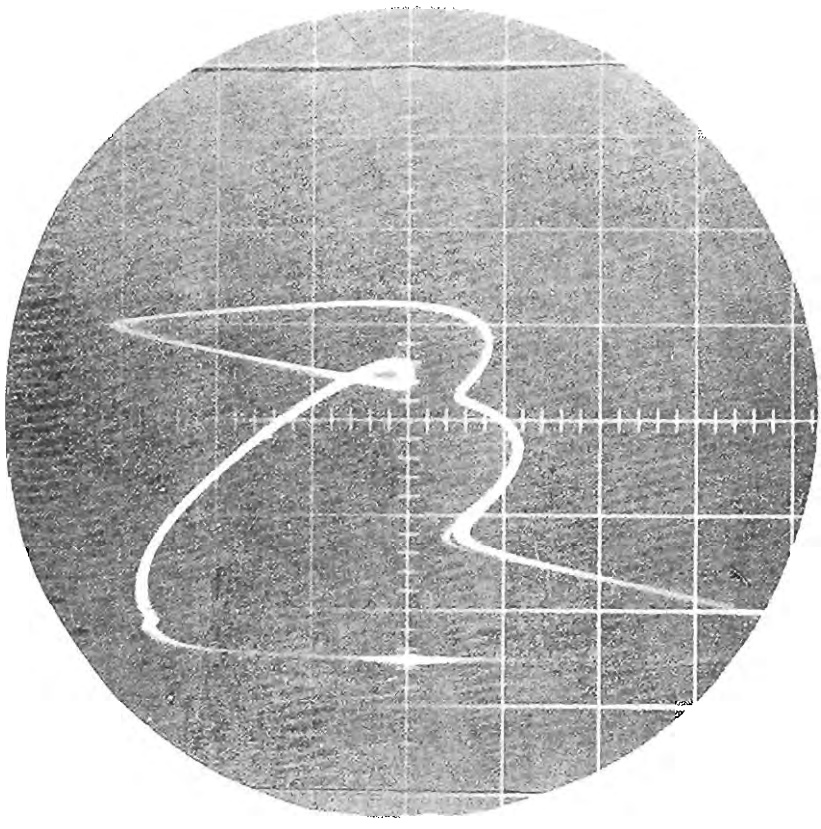
Interface

herausgegeben von
der Abt. Hybridrechenanlage
des Rechenzentrums
der Technischen Hochschule Wien

Seminar über den Einsatz von Hybridrechnern

26. - 28. 5. 1975

Nummer 3
Mai 1975



Phasenbild aus Sprachsignal und Ableitung eines "O"
(aus dem Seminarbeitrag von A. Csandtner und R. Gudonawitschius)

INHALTSVERZEICHNIS

Redaktion: Irmgard Husinsky

Eigentümer, Herausgeber, Verleger: Rechenzentrum der Technischen Hochschule Wien, Abteilung Hybridrechenanlage, Vervielfältigung: Österreichische Hochschülerschaft Technik, für den Inhalt verantwortlich: Dipl.Ing. W. Kleinert, alle: Gußhausstraße 27-29 1040 Wien.

Diese Nummer des Interface erscheint zu dem Zeitpunkt, da die Technische Hochschule in Wien ein

SEMINAR ÜBER DEN EINSATZ VON HYBRIDRECHENSYSTEMEN
IN WISSENSCHAFT, FORSCHUNG UND TECHNIK

veranstaltet, an dem mehr als 70 Wissenschaftler aus Belgien, der BRD, der DDR, Großbritannien, Yugoslawien, der Schweiz und Österreich teilnehmen.
Wir haben daher diesmal dem Abdruck von Kurzfassungen der auf diesem Seminar gehaltenen Referate, soweit sie rechtzeitig eingegangen waren, relativ breiten Raum gewidmet.

D.E. Gilles
Institut für Systemdynamik und Regelungstechnik
Universität Stuttgart

MODELLBILDUNG CHEMISCH-VERFAHRENSTECHNISCHER SYSTEME

Das Ziel des Beitrages besteht zunächst darin, aus einer übergeordneten Sicht die Methodik der Modellbildung für chemisch-verfahrenstechnische Prozesse zu diskutieren. Ausgangspunkt der Betrachtungen sind die physikalisch-chemischen Grundgesetze, aus denen allgemeine Bilanzgleichungen für die Einzelmassen der an der chemischen Umsetzung beteiligten Stoffe, für Energie und Impuls abgeleitet werden. Ihre Formulierung erfolgt möglichst einfach und übersichtlich.

Nach diesen allgemeinen Betrachtungen wenden wir uns der Modellbildung für den katalytischen Festbettreaktor zu. Es handelt sich dabei um den technisch wichtigsten Typ kontinuierlich betriebener chemischer Reaktoren. Dieses heterogene Reaktionssystem mit örtlich verteilten Parametern stellt insbesondere im Falle stark exothermer chemischer Reaktionen ein Beispiel für einen Prozeß dar, der sich nur mit großen Schwierigkeiten mathematisch beschreiben läßt. An Hand von Simulationsergebnissen wird das Verhalten dieses Reaktortyps diskutiert. So wird gezeigt, daß bei stark exothermen Reaktionen mehrfache stationäre Zustände, Hystereseerscheinungen beim Zünden und Verlöschen der Reaktion und wandernde Brennzonen auftreten können.
Die in einem Film festgehaltenen Simulationsergebnisse zeigen auch, daß im Kreisprozeß betriebene Reaktoren leicht zur Instabilität neigen und daß eine rasche Katalysatoralterung zu erheblichen Übertemperaturen führen kann. Dies stimmt mit entsprechenden experimentellen Untersuchungen überein.

W. Claes
EAI Brüssel
European Computation Center

THE HYBRID MACHINES

(Konzept - Struktur - Vorteile - Trends)

Konzept

Moderne Hybridrechner haben ihren Ursprung in der Kombination zweier einander ergänzender Techniken:

- parallele Verarbeitung von Daten, wird normalerweise von Analogrechnern durchgeführt
- serielle Verarbeitung von Daten, entspricht den allseits bekannten Digitalrechnern

Im parallelen Subsystem werden Daten durch Spannungen realisiert, während Bitmuster, zusammengefaßt zu digitalen Worten, Daten in der seriell arbeitenden Maschine darstellen.

Der Vorteil des hybriden Rechnens besteht in der Kombination dieser beiden Techniken, die mit Hilfe von ausführlicher Software zu einem sehr effektiven Instrument für den Einsatz in Technik, Industrie, Forschung und Lehre wird.

Struktur

Die Struktur eines Hybridrechners ist folgendermaßen zu beschreiben: die parallelen und seriellen Subsysteme benötigen eine spezielle Verbindung, die ein effektives Zusammenwirken gewährleistet. Dementsprechend ergibt sich die Notwendigkeit zweier Hauptfunktionen zur Verbindung der beiden Rechner: Daten sollen in beiden Richtungen mit hoher Geschwindigkeit umgewandelt und übertragen werden und eine spezielle Kontrolleinheit soll es erlauben, daß der Digitalrechner den Analogrechner als ein peripheres Gerät benützt und daß dadurch die Notwendigkeit manueller Operationen eliminiert wird.

Die hybride Software, die einzigartig für Systeme dieser Art ist, ermöglicht eine leichte Programmierung der Maschine. Es werden spezielle Softwarepakete angeboten, die für mathematische Modelle geeignet sind und die Gleichungen für eine entsprechende Behandlung an dem System vorbereiten. Es werden Files gebildet, die die Verbindung der Elemente am Steckbrett beschreiben und eine automatische Initialisierung und Überprüfung des Modells am Computer erlauben.

Bequeme periphere Geräte wie Plotter, Videos und graphische Displays ermöglichen es dem Benützer, sein Problem zu überblicken, wobei ihm alle Möglichkeiten zum Zusammenwirken des Modells und der durchgeführten Experimente zur Verfügung stehen.

Vorteile

Der Hybridrechner erweist sich als vorteilhaft für Forschung und Entwicklung, sowie im Bereich der Lehre oder zur Einschulung in das Manövrieren von Schiffen und Flugzeugen, für industrielle Anlagen etc. (Simulation). Generell, wann immer Real-Time Anwendungen auftreten, d.h. wenn ein Mensch mit dem Modell zusammenwirken soll, wird sich der Hybridrechner als überlegen erweisen.

Das System bleibt während des Experiments benützerbezogen und die parallelen Strukturen des analogen Subsystems erweisen sich als nützlich für eine Erhöhung der Geschwindigkeit der gesamten Simulation und für die Gewährleistung von Stabilität.

Trends

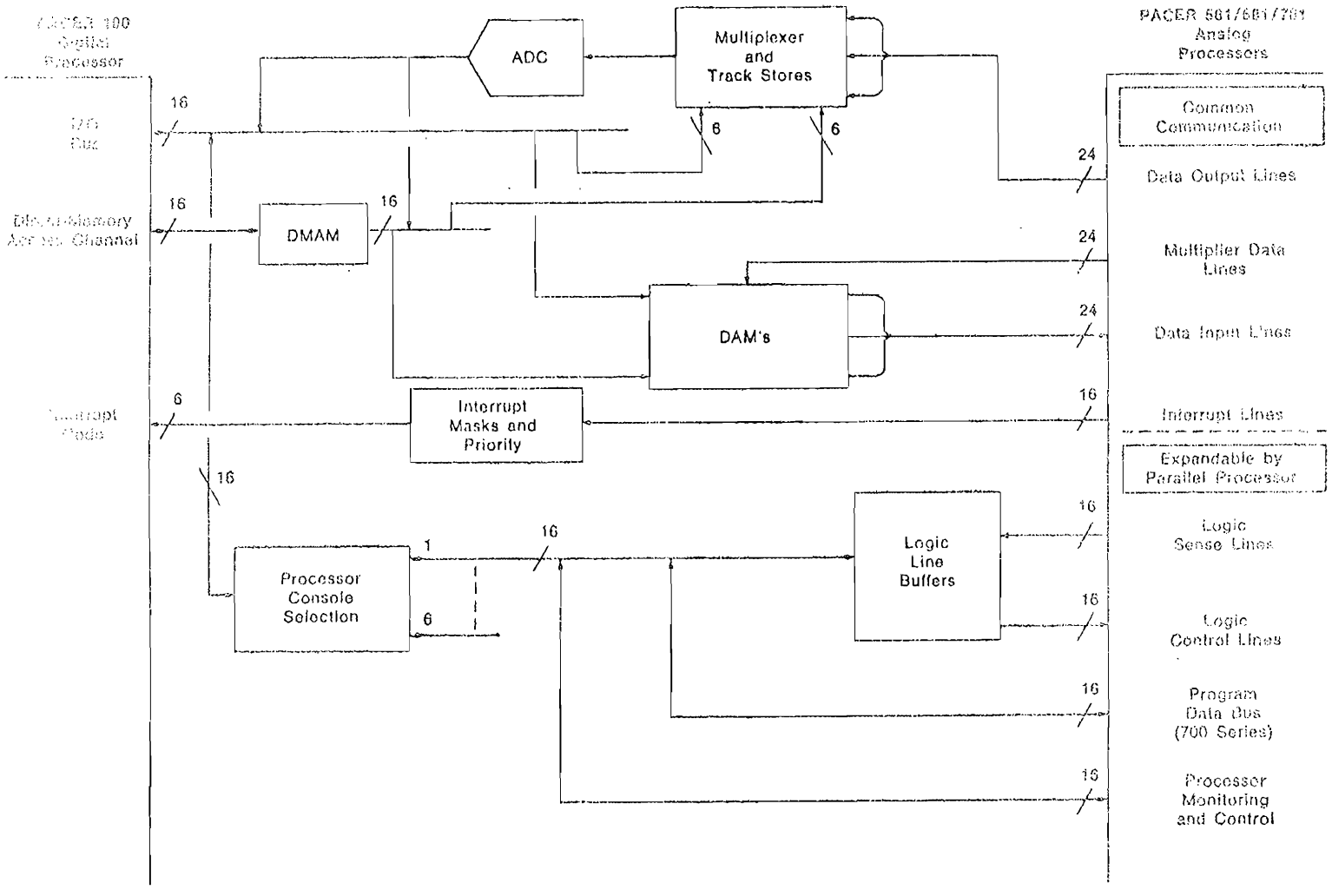
Wie jede in schneller Entwicklung begriffene Technik haben auch die hybriden Computer eine beschleunigte Entwicklung erfahren und sind in Weiterentwicklung.

Das letzte Ziel ist, dem Benützer eine universale Simulationssprache zur Verfügung zu stellen und eine effektive Software, die sich um alle praktischen Details kümmert und dem Anwender ermöglicht, sich nur auf die Betreuung des Modells und seines Verhaltens zu konzentrieren.

Diese Black-box-Methode setzt Neuerungen wie z.B. automatisches Stecken, Fernanschlüsse zum Computer, spezielle interaktive periphere Geräte, einen effektiven hybriden Compiler etc. voraus.

EAI arbeitet an diesen neuen Techniken und an einer optimalen Lösung jeder Situation. Markante Fortschritte sind innerhalb der nächsten 5 Jahre zu erwarten, es wird sich dabei um Weiterentwicklungen auf der Basis des jetzigen Standes handeln.

INTERFACE COMMUNICATION



I. Troch
Institut für Technische Mathematik
TH Wien

EINSATZGEBIETE DES HYBRIDRECHNERS

Hybridsysteme können vor allem in jenen Fällen mit Vorteil eingesetzt werden, in denen auch relativ schnell ablaufende Vorgänge in Echtzeit (oder sogar schneller) simuliert werden müssen. Dies ist stets dann der Fall, wenn z.B. der Mensch selbst Bestandteil des Systems - und damit auch der Simulation - ist, oder aber wenn der Rechner als "Prozeßrechner" eingesetzt wird. Der erste Fall tritt besonders häufig in der Luft- und Raumfahrt auf (z.B. beim Training der Piloten im Hinblick auf die Bewältigung kritischer Situationen), die neben Mechanik, Elektro- und Energietechnik (Reaktortechnik) eines der wichtigsten Einsatzgebiete zunächst für Analog- und dann für Hybridrechner darstellte. Diese Entwicklung erklärt wohl auch die Tatsache, daß es heute in erster Linie die mit diesen Aufgabengebieten befaßten Zweige der Industrie sind, die Hybridsysteme in ihren Entwicklungsabteilungen, und damit kommerziell, einsetzen.

Einige Beispiel mögen dies schlagwortartig illustrieren: Regelungen für maschinelle Anlagen aller Art (Drehzahlregelungen, Werkzeugmaschinen, Wickelmaschinen,...), Strahlführungssysteme für Teilchenbeschleuniger, Regelung von Öfen in Stahl- und Walzwerken, Kühlung von Reaktoren, schwingungstilgende Aufstellung von Motoren und Turbinen, Anpassung von Lastkraftwagen an spezielle Nutzlasten (keine großen Schwingungen während der Fahrt), Simulation von Autounfällen statt crash-tests (Kostenersparnis!),...

Zu diesen klassischen Einsatzgebieten von Hybridrechnern treten in letzter Zeit vor allem auch Probleme aus der Chemie, insbesonders der chemischen Verfahrenstechnik, der Biomedizin und Pharmakologie, sowie der ökologischen und soziologischen Systeme.

Einige Beispiele mögen dies wieder erläutern: Einfluß der Wasserentnahme (etwa durch Brunnen) auf den Grundwasserspiegel, Dimensionierung von Dammbauten an Nebenflüssen im Gebirge zur Verhinderung von Hochwasser (und Überschwemmungen) des Hauptstromes bei starken Niederschlägen, Gewinnung von Trinkwasser aus Stauseen, Auswirkung von Fabrikabwässern, bzw. Kühlwasser auf ein Flußsystem, Auffinden der günstigsten Art der Schädlingsbekämpfung im Obst- oder Getreidebau, Untersuchung bekannter "Weltmodelle" (Forrester) und Ermittlung ihrer Parameter, sowie Entwicklung verbesserter Modelle, Entwicklung und Untersuchung von Wirtschaftsmodellen z.B. zur Erleichterung wirtschaftspolitischer Entscheidungen, die Untersuchung chemischer Vorgänge in Kernreaktoren, Destillationskolonnen, chemische Vorgänge der Biomedizin (z.B. bei der Atmung), Überwachung und Steuerung der Herztätigkeit (Schrittmacherentwicklung), Modelle für wichtige Funktionen des tierischen und menschlichen Körpers (Blutkreislauf, Stoffwechsel,...) zur Erleichterung der Diagnose und Behandlung, Entwicklung von Modellen für die pharmako-kinetischen Vorgänge in Tier und Mensch, sowie für den Ablauf von Arm- oder Beinbewegungen (Prothesenherstellung), Untersuchung des Tag/Nacht-Rhythmus von Lebewesen und dessen Umweltabhängigkeit (z.B. Vorsorgemedizin), Zellteilungsvorgänge,...

Naturgemäß kann die gegebene Aufzählung nicht vollständig die Möglichkeiten eines zweckmäßigen Einsatzes von Hybridsystemen beschreiben, sie will nur beispielhaft aufzeigen, daß derartige Rechner dann mit Vorteil eingesetzt werden können, wenn etwa dynamisch ablaufende Vorgänge neu modelliert oder in der Konzeption vorliegende technische Anlagen einer konkreten Problemstellung angepaßt werden sollen. Beides sind i.a. mit einer Optimierung verbundene Aufgaben, die ein wiederholtes Durchführen der Simulation erfordern, und zwar für eine sehr große Zahl von Parameterwerten, sodaß hier - ebenso wie bei Mensch-Maschine-Systemen - die große Geschwindigkeit beim Berechnen von zeitabhängigen Vorgängen voll zum Tragen kommt.

A. Gsandtner
TH Wien
R. Gudonawitschius
Politechnisches Institut Kaunas, UdSSR

DIE BESTIMMUNG DER MERKMALE FÜR VOKALERKENNUNG

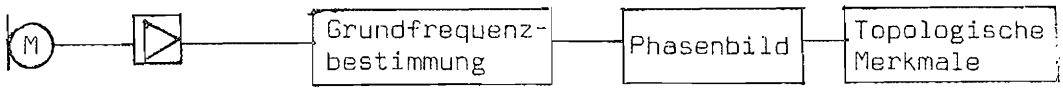
Ein Teilproblem der Analyse der menschlichen Sprache

1. Einleitung

Die automatische Spracherkennung ist eines der schwierigsten Probleme der technischen Kybernetik. Von der Anwendung her scheint sie aber gleichzeitig zu den vordringlichsten Aufgaben in der Datenverarbeitung zu gehören. Eine Lösung dieses Problems, ja selbst eine teilweise Implementierung würde eine weitgehende Rationalisierung etwa der Dateneingabe und Steuerung von Prozessoren ermöglichen.

Die Aufgabe der automatischen Sprachanalyse besteht darin, die Information des Sprachsignals (Abb. 1), das sind bei Berücksichtigung aller Frequenzen bis zu 200 000 Bit/sec, auf die für die Notierung des gesprochenen Textes notwendigen max. 200 Bit/sec zu reduzieren.

In dem hier beschriebenen Verfahren wird die Reduktion der Vokalsignale auf die topologischen Eigenschaften des Phasenbildes des Sprachsignals über seine Ableitung nach der Zeit vorgenommen.Vokalsignale haben eine quasi - periodische Form, das heißt, daß - mit Ausnahme der Störungen - in einem Bereich des Signals (stationärer Bereich) ein periodisches Signal vorliegt. Aus diesem Grund wird hier nur eine Periode des Vokals verarbeitet. In früheren Arbeiten wurde das Phasenbild über einen größeren Bereich (Wort) unabhängig von den Perioden analysiert. Durch die Summierung der Merkmale mehrerer Laute ergaben sich sehr komplexe Ergebnisse, die nur die Analyse eines sehr beschränkten Eingabevokabulars ermöglichte. Das grundfrequenzsynchrone Verfahren hingegen ist unabhängig von Eingabebeschränkungen und daher allgemein verwendbar. Allerdings wird das Sprachsignal nur in die Lautschrift abgebildet.



Blockdiagramm der Vokalanalyse

2. Das Sprachsignal in der Ebene

Im verwendeten Verfahren wird das Sprachsignal über seiner Ableitung aufgetragen (TRAJEKTORIE , Abb. 2). Als Koordinatenfunktionen können auch andere Signale verwendet werden, etwa das Zeitintegral oder höhere Ableitungen.

Um die Eigenschaften der Trajektorie erfassen zu können, sind nicht nur die geometrischen Eigenschaften, sondern auch die Zeitverteilung oder die Geschwindigkeit des laufenden Punktes von Bedeutung.Man steht vor der Aufgabe, eine kontinuierliche Kurve in dirskrete Parameter überzuführen. Zu diesem Zweck wird die Ebene in 2N gleich große Sektoren unterteilt .

Für jeden Sektor werden 2 Parameter bestimmt:
1. die Anzahl der Schnittpunkte mit dem zugehörigen Strahl
2. die Dauer, in der sich die Trajektorie im Sektor befindet.

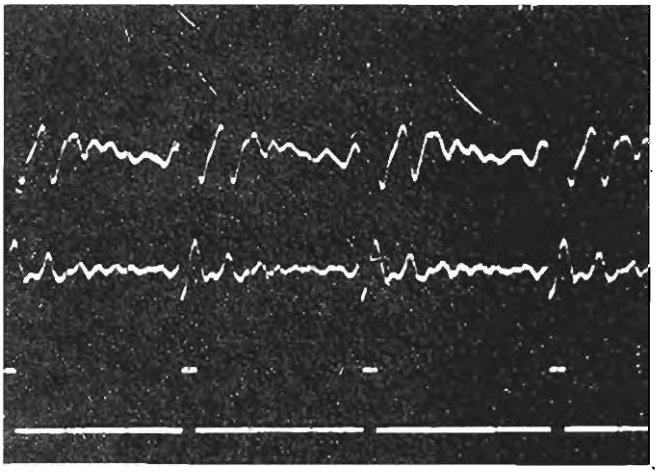


Abb. 1: Sprachsignal, Ableitung und Periodenbeginn für "a"

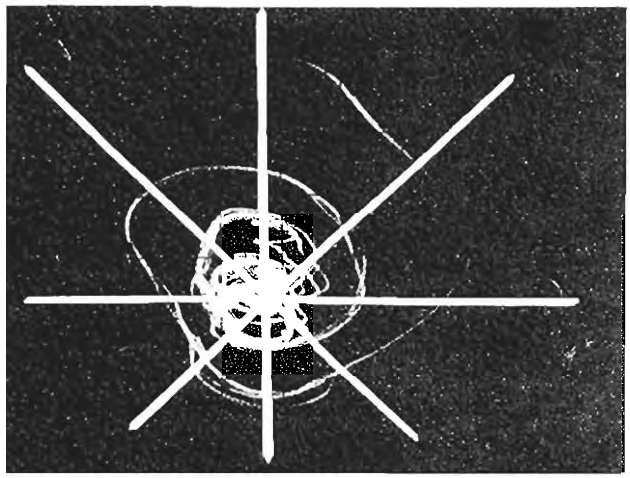


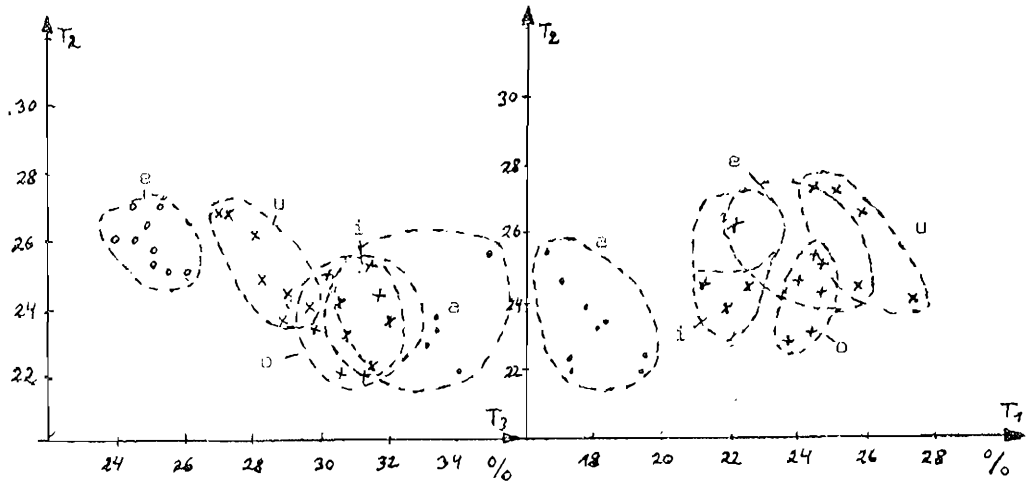
Abb. 2: Trajektorie und 8 Sektoren für "a"

3. Die Messung der Parameter

Abb. 2 zeigt die Unterteilung der Phasenebene durch 4 Strahlen in 8 Sektoren. Die Bestimmung der sich daraus ergebenden 16 Parameter erfolgte rein analog. Für die Kreuzungsparameter wird der Umstand der Kreuzung festgestellt (Nullstellen der Koordinatenfunktionen und deren Summen und Differenzen) und die Kreuzungsimpulse werden verlängert (Hysterese) und aufintegriert.

Für die Bestimmung der Zeitparameter werden die logischen Signale "Trajektorie befindet sich im i-ten Sektor" vom Integrierer Nr. i aufintegriert. So liegen alle 16 Parameter auf parallelen Ausgängen an und werden am Ende einer durch manualle Steuerung ausgewählten Periode digitalisiert und gelistet.

Abb. 3 zeigt die Zeitverteilung bei der Aufteilung in 4 Sektoren (Quadranten), wobei T_i die Zeit bedeutet, in der sich die Trajektorie im i-ten Sektor befindet (in Prozenten). Mehrere Sprecher und verschiedene Lautstärken ergeben schon hier klar unterscheidbare Bereiche, speziell wenn man beide Diagramme zusammen als dreidimensionalen Raum auffaßt.



4. Weiterentwicklung des Verfahrens

Für die zielführende Anwendung des Verfahrens muß die optimale Anzahl der Strahlen festgestellt werden. Weiters ist zu untersuchen, ob nicht andere Koordinatenfunktionen eine bessere Unterscheidung ermöglichen. Diese Untersuchungen setzen eine breite Testserie voraus. Um die empirischen Daten in akzeptabler Zeit ermitteln zu können, muß ein bequem zu handhabendes System geschaffen werden, das die Bewertung der Verfahrensvarianten selbsttätig vornimmt.

Schon jetzt kann man sagen, daß die Trajektorien den Weg zur Lösung des Vokalproblems zeigen. Notwendig dafür ist allerdings die vorhergehende Zeitsegmentation des Sprachsignals in Phoneme, ein Problem, das hier überhaupt nicht behandelt wurde.

W. Claes
EAI Brüssel
European Computation Center

ÜBERBLICK ÜBER DIE ERFAHRUNG DES BRÜSSELER RECHENZENTRUMS IN BIOMEDIZINISCHEN PROBLEMEN

In den letzten Jahren wurde auf dem Gebiet der Pharmakologie und Biomedizin ein erheblicher Fortschritt in der Analyse, Erfassung und Lösung von Problemen mit Hilfe des Computers gemacht.

Dabei müssen zuerst alle Schwierigkeiten überwunden werden, die dadurch entstehen, die auftretenden Phänomene durch mathematische Gleichungen zu beschreiben (Modellbildung). Diese Vorgangsweise ist relativ neu, aber sie verbreitet sich rasch in den Forschungszentren, die sich mit höheren biomedizinischen Studien befassen, hauptsächlich im Hochschulbereich.

Die Modellbildung bringt tatsächlich viele Vorteile, das Verständnis und die Durchführung der Experimente zu erleichtern. Ein Hybridrechner kann den Modellen durch die Simulation Leben geben und erlaubt alle Arten von dynamischer Darstellung, was einen raschen Fortschritt bei vielen schwierigen Aufgaben bringt.

Das europäische Rechenzentrum von EAI in Brüssel wurde mit vielfachen Anwendungen dieser Art in den letzten Jahren konfrontiert, die im folgenden kurz angeführt sind:

1. Vektorkardiographie

Diese Technik stellt Herzschläge als dreidimensionale Bewegung eines elektrischen Vektors dar. Elektroden werden am Körper des Patienten angebracht und ermöglichen die Aufnahme von Signalen, die registriert und analysiert werden sollen, nachdem das System, das für jede Person verschieden ist, entsprechend angeglichen wurde. Körperachsen sollen für jeden Patienten gebildet werden. Das ist eine sehr spezielle Technik, die von Prof. Rijlant, Universität Brüssel, mit analogen und hybriden Geräten seit mehr als 10 Jahren durchgeführt wird.

2. Lungensimulation / erzwungene Ausatmung

Diese Studie wurde im Rechenzentrum für die Leuven Universität in Belgien durchgeführt. Sie bestand aus einem möglichst einfachen Modell der Bronchien und der Luftröhre, sodaß erzwungenes Ausatmen simuliert werden konnte. Speziell die Luftröhre wird auf ihre elastischen Eigenschaften untersucht, die zu einem vollständigen oder teilweisen Zusammenbruch der Röhre unter gewissen Fluß/Druck-Bedingungen führen können. Das Phänomen des Hustens an sich kann als eine Erweiterung der beim Ausatmen auftretenden Bedingungen analysiert werden. Dieses Simulationsprojekt rechtfertigt das Modell für weitere Experimente, die auch jetzt noch durchgeführt werden.

3. Hörfehlererkennung: ein klinischer Apparat

Hybride Simulation hat dem Wissenschaftler erlaubt, eine Regelkreismethode zu entwickeln, die darauf ausgerichtet ist, mit Hilfe von Elektroden die elektrischen Signale zu registrieren, die durch den Hörapparat in das Gehirn geschickt werden. In einem schalldichten Raum werden geeichte Signale erzeugt und dem Patienten übertragen. Die Reaktion des Ohres wird gemessen und unter Berücksichtigung von

qualitativen und quantitativen Kriterien analysiert. Diese Vorgangsweise erlaubt eine frühe Entdeckung von Hörfehlern bei Neugeborenen, lange bevor eine abnormale Entwicklung ihrer Fähigkeiten das Problem aufscheinen läßt, was nur dann auftritt, wenn ein Teil des normalen Lernprozesses bereits vorbei ist. Eine frühe Korrektur ermöglicht eine normale Entwicklung der menschlichen Fähigkeiten.

Die hybride Technik wurde implementiert, um den Eichungsprozeß zu simulieren und eine Spektralnanlyse der niederenergetischen Signale des Meßgerätes zu analysieren. Der daraus resultierende klinische Spezialcomputer könnte wohl ein weit verbreitetes Instrument für Kliniken und Spitäler werden.

4. Modellidentifikation in pharmazeutisch-kinetischen Anwendungen

Man beginnt meist mit der empirischen Untersuchung der Ausbreitungsgeschwindigkeit eines injizierten Präparates in verschiedene zusammenhängende Körperteile. Man vergleicht die Reaktion eines Körpermodells, dem diese Injektion eingegeben wird, mit echten Daten, die an Tieren gemessen wurden. Es wird eine iterative Optimierungsaufgabe gestartet, wobei unbekannte Parameter, die die Wechselwirkung zwischen den verschiedenen Körperteilen beschreiben, variiert werden. Diese Suche nach einem Optimum wird mit Hilfe eines speziellen Softwarepaketes durchgeführt, das dem Benützer eine Reihe von Algorithmen liefert, die es ihm ermöglichen, sein Modell zu steuern, zu verbessern und die unbekannten Parameter zu identifizieren.

Dieses Problem ist sehr typisch für Hybridrechner, da dieselbe Technik auf verschiedenen Gebieten angewendet werden kann. Die Methode ist daher gut bekannt, leicht zu implementieren und ziemlich flexibel.

5. Kapnographische Analyse

(CO_2-Gehalt der ausgeatmeten Luft)

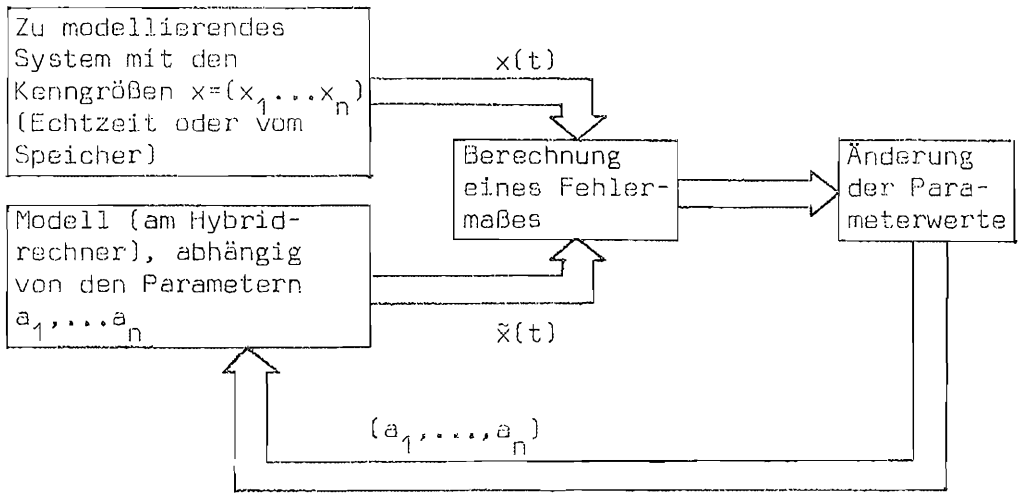
Kinderärzte legen ein Schwergewicht auf die frühe Erkennung von chronischen Atmungsschäden. Kapnogramme registrieren den CO_2-Gehalt der ausgeatmeten Luft; ihre Form wird durch einige Parameter charakerisiert, deren dynamische Messung sehr empfindliche Informationen z.B. über asthmatische Patienten liefert. Experimente, die in Brüssel in Zusammenarbeit mit der Universität Budapest durchgeführt wurden, zeigten, daß ein Hybridprogramm erfolgreich kapnographische Funktionen analysieren und deren signifikante Parameter identifizieren kann.

Dies sind nur einige der neuesten Probleme, die im Brüsseler Rechenzentrum gelöst wurden. Sie sind jedoch beispielhaft für das große Interesse an einer wirksamen Forschungsmethode, die sich rasch verbreitet und geeignete Ergebnisse liefert dank der Fähigkeit von Hybridrechnern, mathematische Modelle lebendig zu machen.

I. Troch
F. Breitenecker
H. Schäfhauser
Arbeitsbereich für Regelungstheorie und Hybridrechentechnik
Institut für Technische Mathematik
TH Wien

ANALYSE- UND SYNTHESEPROBLEME DER REGELUNGSTHEORIE ALS ANWENDUNG DER MATHEMATISCHEN OPTIMIERUNG

Es wurde bereits aufgezeigt, daß gerade der Einsatz eines hybriden Rechnersystems sehr häufig bei der Lösung einer Optimierungsaufgabe für Systeme mit dynamischem Verhalten erfolgt. Dies kann entweder eine Aufgabe der Parameteroptimierung sein, d.h. das Modell, der Regler,... liegt seinem Aufbau nach fest, jedoch sind noch eine gewisse Anzahl von Konstanten den konkret vorliegenden Bedingungen möglichst gut anzupassen, oder aber eine Aufgabe der Strukturoptimierung, d.h der funktionelle Zusammenhang zwischen den einzelnen Systemgrößen ist unbekannt. Die Vorgangsweise zur Lösung des erstgenannten Problems kann im Fall einer Modellbildung (Identifikation) etwa folgendermaßen veranschaulicht werden:



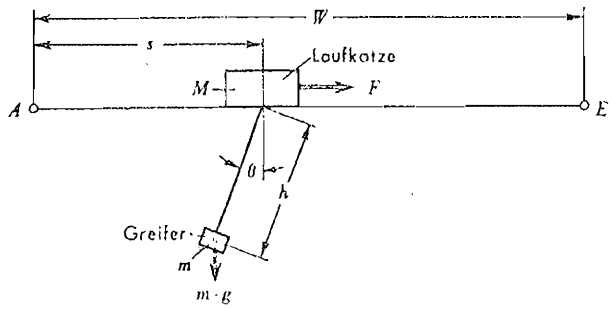
Zur Lösung derartiger Aufgabenstellungen sind eine Reihe von Verfahren entwickelt worden, wie etwa die verschiedenen Gradientenmethoden (Methode des stärksten Anstiegs), zyklische Parametervariation oder Zufallssuche. Methoden, die stets ein relatives Extremum der Gütefunktion liefern, aber nicht sicherstellen, daß dieses auch das absolute Extremum ist, sodaß sie meist mit einer - groben - Abtastung des Parameterbereiches zur Lokalisierung des Extremums kombiniert werden müssen. Abtastmethoden sind meist wegen des hohen Rechenaufwandes (kann jeder der k Parameter N Werte annehmen, so sind N^k Simulationen erforderlich!) nicht durchführbar. Zur Lösung von Aufgaben der Strukturoptimierung sind folgende für Hybridrechner besonders geeignet erscheinende Verfahren entwickelt worden:

Parametrisierung der unbekannten Funktionen: Entwicklung nach den Funktionen eines geeignet erscheinenden Funktionensystems, bzw. die Approximation durch stückweise konstante Funktionen, wodurch eine Rückführung auf die

Aufgabe der Parameteroptimierung - allerdings i.a. mit einer sehr großen Zahl unbekannter Parameter - erfolgt.

Maximumprinzip von Pontrjagin: seine Anwendung führt nach Auswertung der Maximumsbedingung auf ein Zweipunktrandwertproblem, dessen Ordnung doppelt so groß wie die des ursprünglich gegebenen Systems ist, und das i.a. noch mit einer Minimierungsaufgabe verbunden ist.

Die Anwendung des Maximumprinzips - als der für Hybridrechner wohl bedeutungsvollsten Methode wird am Beispiel eines Erzentladers erläutert, der zeitoptimal von A nach E gesteuert werden soll, wobei insbesondere darauf zu achten ist, daß der Greifer in Punkt E in Ruhe ist:



Modell der Anlage. Es bedeuten: s Entfernung der Katze vom Anfangspunkt der Bewegung; B Abstand der Randpunkte der Katzenbewegung; h Pendellänge des Greifers; F Antriebskraft der Katze; M, m Katzen-, Greifermasse; g Erdbeschleunigung.

Die Demonstration am Rechner zeigt einmal die Durchführung dieser Optimierung und ermöglicht auch die Übernahme der Steuerung durch den Menschen. Bezüglich weiterer Einzelheiten sowohl zur Methodik als auch zum konkreten Problem muß auf die Literatur verwiesen werden. Man vergleiche auch die in dieser Auswahl zitierte Literatur:

/1/ M.D. Anderson, S.C. Gupta, Analog computer solution of a third-order Pontryagin optimum control system. Simulation (ed.: McLeod), McGraw-Hill, NewYork, 1968

/2/ M. Athans, P.L. Falb, Optimal Control. McGraw-Hill, NewYork, 1966

/3/ G.A. Bekey, Hybrid Computation. J. Wiley, NewYork, 1968 (dt.:Kohlhammer,Stuttgart, 1971)

/4/ A. Durling, Computational Techniques. Intext, NewYork, 1974

/5/ M. Feilmeier, Hybridrechnen. Birkhäuser, Basel, 1974

/6/ P. Hippe, Zeitoptimale Steuerung eines Erzentladers. Regelungstechnik 18 (1970), 346 - 350

/7/ L.S. Pontrjagin, et al., Mathematische Theorie optimaler Prozesse. Oldenburg, München 1967

/8/ R. Vichnevetsky, J.P. Waha, Dynamic Programming and Hybrid Computation. Ann. AICA (1963) und Simulation (ed.: McLeod), McGraw-Hill, New York, 1968

ÖFFNUNGSZEITEN AN DER HYBRIDRECHENANLAGE:

Montag bis Freitag 8 - 20 Uhr

Jeden Montag von 8 bis 14 Uhr sowie ganztägig jeden ersten Montag im Monat finden Systemtests statt. Zu diesen Zeiten ist kein Benützerbetrieb, Closed-Shop-Eingabe wird jedoch gerechnet.

UMSTELLUNG AUF 29-CODE:

Ab sofort werden an der Hybridrechenanlage Jobs, die im 29-Code gelocht sind, vorrangig behandelt. Bis auf weiteres besteht noch die Möglichkeit, Jobs im 26-Code im Closed-Shop-Betrieb in der dafür vorgesehenen Lade einzugeben. Die beiden Codes können an der Hybridrechenanlage nicht in einem Job gemischt werden.

Es besteht die Möglichkeit, bereits im 26-Code existierende Programme auf 29-Code umstanzen zu lassen. Dies ist auf der Jobkarte anzugeben. Die Turn-around-Zeit beträgt dafür etwa 48 Stunden.

JOBS MIT GROSSEM KERNSPEICHERBEDARF:

Im Closed-Shop-Betrieb werden Jobs, die mehr als 12 k Kernspeicherplatz benötigen, nur einmal pro Tag gerechnet, da sie den Multiprogrammingbetrieb behindern.
Es besteht die Möglichkeit, solche Programme in Teilen hintereinander zu exekutieren (siehe Interface 2).
Der für ein Programm benötigte Kernspeicherplatz ist auf der Core-Load-Map des Core Image Generators (JCSCIG) unter CORE: oktal angegeben. Für den Standard-Multiprogrammingbetrieb darf dieser Wert nicht größer als 27377 sein (entspricht 12 k für Benützerprogramm). FREE: gibt den Speicherplatz an, der noch im Kernspeicher frei ist, wenn das Programm geladen ist. Falls dieser Wert negativ ist, benötigt das Programm mehr als 12 k. Damit der Core Image Generator ein solches Programm exekutabel binden kann, muß vom Operator eine entsprechende Option gesetzt werden. Dazu sollte auf der Jobkarte der Kernspeicherbedarf oktal in einer kurzen Notiz angegeben werden. Z.B. CORE 37777 für max. 16 k Speicherbedarf.

KURSE

Folgende Kurse werden im kommenden Wintersemester 1975/76 an der Hybridrechenanlage abgehalten werden:

RH1 GERÄTETECHNIK EAI PACER 600

Zentraleinheit, Interruptsystem, I/O System (inklusive Interface und Analogrechner)
Dauer: 1 Woche
Termin: 1975-09-22 bis 1975-09-26
Vortragender: Anton Gsandtner
Die genauen Kurszeiten und der Kursort werden noch rechtzeitig im Schaukasten der Hybridrechenanlage bekanntgegeben.

RH2 BENÜTZUNG DES BETRIEBSSYSTEMS JCS/MP

Optimale Ausnützung der Möglichkeiten des Systems, Systemstruktur
Dauer: 1 Woche
Termin: 1975-11-10 bis 1975-11-14
Vortragender: Dipl.Ing. A.Blauensteiner
Die genauen Kurszeiten und der Kursort werden rechtzeitig im Schaukasten der Hybridrechenanlage bekanntgegeben.

RH3 BESONDERHEITEN IN FORTRAN AN DER HYBRIDRECHENANLAGE

1 Vortrag
Termin: 1975-10-10
Vortragender: Fritz Blöser
Die genaue Zeit und der Ort werden rechtzeitig im Schaukasten der Hybridrechenanlage bekanntgegeben.

RH4 HYBRIDE FORTRAN-PROGRAMMIERUNG MIT ÜBUNGEN

Einführung in das Hybrid Linkage Softwarepaket
Dauer: 1 Woche
Termin: 1975-10-13 bis 1975-10-17
Vortragender: Helmut Hummer
Die genauen Kurszeiten und der Kursort werden rechtzeitig im Schaukasten der Hybridrechenanlage bekanntgegeben.

RH6 EAI ASSEMBLER MIT ÜBUNGEN

Dauer: 1 Woche

Termin: 1975-10-20 bis 1975-10-24
Vortragender: Dipl.Ing. A.Blauensteiner
Die genauen Kurszeiten und der Kursort werden rechtzeitig im Schaukasten der Hybridrechenanlage bekanntgegeben.

RH8 TERMINALPROGRAMMIERUNG

Benützung des CURSOR-Softwarepaketes für Video Input/Output, Edition von Sourceprogrammen, FORTRAN READ/WRITE
1 Vortrag
Termin: 1975-10-29
Vortragender: Dipl.Ing. A.Blauensteiner
Die genaue Zeit und der Kursort werden rechtzeitig im Schaukasten der Hybridrechenanlage bekanntgegeben.

Die Kurse werden aufbauend gestaltet, d.h. es wird von Fragen allgemeinen Interesses zu speziellen Problemen fortgeschritten, sodaß ein Kursbesucher, der nur einige allgemeine Informationen benötigt, auch bei vorzeitiger Beendigung des Kurses die für ihn notwendigen Informationen erhält.
Nähere Auskünfte und Anmeldungen zu den Kursen telefonisch oder persönlich bei Herrn Schandl, Rechenraum Zimmer 1404/05, Tel. 65-37-85/803 Durchwahl.

Vom Institut für Technische Mathematik wird folgender Kurs abgehalten:

AH3
AH4 HYBRIDE SPRACHEN UND VERFAHREN MIT ÜBUNGEN

Grundprinzipien des Analog- und Hybridrechners, kontinuierliche Simulationssprache, benützerorientierte hybride Programmiersprache für Dialogbetrieb (HOI).
Lösung charakteristischer Anwendungsbeispiele mit Hilfe von HOI.
Dauer: 1 Woche
Termin: 1975-06-30 bis 1975-07-04
Die Festlegung der Kurszeiten erfolgt am 30. Juni 1975 um 9 Uhr im Seminarraum der mathematischen Institute (Zimmer 1415) im Einvernehmen mit den Teilnehmern.
Auskünfte und Anmeldungen bei Dipl.Ing. Frank Rattay (Klappe 899), Institut für Technische Mathematik, Gußhausstraße 27-29, 1040 Wien

DAS BETRIEBSSYSTEM JCS/MP

ENTWICKLUNG

An der Hybridrechenanlage wurde vor zwei Jahren mit der Entwicklung eines Multiprogramming Job Control Systems (JCS/MP) begonnen. Die Gründe für die Entwicklung eines eigenen Betriebssystems lagen hauptsächlich darin, daß von der Herstellerfirma EAI kein eigentliches Betriebssystem angeboten wurde, die angebotene Software sich nur auf Standardkonfiguration bezog und außerdem nur minimal die gegebenen Hardwaremöglichkeiten ausnützte. Auftretende Fehler konnten praktisch nicht behoben werden und erforderten einen erheblichen Testaufwand. Das später von EAI angebotene RTOS (Real Time Operating System) stellt zwar eine wesentliche Erweiterung der Möglichkeiten dar, ist aber eher schwerfällig im Multiprogramming und entspricht nicht ganz den Vorstellungen eines Real Time Processing. In JCS/MP wird ein hybrider Real-Time-Betrieb gewährleistet (Prioritäten), sowie der PACER 1ØØ als Digitalrechner mittlerer Größe als solcher verwendet.

JETZIGER STAND DES BETRIEBSSYSTEMS

Seit November 1974 ist die Version 6 von JCS/MP installiert. JCS/MP benötigt als Minimalkonfiguration nur 4 k Kernspeicher, eine Fixed-Head Disk und eine Systemkonsole, ist aber durch ein vollkommen allgemeines und variables Input/Output Konzept auf Interruptbasis mit In-core Buffer von den tatsächlichen Ein- und Ausgabeeinheiten unabhängig. So können in JCS/MP z.B. an der Hybridrechenanlage 7 Ausgabeeinheiten und 5 Eingabeeinheiten gleichzeitig arbeiten und logisch miteinander vertauscht werden. Dadurch belasten I/O Operationen auch bei einem Real-Time Job das System nur unmerklich. Datentransfers mit Massenspeicher (Fixed Head Disk) werden in Warteschlangen angereiht und nacheinander parallel zum Rechenbetrieb abgearbeitet.
JCS/MP zeichnet sich durch ein einfaches Operating aus, das keine besondere Einschulung erfordert. In JCS/MP ist ein vollständiges Job Accounting implementiert. Jobs können im Stapelbetrieb verarbeitet werden. Jobs mit zu großem Kernspeicherbedarf können segmentiert werden und es besteht die Möglichkeit, Jobs während ihrer Exekution auf die Platte zu swappen und zu einem beliebigen späteren Zeitpunkt die Exekution fortzusetzen.

Ferner existiert Softwareunterstützung, die ein bequemes Disk File Handling, sowie die Verwendung von virtuellen Feldern auf der Platte ermöglicht.
Durch umfangreiche Run-Time Informationen und bequeme Systemutilities kann der Status des Betriebssystems jederzeit überblickt werden.

MULTIPROGRAMMING

Das Multiprogramming in JCS/MP ermöglicht einen dynamischen Multi-Access Betrieb von zwei Jobs, wobei vom Benützer Prioritäten gesetzt werden können. Dabei kann einer der Jobs eine Echtzeit-simulation am Hybridrechner sein, wobei in diesem Fall die höchste Priorität gesichert ist. Durch ein Scratch/Spooling System können beide zugleich laufenden Jobs sowohl den Card Reader als auch den Line Printer verwenden. Das anfallende Datenmaterial wird vom System geordnet.

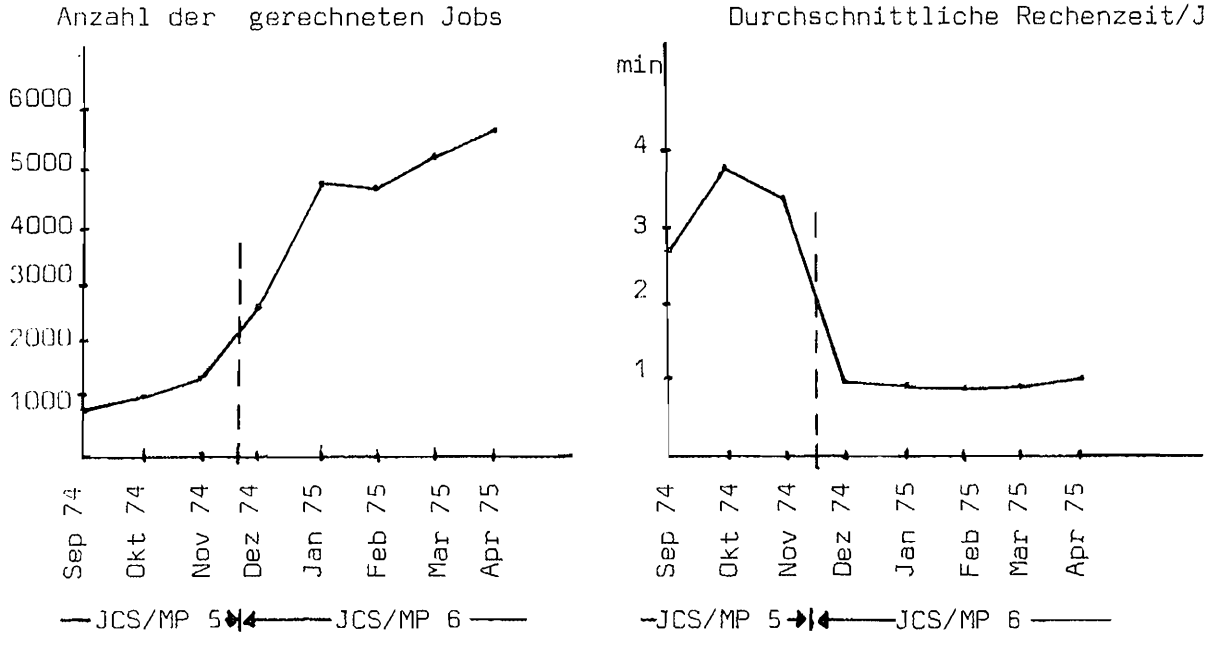
Durch umfangreiche Softwareunterstützung kann das Betriebssytem JCS/MP allen gestellten Anforderungen von Benützern der Hybridrechen-anlage leicht gerecht werden und gibt auch dem Systemprogrammierer alle Möglichkeiten, im Rahmen des Systems Programme zu erstellen, die in JCS/MP eingreifen, bzw. mit JCS/MP arbeiten.

Die Hybridrechenanlage wird sehr intensiv im Lehr- und Ausbildungs-betrieb eingesetzt; das bedeutet gerade bei Hybridanwendungen eine große Anzahl von Vorbereitungsarbeiten wie Ausbessern von syntaktischen und logischen Fehlern im Programm. Das Betriebssystem JCS/MP bietet nun den enormen Vorteil, daß das Austesten der analogen Schaltung unter Kontrolle des Hybridprogramms gleichzeitig mit der Erstellung einer neuen Version dieses Programms am Digitalrechner geschehen kann.

DURCHSATZSTEIGERUNG SEIT DER INSTALLATION VON JCS/MP

Durch das Input/Output Konzept auf Interruptbasis erhöht sich allein der Durchsatz während Outputoperationen um das 1ØØ bis 1ØØØfache.
Im April 1975 wurden in jeder verrechneten CPU-Sekunde 197.13 Input/Output Operationen durchgeführt.

Bei Datentransfers mit der Fixed Head Disk konnten Durchsatzsteigerungen bis zu 93% erreicht werden. Für eingabeintensive Programme (z.B. HOI) ergibt sich im Multiprogrammingbetrieb eine Durchsatzsteigerung von 95%, da die wechselweise Exekution zweier Jobs aufgrund von Kriterien erfolgt, die sowohl vom System (Timer) als auch von der Exekutionsnachfrage der Jobs bestimmt werden. Das bedeutet, daß z.B. bei einem Job, der auf Eingabe wartet, keine Notwendigkeit besteht, ihn aktuell zu exekutieren. Bei günstiger Jobeinteilung im Multiprogrammingbetrieb kann man daher eine optimale Auslastung der Hardware erreichen. Insgesamt wurde an der Hybridrechenanlage der Durchsatz allein durch den Übergang von JCS/MP Version 5 auf JCS/MP Version 6 vervierfacht.



WEITERENTWICKLUNG

Analysen zeigen, daß trotz allem noch CPU-Kapazität frei ist, um weitere Hintergrundapplikationen im Betriebssystem durchzuführen. Geplant für JCS/MP Version 7 ist vor allem ein variables Massenspeicherkonzept, sowie zusätzlich die Simulation eines 200 UT zur Rechenanlage CDC CYBER 74.

A. BLAUENSTEINER

IMPLEMENTIERUNG DER DIGITALEN SIMULATIONSSPRACHE CSMP AN DER HYBRIDRECHENANLAGE

CSMP ist die Abkürzung für "Continuous Systems Modelling Program", ein neu an der Abteilung Hybridrechenanlage installiertes Programm aus der Gruppe der digitalen Simulationssprachen.
Simulation bedeutet allgemein die Ersetzung eines Systems durch ein Modellsystem, hier mathematischer Art. Diese für die Behandlung durch Rechner geeigneten "Rechenmodelle" bestehen aus Systemen gekoppelter Differentialgleichungen und können durch Strukturbilder beschrieben werden.
Darin werden sogenannte Blöcke definiert, black boxes, denen eine mehr oder weniger komplizierte Funktion zugeordnet wird. Das Schaltbild der Verknüpfungen dieser Blöcke wird je nach dem vorher definierten Inhalt dieser Elemente das tatsächliche Modell gröber oder feiner beschreiben können. Der Zusammenhang zwischen diesen Strukturbildern und Analog-Schaltbildern ist also offensichtlich. Letztere sind spezielle Strukturbilder, da sie sich auf die am Analogrechner verfügbaren Blockarten beschränken müssen, die durch technische Eigenschaften des Gerätes vorgegeben sind (z.B. Vorzeichenumkehr bei allen Verstärkern des Analogrechners).
Prinzipiell können zur Lösung von Simulationssystemen sowohl Analog- als auch Digitalrechner verwendet werden, die Arbeitsweise des Digitalrechners bedingt jedoch Zeitdiskretisierung und numerische Integration. Diese serielle Arbeitsweise der Digitalrechner bewirkt, daß die Lösungszeiten bei der digitalen Simulation im Gegensatz zur analog/hybriden mit dem Modellumfang nichtlinear wachsen. Dies kann bei Echtzeitsimulation (die Rechnungen müssen mit externen Prozessen synchron ablaufen) zu großen Problemen für einen reinen Digitalrechner führen. Bei gegebenem Problemumfang resultiert daher eine obere Schranke für die höherfrequenten Bestandteile der Systemvariablen. Diese maximale digitale Bandbreite ist also wesentlich geringer als die analog/hybride.

WOZU DIENT NUN AN EINER HYBRIDRECHENALAGE, DIE BEREITS ÜBER DIE BESTEN VORAUSSETZUNGEN ZUR BEHANDLUNG VON ECHTZEITPROBLEMEN VERFÜGT, EIN REIN DIGITALES SIMULATIONSPROGRAMM ?

Die Aufbereitung vieler hybrider Probleme ist umfangreich und hängt ganz wesentlich von den Größenordnungen ab, in denen sich die Lösungsvariablen bewegen.
Der Analogrechner ist an eine maximale maschineninterne Spannung gebunden, und daher sind alle Problemvariablen zuerst einer Transformation auf dimensionslose Rechnervariablen zu unterziehen. Ist das Lösungsverhalten unbekannt, können große Skalierungsprobleme auftreten.
Um diese bei Neuaufbauten oder Änderungen der Schaltung besonders in der Entwicklung von Modellen häufig auftretenden Fälle leichter bewältigen zu können, kann man das grundlegende Arbeitsmodell und später folgende größere Änderungen zuerst digital austesten (bei Gleitkomma gibt es natürlich keine Skalierungsprobleme!). Man kann so einen Überblick über das grundsätzliche Verhalten des Modells bei verschiedenen Anfangswerten und Parametern erhalten.
Jede Fehlersuche im später aufgebauten Analogrechner-Programm wird dadurch außerdem beschleunigt, da aus den bereits ungefähr bekannten Ergebnissen Rückschlüsse auf falsche Steckverbindungen oder defekte Elemente gezogen werden können.
Bei einem noch völlig unbekannten Modellverhalten ist diese Suche sehr viel schwieriger. Die digitalen Tests können wie alle anderen rein digitalen Programme im Batch-Betrieb laufen und beanspruchen daher auch nicht die Kapazität des gesamten Hybridsystems. Die vom digitalen Simulationsprogramm nicht nachgebildeten höchstfrequenten Anteile der Lösung sind in diesem Anfangsstadium bei der Festlegung der Skalierung in den wenigsten Fällen von Interesse.

Die Verwendung digitaler Simulationssprachen erfordert meist die Installation aufwendiger Compiler und viel Kernspeicherplatz. Ein Beispiel hiefür ist CSMP III, eine sehr leistungsfähige Sprache von IBM, welche als Minimum immerhin 120 K Kernspeicherworte benötigt. Das an der Hybridrechenanlage verwendete EAL-CSMP von wesentlich einfacherer Struktur wurde von EAL speziell für kleinere Rechner entwickelt und benötigt etwa 20 K.

Komplizierte Simulationssprachen dieser Art beschreiben oft nur die Modellstruktur, während der Simulationsablauf in einer algorithmischen Sprache wie ALGOL oder FORTRAN geschrieben sein muß. CSMP III zum Beispiel stellt eine Untermenge von FORTRAN dar.

Im Gegensatz zu diesen Sprachen, in denen die Definition des Modells in leicht überschaubaren algebraischen Anweisungen erfolgt, verlangt das an der Hybridrechenanlage laufende und auf minimalen Kernspeicherplatz reduzierte CSMP diese Modelldefinitionen auf Datenkarten in einem Code. Dieser beschreibt ein dem Analogrechner-Blockschaltbild sehr ähnliches CSMP-Blockdiagramm des Problems.

Für Benützer, die mit diesen Schaltbildern des Analogrechners bereits vertraut sind, bedeutet dies eine leichte Übertragbarkeit ihrer Schaltung auf die CSMP-Datenkarten, es wird nichts prizipiell Neues gefordert.

Andere Benützer benötigen allerdings gewisse Einarbeitungszeiten, um sich daran zu gewöhnen, ihre Problem-Differentialgleichungen nicht mehr in algebraischer Form darzustellen, sondern als Blockstrukturen eines geschlossenen Regelkreises.

CSMP liegt einstweilen in der Form eines Hauptprogrammes vor, das selbständig eine beliebig lange Folge von Kontroll- und Datenkarten verarbeitet.

Die Kontrollkarten enthalten Angaben über das Integrationsintervall, Genauigkeitsschranken und andere Steuerparameter. Eine bereits definierte Schaltung kann über einen dieser Parameter z.B. sehr leicht mit geringfügigen Veränderungen nochmals gerechnet werden, ohne daß man die ganze Schaltung neu definieren müßte.

BEISPIEL

Als Beispiel für die Kontroll- und Datenkartenfolge eines CSMP-Datendecks sei hier die Eingabe für die Simulation der Flugbahn eines geworfenen Körpers im Schwerefeld angegeben, ein nur wegen seiner Kürze gewähltes sehr einfaches Beispiel.

Die Bewegung des Körpers wird durch die Differentialgleichungen

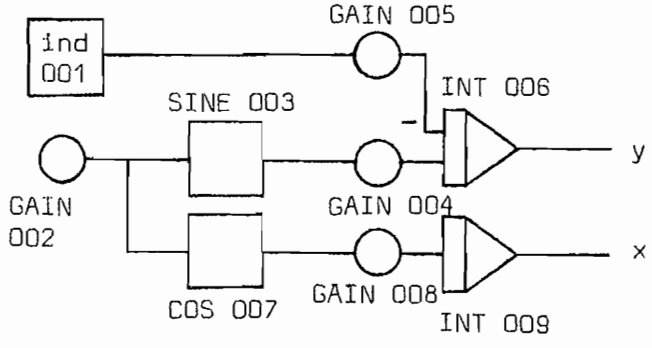
$$\frac{dx}{dt} = v_o \cos \theta_o \qquad x(0) = 0$$

$$\frac{dy}{dt} = v_o \sin \theta_o - gt \qquad y(0) = 0$$

beschrieben.

x ... horizontale Koordinate
y ... vertikale Koordinate
v_o ... Anfangsgeschwindigkeit (m/sec)
θ_o ... Wurfwinkel (auf die Horizontale bezogen)
g_o ... Erdbeschleunigung (m/sec^2)

Das Blockdiagramm hat folgende Form:



Lineprinterausdruck:

EAI PACER 100 TR WIEN CSMP - RUN NO. 1

TRAJECTORY OF A BODY

ELEMENT TYPE	BLOCK NO.	INPUT A	INPUT B	INPUT C	I/C,CONSTANT,GAIN	COMMENTS
IC	1	0	0	0	.0000E+00	
GAIN	5	1	0	0	.3200E+02	
COS	2	0	0	0	.5236E+00	
SINE	3	2	0	0	.0000E+00	
GAIN	4	3	0	0	.1000E+03	
INT	6	4	-5	0	.0000E+00	
COS	7	2	0	0	.0000E+00	
GAIN	8	7	0	0	.1000E+03	
INT	9	8	0	0	.0000E+00	

IND. VAR.	BLOCK 6	BLOCK 9	BLOCK 2	BLOCK 7
.000E+00	.0000E+00	.0000E+00	.5236E+00	.8660E+00
.500E+00	.2140E+02	.4334E+02	.5236E+00	.8660E+00
.100E+01	.3400E+02	.8660E+02	.5236E+00	.8660E+00
.150E+01	.3900E+02	.1299E+03	.5236E+00	.8660E+00
.200E+01	.3980E+02	.1732E+03	.5236E+00	.8660E+00
.250E+01	.2500E+02	.2165E+03	.5236E+00	.8660E+00
.300E+01	.0340E+01	.2598E+03	.5236E+00	.8660E+00
.350E+01	-.2089E+02	.3031E+03	.5236E+00	.8660E+00
.400E+01	-.5929E+02	.3464E+03	.5236E+00	.8660E+00
.450E+01	-.9499E+02	.3897E+03	.5236E+00	.8660E+00
.500E+01	-.1499E+03	.4330E+03	.5236E+00	.8660E+00
.550E+01	-.2089E+03	.4763E+03	.5236E+00	.8660E+00
.600E+01	-.2759E+03	.5196E+03	.5236E+00	.8660E+00

EAI PACER 100 TR WIEN CSMP - RUN NO. 2

TRAJECTORY OF A BODY

ELEMENT TYPE	BLOCK NO.	INPUT A	INPUT B	INPUT C	I/C,CONSTANT,GAIN	COMMENTS
COS	2	0	0	0	.7854E+00	TRAJECT

IND. VAR.	BLOCK 6	BLOCK 9	BLOCK 2	BLOCK 7
.000E+00	.0000E+00	.0000E+00	.7854E+00	.7071E+00
.500E+00	.3135E+02	.6335E+02	.7854E+00	.7071E+00
.100E+01	.3471E+02	.1271E+03	.7854E+00	.7071E+00
.150E+01	.7506E+02	.1906E+03	.7854E+00	.7071E+00
.200E+01	.7742E+02	.1414E+03	.7854E+00	.7071E+00
.250E+01	.7677E+02	.1767E+03	.7854E+00	.7071E+00
.300E+01	.5813E+02	.2121E+03	.7854E+00	.7071E+00
.350E+01	.3140E+02	.2474E+03	.7854E+00	.7071E+00
.400E+01	.0340E+02	.2828E+03	.7854E+00	.7071E+00
.450E+01	-.3041E+01	.3181E+03	.7854E+00	.7071E+00
.500E+01	-.4044E+02	.3535E+03	.7854E+00	.7071E+00
.550E+01	-.9549E+02	.3889E+03	.7854E+00	.7071E+00
.600E+01	-.1517E+03	.4242E+03	.7854E+00	.7071E+00

EAI PACER 100 TR WIEN CSMP - RUN NO. 3

TRAJECTORY OF A BODY

ELEMENT TYPE	BLOCK NO.	INPUT A	INPUT B	INPUT C	I/C,CONSTANT,GAIN	COMMENTS
SIN	2	0	0	0	.1047E+01	TRAJECT

IND. VAR.	BLOCK 6	BLOCK 9	BLOCK 2	BLOCK 7
.000E+00	.0000E+00	.0000E+00	.1047E+01	.5001E+00
.500E+00	.3229E+02	.6534E+02	.1047E+01	.5001E+00
.100E+01	.7259E+02	.1301E+03	.1047E+01	.5001E+00
.150E+01	.9509E+02	.1952E+03	.1047E+01	.5001E+00
.200E+01	.1091E+03	.2603E+03	.1047E+01	.5001E+00
.250E+01	.1104E+03	.3254E+03	.1047E+01	.5001E+00
.300E+01	.1157E+03	.3905E+03	.1047E+01	.5001E+00
.350E+01	.1077E+03	.4556E+03	.1047E+01	.5001E+00
.400E+01	.9137E+02	.5207E+03	.1047E+01	.5001E+00
.450E+01	.6500E+02	.5858E+03	.1047E+01	.5001E+00
.500E+01	.3290E+02	.6509E+03	.1047E+01	.5001E+00
.550E+01	-.7749E+01	.7159E+03	.1047E+01	.5001E+00
.600E+01	-.5644E+02	.7810E+03	.1047E+01	.5001E+00

Die angegeben Codes für die Blöcke bedeuten:

IND ... das Blockelement erzeugt die unabhängige Problemvariable "t" (Zeit)
GAIN ... "Potentiometer" zur Einstellung multiplikativer Faktoren, diese können im Gegensatz zum Analogrechner Werte größer als 1 annehmen
CON ... Konstantengeber
SINE ... Sinus-Funktionsgeber
COS ... Cosinus-Funktionsgeber
INT ... Integrierer
LAST ... steht für das Ende der Datenkartenfolge eines Problems

Die Flugsimulation soll für drei verschiedene Wurfwinkel durchgeführt werden, wobei der Winkel vorher in das Bogenmaß zu übertragen ist.

30^{o} = 0.5236 rad
45^{o} = 0.7854 rad
60^{o} = 1.047 rad

Es müssen somit drei Runs mit jeweiliger Veränderung des Elements CON 002 (θ) durchgeführt werden. Die übrige Schaltung bleibt erhalten.
Die Beobachtungszeit wird von t=0 bis t=6 Sekunden gewählt, mit einem Protokollausdruck der Werte in jeder halben Sekunde.

Kontrollkarte 1	0.50E+00	0.60E+01	006009002007	03
Datenkarten:	IND 001			
	GAIN 005	+001		+0.3200E+02
	CON 002			+0.5236
	SINE 003	+002		
	GAIN 004	+003		+0.1000E+03
	INT 006	+004,-005		
	COS 007	+002		
	GAIN 008	+007		+0.1000E+03
	INT 009	+008		
	LAST			
Kontrollkarte 2	0.50E+00	0.60E+01	006009002007	
Datenkarte:	CON 002			+0.7854E+00
	LAST			
Kontrollkarte 3	0.50E+00	0.60E+01	006009002007	
Datenkarte:	CON 002			+0.1047E+01
	LAST			

Die Werte auf den Kontrollkarten:
1. Intervall für Ausgabe am Printer,
2. Totale Zeit,
3. auszudruckende Blöcke (vier Dreiergruppen),
4. Zahl der Runs.

Die Werte auf den Datenkarten:
1. Code für Blockelement
2. eindeutige Nummer
3. Nummern aller Blöcke, die Eingänge für das gerade spezifizierte Element liefern
4. numerischer Wert des Elementes, falls erforderlich

Eine kurzgefaßte Beschreibung aller für CSMP notwendigen Steuer- und Datenkarten sowie der jeweiligen Datenformate ist in Ausarbeitung und wird den Benützern in Kürze zur Verfügung stehen. Das Programm selbst kann ab sofort verwendet werden.

Bei eventuell auftretenden Fragen wenden Sie sich bitte an H. Hummer, Abt. Hybridrechenanlage, Gußhausstraße 27-29, Tel. 653785/901.

WEITERENTWICKLUNG

Die derzeitige Ausgabe der Ergebnisse in Tabellenform wird durch folgende Erweiterungen ergänzt werden:

- o die Tabelle kann auf Wunsch unterdrückt werden und
- o die errechneten Daten werden in diesem Fall auf Datenfiles der Systemplatten geschrieben, von wo sie der Benützer mit Hilfe einfacher Routinen beliebig verarbeiten kann
- o CSMP wird als Unterprogramm gefaßt werden, sodaß es in hybriden Programmen zur Parameterabschätzung und On-line Skalierung dienen kann. Datenausgabe wahlweise über Datenfiles der Systemplatte oder Parameterliste.
- o Erweiterung der Ausgabemöglichkeiten auf automatische Erzeugung von Kurven als Printout und Einbau eines Zufallszahlengenerators für Programme, die am Analogrechner den Rauschgenerator verwenden.

H. HUMMER

DAS RECHNEN MIT INTERRUPTUNTERPROGRAMMEN

Ein wesentlicher Unterschied zwischen rein digitaler und hybrider Programmierung liegt darin, daß bei hybriden Programmen neben Unterprogrammen, die vom Programm aufgerufen werden, auch Unterprogramme exekutiert werden können, falls der Analogrechner gewisse Impulse gibt. Diese sogenannten Interruptunterprogramme unterscheiden sich formal nicht von den anderen; der funktionelle Unterschied besteht aber darin, daß ein Interruptunterprogramm zu einem beliebigen Zeitpunkt aufgerufen werden kann und dabei das laufende Programm kurzzeitig unterbricht.

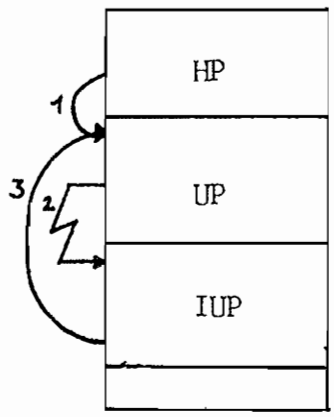
Verwendet nun das Interruptunterprogramm weitere Unterprogramme (Systemunterprogramme werden aufgerufen ohne daß der Benützer ein CALL programmiert), die das laufende Programm auch aufruft, so kommt es beim normalen Bindevorgang des Programmes zu folgender Schwierigkeit:

Beispiel:

```
/FOR HP
EXTERNAL IUP
CALL JOB (777,5)
CALL HYBRID
CALL INTER (3,IUP)
1 CALL UP
GOTO 1
END

/FOR UP
SUBROUTINE UP
RETURN
END

/FOR IUP
SUBROUTINE IUP
CALL UP
RETURN
END
```

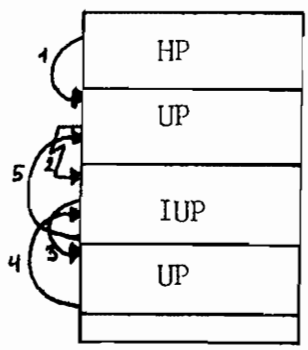



```
IN HP
LOAD UP
LOAD IUP
```

Sollte das Interruptunterprogramm IUP gerade die Exekution des Unterprogramms UP unterbrechen und selbst danach UP aufrufen, so wird durch den neuerlichen Aufruf von UP die Information des Rücksprunges nach dem rufenden Programm HP mit der Information des Rücksprunges nach dem rufenden Programm IUP überschrieben. Damit ist eine Rückkehr in das ursprüngliche Programm HP nach Beendigung der Exekution von IUP und danach von UP nicht mehr möglich. Vielmehr würde wieder nach IUP zurückgesprungen werden.

Dieser Schwierigkeit begegnet man beim Binden des Programmes mit der SPECIAL-Steuerkarte, die veranlaßt, daß das angegebene Interruptunterprogramm alle danach folgenden Files speziell dazuladet. Im allgemeinen wird nämlich ein benötigtes Unterprogramm nur einmal zum Programm gebunden, gleichgültig wie viele LOAD-Karten dasselbe Unterprogramm laden.

Im allgemeinen tritt diese Notwendigkeit zumindest für Systemunterprogramme auf, wobei entsprechende LOAD RTL-Karten einzufügen sind.




```
IN HP
LOAD UP
SPECIAL IUP
LOAD UP
```

Allgemeine Steuerkartenreihenfolge:

```
/BUILD MAINCI
IN PRO                  Hauptprogramm
LOAD FILE1
  .                     benötigte
  .                     Subroutinen
LOAD FILEX
LOAD RTL                Systemfile
SPECIAL INTERS          Interruptroutine
LOAD FILE1
  .                     benötigte
  .                     Subroutinen
LOAD FILEX
LOAD RTL                Systemfile
```

A. BLAUENSTEINER

ANREGUNGEN FÜR DEN FORTGESCHRITTENEN FORTRAN-BENÜTZER BEI VERWENDUNG DES EAI-COMPILERS

Nachdem bereits in früheren Artikeln auf die wesentlichsten Besonderheiten des EAI-FORTRAN eingegangen wurde (Interface Nr. 1 und 2), sollen nun einige Hinweise gegeben werden, die es dem FORTRAN-Programmierer gestatten, effektivere und schnellere Programme zu schreiben.

Besonders bei Schleifen kann oft viel Zeit eingespart werden. Ein eleganter Algorithmus muß nicht unbedingt optimal sein. Die Befehlsfolge

```
      DO 10 I=1,1000
   10 A(I)=I
```

bewirkt 1ØØØ Konvertierungen des Wertes I von INTEGER auf REAL. Das folgende Programmstück ist zwar länger, aber in der Ausführung schneller:

```
      B=0.
      DO 10 I=1,1000
      B=B+1.
   10 A(I)=B
```

Berechnungen mit Größen, die nicht mit der Laufvariablen indiziert sind, sollten nach Möglichkeit vor der Schleife durchgeführt werden. Man schreibt also statt

```
      DO 1 I=1,100
      A(I)=0.
      DO 1 J=1,100
    1 A(I)=A(I)+B*C*D(I,J)
```

besser

```
      X=B*C
      DO 1 I=1,100
      SUM=0.
      DO 2 J=1,100
    2 SUM=SUM+D(I,J)
    1 A(I)=SUM*X
```

Im ersten Fall wird nämlich B*C 1ØØØØmal und auch die Adresse von A(I) innerhalb der J-Schleife jedesmal neu berechnet.

Die Adressenberechnung wird bei einem mehrdimensionalen Feld wesentlich aufwendiger als bei einem eindimensionalen, sodaß man besser mehrere eindimensionale Felder verwenden bzw. nach Möglichkeit ein Feld "linearisieren" sollte, also statt

```
      DIMENSION D(10,50)
      DO 10 I=1,10
      DO 10 J=1,50
   10 D(I,J)=0.
```

besser

```
      DIMENSION D(10,50),DE(500)
      EQUIVALENCE (D(1,1),DE(1))
      DO 10 I=1,500
   10 DE(I)=0.
```

Interessant ist im Zusammenhang mit der Auswertung arithmetischer Ausdrücke die Überlegung, was zeitmäßig günstiger ist, A*A*...*A (N mal) oder A**N. Es zeigt sich, daß bis einschließlich N=6 die fortgesetzte Multiplikation schneller als die Exponentiation ist.

Es ist erlaubt, aus einer DO-Schleife zu einem anderen Programmteil herauszuspringen und danach von dort wieder in die Schleife zurückzuspringen. Dies kann gewisse Vorteile bringen, wenn eine Befehlsfolge nicht nur aus einer Schleife, sondern auch von anderen Stellen des Programms aus angesprungen wird. Da die Realisierung von Schleifen nicht mit eigenen Indexzellen vorgenommen wird, ist auch die folgende Befehlsfolge möglich:

```
      I=5
      GOTO 10
         .
         .
      DO 20 I=1,50
         .
   10 K=I
         .
   20 CONTINUE
```

Die Schleife wird für I=5 ab dem Statement 1Ø und für I=6(1)5Ø vollständig durchlaufen.

FORTRAN bietet dem Programmierer zwei Möglichkeiten für Abfragen:
die arithmetische Abfrage IF (A) N1,N2,N3 und
die logische Abfrage IF (B) S
Es stellt sich hier die Frage, welche der beiden Möglichkeiten wann vorzuziehen ist. Beim arithmetischen IF-Statement ist für das darauffolgende Statement unbedingt eine Statementnummer notwendig. Vergleiche sind dabei nur mit Null möglich, sodaß eine solche Abfrage meist schlechter lesbar ist als eine logische. Außerdem benötigt die arithmetische Abfrage mehr Rechenzeit. Die logische Abfrage ist daher vorzuziehen; die arithmetische sollte nur dann verwendet werden, wenn in Abhängigkeit von einem arithmetischen Ausdruck einer von drei möglichen Wegen weiterverfolgt werden soll, also nicht zwei der drei auftretenden Statementnummern gleich sind.

Bei der Verwendung eines Computed GOTO der Form GOTO (K1,...,KN),I könnte es vorkommen, daß I außerhalb des Bereichs zwischen 1 und N liegt. Für I≤Ø erhält man die Fehlermeldung

```
COMP GOTO INDX .LE. Ø, EXIT 1 TKN AT XXXXXX .
```

das Programm läuft aber mit I=1 weiter. Ein Wert I>N hingegen führt zu einem RANGE VIOLATION Fehler und zum Abbruch des Programms.

Dem Computed GOTO fast gleichwertig ist das Assigned GOTO Statement, GOTO I,(K1,...,KN). Ein Programm arbeitet auch dann richtig, falls irrtümlich I eine Statementnummer zugewiesen worden ist, die nicht in der Liste K1,...,KN vorkommt.

Obwohl in Ein- und Ausgabebefehlen Feldnamen (ohne Indizes) angeführt werden dürfen, ist es doch besser, statt

```
WRITE (6,100) A
```

den Befehl

```
WRITE (6,100) (A(I), I=1,100)
```

zu verwenden, da dieser (für den Programmierer aufwendigere) Befehl weniger Rechenzeit benötigt.

Die Datenübergabe an ein Unterprogramm kann über Parameter sowie über COMMON-Größen vorgenommen werden, wobei das Rechnen mit COMMON-Daten schneller ist als das Arbeiten mit Parametern, zu denen der Zugriff wesentlich aufwendiger ist. Es erscheint daher sinnvoll, alle Größen, die in einem Unterprogramm mehrfach benützt werden, zu Beginn lokalen Größen zuzuweisen und in der weiteren Folge auf diese zuzugreifen. Man benötigt zwar zusätzlichen Speicherplatz, die Verarbeitung wird jedoch schneller. Dieses Umspeichern ist aber nur für Variable zu empfehlen, da bei Feldern der Aufwand zu groß wird. Bei Ein- und Ausgangsparametern muß der Wert der lokalen Größe vor dem Rücksprung in das rufende Programm wieder dem entsprechenden Parameter zugewiesen werden.

Bei FUNCTION-Unterprogrammen ist besonders darauf zu achten, ob der Typ des Ergebnisses implizit durch den Namen der FUNCTION oder explizit festgelegt wird.

Besonders wichtig ist die Überprüfung der Parameter auf ihre Gültigkeit (auch bei FUNCTION-Unterprogrammen). Wird ein Fehler festgestellt, so sollte dies durch einen Fehlerparameter oder eine COMMON-Größe dem rufenden Programm mitgeteilt werden.

COMMON-Blöcke müssen nicht in allen Programmteilen, in denen sie vorkommen, gleich lang sein. Allerdings muß im Hauptprogramm der volle COMMON-Block stehen, d.h. ein COMMON-Block darf nicht in einem Unterprogramm vergrößert werden.

In Standard-FORTRAN nicht vorgesehen, aber vom Compiler toleriert und richtig verarbeitet werden DATA-Statements für COMMON-Variable. Solche DATA-Statements können auch in Unterprogrammen stehen. Die Vorbesetzung der Variablen mit gegebenen Werten wird aber nur einmal zu Beginn und nicht bei jedem Unterprogrammeinsprung vorgenommen.

Abschließend sei noch auf zwei Möglichkeiten des Compilers hingewiesen, einen effektiveren und rascheren Code zu generieren, was besonders bei Hybridprogrammen zu bedenken ist.

Die erste Möglichkeit, einen schnelleren Code zu erhalten, besteht in einer speziellen Option des FORTRAN-Compilers, nämlich der Erzeugung von In-line Floating Point Processor und In-line Integer Code. Es werden hier für die Basisrechenoperationen mit Real bzw. Integer Zahlen keine Systemprogramme aufgerufen, sondern die entsprechenden Maschineninstruktionen direkt in das entstehende Objektprogramm hineincodiert. Das bedeutet natürlich einen erheblich größeren Speicherbedarf, aber die Durchführungszeit des Programmes wird bedeutend verkürzt.

Die zweite Möglichkeit besteht im In-line Assembly Codieren, d.h der Programmierer kann in seinem FORTRAN-Programm zeitkritische Teile direkt im Assembler codieren. Die gebräuchlichsten Maschineninstruktionen werden mit mnemotechnischen Abkürzungen, die weniger gebräuchlichen direkt mit dem oktalen Instruktionscode bezeichnet. Das In-line Assembly Codieren kann auch bei reinen Digitalprogrammen große Vorteile bei Bitoperationen, Shifts etc. bringen.

Abschließend sollen an einem Beispiel die Vorteile der beiden Methoden gezeigt werden, und zwar betrachten wir nur den Befehl I=I+1. Mit den üblichen Compiler-Options erhält man den kürzeren, aber zeitmäßig langsameren Code a.), bei Erzeugung von In-line Integer Code den längeren, dafür aber schnelleren Code b.):

a.)

```
L       .L11
ADR     $5
L       .A11
ADR     I
L       .H11
ADR     I
```

b.)

```
LA      *+2,2
J       *+2
ADR     $5
A       *+2,2
J       *+2
ADR     I
STA     *+2,2
J       *+2
ADR     I
```

(.L11, .A11, .H11 sind Systemroutinen zum Laden, Addieren und Abspeichern von Integerzahlen, $5 der Name einer Speicherzelle mit dem Wert 1). Die Zeiteinsparung beträgt im zweiten Fall etwa 35%. Zur In-line Assembly Codierung des Befehls I=I+1 genügt der Befehl AOM I (Add One to Memory location I). Zeitmäßig ergibt sich hier eine Rechenzeitersparnis von rund 5Ø%.

Natürlich muß man beim In-line Assembly Codieren eine große Maschinenabhängigkeit in Kauf nehmen, die aber bei Hybridprogrammen ohnehin schon gegeben ist. Interessenten, die von dieser Möglichkeit Gebrauch machen wollen, können nähere Informationen und Unterlagen in der Programmberatung bekommen.

F. BLÖSER

FOLGENDE SCHRIFTLICHE UNTERLAGEN FÜR BENÜTZER DER HYBRIDRECHENANLAGE SIND IN ZIMMER 1403 ERHÄLTLICH:

Bedienungsanleitung für den Analogrechner	10.-
Benützung des Betriebssystems JCS/MP 6	kostenlos
Steuerkarten in JCS/MP 6	kostenlos
Besonderheiten des EAI-FORTRAN	kostenlos
Einführung in die hybride Programmierung mit FORTRAN	5.-
Kurzbeschreibungen der Hybrid Linkage Subroutines	15.-
Programmieranleitung für EAI 140 Data Plotter	10.-
Disk Input/Output	kostenlos
Input/Output auf der Moving Head Disk	kostenlos
Exekutieren von Programmen mit sehr großem Kernspeicherbedarf, Segmentieren in JCS/MP 6	kostenlos
EAI Assembler	40.-
Kurzbeschreibungen von allen Bibliotheksprogrammen	kostenlos

INTERFACE Mai 1975