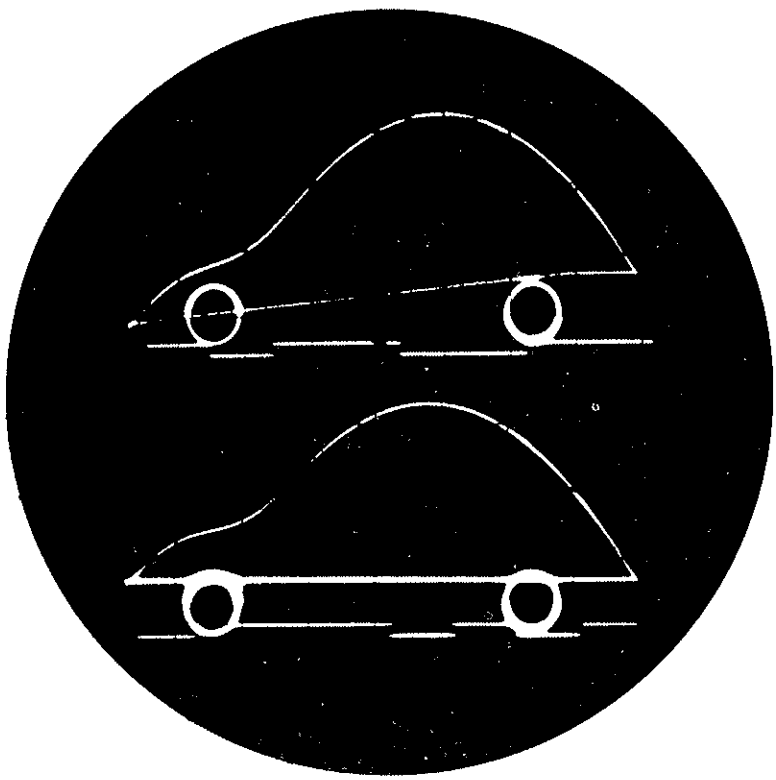
Interface

herausgegeben vom
Hybridrechenzentrum
des EDV-Zentrums
der Technischen Universität Wien

Nummer 11
April 1978



Schwingungsverhalten eines Autos
auf einer Teststrecke
(Analogsimulation und rein analoge Bilddarstellung)

INHALTSVERZEICHNIS

Redaktion: Irmgard Husinsky

Eigentümer, Herausgeber, Verleger: EDV-Zentrum der Technischen Universität Wien, Hybridrechenzentrum, Vervielfältigung: Österreichische Hochschülerschaft Technik, für den Inhalt verantwortlich: Dipl.Ing.W. Kleinert, alle Gußhausstraße 27-29, 1040 Wien

HYSMPK - PROGRAMMPAKET ZUR DIGITALEN UNTERSTÜTZUNG VON ANALOG- UND HYBRIDSIMULATIONEN MIT AUTOMATISCHER ON-LINE-SKALIERUNG

1. EINLEITUNG

Die Anforderungen der Benützer, die in den letzten Jahren mit ihren Problemen zum Hybridrechenzentrum der TU Wien kamen, zeigten die Notwendigkeit der Entwicklung hybrider Unterstützungssoftware für die Vorbereitungsarbeiten und die Dokumentation von analogen und hybriden Simulationen. So wurde mit dem Aufbau einer hybriden Programmbibliothek begonnen, die folgende Zielsetzung hatte:

- Unterstützung der Vorbereitungsarbeiten am Analogrechner (Skalierung, statischer Test)
- Dokumentation der Analogschaltung
- Dokumentation von Simulationsergebnissen
- vollständige Automatisierung von Standardsimulationen
- Bereitstellung von Standardverfahren zur Parameter-Optimierung, etc.

Diese Anforderungen führten zur Entwicklung eines digital verarbeitbaren Modells aller möglichen Analogschaltungen als Informationsgrundlage, die einerseits den Aufbau der geforderten hybriden Programmbibliothek mit Standardfunktionen und -verfahren, andererseits den Anschluß von Benützerunterprogrammen mit speziellen Funktionen und Verfahren ermöglicht. Die wesentlichen Schwierigkeiten bei der Entwicklung eines funktionsfähigen Programmpakets lagen dabei an der Schnittstelle zwischen automatischem Simulationsablauf und manuellem Eingriff in den Simulationsvorgang, die speziell das Erstellen der Schaltung am Analogrechner betrifft. Dieser fehlende Modul, der bereits als AutoPATCH-Erweiterung existiert, stellt das letzte Hindernis zur Entwicklung einer vollständigen hybriden Programmiersprache dar. Das Programmpaket wurde daher so konzipiert, daß es problemlos an die AutoPATCH-Erweiterung angeschlossen werden kann. Das Fehlen dieses AutoPATCH fand in der vorliegenden Version von HYSMPK speziell in den Ein- und Ausgabemoduln (siehe Abschnitte 2.1 und 2.2) insofern seinen Niederschlag, als die Problemgleichungen nur in zerlegter Form (Variablenvektor und Verbindungsvektor) eingegeben werden können. An der Entwicklung eines hybriden Compilers wird bereits gearbeitet, sein Einsatz wird jedoch erst mit der Installation eines AutoPATCH sinnvoll.

Das Hybrid Simulation Package HYSMPK ist ein Programmpaket zur Unterstützung von Analog- und Hybridsimulationen auf FORTRAN-Ebene, das aus einer Reihe von FORTRAN- und EAI-Assembler-Unterprogrammen, den Hybrid Simulation Subroutines HYSMSR und einem Prozessor HYBSYS zur interaktiven Bedienung der HYSMSR auf Overlay-Basis besteht. Mit diesem Programmpaket können sämtliche aus folgenden Komponenten bestehende Analogschaltungen verarbeitet werden:

Integrierer	Digital setzbare Funktionsgeber (DCFG)
Summierer	Komparatoren
Multiplizierer	Externe Funktionen (Komponenten, für welche
Quadrierer	zwar Ausgänge, aber keine Eingänge definiert
Quadratwurzelschaltung	sind)
Divisionsschaltung	Konstante Parameter
Punktspeicher	Sinus-, Cosinus-, Logarithmus- und
Switches	Exponentialfunktionsgeber
ADC	Limiter
DAC	

Als "Potentiometer" sind Servo Potentiometer, DCA, DAC und DAM zugelassen.

2. FUNKTIONEN

Die vorliegende Version des Hybrid Simulation Package HYSMPK beinhaltet die folgenden Moduln:

- Definition und Eingabe der Analogschaltung
- Dokumentation der Analogschaltung (inklusive Schaltplanzeichnung)
- Speichern der Analogschaltung auf externe Speicher
- Berechnung der Potentiometerwerte bei vorgegebener Skalierung
- Optimale oder überlauffreie On-Line-Skalierung
- Statischer Test
- Graphische Ausgabe unskalierter Variablen
- Interaktive Bedienung aller Moduln (HYBSYS)

2.1 Definition bzw. Eingabe der Analogschaltung, bestehend aus

- Definition von analogen, digitalen und hybriden Variablen
- Definition der Verbindungen von Variablen
- Löschen von Variablen
- Löschen von Schaltverbindungen
- Änderung von Potentiometer- und Verstärkeradressen und von Eingangswiderständen

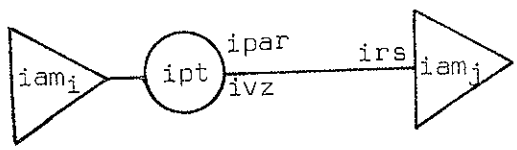
Die Analogschaltung wird durch einen Variablenvektor und einen Vektor der Schaltverbindungen festgelegt. Jeder Variablen entspricht am Analogrechner ein Verstärkerausgang oder (bei konstanten Variablen = Parametern) die Referenzspannung. Die Elemente x_i des Variablenvektors werden definiert durch die Tupel

$$x_i = (iam_i, iop_i, sx_i, name_i)_i$$

i	Index im Variablenvektor
iam	Verstärkeradresse der Variablen
iop	Operation, bzw. Typ des Verstärkers
sx	Skalierungsfaktor
name	problembezogener Name der Variablen

und die Elemente des Verbindungsvektors durch die Tupel

(i,j,ipt,irs,ipar,ivz,pw)



i	Index der Variablen, von der die Verbindung ausgeht
j	Index der Variablen, zu der die Verbindung führt
ipt	Adresse des Potentiometers der Schaltverbindung
irs	Eingangsverstärkung bei dem Verstärker der Variablen j oder Adresse eines Relays bei automatischer Anwahl des Eingangswiderstandes
ipar	Index des Parameters, mit dem die Eingangsvariable multipliziert wird
ivz	Vorzeichen der Eingangsvariablen in den Problemgleichungen (qualitativ)
pw	Wert des Potentiometers ipt

$$pw = f(iop_i, iop_j, ivz, sx_i, sx_j, sx_{ipar}, irs)$$

2.2 Dokumentation der Analogschaltung

- Liste der analogen Variablen einschließlich Maximum und Minimum der dazugehörigen Verstärkerausgänge nach erfolgter Skalierung (siehe Abb.4)
- Liste der Schaltverbindungen und Potentiometerwerte (siehe Abb.4)
- Automatische Schaltplanzeichnung am Data Plotter (siehe Abb. 5)

2.3 Speichern der Analogschaltung auf externe Speicher

- Datenfile auf Platte oder Magnetband
- auf Papertape

2.4 Berechnung der Potentiometerwerte bei vorgegebener Skalierung und Setzen der Potentiometer mit automatischer Steuerung der Maschinenzeit durch die Real-Time-Clock und automatischer Zeittransformation zur Realisierung der Potentiometerwerte bzw. um eine Änderung von Eingangswiderständen zu vermeiden (steuerbar durch Wahl der Art der Zeittransformation und einer Schranke für den kleinsten erlaubten Potentiometerwert). Potentiometer, deren Werte sich seit dem letzten Setzen nicht verändert haben, werden nicht gesetzt.

2.5 Optimale oder überlauffreie On-Line-Skalierung der Analogschaltung mit oder ohne automatische Zeittransformation und mit dauernder Fehlerkontrolle durch den ersten Übersteuerungszeitpunkt. Jeder Verstärker wird über das Amplifier Select System angewählt und über einen schnellen ADC in Abständen von ca. 80 µs bei gleichzeitiger Berechnung der Extremwerte bis zur ersten Übersteuerung desselben Verstärkers ausgelesen. Aus den so erhaltenen Extremwerten wird durch die Iterationsvorschrift

$$sx_{i+1} = sx_i * (\max|x(t)|)/r$$

mit

sx_{i+1}	neuer Skalierungsfaktor
sx_i	voriger Skalierungsfaktor
$\max\|x(t)\|$	absolutes Maximum des dazugehörigen Verstärkerausganges
r	Verfahrensparameter zur Steuerung der Konvergenzgeschwindigkeit und des Bereiches der überlauffreien Skalierung

ein neuer Skalierungsfaktor berechnet. Dieser Vorgang wird bis zum Erreichen einer optimalen oder überlauffreien Skalierung wiederholt. Die Skalierungsfaktoren werden dabei jeweils auf das nächstgrößere Vielfache einer frei wählbaren Skalierungseinheit aufgerundet.
Die nach erfolgreicher Skalierung erhaltenen Extremwerte der Verstärkerausgänge werden gespeichert und auf der Variablenliste ebenfalls ausgegeben.

2.6 Vollständiger statischer Test zu einem beliebigen Problemzeitpunkt mit vorzugebender Fehlerschranke und Ausgabe der erhaltenen Fehler oder des gesamten Testprotokolles. Zur Durchführung eines statischen Tests zum Zeitpunkt t_1 werden die durch einen Analoglauf ermittelten Integriererausgänge zum Zeitpunkt t_1 als neue Anfangswerte gesetzt.

2.7 Graphische Ausgabe (Data Plotter oder Graphic Display) einer oder mehrer unskalierter Variablen mit Parametervariation über der Zeit oder einer Variablen. Die Funktionen werden in einem beschrifteten Achsenkreuz gezeichnet und können wahlweise mit ihren Namen oder entsprechenden Parameterwerten beschriftet werden. Zusätzlich kann die Zeichnung mit einem Text und den Namen und Werten von bis zu 12 Parametern beschriftet werden (siehe Abb. 6 und 7). Die auszulesenden Verstärker werden über das Amplifier Select System angewählt und von einem ADC über einen direkten Datenkanal ausgelesen. Die Anzahl der auszulesenden Werte wird von der Maschinenzeit gesteuert (Standardfrequenz bei Time Scale Normal Millisec: 10^4).

2.8 Interaktive Bedienung aller erwähnten Moduln und angeschlossener User-Unterprogramme und Parameteränderung mit dem Prozessor HYBSYS. HYBSYS besteht aus einer festen Programmzone (14 k Worte) und einer Overlay Zone (variabler Größe) und wird mit Commands bedient. Bei der Exekution eines Commands wird die Overlay Zone mit den speziell benötigten Unterprogrammen geladen (siehe Abb. 3). HYBSYS erlaubt die Errichtung beliebig vieler User-Overlays, die

nach erfolgter Initialisierung als Overlay ebenfalls durch ein vom User definiertes Command aktiviert werden können. Auf Grund der Overlay-Struktur des Prozessors ist für das (momentan im Gesamten ca. 50 k Worte umfassende) Programmpaket eine maximale Kernspeicherbelegung von 24 k Worten erforderlich, die durch den größten Overlay (graphische Ausgabe der Variablen, 9.5 k Worte) bestimmt ist. Die durchschnittliche Größe der restlichen Overlays beträgt ca. 3 - 4 k Worte.

Befehlssatz von HYBSYS:

INPUT,OUTPUT	Definition der Ein- und Ausgabeeinheit
READ	Einlesen einer Schaltstruktur
DEFINE	Definition einer Variablen
DELETE	Löschen einer Variablen
CHANGE	Adressenänderung analoger Komponenten
LIST	Liste der Schaltstruktur
PLOTP	automatische Schaltplanzeichnung
GET	Lesen einer Schaltstruktur von externem Speicher oder Laden eines Overlays
STORE	Abspeichern einer Schaltstruktur auf externem Speicher oder Erzeugen eines Overlays
SET	Berechnen und Setzen der Potentiometerwerte
SCALE	Automatische Skalierung
PLOT	Zeichnen von analogen Variablen am Data Plotter
PLOTCP	Zeichnen von analogen Variablen am Graphic Display mit Copy Option
TEST	Statischer Test

3. NEUDEFINITION ANALOGER KOMPONENTEN

Bedingt durch ein allgemeines Modell aller möglichen (und sinnvollen) Analogschaltungen, das als Informationsgrundlage des Programmpaketes dient, müssen die analogen Komponenten folgende Bedingungen erfüllen:

- Jedem Integriereingang (einschließlich IC) muß ein Potentiometer (auch DCA, DAC, OAM) vorgeschaltet sein.
- Jedem Summiereingang muß ein Potentiometer vorgeschaltet sein, Feedback-Potentiometer können ebenfalls verwendet werden.
- Jedem Komparatoreingang ist ein Potentiometer vorzuschalten.
- Bei der Divisionsschaltung muß am Eingang des Zählers ein Potentiometer vorgeschaltet sein.
- Bei Multiplizierern kann ein Feedback-Potentiometer verwendet werden.
- Den Eingängen der digital setzbaren Funktionsgeber (DCFG) ist ein Potentiometer vorzuschalten, die Ausgänge der Funktionsgeber müssen an einem vom Amplifier Select System anwählbaren Verstärker anliegen.
- Bei den Eingängen der folgenden Komponenten, bzw. Schaltungen darf kein Potentiometer verwendet werden:
  - Inverter
  - Absolutbetragsschaltung
  - Multiplizierer
  - Nenner der Divisionsschaltung
  - Quadrierer
  - Quadratwurzelschaltung

Potentiometer werden nicht als eigene Komponenten, sondern als Teil des Eingangs- und Rückführnetzwerkes eines Verstärkers angesehen. Die Multiplikation einer analogen Variablen mit einem konstanten Parameter ist nicht an die Existenz eines Potentiometers gebunden, sondern wird digital durch Multiplikation der Skalierungsfaktoren durchgeführt.

4. ANALOGE UND LOGISCHE GRUNDSCHALTUNG

Von den hybriden Unterprogrammen des Programmpaketes wird folgende analoge und logische Schaltung benötigt:

4.1 Logische Schaltung (Steuerung)

Repop Timer: A = 1.
B = 999.9
C = 999.9

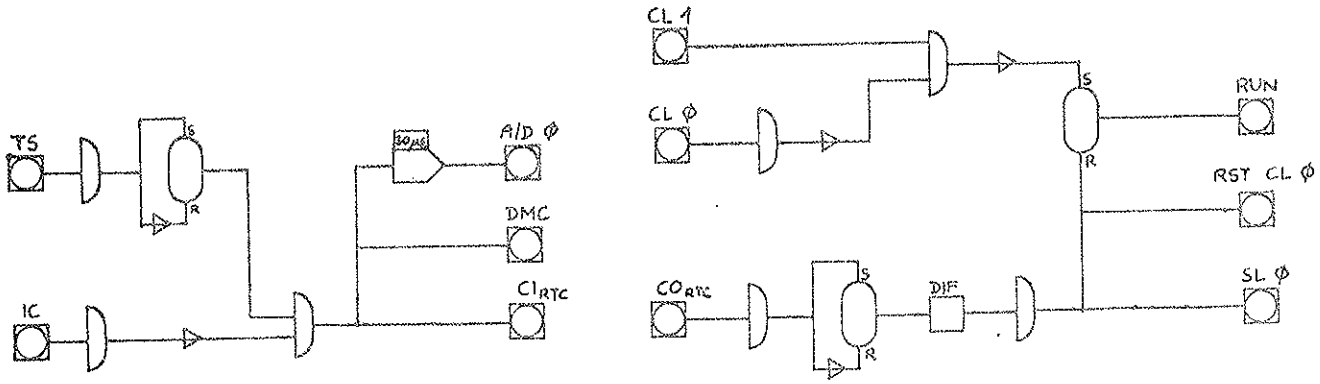


Abb. 1

Der Repop Timer wird auf die volle Dauer eingestellt, die benötigte Maschinenzeit wird von den Programmen mit der Real-Time-Clock (RTC) gesteuert.

Controlline 1: dient zur Wahl von Single oder Repop Betrieb.
Cl 1 = Ø Repetierendes Rechnen unabhängig von Cl Ø
Cl 1 = 1 Single Mode, einmal Rechnen, Start durch Set Cl Ø.

Controlline Ø: dient zur Steuerung eines einzelnen Analoglaufes und hat nur bei Single Mode Bedeutung (Cl 1 = 1). Durch Setzen von Cl Ø wird ein Analoglauf gestartet, am Ende des OP-Intervalles, das von der Real-Time-Clock angezeigt wird, wird Cl Ø automatisch zurückgesetzt.

Senseline Ø: teilt dem digitalen Programm das Ende eines Analoglaufes mit.

Die mit TS (Timescale) und IC (Initial Condition) erzeugte Impulsfolge während des OP-Intervalles dient zum Mitzählen der Rechenzeit und wird beim Auslesen von Verstärkern für die graphische Darstellung als Transfersignal verwendet.

4.2 Analoge Schaltung

Diese Schaltung dient zum Auslesen eines beliebigen Verstärkers über einen schnellen ADC. Der Verstärker wird über das Amplifier Select System angewählt und kann dann an der Buchse A-SEL weiterverwendet werden. Diese Schaltung wird bei der automatischen Skalierung und beim Zeichnen von Verstärkerausgängen verwendet.

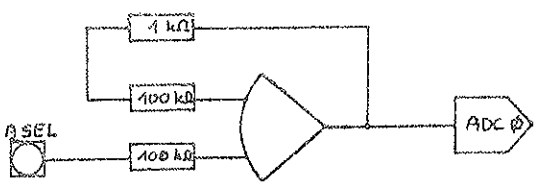


Abb. 2

Da der A-SEL-Ausgang einen 1kΩ Innenwiderstand hat, muß in der Rückführung des Verstärkers ebenfalls ein 1kΩ Widerstand in Serie geschaltet werden, um einen Verstärkungsfaktor von 1 zu erhalten.

5. BEISPIEL: PHARMAKOKINETISCHES MODELL

Der Einsatz des Programmpakets sei nun an einem pharmakokinetischen Modell mit zwei parallelen Verteilungsräumen demonstriert, das durch das folgende Gleichungssystem beschrieben wird

$$\frac{dm}{dt} = -k_1 m$$

$$\frac{db}{dt} = k_1 m + (k_3 g - k_2 b) - (k_5 b - k_6 l) - k_4 b$$

$$\frac{dg}{dt} = (k_2 b - k_3 g)$$

$$\frac{dl}{dt} = (k_5 b - k_6 l)$$

$$\frac{du}{dt} = k_4 b$$

$$m(0) = 100$$
$$b(0) = g(0) = l(0) = u(0) = 0$$

wobei m(t), b(t), g(t), l(t) und u(t) die Konzentrationen des Pharmakons in Magen, Blut, Gewebe, Leber und Urin darstellen, k_1 bis k_6 sind die Reaktionskonstanten.

Nach Entwerfen des Schaltplans und Erstellen der Schaltung ist eine Liste aller Variablen (einschließlich Parameter) und aller Schaltverbindungen anzulegen, bzw. bei Eingabe vom Card Reader zu lochen.
Bei der Definition der Schaltverbindungen wird die invertierende Wirkung der Verstärker nicht berücksichtigt. Soll an einem Verstärkerausgang die invertierte Variable anliegen, wird ihr ein negativer Skalierungsfaktor zugeordnet (z.B. bei der Variablen BLUT).

Die folgenden Abbildungen zeigen das Protokoll eines interaktiven Laufs von HYBSYS und die dabei erzielten Ergebnisse.

```
HYBSYS
-GET,PHARMA;
-EPS=.1;
-SET;
 VALUE  .7722E-01 OF POT  73 LESS EPS
-SCALE;
 SCALING PROCEDURE  3
 TIME TRANSFORMATION: M-TIME = 1403; BETA = .1403E+02
 CONTROL RUN
 COMPLETED AFTER  4 RUNS
-LIST,LP;
-STORE;
-TEST,.0005;GET;
 AMPLIFIER OUTPUT CHECK
 POT-TEST / OFF-LINE
 POT-TEST / ON-LINE
 DERIVATIVE CHECK
 AMPLIFIER - POT CONNECTION CHECK
 POT - AMPLIFIER CONNECTION CHECK
-XTEXT=TAGE;YTEXT=KONZENTRATION;
-TEXT=MODELL 37;
-PARTXT=K1,K2,K3,K4,K5,K6;
-PARTXT:
   K1      = .2500E+01
   K2      = .1500E+01
   K3      = .3000E+01
   K4      = .3000E+01
   K5      = .2000E+01
   K6      = .6000E+01
-TMTXT=.8;TIME=5.;SET;
-PLOT URIN,GEWEBE,BLUT,LEBER,MAGEN;
-YTEXT=URIN;
-PLOT URIN,K1=1.,5.1,1.;
 POT  90 TO GAIN  10 AT AMPL.  95
 AFTER PATCHING SET SL(5)
-K2=2.,K6=4.;SET;REPOP;
-PLOT GEWEBE,BLUT;
-PLOTP;
-EXIT;
```

Abb. 3
HYBSYS-Protokoll:
die Zeilen für Benützereingabe werden von HYBSYS mit '-' markiert. Ferner gibt HYBSYS Kontrollmeldungen über den Verlauf der einzelnen Funktionen sowie Fehlermeldungen.

HYBSYS

PROJECT PHARMA

PROBLEM-TIME = .1000E+02 MACHINE-TIME = 1403

TIME-TRANSFORMATION BETA = .1403E+02

VARIABLE	ADDRESS	TYPE	NAME	SCAL.FACTOR	MINIMUM	MAXIMUM
X(1)	90	INT	MAGEN	.100000E+03	-.0007	.9899
X(2)	95	INT	BLUT	.360000E+02	-.9851	.0005
X(3)	70	INT	GEWEBE	.120000E+02	-.0007	.9893
X(4)	100	INT	URIN	.100000E+03	.0000	.9912
X(5)	75	INT	LFBER	.100000E+02	-.0007	.9746
Y(1)	96	SUM	B-G	.390000E+02	-.2252	.9742
Y(2)	76	SUM	B-L	.420000E+02	-.2198	.9773
Z(1)	71	INV	G-B	-.390000E+02	-.9744	.2251
Z(2)	80	INV	L-B	-.420000E+02	-.9773	.2198

P(1)	511	PAR	T0	.000000E+00
P(2)	512	PAR	TIME	.100000E+02
P(3)	509	PAR	BETA	.140300E+02
P(4)	508	PAR	M(0)	.100000E+03
P(5)	507	PAR	K1	.500000E+01
P(6)	506	PAR	K2	.150000E+01
P(7)	505	PAR	K3	.300000E+01
P(8)	504	PAR	K4	.300000E+01
P(9)	503	PAR	K5	.200000E+01
P(10)	502	PAR	K6	.600000E+01
P(11)	501	PAR	K1M	-.100000E+01
P(12)	500	PAR	Z1	.100000E+01
P(13)	497	PAR	L(0)	.100000E+02
P(14)	498	PAR	G(0)	.120000E+02
P(15)	499	PAR	U(0)	.500000E+02
P(16)	496	PAR	B(0)	.360000E+02

PATCHING CONNECTIONS / POTENTIOMETER SETTINGS

OUTPUT		PAR	TYPE	NUMBER	VALUE	INPUT	AMPLIFIER		NAME
INT	90	K1	POT	91	.3564F+00	1	INT	90	MAGEN
	4		DAC	1	-.1000E+01	IC			
INT	90	K1	POT	90	.9899E+00	1	INT	95	BLUT
INT	95	K4	POT	95	.2138E+00	1			
INV	71		POT	73	.7722E-01	1			
INV	80		POT	80	.8316E-01	1			
	16		DAC	2	.1000F+01	IC			
INV	71		POT	71	.2316E+00	1	INT	70	GEWEBE
	14		DAC	4	-.1000F+01	IC			
INT	95	K4	POT	94	.7698E-01	1	INT	100	URIN
	15		DAC	3	-.5000E+00	IC			
INV	80		POT	82	.2994E+00	1	INT	75	LFBER
	13		DAC	5	-.1000F+01	IC			
INT	95	K2	POT	97	.1385F+00	10	SUM	96	B-G
INT	70	K3	POT	72	.9231E+00	1			
INT	95	K5	POT	99	.1714E+00	10	SUM	76	B-L
INT	75	K6	POT	77	.1429E+00	10			
SUM	96			-99	.1000E+01	1	INV	71	G-B
SUM	76			-99	.1000F+01	1	INV	80	L-B

Abb. 4

Liste der Variablen und Schaltverbindungen
(Command LIST in HYBSYS)

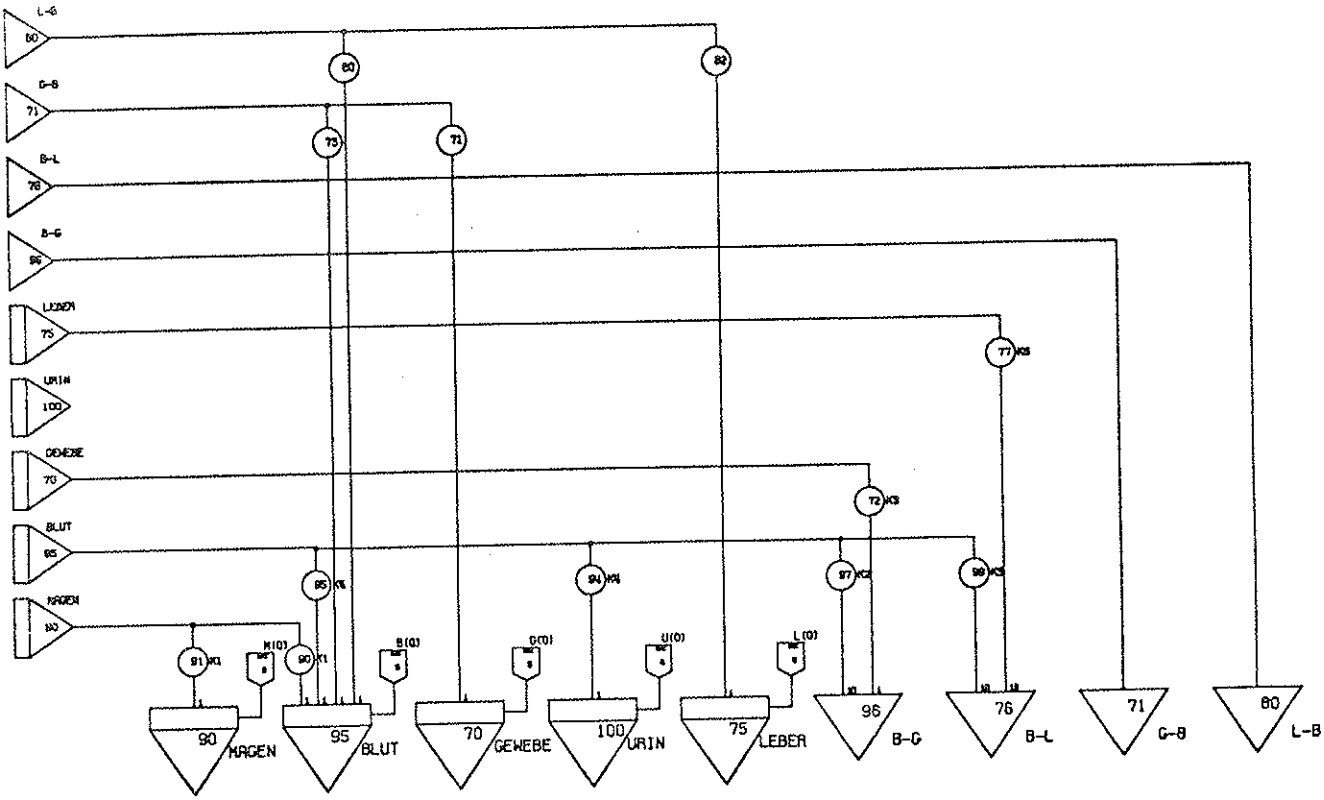


Abb. 5

Darstellung des Schaltplans am Data Plotter
(Command PLOTP in HYBSYS)

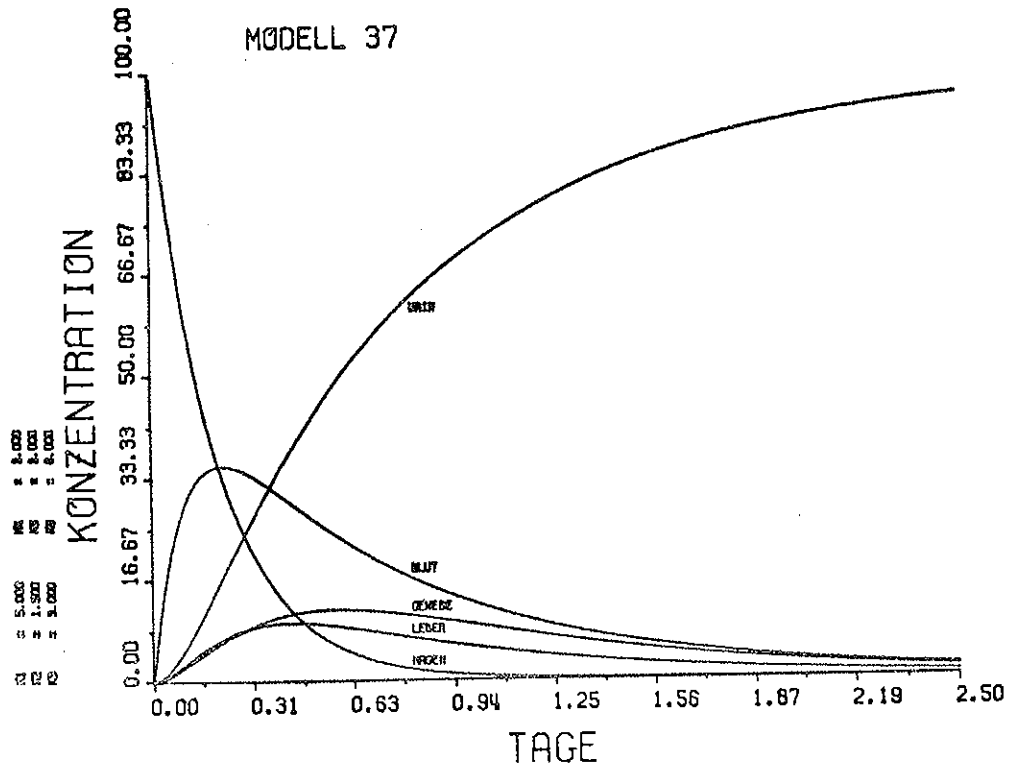


Abb. 6

Darstellung der Lösung des Differentialgleichungssystems am Data Plotter
(Command PLOT in HYBSYS)

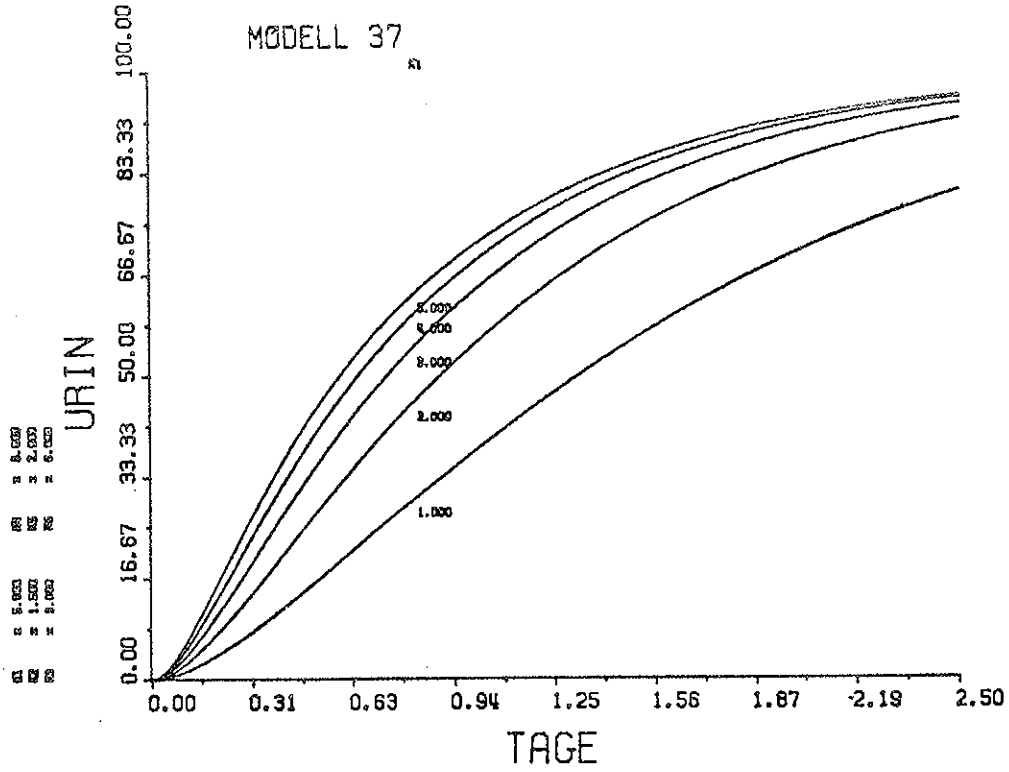


Abb. 7

Darstellung einer Variablen für verschiedene Parameterwerte am Data Plotter (Command PLOT in HYBSYS)

6. ZUSAMMENFASSUNG

Die Programme des Pakets HYSMPK sind bereits seit zwei Jahren intern im Einsatz und werden ständig verbessert und erweitert. Die vorliegende Version stellt wegen der vorläufig unvollständigen Automatisierbarkeit des Analogrechners nur eine Zwischenlösung dar, die jedoch auf Grund ihres Konzeptes allen Erweiterungen zugänglich ist. Eine vollständige digitale Unterstützung analoger und hybrider Simulationen wird erst durch die Installation eines AutoPATCH ermöglicht werden.

SUMMARY

The requirements of the users of our hybrid computation centre showed the necessity of a software system which supports preparation and documentation of analog simulations. The aims of the hybrid program library which has been developed were the following: support for preparations on the analog computer (scaling, static test), documentation of the analog patchings (incl. drawing of patch panel wiring), documentation of simulation results, full automation of standard simulations, support of standard algorithms for parameter optimizations, etc. The Hybrid Simulation Package HYSMPK is a program package consisting of several FORTRAN and Assembler Subroutines (Hybrid Simulation Subroutines HYSMSR) and a hybrid processor HYBSYS, based on overlays, which enables interactive usage of the HYSMSR. The characteristics of the system are described and demonstrated by an example (pharmacocinetic model).

D. Solar

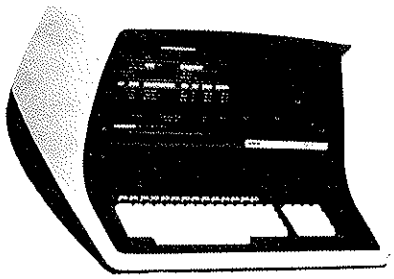
aktuelle mitteilungen

ÖFFNUNGSZEITEN:

Die Hybridrechenanlage ist von

Montag bis Freitag
von
8 Uhr bis 20 Uhr

geöffnet. Die Closed-Shop-Eingabe ist zu jeder vollen Stunde.



NEUES DATENSICHTGERÄT REGENT 100

Am Hybridrechenzentrum wurde ein neues Datensichtgerät, ein REGENT 100 der Firma KONTRON, als Video 2 installiert. Dieses löst das alte Terminal EAI C101 ab, das jetzt als Video 4 angeschlossen ist. Das neue mikroprozessorgesteuerte Gerät verfügt über einen Bildschirmspeicher von 1920 Zeichen zu 24 Zeilen mit je 80 Zeichen. Eine Statuszeile (25. Zeile) zeigt den jeweiligen Bedienungsstatus an.

Die wesentlichen Eigenschaften des REGENT 100 sind:

- erweiterter Sonderzeichensatz,
- Groß- und Kleinschreibung,
- graphische Sonderzeichen für rechtwinkelige Strukturen (Tabellen, Gitter, Statistiken, etc.),
- wahlweise dunkler oder heller Hintergrund,
- Möglichkeit, jeden Bildschirmbereich blinken zu lassen,
- Unterstreichen von Texten,
- halbe oder volle Intensität,
- absolute Cursorpositionierung.

Alle diese Eigenschaften werden softwaremäßig unterstützt. Die entsprechenden Routinen sind über den File REGENT auf District 9 ansprechbar (siehe auch Seite 25).

NEU: SWITCH PANEL

Dieses Panel gestattet neben verschiedenen Schaltmöglichkeiten zwischen EAI PACER 1ØØ, CYBER 74 und Prozeßrechner DEC 2Ø innerhalb des Hybridrechenzentrums nun auch dem Institut für Numerische Mathematik das Arbeiten am PACER 1ØØ über Terminal, sowie den Anschluß des Instituts für Numerische Mathematik an die Prozeßrechenanlage und den Aufbau von Datenfernverbindungen von und zum PACER 1ØØ über das Telefonnetz. Das Institut für Regelungstechnik wird ebenfalls in diesen Verbund integriert werden. Siehe auch Seite 23

NEU: TERMINAL ZUR PROZESSRECHENANLAGE

An der Hybridrechenanlage kann ab sofort ein Terminal zur Prozeßrechenanlage eingesetzt werden. Auskünfte können beim Operator eingeholt werden. Im nächsten INTERFACE soll mehr darüber berichtet werden.

RECON - ERÖFFNUNG

Nach dankenswerter Unterstützung des Hybridrechenzentrums durch die Austrian Solar and Space Agency (ASSA), vertreten durch Frau Dr. Müller, sowie durch die zuständigen Stellen des Bundesministeriums für Wissenschaft und Forschung, insbesondere Herrn Dr. Rozsenich, konnte die Benützung des Literaturinformationssystems RECON des Space Documentation Service ESRIN der European Space Agency am 3. März 1978 im Beisein der Vertreter einiger interessierter Institute aufgenommen werden. Bis zum jetzigen Zeitpunkt wurden Recherchen auf den Gebieten

- o Terrestrische Photogrammetrie
- o Supraleitung
- o Physik der Verbindungen seltener Erdmetalle
- o Redundante Multi-Microprozessor-Systeme
- o Vichnevetsky-Verfahren zur Lösung partieller Differentialgleichungen

durchgeführt.

Alle interessierten Institute der Technischen Universität werden gebeten, an das Hybridrechenzentrum (Herrn H. Hummer) heranzutreten. Da keine Ansuchen um Finanzierung der Suchläufe von den Einzelinstituten mehr gestellt werden müssen, kann mit den jeweiligen Suchläufen praktisch sofort begonnen werden.

NEU IN DER PROGRAMMBIBLIOTHEK

Das Ende der Datenkarten für ein Programm kann nun mit der REAL FUNCTION EOF abgefragt werden.
Verwendung: EOF (I)
Für I = 5 bedeutet EOF = Ø. Ende der Datenkarten noch nicht erreicht
EOF = 1. es sind keine Datenkarten mehr vorhanden

Für I ≠ 5 ist EOF immer Ø.
Bei Verwendung von EOF ist eine LOAD,D2 EOF Steuerkarte nach der BUILD-Karte erforderlich. LABEL 1 und REWIND 1 dürfen dann nicht verwendet werden.

Ein neues Unterprogramm LING ersetzt die bisher vorhandenen Programme CROUT und SIMQ zur Lösung eines linearen Gleichungssystems. Beschreibungen sind in der Programmberatung erhältlich.

GROSSES INTERESSE AM IPP1

Das im Rahmen der Projektgruppe Mathematische Software des Instituts für Numerische Mathematik der TU Wien am Hybridrechenzentrum entwickelte Interpolations- und Plotterprogrammpaket IPP1 läuft seit Mitte letzten Jahres unverändert mit großem Erfolg am EAI PACER 600. Bereits im September wurden die 48 FORTRAN-Programme des IPP1 auch am Rechenzentrum der Universität Karlsruhe implementiert, was auf Grund der Maschinenunabhängigkeit und Transportabilität der Programme ohne Schwierigkeiten möglich war. Seit Ende letzten Jahres wird das IPP1 auch am Regionalen Rechenzentrum für Niedersachsen in Hannover auf einer CDC CYBER 73 und 76 Rechenanlage verwendet. Außerdem wurde das IPP1 am Digitalrechenzentrum der TU Wien für das Institut für Festigkeitslehre installiert. Der an diesem Institut verwendete Houston Instrument Plotter konnte nach der Änderung weniger COMMON-Größen (Zeicheneinheit, Zeichengröße, etc.) direkt angesprochen werden, da eine CalComp-kompatible Plottergrundsoftware, wie sie das IPP1 voraussetzt, vorhanden war.

RH9 EINFÜHRUNGSKURS IN DIE HYBRIDE PROGRAMMIERUNG

Dieser Kurs wird nach Bedarf für wissenschaftliche Benützer, die keinerlei Vorkenntnisse auf dem Gebiet der hybriden Programmierung besitzen und an einer möglichst raschen Problemlösung interessiert sind, abgehalten.
Dauer: 2 Tage
Vortragender: D. Solar
Nähere Auskünfte bei Herrn D. Solar (1040 Wien, Gußhausstraße 27-29, 4. Stock, Zimmer 1401, Tel.: 65-37-85 / 747 DW).

Der Kurs RH9 hat bis jetzt 6 mal stattgefunden, wobei jedesmal 4-5 Interessenten teilgenommen haben. Anhand von schriftlichen Unterlagen und eines Demonstrationsbeispiels wurde die hybride Rechentechnik erklärt. Ferner wurde auf Wunsch von Teilnehmern auf spezielle Problemkreise eingegangen. Viele Kursteilnehmer verwerteten später die erworbenen Kenntnisse bei der Lösung ihres wissenschaftlichen Problems am Hybridrechner.

Die Kurse

RH3 Hinweise für FORTRAN-Programmierer an der Hybridrechenanlage
RH4 Hybride FORTRAN-Programmierung
RH7 Softwareunterstützung für die Benützung des PACER 600 als Plotter System
RH10 Datenverbindung mit CYBER 74

werden wieder im Herbst 1978 stattfinden. Genaue Termine werden im nächsten INTERFACE bekanntgegeben werden.

HYBRIDES MULTIPROGRAMMING FÜR HOI-BENÜTZER MIT PARALLELEM ZUGRIFF ZUM ANALOGRECHNER

Das Betriebssystem JCS/TS 7 ermöglicht Multiprogramming prinzipiell nicht nur für digitale Jobs, sondern auch für hybride. Diese Möglichkeit ist insbesondere für hybride HOI-Jobs interessant, da Zeitverluste durch das Cycling (6ms) auf Grund der Konzeption der Interpretersprache HOI keine wesentliche Rolle spielen.

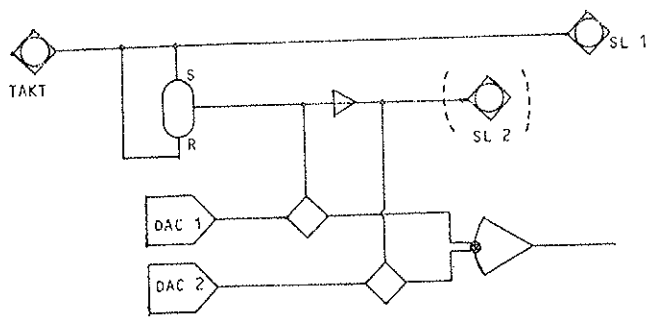
Im Hybridpraktikum I (WS 77/78) bot sich nun die Möglichkeit, ein echtes hybrides Multiprogramming zu versuchen. Die Möglichkeit des Multiprogrammings wurde durch die große Praktikumsgruppenanzahl sogar zu einer Notwendigkeit. Im folgenden sollen kurz die dabei auftretenden Schwierigkeiten behandelt werden.

Es rechneten jeweils zwei Gruppen parallel am Hybridrechner einen HOI-Job. Dabei wurde der einen Gruppe als Ein/Ausgabeeinheit das Tektronix, der anderen das Teletype zugeordnet. Die "Aufteilung" des Analogrechners gestaltete sich schwieriger, insbesondere da ja nur eine Zeitgebereinheit zur Verfügung steht. Um beide Programme voll kompatibel zu erhalten, wurde prinzipiell auf die Verwendung des Timers verzichtet. Beide Benutzer verwendeten Einzelsteuerung. Damit wurde außerdem vermieden, daß beide Benutzer einander bei Verwendung der zentralen analogen Steuermodes IC, OP, PP, H ... behindern.
Von Anfang an wurde der Analogrechner in den PP-Mode gesetzt, um sämtliche Taktfrequenzen zur Verfügung zu stellen; die dabei am Timer eingestellte Zeit ist bedeutungslos. Der Zugriff zu allen Taktfrequenzen war deshalb oft nötig, weil durch Herabzählen diverser Frequenzen Zeitpunkte für den Ablauf der jeweiligen Einzelsteuerung bestimmt werden mußten.

Die zeitlichen Synchronisationsprobleme zwischen Digital- und Analogrechner, die durch das Cycling entstehen, sind die folgenden:

1) Soll ein Wert in den Digitalrechner übertragen werden (z.B. über einen ADC), so kann bis zur tatsächlichen Durchführung des Auslesebefehles, gemessen vom gewünschten Übertragungszeitpunkt an, eine Zeit von maximal 18ms verstreichen.
   Abhilfe: Verwenden von T/S-Einheiten oder Setzen der Integrierer in H-Mode.
   Konsequenz: Soll eine ganze Funktion stützstellenweise übertragen werden, so kann der Abtastfrequenzabstand eine Zeit von ca 50ms nicht unterschreiten.

2) Soll ein Wert dem Analogrechner zu einem bestimmten Zeitpunkt zur Verfügung gestellt werden, so kann es dabei ebenso zu einer Verzögerung von 18ms kommen.
   Abhilfe: Besonders beim Übertragen ganzer Funktionen ist zum Ausgleich des Cyclings einiger Aufwand nötig. Eine Möglichkeit, die von zwei Gruppen realisiert wurde, ist, zwei DAC's oder DCA's zu verwenden, wovon einer jeweils schon den zum Zeitpunkt t_i gewünschten Wert enthält, während der andere zum Zeitpunkt t_i mit dem Wert zum Zeitpunkt t_{i+1} geladen wird. Ein Schalter am Analogrechner und eine logische Variable bzw. eine weitere Senseline im Digitalrechner schalten den aktuellen Wert durch, bzw. adressieren den aktuellen zu ladenden DAC (DCA).
   Die folgende Schaltung veranschaulicht die Vorgangsweise:



3) Beim Abfragen einer Senseline muß beachtet werden, daß sie ebenso 18ms den gewünschten logischen Zustand behalten muß.

Prinzipielle Probleme allerdings bildet beim parallelen Hybridrechnen das Potentiometersetzen. Durch den Aufruf eines Befehles zum Setzen eines Servopots wird der Analogrechner bekanntlich in den SP-Mode, wenn auch nur für kurze Zeit, gesetzt, wodurch der zweite Benutzer mehr als empfindlich gestört wird: sein Programm rechnet falsch.

Abhilfe ist hier nur durch Verwendung elektronischer Potentiometer (DCA's) möglich, allerdings begrenzt durch die Anzahl der DCA's (derzeit 2Ø).

Benötigen beide Benutzer daher mehr als 2Ø Potentiometer, so war eine Aufteilung in "fixe" und "variable" Potentiometer nötig. Koeffizienten, die während der gesamten Rechenzeit unverändert blieben, galten als "fixe" und wurden durch Servopots realisiert. Diese Pots wurden am Anfang der Rechenzeit beider Benutzer gesetzt, noch bevor der Analogrechner in den generellen PP-Mode gesetzt wurde. Variable Pots wurden als DCA's gesteckt und konnten damit während der Rechnung beliebig von beiden Benutzern verändert werden.

Zusammenfassend müssen beim begrüßenswerten Multiprogramming in HOI folgende Punkte beachtet werden:

- o möglichst Einzelsteuerung verwenden
- o möglichst DCA's und keine Servopots verwenden
- o günstigerweise generell den PP-Mode verwenden
- o Zeitverzögerung von 18ms bei Datenübertragung ausgleichen

Die Vorteile des Multiprogrammings liegen auf der Hand:
bessere Ausnutzung des Hybridrechners,
beide HOI-Benutzer können im Rahmen des Spoolings des Betriebssystems gleichzeitig Line Printer, Card Reader, Data Plotter sowie Datenfiles verwenden.

Die Nachteile sind:
vermehrter Steuerungsaufwand (kaum abschaffbar),
stark begrenzte Potentiometeranzahl (durch Anschaffung weiterer DCA's behebbar).

Sinnvolles Multiprogramming in HOI kann erst mit AutoPATCH zur vollsten Zufriedenheit gelöst werden, da dann durch das Gerät selbst für beide Programme die Schaltungen definiert werden.

F. Breitenecker
Institut für Technische Mathematik
Arbeitsbereich Regelungstechnik
und Hybridrechentechnik
Technische Universität Wien

I P P 2 - INTERPOLATIONS- UND PLOTTERPAKET FÜR ZWEIDIMENSIONALE PROBLEME (VORLÄUFIGE VERSION)

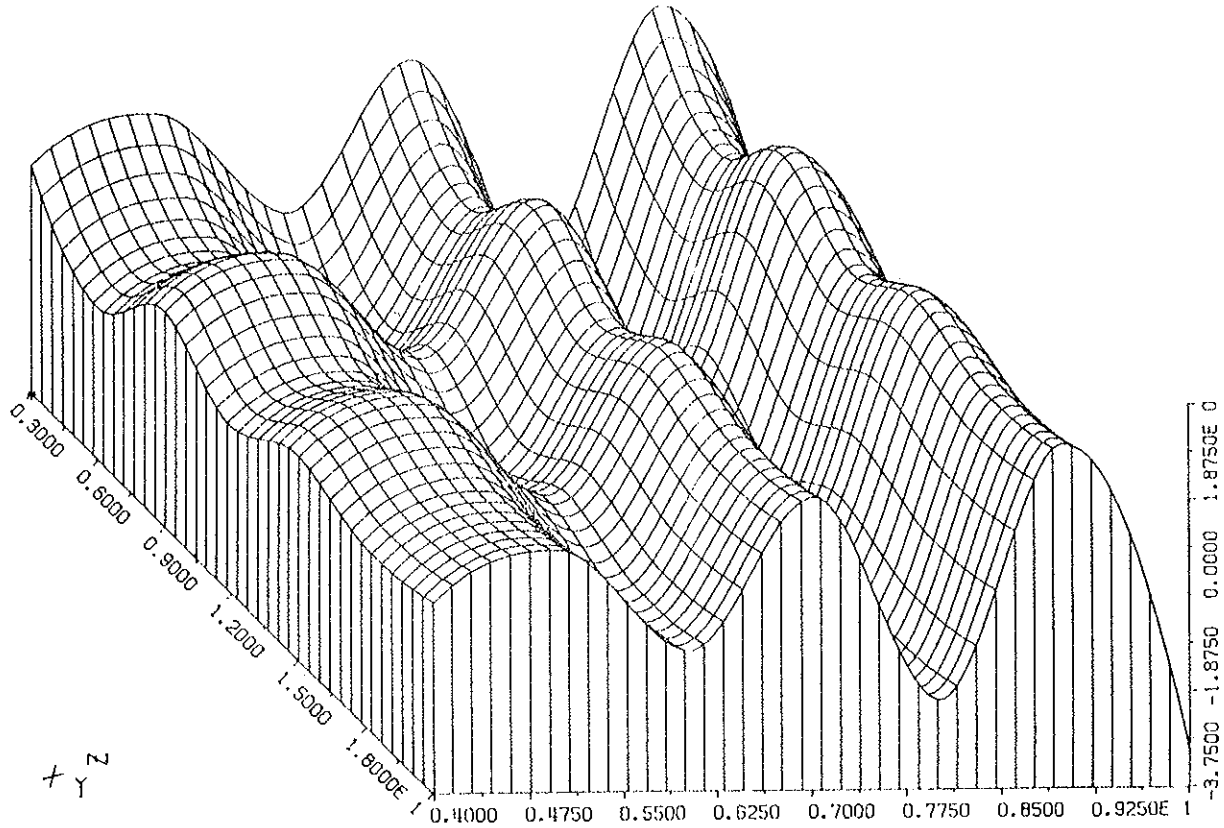
In Zusammenarbeit mit dem Institut für Numerische Mathematik wurde von Herrn Wolfgang Remp am Hybridrechenzentrum ein Programmpaket für die Interpolation und graphische Darstellung von Flächen entwickelt. Das Paket umfaßt 18 FORTRAN-Programme und stützt sich auf die CalComp-kompatible Plottergrundsoftware, die für den Data Plotter und das Graphic Display Terminal am Hybridrechenzentrum zur Verfügung steht. Außerdem werden zwei Programme aus dem IPP1 (Interpolations- und Plotterpaket für Funktionen und ebene Kurven) verwendet, das schon seit längerer Zeit am EAI PACER in Verwendung ist. Es ist geplant, die IPP2-Programme noch einmal zu überarbeiten und auf den Standard des IPP1 zu bringen. Dennoch sollen die Programme schon jetzt in einer *vorläufigen* Version interessierten Benutzern zur Verfügung gestellt werden.

Bei Verwendung der Programme des IPP2 ist die darzustellende Fläche des dreidimensionalen Raumes durch eine Matrix von Funktionswerten vorzugeben. Je nach der gewünschten Darstellung können die Werte dann interpoliert werden, und zwar für die Darstellung mittels Schnittlinien im wesentlichen nur in einer Richtung mithilfe stückweise kubischer Funktionen bzw. für die Darstellung mittels Netz mithilfe stückweise bikubischer Funktionen nach H. Akima. Die Darstellung der Fläche kann entweder als Projektion (Blickwinkel fix oder frei wählbar) oder als Schrägriß (mit oder ohne Interpolation) erfolgen.

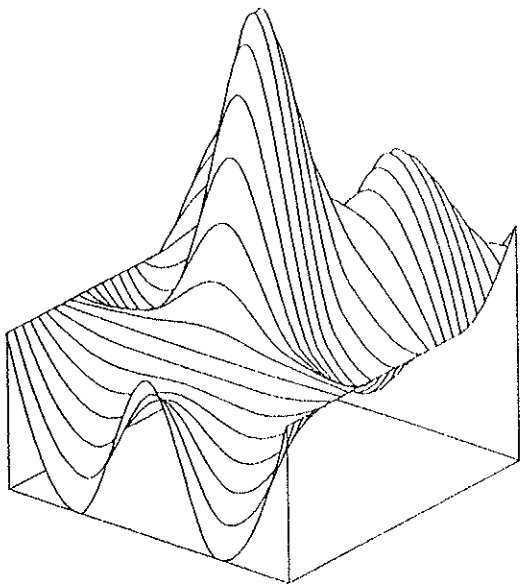
Um den Benutzern die Verwendung des IPP2 zu erleichtern, gibt es vier "Zwischenprogramme", die ohne jede Kenntnis der Plottergrundsoftware aufrufbar sind (Kurzbeschreibungen in der Programmberatung erhältlich). Es sind dies:

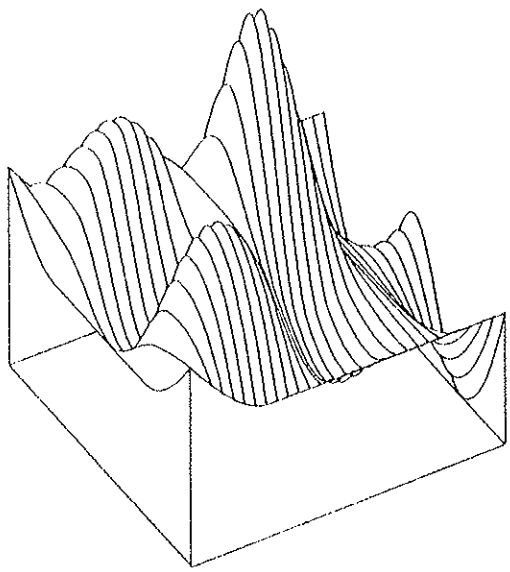
PZUEGE - Schrägriß ohne Interpolation
FIGUR - Schrägriß mit Interpolation
MAPLOT - Projektion mit fixen Blickwinkeln
DREHEN - Projektion mit frei wählbaren Blickwinkeln

Beispiel für eine Zeichnung mit FIGUR:



Beispiel für Zeichnungen mit DREHEN:





F. Blöser

D R A G O N

Die abgebildete "Drachenfigur 12. Ordnung" beweist, daß der EAI 145 Data Plotter sehr gut geeignet ist, auch für mathematische Spiele und Ästhetisches eingesetzt zu werden.

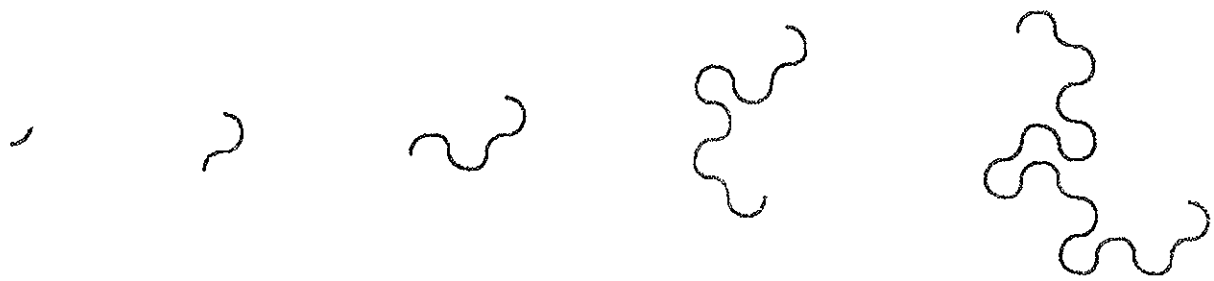
Interessierten Lesern sei kurz das Bildungsgesetz der Drachenfigur vorgestellt. Ursprünglich von J.E. Heighway bei Untersuchungen an Falzlinien speziell gefalteten Papiers entdeckt, wurden dafür bald sowohl eine geometrische Konstruktionsmethode als auch eine Rekursionsformel auf rein digitaler Basis entdeckt.

Die den Drachen beschreibende Formel besteht aus einer Kette von 1 und 0, wobei diesen Werten Viertelbögen nach rechts und links (oder links und rechts) zugeordnet werden. Die Länge des Drachens ist durch seine Ordnung bestimmt.
Der Drache mit der Ordnung 1 hat die Formel 1 und somit die Länge 1. Weiters gilt: hat der Drache der Ordnung N die Länge L, so hat die Ordnung N+1 die Länge 2L+1. Um die Formel für die Ordnung N+1 zu erhalten, werden alle Ziffern der Formel für N angeschrieben, eine 1 angefügt und nochmals alle Ziffern der Formel für N angehängt, wobei die Ziffer im Zentrum der Ordnung N zu invertieren ist.

Beispiel:

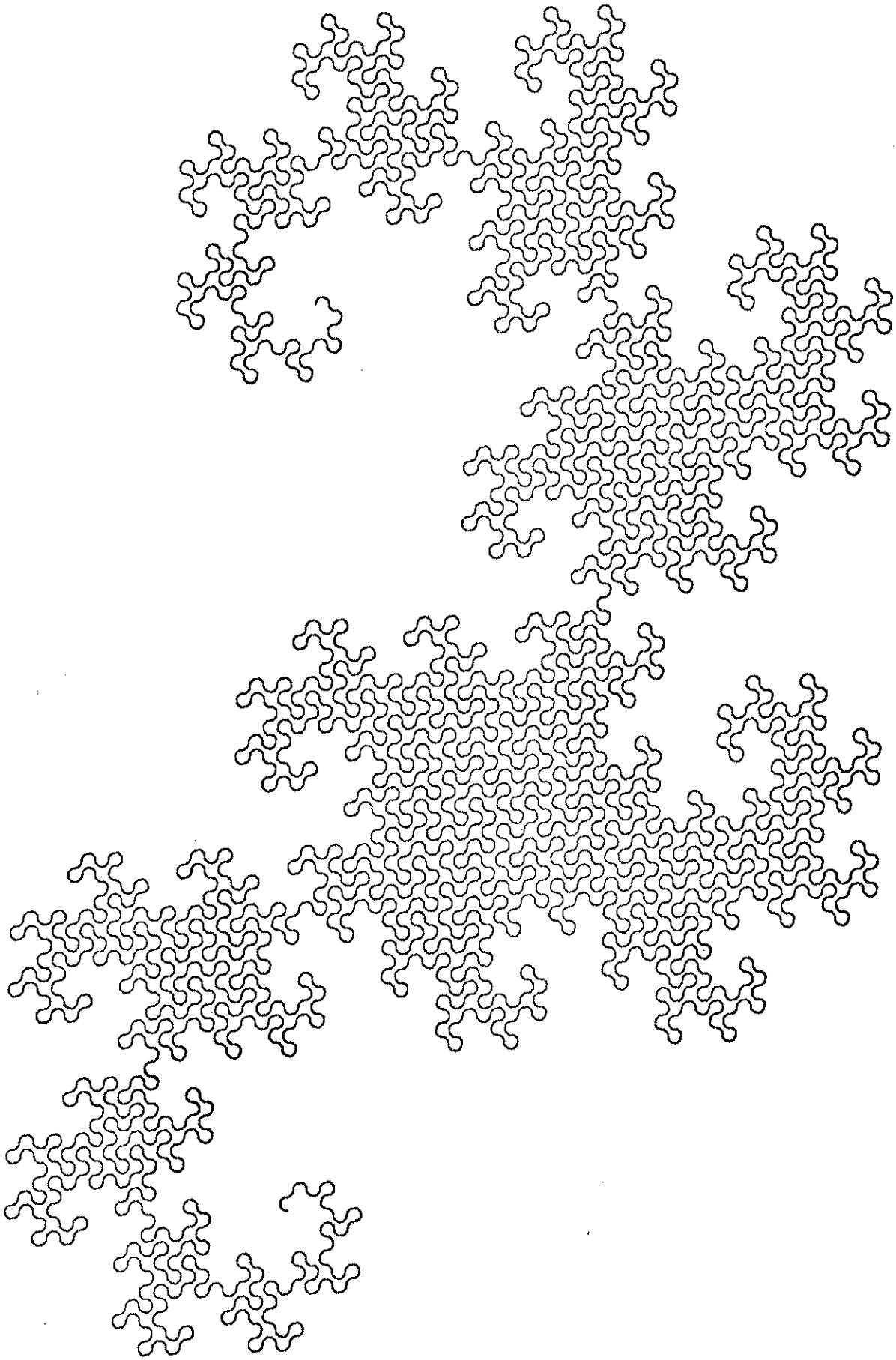
1. Ordnung	1
2. Ordnung	110
3. Ordnung	1101100
4. Ordnung	1101100111001 00
5. Ordnung	11011001110010011101100011001 00

Im folgenden seien die Figuren zu diesen einfachen Formeln wiedergegeben:



Obwohl die Figuren prinzipiell ebensogut von Hand auf Millimeterpapier gezeichnet werden könnten, erweist sich dies doch spätestens ab Ordnung 5 als mühsam und die Berechnung der Formel als kaum mehr fehlerfrei möglich. Die abgebildete Figur 12. Ordnung setzt sich immerhin aus 4095 Viertelbögen zusammen.
Werden Figuren dieser Art an Kopf oder Schwanz verbunden, entstehen durch verschiedene Anfangsblickrichtungen (aber gleiche Anfangspositionen) reizvolle kombinierte Strukturen (Verwendung verschiedener Farbstifte), die alle kreuzungsfrei sind.
Eine der seltsamsten Eigenschaften der Figuren ist, daß die Zentrums-Einser aller Ordnungen kleiner als N, die ja sämtlich in N enthalten sind, auf einer logarithmischen Spirale liegen, deren Zentrum der Anfangspunkt der Gesamtfigur ist.
Das zum Zeichnen der Figuren entwickelte FORTRAN-Unterprogramm gestattet die Wahl der Ordnung, der Anfangslage und -position, der Zeichensequenz und des Kurvenradius der Figuren und steht interessierten Benützern auf Anfrage gerne zur Verfügung.

H. Hummer



NEUE HARDWAREMODIFIKATIONEN IM 693-INTERFACE

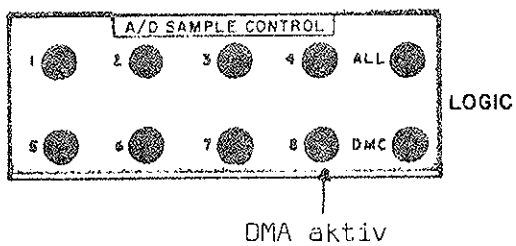
DAS SETZEN VON SENSELINES FUNKTIONIERT JETZT EINWANDFREI

Bei vielen hybriden Synchronisationsproblemen werden die zentralen Frequenzen ($10, 10^2, 10^3, 10^4, 10^5$) am 680 Analogrechner zur Erzeugung von äquidistanten Steuersignalen verwendet. Seit Jahren tauchten aber bei Synchronisation mit Senselines Probleme auf, da diese erst auf Impulse von mehr als 1 µs Dauer ansprachen, während die Impulsbreite der zentralen Frequenzen nur 500 ns beträgt. Bisher mußte jeweils eine Monostabile mit 1.2 µs dazwischengeschaltet werden, um eine saubere Synchronisation zu erreichen.
Eine genaue Analyse des Problems ergab, daß die für Senselines intern verwendeten J-K Flip Flops vom Herstellerdesign her mit derselben steigenden Flanke der zentralen Clock wie die zentralen Frequenzen getriggert wurden. Die Verwendung der fallenden Flanke der zentralen 1MHz Clock des 680-Analogrechners garantiert nun in allen Fällen einwandfreies Setzen der Senselines.

VERBESSERTE SYNCHRONISATIONSMÖGLICHKEITEN BEI HYBRIDEM DMA-TRANSFER

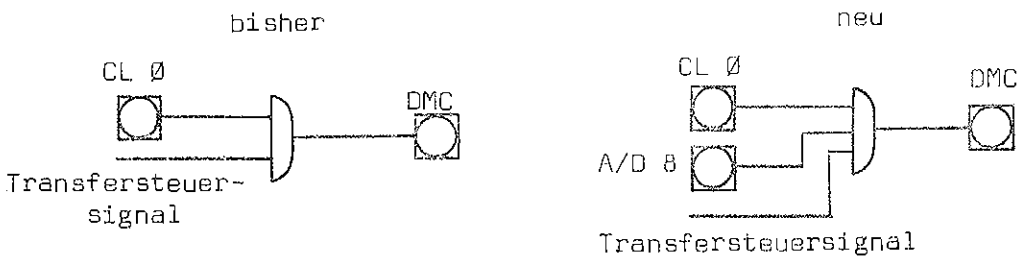
Mit den Systemroutinen TD680 und DF680 kann ein Block von hybriden DMA-Transfers initialisiert werden. Nach Verlassen dieser Routinen ist aber der DMA-Kanal noch nicht unbedingt wirklich zur Datenübertragung bereit, da ein gerade laufender Systemzugriff zu einem Record auf der Fixed-Head-Disk am selben Datenkanal nicht von diesen Routinen abgebrochen oder sein Ende abgefragt wird.
Um allenfalls auftretende Synchronisationsprobleme zu vermeiden, wurde jetzt den Benutzern ein neues Signal am Steckbrett zur Verfügung gestellt. An der Buchse "8" (ehemals Steuer-Eingang für den bei uns nicht existenten A/D Sample Holdverstärker 7) neben dem DMC-Hole im Logikteil liegt jetzt genau dann eine logische "1" an, wenn der DMA1 mit einem hybriden Datentransfer aktiv ist.

Buchsen am Steckbrett:



Beispiel für sichere Synchronisationen:

```
CALL  TD680 (...)
CALL  SCL (0)
```



W. Kleinert

EIGENENTWICKLUNG: SWITCH PANEL ZUR REALISIERUNG VERSCHIEDENER RECHNERVERBINDUNGEN MIT VORHANDENER HARDWARE

Im Zuge der Ausweitung des Verbundes des Hybridrechenzentrums mit anderen Instituten und Rechenanlagen wurden bisher mehrere Kabelverbindungen geschaffen, die unserem Rechenzentrum den Anschluß an andere Anlagen bieten und anderen Instituten erlauben, ihre Terminals an der Hybridrechenanlage zu betreiben.

Der Gedanke lag nahe, durch Zusammenfassung sämtlicher Kabel für Asynchron-Verbindungen in einer ausbaufähigen Schaltmatrix den Teilnehmern am Verbund völlig neue Möglichkeiten zu geben. Das folgende Schaltschema (Abb.1) zeigt sämtliche Möglichkeiten des entwickelten "Switch Panels", wobei Kreise umschaltbare Devices, Terminals oder Rechner symbolisieren, Rechtecke die nicht schaltbaren. Fett gezeichnete Schalterstellungen bedeuten "no connection", diejenigen, von denen keine Linien weiterführen, sind noch nicht belegt.

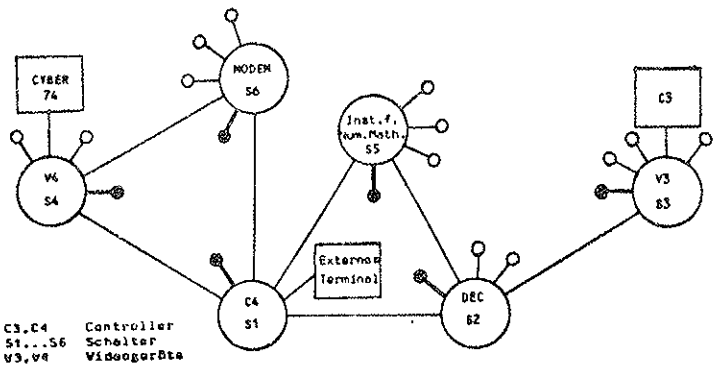
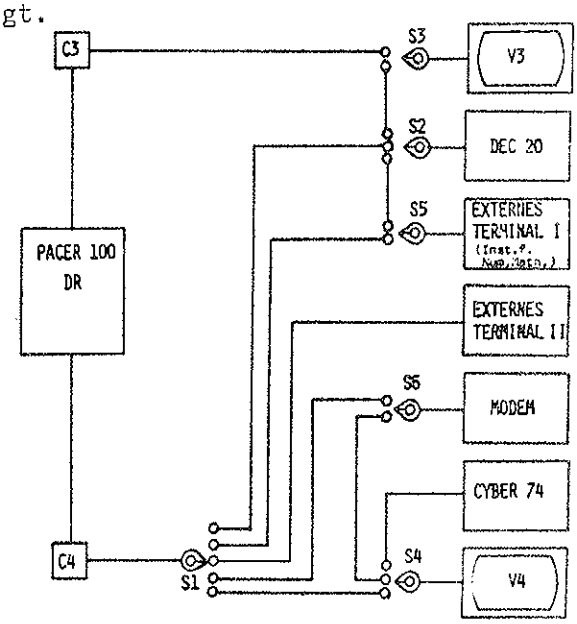


Abb. 1



LOGISCHE SCHALTSKIZZE IM GERÄTEVERBUND

Abb. 2

GRUNDGEDANKEN ZUR SCHALTSTRUKTUR

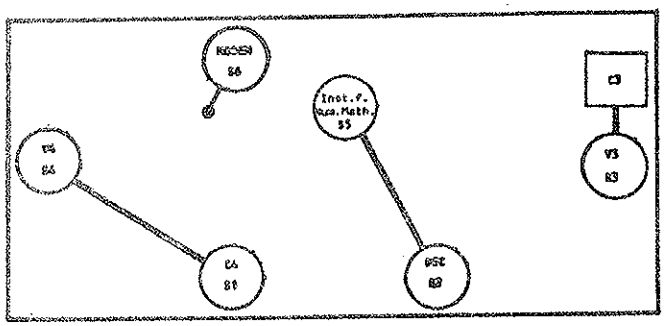
Ursprünglich bestand die Möglichkeit, Video Nr. 4 (V4) von Controller 4 (C4) des Digitalrechners PACER 100 abzuschalten und über eine Kabelverbindung an ein asynchrones Board der CYBER 74 zu legen. Diese Aufgabe bewältigt nun Schalter 4 (S4). Als eine weitere Kabelverbindung zum Institut für Numerische Mathematik gelegt wurde, mußte auch diese bei den Schaltmöglichkeiten vom C4 berücksichtigt werden. Diese Aufgabe bewältigt S1, sodaß das Institut nun direkt über Terminal an der Hybridrechenanlage arbeiten kann. Um Controller 4 an die verschiedenen Einsatzmöglichkeiten optimal anpassen zu können, wurde er mit einer selbstentwickelten programmierbaren Multibaudrateneinstellung versehen (300,1200,2400,4800 Bd). Eine weitere Verbindung zum Institut für Regelungstechnik mit Hilfe eines dafür reservierten Controllers ist an diesem Schaltknoten jederzeit leicht ins Gesamtsystem integrierbar. Bei der Anschaltung eines Modems (300 Bd) zur Datenfernübertragung über das Telefonnetz bestand der Wunsch, das Gerät sowohl im stand-alone Betrieb mit V4 bei der Benützung der Ports eines anderen Rechners (CYBER 74, CYBER 73 an der Universität, Konzentrator des RECON-Systems Frascati/Italien in Darmstadt) als auch zur Vorschaltung vor C4 des PACER zu verwenden, sodaß andere Terminals über

Telefonleitungen bedient werden können. Diese Umschaltung erledigt S6. Arbeitet V4 mit dem Modem, kann C4 durchaus für andere Zwecke eingesetzt werden. Eine weitere Ausbaumöglichkeit ergab sich durch das Verlegen einer Kabelverbindung zwischen Prozeßrechner DEC 20 und CYBER 74. Entsprechende Signale wurden am Verteiler der Digitalrechenanlage zu noch unbenützten Adern des bereits existierenden Kabels zur Hybridrechenanlage durchgeschleift und gestatten nun den Betrieb von V3 als Terminal des Prozeßrechners sowie die direkte Verbindung Prozeßrechner - PACER 100 (Schalter S2). Um V3 sowohl im normalen Betrieb an C3 betreiben als auch auf DEC 20 umschalten zu können, benötigt man S3.

Da das Institut für Numerische Mathematik zwar bereits über eine Verbindung zur CYBER 74, nicht aber über eine solche zur DEC 20 und zur Hybridrechenanlage verfügte, konnten diese durch Einbau von S5 über das Kabel der Hybridrechenanlage zur Verfügung gestellt werden. Diese logische Schaltstruktur wird in Abb.2 wiedergegeben.

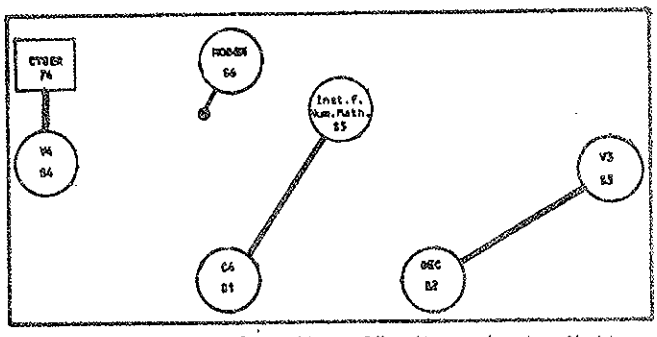
Die Beispiele 1 und 2 zeigen typische Einsatzfälle des Panels.

Beispiel 1:



Normale Rechnerkonfiguration plus einer Leitung vom Institut für Numerische Mathematik zur DEC 20

Beispiel 2:



Das Terminal am Institut für Numerische Mathematik ist über den Controller 4 am PACER angeschlossen. Ein Terminal ist zur CYBER und eines zur DEC 20 geschaltet.

TECHNISCHE REALISIERUNG

Was die technische Realisierung betrifft, so entsprechen die Kreise der Schemazeichnung Stufenschaltern mit mehreren Ebenen, die den durch die Beschriftung angegebenen Geräten zugeordnet sind. Die Zahl der Ebenen ist durch die Zahl der Signale bestimmt, die pro Verbindung zugleich geschaltet werden müssen. Es sind dies mindestens TX (Transmit), RX (Receive), DTR (Data Terminal Ready) und Ground.
Alle benötigten Kabel wurden im Kabelboden zu einem Schaltkasten geführt und an eine 50-polige Steckerleiste geschaltet. Da diese 50 Pole mehrfach ausgeführt sind, können an dieser Stelle mit Steckerschuhen bereits feste Verbindungen definiert werden. Sämtliche Signale werden dann über ein Flachbandkabel zu dem in einer für den Operator günstigen Sichtposition montierten Switch Panel geführt. An diesem befinden sich neben den bereits erwähnten Stufenschaltern noch Indikator-Leuchtdioden, die den Aufbau geschalteter Verbindungen signalisieren. Zudem sind sämtliche Signale an Normsteckbuchsen an der Frontseite des Panels abgreif- und steckbar, und die Anordnung der Schalter wurde kreuzungsfrei und ausbaubar vorgenommen.
Die 200 UT Emulation mit JCS/TS über den CCM steht weiterhin parallel zur asynchronen 300 Baud Verbindung zur Verfügung.

AUSBLICK

Das Schaltsystem ist einfach an alle zukünftigen Aufgaben anzupassen und zu erweitern, da es sowohl vom Schaltprinzip als auch vom elektronischen Aufbau her zur Gänze modular ausgelegt ist.

H. Hummer

PROGRAMMIERUNG DES REGENT 1ØØ DATENSICHTGERÄTS

Das REGENT 1ØØ Datensichtgerät ist als Video 2 im Betriebssystem JCS/TS 7 programmierbar. Die Bildschirmgröße beträgt 24 Zeilen zu je 8Ø Zeichen. Die Einheitennummer für Ein/Ausgabe mit FORTRAN READ/WRITE ist 26.

Außerdem steht ein Programmpaket REGENT zur Verfügung, das in Anlehnung an das Softwarepaket CURSOR (für das EAI C1Ø1 Video) geschrieben wurde und ein bequemes Bedienen des Videos ermöglicht:

- Löschen des Bildschirms, Löschen einzelner Zeilen,
- Positionieren des Cursors,
- Umschalten on-line/off-line,
- Ein/Ausgabe von ASCII-Zeichen,
- Programmierung graphischer Strukturen,
- Steuerung des Hintergrunds und der Intensität.

Im folgenden sind die Funktionen der FORTRAN-aufrufbaren Unterprogramme des Programmpakets REGENT kurz beschrieben. Detaillierte Programmbeschreibungen sind in der Programmberatung erhältlich. Zu einem Programm, das die angegebenen Unterprogramme aufruft, muß eine LOAD,D9 REGENT Steuerkarte verwendet werden.

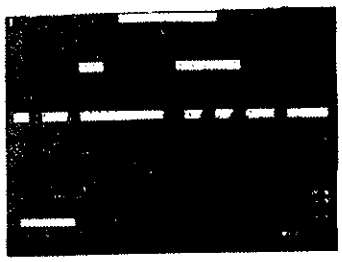
ALPHANUMERISCHE PROGRAMMIERUNG UND KONTROLLPROGRAMME

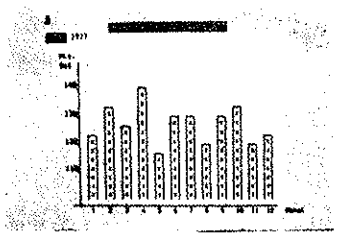
Programm	Funktion
CLEAR	Löschen des gesamten Bildschirms
CLEARL	Löschen der restlichen Zeile } ab der Cursor-
CLEARP	Löschen des restlichen Bildschirms } position
HOME	Löschen des Bildschirms, der Cursor wird in die linke obere Bildschirmecke versetzt
LOCAL	Versetzen des Videos in den LOCAL-Mode
REMOTE	Versetzen des Videos in ON LINE-Mode
MARK	Absolute Positionierung des Cursors
CURSOR	Relative Positionierung des Cursors

Programm	Funktion
STAR	Ausgabe eines Zeichens im Hollerithformat
HERE, GET	Einlesen des an der Position des Cursors befindlichen Zeichens
RLINE	Einlesen einer Zeile
RPAGE	Einlesen einer Seite (gesamter Bildschirm)
REGENT	Angabe der momentanen Position des Cursors
RING	Pfeifsignal
HCOPY	Ausdrucken des Videobildes am Line Printer

GRAPHISCHE PROGRAMMIERUNG

Alle visuellen Merkmale des REGENT 1ØØ wie Umkehrbarkeit des gesamten Bildschirms oder einzelner Felder, Halb- oder Vollintensität, Blinken und Unterstreichen werden vom Programmpaket REGENT softwaremäßig unterstützt. Die Abbildung zeigt ein Beispiel für eine übersichtliche Gliederung des Bildschirms durch graphische Zeichen und Feldgestaltung sowie ein Beispiel für eine Histogrammdarstellung.





Programm	Funktion
VIDEO	Gestaltung eines Feldes am Bildschirm beginnend bei der momentanen Position des Cursors
FIELD	Gestaltung eines absolut positionierten Feldes
DRAW	Ausgabe graphischer Zeichen mit Wahl der visuellen Darstellung

Als graphische Zeichen können folgende Zeichen programmiert werden (die rechteckige Umrahmung stellt den Bereich einer Cursorposition dar):

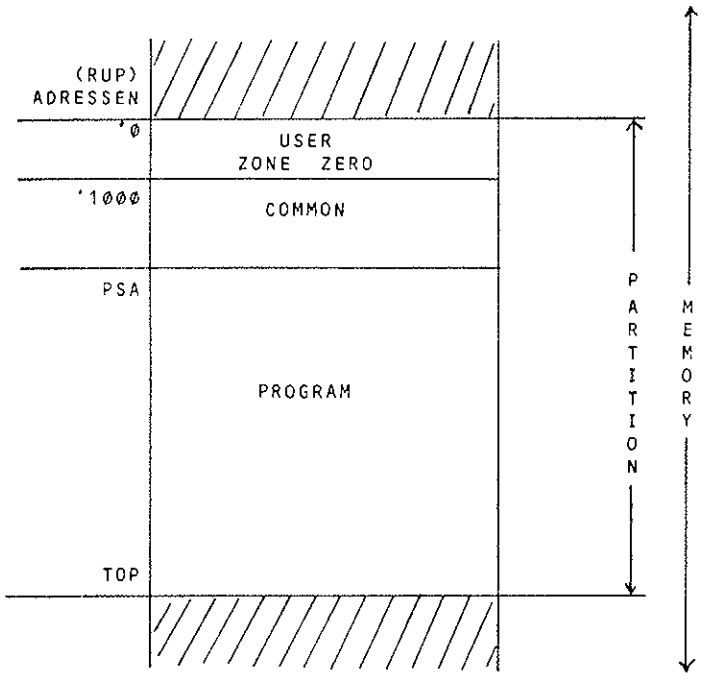
I. Husinsky

LOKALISIEREN VON PROGRAMMFEHLERN ANHAND VON ADRESSEN UND MAP-LISTEN

EINLEITUNG

Bei der Durchführung eines Jobs an der Hybridrechenanlage unterstützt das Betriebssystem den Benützer beim Auftreten von Fehlern bzw. Spezialfällen durch entsprechende Meldungen, die im Line-Printer-Ausdruck an der Stelle des Auftretens in logisch richtiger Reihenfolge den vom Benützer programmierten Ausdruck unterbrechen. Je nachdem, wie schwerwiegend der Fehler ist, wird der laufende Task nach der entsprechenden Meldung fortgeführt oder abgebrochen. Bei einem Abbruch wird mit dem Task der nächsten Steuerkarte (z.B. /RUN...) fortgesetzt.
Vor diesen Meldungen werden Adressen ausgedruckt, die auf die Position des Auftretens des Fehlers im Programmablauf hinweisen können und dadurch die Fehlersuche konkreter und schneller gestalten können. Um diese Adressen interpretieren zu können, sei hier auf die Übersetzungs-, Binde-, und Ladestruktur normaler Benützer-Tasks eingegangen.

SPEICHEREINTEILUNG



EINTEILUNG EINER USER PARTITION

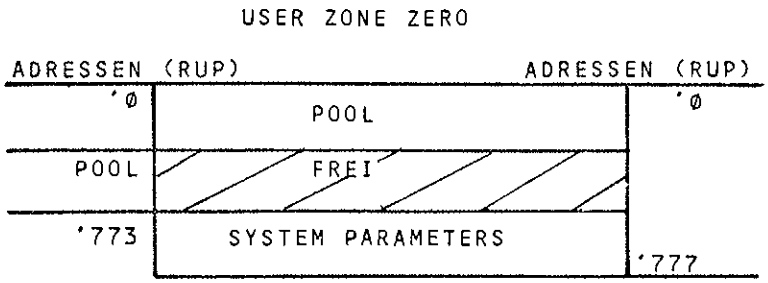
Bevor ein Programm durchgeführt wird (/RUN...), muß es übersetzt (/FOR... bzw. /ASM...), anschließend mit allen benötigten Unterprogrammen und Programmen der Bibliothek gebunden (/BUILD...) und danach in das Memory geladen werden. Sowohl beim Übersetzen als auch beim Binden eines Programmes erhält man im Standardfall MAP-Listen ausgedruckt, die Adressen der Programmstrukturen enthalten. Mit den MAP-Listen der FORTRAN-Programme und der MAP-Liste des gebundenen Tasks (/BUILD) kann man nun eine auftretende Fehlermeldung lokalisieren, wobei alle Adressen oktal angeschrieben werden.

Ein fertig gebundenes Programm wird durch /RUN in eine Partition des Memorys geladen. Diese Partition wird vom Betriebssystem JCS/TS zugeteilt und variiert im allgemeinen. Alle Programmadressen werden jedoch relativ dieser zugeteilten Partition (RUP-Adressierung) notiert, sind also von dieser Wahl unabhängig. Der restliche Teil des Memorys spielt für das Programm keine Rolle und steht im Rahmen des Multiprogrammings anderen Benützern zur Verfügung.

Der gesamte in die Partition geladene Task setzt sich aus drei unterschiedlichen Strukturen zusammen:

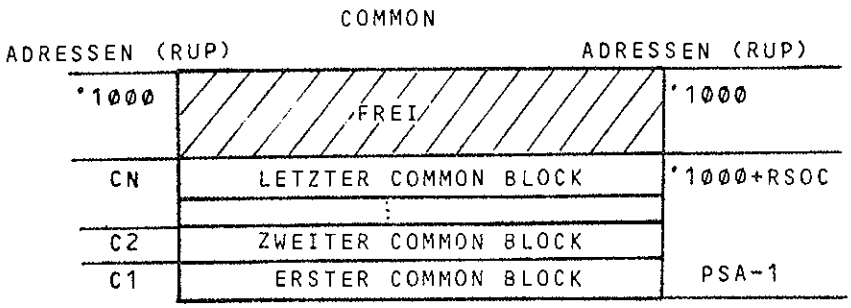
- Die konstant große USER ZONE ZERO, die im allgemeinen vom Benützer nicht angesprochen wird, in der sich im wesentlichen Systeminformationen befinden, um die Durchführung des Programms zu gewährleisten, reicht von 'Ø bis '777.
- Ab der Adresse '1ØØØ beginnt der COMMON-Bereich, der dem Benützer programmäßig zur Verfügung steht. Die Größe des COMMON-Bereiches (COM) ist durch die vom User auf der /BUILD-Steuerkarte angegebene COMMON-Option bereits beim Binden des Programmes definiert. Ohne diese COMMON-Option wird ein Standardwert (derzeit COM = 2ØØ) angenommen.
- Anschließend an den COMMON-Bereich gliedert sich der eigentliche PROGRAM-Bereich mit den Programmen und Unterprogrammen. Dieser PROGRAM-Bereich reicht von der durch die COMMON-Option definierten Programm-Startadresse PSA = '1ØØØ+COM bis zum Ende der Partition TOP. Diese Adresse TOP wird vom Betriebssystem so gewählt, daß der gebundene Task noch in die Partition geladen werden kann. USER ZONE ZERO, COMMON-, und PROGRAM-Bereich lassen sich jedoch noch differenzierter gliedern.

POOL-BEREICH



Im untersten Teil der USER ZONE ZERO befindet sich der POOL-Bereich, der von Adresse 'Ø bis POOL-1 reicht. Dieser Grenzwert POOL wird auf der MAP-Bindeliste (/BUILD) nach POOL: ausgegeben. Im POOL-Bereich befinden sich alle Unterprogrammadressen (und zwar deren ENTRY POINTS, das sind die Adressen der Unterprogramme, bei denen die Durchführung beginnt), sowie die Adressen der programmierten COMMON-Blöcke. Die Namen der Unterprogramme und COMMON-Blöcke sind ebenfalls auf der MAP-Bindeliste mit den jeweiligen Adressen angegeben.
Im oberen Teil der USER ZONE ZERO befinden sich die SYSTEM PARAMETERS und zwar im konstanten Bereich der Adressen '773 bis '777. Hier sind die Werte der PARAMETER- und DISTRICT-Option vermerkt, sowie die Adresse der Partition selbst (absolut im Memory) und die Adressen TOP und PSA für das geladene Programm. Der Bereich der Adressen POOL bis '772 ist im allgemeinen frei, wobei der Bereich '742-'772 von manchen Bibliotheksprogrammen verwendet wird.

COMMON-BEREICH

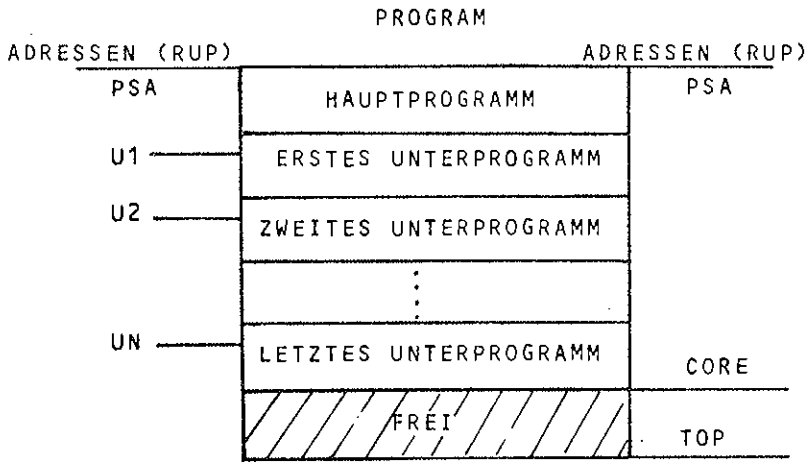


Der von der Adresse '1ØØØ an beginnende COMMON-Bereich wird von oben her belegt und zwar von den programmierten COMMON-Blöcken in der Reihenfolge ihres Auftretens beim Binden. Die Größe der COMMON-Blöcke

richtet sich nach der Deklaration im Programm und die Startadresse jedes COMMON-Blockes C1, C2, ... CN ist in der MAP-Bindeliste mit dem jeweiligen Namen vermerkt (für den BLANK COMMON-Block wird hier .C geschrieben).

Der verbleibende Platz im COMMON (RSOC) ist frei und wird ebenfalls auf der MAP-Bindeliste ausgedruckt. Die Startadresse des letzten COMMON-Blocks und daher erste Adresse im COMMON-Bereich ergibt sich daher aus CN = '1ØØØ+RSOC.

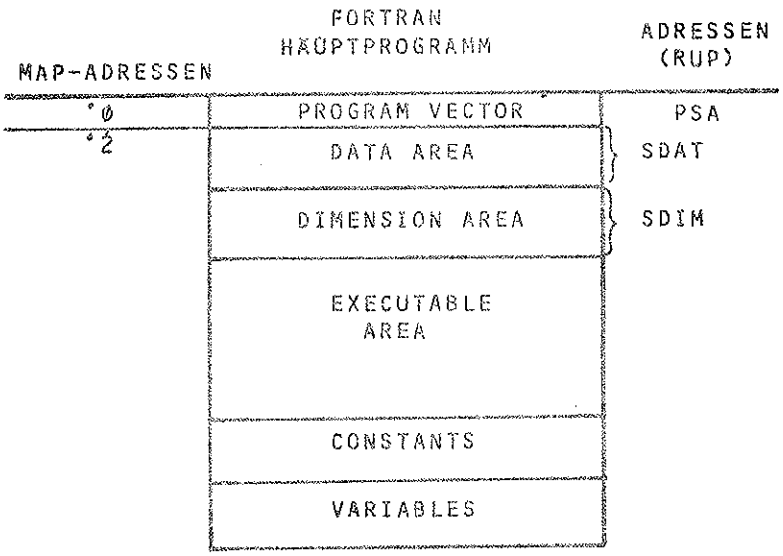
PROGRAM-BEREICH



Der PROGRAM-Bereich selbst wird von der kleinsten Adresse beginnend (PSA) durch das Hauptprogramm und daran folgend von allen Unterprogrammen, die zum Binden des Programmes benötigt werden, belegt. Diese Unterprogramme werden in der Reihenfolge ihres Antreffens auf den Files, in der Reihenfolge der jeweiligen LOAD-, FORCE-, bzw. SPECIAL-Karten beim Binden erstellt. Die letzte dabei benötigte Partition-Adresse CORE wird auf der MAP-Bindeliste ausgegeben. Der Bereich von den Adressen CORE+1 bis TOP bleibt systembedingt frei und variiert in seiner Dimension von 'Ø bis '377.
Innerhalb eines Unterprogrammes gibt es zumindest einen ENTRY POINT (FORTRAN); Diese Adressen der ENTRY POINTS (U1,U2,...UN) sind mit den entsprechenden Routine-Namen in der MAP-Bindeliste angegeben. Diese MAP-Bindeliste ist nach der Belegung in der Partition, mit den COMMON-Blöcken beginnend und den Unterprogrammen folgend, geordnet.
Während bisher für die groben Strukturen die Adressen der MAP-Bindeliste für die einzelnen Positionen innerhalb der Partition heranzu-

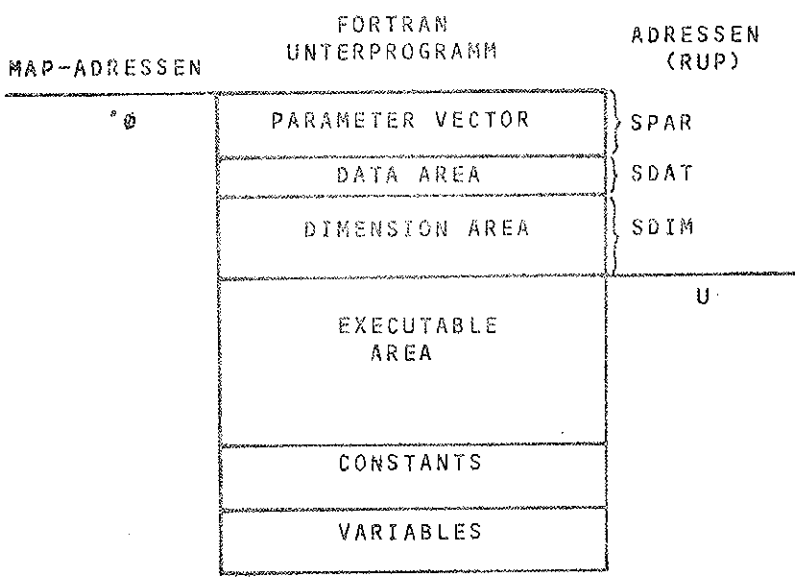
ziehen sind, werden bei dem detaillierten Aufbau eines FORTRAN-Haupt- oder Unterprogrammes durch die der FORTRAN-Liste anschließende zweiteilige FORTRAN-MAP-Liste die Adressen innerhalb dieses FORTRAN-Programmes, also relativ vom Ladepunkt innerhalb der Partition notiert. Alle Programme werden entsprechend ihrer Programm- und Datenstruktur nach aufsteigenden Adressen aufgebaut.

FORTRAN-HAUPTPROGRAMM



Beim Hauptprogramm beginnt die Zählung mit der MAP-Adresse 'Ø an der Adresse PSA. Hier befindet sich der PROGRAM VECTOR, eine Information, die zum eigentlichen Hauptprogramm-Beginn in der EXECUTABLE AREA verzweigt.
Anschließend mit der relativen Adresse '2 beginnt die FORTRAN-Programmstruktur, die genauso bei einem FORTRAN-Unterprogramm anzutreffen ist.

FORTRAN-UNTERPROGRAMM



Das FORTRAN-Unterprogramm beginnt allerdings mit dem PARAMETER VECTOR, einer Zone, in der die Adressen der Parameter des Unterprogrammes abgespeichert sind. Daher berechnet sich die Länge des PARAMETER VECTORs (SPAR), der bei der relativen Adresse 'Ø des Unterprogramms beginnt, genau aus der Anzahl der Parameter.
Die DATA AREA enthält alle durch DATA-Befehle konstant definierten Variablen, die in ihrer programmierten Reihenfolge generiert werden. Die relativen Adressen dieser DATA-Variablen erhält man unter deren programmierten Namen im zweiten Teil der FORTRAN-MAP-Liste.
Anschließend an die DATA AREA grenzt die DIMENSION AREA, in welcher die Felder der durch die DIMENSION-Deklaration definierten Variablen aufgebaut sind. Die Reihenfolge der Felder richtet sich wieder nach dem Auftreten in der DIMENSION-Deklaration, die relativen Startadressen der Felder findet man wieder zusammen mit dem Feldnamen im zweiten Teil der FORTRAN-MAP-Liste. Allgemein sind diese Felder unbestimmten Inhaltes. Bei der Anlage mehrdimensionaler Felder variiert zuerst der erste Index. Variable bzw. Felder, die als COMMON deklariert sind, befinden sich in ihrer definierten Reihenfolge in einem der vorher erwähnten COMMON-Blöcke. Ihre ebenfalls im zweiten Teil der FORTRAN-MAP-Liste angegebenen relativen Adressen beziehen sich daher auf den Beginn des jeweiligen COMMON-Blockes C1, C2, bzw. CN.
Die in der MAP-Bindeliste eingetragene ENTRY-POINT-Adresse U dieses Unterprogrammes befindet sich genau am Beginn der EXECUTABLE AREA, an der mit der eigentlichen Unterprogrammdurchführung begonnen wird. Ab hier sind die eigentlichen Befehle des Programmes bzw. Unterprogrammes codiert und hier erfolgt auch das Ende des Programmes bzw. der Rücksprung in das ein Unterprogramm rufende Programm. Lokalisierte Adressen von Fehlermeldungen müssen sich demnach immer in der EXECUTABLE AREA eines Programmes befinden. Findet man solch eine Adresse außerhalb, deutet dies bereits auf einen schwerwiegenden strukturellen Programmfehler hin (Indexüberläufe bei Feldern, Fehlsprünge durch falsche Parameterlisten, etc.).
Anschließend an die EXECUTABLE AREA eines Programmes werden in CONSTANTS alle im Programm auftretenden Konstanten generiert. Diese werden im ersten Teil der FORTRAN-MAP-Liste ausgegeben.
Daran gliedern sich die VARIABLES, das sind alle weiteren benötigten, deklarierten bzw. implizit typisierten Variablen, die im Laufe des Programmes vorkommen, und zwar in der Reihenfolge ihres syntaktischen Auftretens. Dazu gehören auch vom FORTRAN-Compiler benötigte Hilfsvariable. All diese Variablen sind zusammen mit ihren relativen Adressen ebenfalls im zweiten Teil der FORTRAN-MAP-Liste zu finden.
Ebenso finden sich hier alle Befehlsnummern des Programms (mit voran-

gehendem Punkt) und ihre Adressen, durch die die EXECUTABLE AREA weiter unterteilt werden kann. Unterprogrammnamen werden mit X anstelle einer Adresse angegeben, bei nicht existierenden, aber angesprochenen Befehlsnummern wird ein U statt der Adresse ausgegeben.

FEHLERSUCHE

Durch die dem Programmierer bekannten Werte und Dimensionen von Parameteranzahl, DATA-Befehlen sowie DIMENSION-Deklarationen sind die Größen des PARAMETER VECTORs, der DATA AREA bzw. der DIMENSION AREA leicht zu bestimmen (ohne COMMON!). Hiebei ist zu beachten, daß INTEGER, LOGICAL, SCALED FRACTION eine, REAL zwei, COMPLEX, DOUBLE PRECISION vier Memory-Zellen, also ebensoviele Adressen belegen. Aus der Kenntnis dieser Dimensionen läßt sich auch der Zusammenhang zwischen der aus der MAP-Bindeliste gewonnenen Adresse U des ENTRY POINTs eines Unterprogrammes und der Position der relativen MAP-Adresse Ø herstellen.

Die Adressangabe bei einer Fehlermeldung selbst ist RUP, also relativ der gesamten Partition, laßt sich also vorerst nur mit den Adressen der MAP-Bindeliste in Relation bringen. Dort sucht man den vorangehenden ENTRY POINT und erhält den Namen des Programmes, in dem der Fehler aufgetreten ist. Ist dieser Name bereits der eines COMMON-Blockes, so ist der Fehler im Hauptprogramm aufgetreten. Nach dieser ersten Eingrenzung rechnet man die Fehleradresse auf die relativ vom Ladepunkt rechnende Adresse, die MAP-Adresse, um, welche dann mit den in der FORTRAN-MAP-Liste angegebenen Adressen in gleiche Relation gebracht werden kann (z.B. Statement-Nummern). Diese Umrechnung ist bei Haupt- und Unterprogramm verschieden: Beim Hauptprogramm muß man für die Fehlerposition relativ der MAP-Adressen (FMAP) von der ausgedruckten Fehleradresse (FRUP) den Wert der Startadresse des Hauptprogrammes (PSA) subtrahieren:

$$FMAP = FRUP - PSA$$

wobei PSA ='1ØØØ + COM mit COM = COMMON-Option. Beim Unterprogramm ist man genötigt, die Größen des PARAMETER VECTORs (SPAR), der DATA AREA (SDAT) sowie der DIMENSION AREA (SDIM) zu ermitteln, um die Beziehungen herzustellen:

$$FMAP = (FRUP - U) + (SPAR + SDAT + SDIM)$$

wobei U die Adresse des ENTRY POINTs des entsprechenden Unterprogrammes ist.

Enthält ein Programm eine Statement-FUNCTION, so wird diese genau so wie ein FORTRAN-Unterprogramm direkt zwischen DIMENSION AREA und EXECUTABLE AREA einkodiert, also mit einem PARAMETER VECTOR.
Falls die Lokalisierung eines Fehlers allein anhand der Statement-Nummern unzureichend erscheint, können mithilfe der SYMBOL-Option auf der /FOR-Steuerkarte die vom FORTRAN-Compiler generierten Befehle mit ihren Adressen ausgedruckt werden. Diese Adressen haben den gleichen Bezugspunkt wie die MAP-Adressen. Die INTEGER FUNCTION IADR (VAR) liefert die Adresse (RUP) einer Variablen oder Konstanten VAR.

BEISPIEL

Hauptprogramm:

```
PAGE  1        RUN/AK   FORTRAN COMPILER  7.6  JCS/TS

C       HAUPTPROGRAMM LIEST SIEBEN WERTE
        INTEGER IWERT(7)
        DATA INPUT/5/
        READ(INPUT,100) IWERT
        CALL DIST(IWERT,7)
        CALL EXIT
  100   FORMAT(7I10)
        END


34    $36      OCT   7

 3      IWERT      2      INPUT      30     .100       X      DIST       X      EXIT
```

Unterprogramm:

```
PAGE  1        RUN/AK   FORTRAN COMPILER  7.6  JCS/TS

C       UNTERPROGRAMM BESTIMMT WURZEL AUS ABSTAND ZUM MITTEL
        SUBROUTINE DIST(X,N)
        INTEGER X(1)
        REAL D(7)
        DATA IOUT/6/
        IS=0
        WRITE(IOUT,100) (X(K),K=1,N)
        DO 1 I=1,N
    1   IS=IS+X(I)
        IS=IS/N
     10 D(1)=SQRT(FLOAT(IS-X(1)))
     20 D(2)=SQRT(FLOAT(IS-X(2)))
     30 D(3)=SQRT(FLOAT(IS-X(3)))
     40 D(4)=SQRT(FLOAT(IS-X(4)))
     50 D(5)=SQRT(FLOAT(IS-X(5)))
     60 D(6)=SQRT(FLOAT(IS-X(6)))
     70 D(7)=SQRT(FLOAT(IS-X(7)))
    300 FORMAT(7F10.3)
    100 FORMAT(7I10)
        WRITE(IOUT,300) (D(K),K=1,N)
        RETURN
        END


407   $5       OCT   1
410   $106     OCT   0

 0      X          1      N          3      D          2      IOUT       411    IS
 351    .100       412    K          413    IU1000     67     .1         414    I
 121    .10        X      SQRT       X      FLOAT      415    IU1010     416    .U2020
 145    .20        172    .30        217    .40        244    .50        271    .60
 316    .70        344    .300
```

Bindeliste:

```
RUN/AL  CORE IMAGE GENERATOR  7.10  JCS/TS

RUN/AK 13 LD
JCSRTL 01 LD
JCSLIB 01 DN

POOL: 000023 RSOC: 000200 CORE: 004000 FREE: 028670

  DIST 01366  FLOAT 01765  .C12 01774  SQRT 02007  .L22 02172  .M22 02207
  .A22 02221  .S22 02237  .M22 02255  .D22 02273  .N22 02320  .L11 02330
  .L33 02330  .H11 02335  .H33 02335  .A11 02342  .S11 02357  .M11 02405
  .D11 02437  .N11 02460  .A33 02465  .M33 02472  .FIO 02477  .FAR 03036
  .FCB 03043  .FDT 07072  .HRW 07112  .FER 07142  .CKS 07152  .FLT 07160
SYS040 07201 SYS034 07476  .FWR 07634  .FRE 07671  .FAT 07711 RTLERR 07749
  EXIT 07766
```

Execution 1:

```
      12        45        23        56        89        45        43
 1533 SQUARE-ROOT OF NEGATIVE ARGUMENT
```

Execution 2:

```
       1         2         3         4         5         6         7
 1632 SQUARE-ROOT OF NEGATIVE ARGUMENT
```

Fehlerlokalisierung:

Der Fehler tritt im ersten Beispiel bei der Adresse FRUP = '1533 auf. Anhand der MAP-Bindeliste stellt sich heraus, daß der vorangehende ENTRY POINT, der von DIST mit der Adresse U = '1366 ist.

In der Routine DIST hat der PARAMETER VECTOR die Größe SPAR = '2 (2 Parameter), die DATA AREA die Größe SDAT = '1 (eine INTEGER Größe) und die DIMENSION AREA die Größe SDIM = '16 (ein lokales REAL-Feld der Größe '7).

Aus FMAP = (FRUP-U) + (SPAR + SDAT + SDIM) für dieses Unterprogramm ergibt sich FMAP = ('1533 - '1366) + ('2+'1+'16) = '145+'21 ='166. Der Fehler tritt also relativ der FORTRAN-MAP-Liste von DIST an der Adresse '166 auf, also zwischen den Adressen von '145 für .2Ø und '172 von .3Ø, das heißt zwischen den Positionen der Befehlsnummern 2Ø und 3Ø.
Tatsächlich bringt die Auswertung der Wurzel von D(2) im ersten Beispiel den Fehler. Das zweite Beispiel ist analog zu behandeln.

A. Blauensteiner

POINT - EIN NEUES PROGRAMMPAKET ZUR NUMERISCHEN INTEGRATION

KONZEPT

Zur numerischen Berechnung bestimmter Integrale

$$\int_a^b f(x)dx$$

gibt es auf Hybridrechnern zwei Möglichkeiten - man kann entweder den Analogrechner oder den Digitalrechner verwenden (auch Kombinationen sind möglich, wenn etwa die Berechnung der Werte des Integranden f am Analogrechner und die Integration am Digitalrechner erfolgt). Das Programmpaket POINT (=Package for Onsdimensional INTegration) ist für jene Fälle gedacht, wo die Berechnung am Analogrechner z.B. aus einem der folgenden Gründe ausscheidet:

a) es wird ein Näherungswert benötigt, dessen Genauigkeit größer sein soll als jene Genauigkeit, die am Analogrechner erzielbar ist (am EAI 680 Analogrechner sind das ca. 4 Dezimalstellen)

b) die Werte des Integranden f im Integrationsintervall [a,b] weisen sehr unterschiedliche Größenordnungen auf (ein Extremfall liegt dann vor, wenn f Singularitäten an den Stellen a, b oder im Inneren von [a,b] aufweist).

Wegen a) wurden alle Programme von POINT für doppelt-genaue Rechnung ausgelegt, was auf dem EAI PACER 100 einer Rechengenauigkeit von etwa 16 Dezimalstellen entspricht.

Die Auswahl der Algorithmen und die Erstellung der Programme erfolgte entsprechend den folgenden Gesichtspunkten:

a) Die Programme sollen jene Probleme besonders effizient lösen, die sich aus den technisch-naturwissenschaftlichen Problemstellungen des Benutzerkreises des EDV-Zentrums der TU-Wien ergeben. Die Bewertung der Programme hinsichtlich dieser Forderung konnte auf Grund der Ergebnisse einer Studie über das Benutzerprofil von Quadraturprogrammen am Digitalrechenzentrum der TU-Wien vorgenommen werden (vgl. z.B. Stetter, Überhuber [1], Schulner, Überhuber [2]).

Aus der obigen Forderung und den Ergebnissen der zitierten Studie ergab sich:

b) Die Programme sind primär als Mehrzweckprogramme auszulegen, wobei die höchste Effizienz bei "glatten" Integranden f erreicht werden soll. Sonderfälle, die (mit verminderter Effizienz) gelöst werden sollen:

1) Randpunktsingularitäten und Singularitäten im Inneren von [a,b] vom Typ x^{α}, $\alpha > -0.9$
2) Oszillierende Integranden vom Typ $\sin \omega x$, $\omega < 50$, $0 \leq x \leq 1$
3) "Verrauschte" ("noisy") Integranden vom Typ $f(x) + \delta(x)$, wobei $\delta(x)$ eine "zufällige" Störung im Sinne einer Zufallsvariablen mit dem Mittelwert $\emptyset$ darstellt (z.B. kann $\delta(x)$ die Fehler symbolisieren, die etwa bei der Berechnung von $f(x)$ am Analogrechner auftreten).
4) Unendliche Integrationsbereiche $(-\infty,b]$, $[a,\infty)$ oder $(-\infty,\infty)$ bei Integranden, die exponentiell oder algebraisch abklingen für $x \to \pm\infty$.

Jene Fälle, die von den Programmen nicht verarbeitet werden können (z.B. $-1 < \alpha < -0.9$ bei 1) sollen dem Benutzer über einen Parameter angezeigt werden.

c) Bedienung: Vom Benutzer sollen nur die absolut notwendigen Informationen vorgegeben werden müssen:

1) der Integrand f in Form eines Funktions-Unterprogramms
2) die Integrationsgrenzen a und b
3) die gewünschte Genauigkeit

$$|I - \int_a^b f(x)dx| \leq \varepsilon_{abs} + \varepsilon_{rel} I$$

wobei I die numerische Näherung bezeichnet

4) eine Rechenaufwandsbeschränkung (max. Anzahl von Aufrufen des "Integranden-Unterprogramms").

Als Ergebnisse soll der Benutzer die folgenden Informationen erhalten:

1) die numerische Näherung I
2) eine Fehlerschranke τ

$$|I - \int_a^b f(x)dx| \leq \tau$$

3) eine Mitteilung (über einen Parameter), ob die Integration erfolgreich abgelaufen ist, bzw. ob und welche Schwierigkeiten das Erreichen der geforderten Genauigkeit verhinderten.

d) Programmiersprache: Standard FORTRAN, das überdies soweit einzuschränken ist, daß die Programme problemlos auf verschiedenen Rechenanlagen zu implementieren sind.

REALISIERUNG DES KONZEPTS

Der obige Forderungskatalog wurde in POINT nicht mit einem Programm, sondern mit mehreren Programmen realisiert. Als Grundlage aller Programme wurden die Gauss'schen Quadraturformeln gewählt (kombiniert mit den entsprechenden Kronrodformeln zur Fehlerschätzung). Diese Formeln werden auf zwei Arten eingesetzt:

1) adaptiv: hier wird die Auswahl der verwendeten Werte von f nach dem "Verhalten" der Funktion getroffen: bei "glatten" Abschnitten von f werden vergleichsweise weniger Werte benötigt als z.B. an Stellen, wo f singulär ist.

2) nicht adaptiv: hier erfolgt die Wahl der verwendeten Werte von f unabhängig von den "lokalen" Eigenschaften des Integranden.

Für den adaptiven Einsatz der Gauss-Kronrod-Formeln wurde ein Steuerungsmechanismus gewählt, der auf einer "globalen" Fehlerschätzung beruht (vgl. Malcolm, Simpson [3]), da diese Art der Steuerung zu den zuverlässigsten Integrationsprogrammen führt (vgl. z.B. Rice [4]).

Für die Verarbeitung von Endpunktsingularitäten und für unendliche Integrationsbereiche wurden eigene Programm-Module vorgesehen, in denen eine Transformation des Integranden vorgenommen wird, so daß der transformierte Integrand keine Singularität aufweist bzw. über ein endliches Intervall integriert werden kann.

Die vier Grundroutinen von POINT sind:

"QNG" (quadrature code, non-adaptive, general purpose)
"QAG" (quadrature code, adaptive, general purpose)
"QAGS" (quadrature code, adaptive, general purpose, with special provision for endpoint singularities)
"QAGI" (quadrature code, adaptive, general purpose, with special provision for an infinite range)

TESTS

Jede der vier Grundroutinen von POINT ("QNG", "QAG", "QAGS", "QAGI") wurde mit mehr als 9000 Testintegranden auf der Rechenanlage CDC CYBER 74 der TU-Wien getestet. Im Vergleich mit den dort implementierten Programmen der NAG- und der IMSL-Bibliothek stellen die Programme von POINT bei manchen Testsituationen (z.B. bei Integranden mit einem steilen, stark ausgeprägten "Gipfel") eine beachtliche Verbesserung dar. Die POINT-Programme waren auch in weniger günstigen Situationen nie schlechter als die NAG- oder IMSL-Programme.

Die Testroutinen (inkl. einem Programm zum Erstellen detaillierter Test-Protokolle) sind im Paket POINT enthalten. Es kann sich also jeder Benutzer auf sehr einfache Weise von der Qualität der POINT-Programme überzeugen, oder zu Vergleichszwecken eigene Quadraturprogramme testen.

BENUTZERKOMFORT

Da die POINT-Grundroutinen frei von Input/Output-Statements sind (die Kommunikation mit dem Benutzer erfolgt ausschließlich über Parameter), bereitet die Analyse jener Fälle, wo die POINT-Programme zu keiner befriedigenden Antwort gelangten, für den Benutzer Schwierigkeiten. Für diese Situationen stehen Interface-Programme zur Verfügung:

"QUICG" (quadrature interface-code for "QAG")
"QUICGS" (quadrature interface-code for "QAGS")

Diese Programme bieten die Möglichkeit, wahlweise Informationen auszudrucken, aus denen ersichtlich ist, ob die Berechnung problemlos zu Ende geführt werden konnte

```
**********************************************************************
                                                  CODE    0.00000D-01

NUMERICAL CALCULATION OF A ONE-DIMENSIONAL
DEFINITE INTEGRAL WITH SUBROUTINE "QUICG"

LIMITS OF INTEGRATION          A       =   1.30000000000D-01
                               B       =   1.00000000000D+00

REQUIRED ACCURACY OF RESULT    DIGIT   =                5  DEC. DIGITS
MAX. NUMBER OF F-EVALUATIONS   MAXF    =             5000

R E S U L T
ESTIMATE OF THE INTEGRAL       RESULT  =   1.86663031500D+00
ESTIMATED BOUND ON ABS. ERROR  ABSERR  =            1.80D-05  ( 5 DIGITS)
NUMBER OF F-EVALUATIONS        NFCALL  =             3813

**********************************************************************
```

oder ob Schwierigkeiten aufgetreten sind

```
**********************************************************************
                                                    CODE    1.00000D+00

NUMERICAL CALCULATION OF A ONE-DIMENSIONAL
DEFINITE INTEGRAL WITH SUBROUTINE "QUICG"

LIMITS OF INTEGRATION          A      =   1.30000000000D-01
                               B      =   1.00000000000D+00
REQUIRED ACCURACY OF RESULT    DIGIT  =                 10   DEC. DIGITS
MAX. NUMBER OF F-EVALUATIONS   MAXF   =               5000

R E S U L T
ESTIMATE OF THE INTEGRAL       RESULT =   1.86662918750D+00
ESTIMATED BOUND ON ABS. ERROR  ABSERR =            3.2D-06   ( 5 DIGITS)
NUMBER OF F-EVALUATIONS        NFCALL =               4991

F A I L U R E
MORE THAN MAXF F-EVALUATIONS ARE NECESSARY FOR THE REQUIRED ACCURACY


SUBINTERVALS OF (A,B) WITH LARGE ERROR ESTIMATES

                                                    ERROR      PERC. OF
      SUBINTERVAL : ENDPOINTS                WIDTH  ESTIMATE   ABSERR

  1   6.99981689453 1D-01   7.00000D-01   1.1D-04   6.7D-07    20.85
  2   5.99940185546 90D-01   6.00050D-01   1.1D-04   5.8D-07    17.93
  3   8.99985046386 70D-01   9.00010D-01   2.7D-05   4.0D-07    15.13
  4   2.99021875000 0D-01   3.01820D-01   1.7D-03   4.6D-07    14.25
  5   4.99580078125 0D-01   5.00000D-01   4.2D-04   2.8D-07     8.58
  6   1.99667968750 0D-01   2.01370D-01   1.7D-03   2.7D-07     8.50
  7   3.99063378906 2D-01   4.00180D-01   2.1D-04   2.5D-07     7.78
  8   7.99996643066 4D-01   8.00020D-01   2.7D-05   2.2D-07     6.99

**********************************************************************
```

Aus dem zweiten Printout kann man z.B. entnehmen, daß an den Stellen 0.2, 0.3,.. .., 0.9 der Integrand dem Programm Schwierigkeiten bereitet hat.

PROGRAMMAUSWAHL

Bezüglich der Auswahl eines geeigneten Unterprogramms für eine konkrete Anwendung kann folgende Empfehlung gegeben werden:
Wenn über den Typ des Integranden keine speziellen Informationen vorliegen, so wird die Verwendung von "QAG" empfohlen. Falls sich auf Grund der benötigten f-Auswertungen herausstellen sollte, daß der Integrand hinreichend "glatt" ist (falls die Berechnung etwa nicht mehr als rund 100 f-Auswertungen benötigt), so kann bei späteren Berechnungen ähnlicher Integranden "QNG" verwendet werden, was sich vor allem in einer Speicherplatzersparnis auswirkt, aber auch bei oftmaligen Integralauswertungen bzgl. der Rechenzeit von Vorteil sein kann. Wenn sich jedoch bei der Verwendung von "QAG" herausstellt, daß Endpunktsingularitäten vorliegen

(was z.B. im Printout des Interface-Programms "QUICG" erkannt werden kann), so empfiehlt es sich, "QUAGS" zu verwenden. "QUAGI", schließlich, stellt im Fall eines unendlichen Integrationsbereichs die einzige Wahlmöglichkeit dar.

DOKUMENTATION

Zu den POINT-Programmen sind Kurzbeschreibungen erhältlich. Eine ausführliche Beschreibung ist in Ausarbeitung.

LITERATUR

[1] H.J. Stetter, Chr.W. Überhuber
Proposal for Population Studies in Numerical Quadrature
Journal of Computational and Applied Mathematics
Vol.1 (1975), pp. 213-215

[2] W. Schulner, Chr.W. Überhuber
Monitorprojekt Numerische Quadratur - Datengewinnung, Datenspeicherung
Institut für Numerische Mathematik, TU-Wien, Bericht Nr. 11, 1975

[3] M.A. Malcolm, R.B. Simpson
Local versus Global Strategies for Adaptive Quadrature
ACM Trans. on Math. Software Vol.1 (1975), pp. 129-146

[4] J. Rice
A Metalgorithm for Adaptive Quadrature
J.ACM Vol.22 (1975), pp. 61-82

Friedrich Blöser
Christoph W. Überhuber (Inst. f. Num. Math.)

Das vorliegende Programmpaket stellt einen Teil der Ergebnisse einer Zusammenarbeit der TU-Wien mit der Universität Leuven (Belgien) - insbesondere mit Robert Piessens und Elise de Doncker - dar.

ANALOGE SIMULATION DER BEWEGUNG EINES AUTOS BEI DER FAHRT ÜBER EINEN KNÜPPELWEG MIT ANALOGER FAHRZEUGDARSTELLUNG

Ohne digitale Unterstützung wurde am Analogrechner von Teilnehmern des "Einführungspraktikums für hybride Analogrechner" eine Echtzeitsimulation eines fahrenden Autos mit Display am Oszillographen durchgeführt. Die Aufgabe wurde so ausgelegt, daß die erforderlichen Schaltungen die Kapazität der drei verkoppelten Kleinrechner EAI-180 auslasten, die Rechnung aber auch an der Anlage EAI-680 durchgeführt werden kann.

Die Aufgabe wurde in drei Teilprobleme gegliedert, die von je zwei Praktikanten zu lösen waren, sodaß jede Gruppe wegen der Schnittstellen der Teilprobleme den beiden anderen Teams genaue Angaben über zu übergebende Daten zu machen hatte. Durch diese notwendigen Absprachen blieb jedem Teilnehmer der Überblick über das gesamte Problem erhalten.

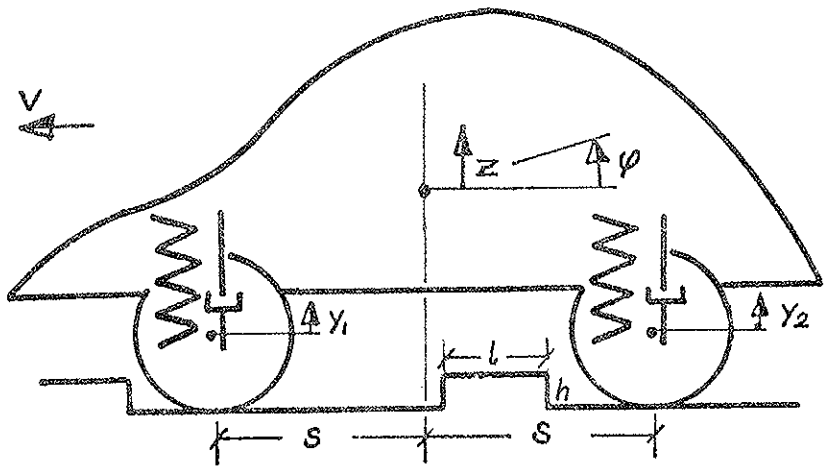


Abb.1

Ersatzsystem für das Fahrzeug, bestehend aus Wagenkasten mit der Masse m, masselosen Rädern und Achsen, die über Federn und Stoßdämpfer mit richtungsabhängigen Dämpfungswerten mit dem Kasten verbunden sind.

Eine aus Schwellen bestehende Fahrbahn bedingt das gleichartige Verhalten der Vorder- bzw. der Hinterräder. Diese Symmetrie erlaubt eine angenäherte Darstellung des Bewegungsablaufes mittels zweier Differentialgleichungen:

$$\ddot{z} = -\frac{2c}{m}\left(z - \frac{y_1 + y_2}{2}\right) - \frac{2B}{m}\left(\dot{z} - \frac{\dot{y}_1 + \dot{y}_2}{2}\right)$$

$$\ddot{\phi} = -\frac{2cs^2}{I}\left(\phi + \frac{y_2 - y_1}{2s}\right) - \frac{2Bs}{I}\left(\dot{\phi} + \frac{\dot{y}_2 - \dot{y}_1}{2}\right)$$

mit

z: Bewegung des Schwerpunktes
y_1, y_2: Bewegung der Räder
ϕ: Verdrehung des Fahrzeuges (Nicken)

Für die Konstanten wurden die folgenden Werte eingesetzt:

Masse m	1000 kg
Trägheitsmoment I	1000 kg m^2
Halber Radstand s	1.5 m
Schwellenbreite l	.5 m
Schwellenhöhe h	.1 m
Raddurchmesser	.55 m
Federkonstante c	100 kp cm^{-1}
Dämpferkonstante B_0	25 kp cm^{-1} sec
$B = B_0$ für $\dot{z} - \frac{\dot{y}_1 + \dot{y}_2}{2} < 0$	
$B = 10\ B_0$ sonst	
Fahrzeuggeschwindigkeit v	20 km/h

Die Simulation des momentan unterhalb des Autos liegenden Fahrbahnstückes in der Länge von acht Schwellenbreiten erfolgte durch ein Schieberegister. Jedes der acht Flip-Flops des Registers repräsentiert ein Stück Fahrbahn der Breite s und der Höhe h bzw. 0, je nachdem ob das Flip-Flop gesetzt oder zurückgesetzt ist. Die stetige Bewegung der Räder ist aus diesen unstetigen Signalen abzuleiten. Die sich ergebende kreisförmige Bahn des Rades beim Hinauf- und Hinunterfahren an einer Schwelle wurde durch eine lineare Funktion ersetzt, sodaß die Bewegung eines Radmittelpunktes einen Polygonzug beschreibt.

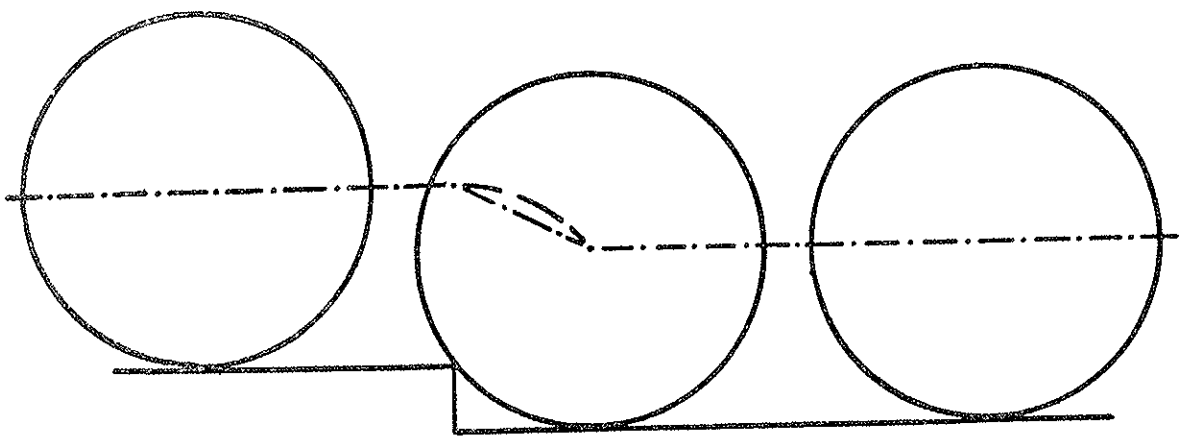


Abb.2

Exakter (---) und simulierter (-·-·-) Weg des Radmittelpunktes beim Auffahren an einer Schwelle.

Der Knüppelweg wurde also durch ein Schieberegister mit dem Inhalt $(a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8)$ simuliert, wobei nach der Zeit Δt, die dem Überfahren einer Schwellenbreite entspricht, der Inhalt des Registers einmal geschoben wird und ein neuer Wert a_1 eingeht:

$$a_i = a_{i-1} \qquad i = 2,3,\ldots 8$$

$$a_1 = (a_3 \wedge a_8) \vee (\bar{a}_3 \wedge \bar{a}_8)$$

d.h. der neue Wert ist '1 wenn a_3 und a_8 gleich waren und sonst '0. Das durch den Registerinhalt gewonnene Fahrbahnstück wurde während der Simulation der Fahrzeugbewegung dem Auto mit konstanter Geschwindigkeit unterschoben.

Nach dem Lösen der Gleichungen und der Modellbildung des Knüppelwegs blieb für die letzte Gruppe noch die Darstellung der gesamten Simulation am Oszillographen. Auch diese Aufgabe wurde unter der Auflage, Rechenelemente zu sparen, gelöst. Es konnte beispielsweise die Darstellung des Autos mit lediglich drei Integrierern und ohne Verwenden von Funktionsgebern erreicht werden. Das gesamte Display wurde in mehrere Teilbilder zerlegt, die entsprechend schnell, repetierend geschaltet wurden, sodaß für den Betrachter am Oszillographen nur mehr der Eindruck eines Gesamtbildes wahrzunehmen war. Die in dem für die Kleinrechner vorgesehenen Schaltplan enthaltenen Relais erzeugten allerdings während des Schaltvorgangs gewisse zusätzliche Störablenkungen am Oszillographen. Dieser Effekt konnte am Rechner EAI-680 vermieden werden, da hier genug elektronische Schalter zur Verfügung stehen.

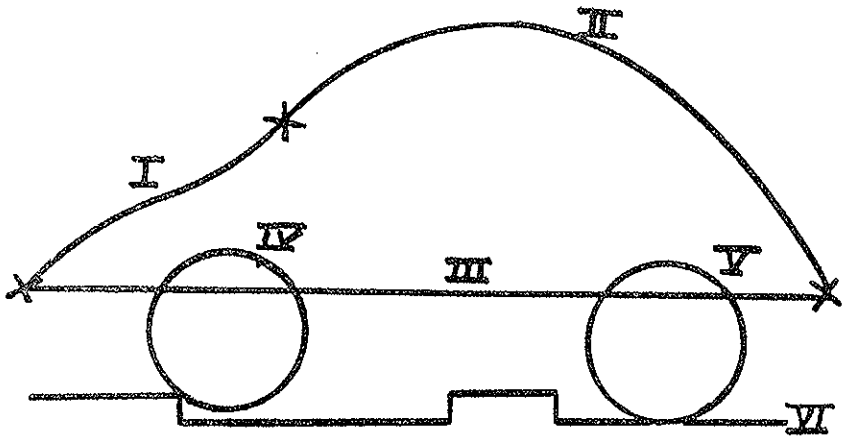


Abb. 3

Die Darstellung des Fahrzeuges in sechs Phasen:

$$\left.\begin{array}{ll} \text{I} & y = a\sin\omega t + bt \\ \text{II} & y = -ct^2 + dt + e \\ \text{III} & y = \text{konst.} \end{array}\right\} + \phi\bar{t} + z$$

IV,V Phasenbild des in I verwendeten Sinus
VI Inhalt des Schieberegisters

Bei der Simulation mit den gegebenen Werten konnte infolge der häufigen Erregung des Fahrzeuges durch die Schwellen eine Dämpfung kaum beobachtet werden. Bei eingehenderen Untersuchungen dieses Modells könnten noch gewisse Optimierungen für die Feder- und Dämpferwerte durchgeführt werden, oder etwa die Frage geklärt werden, ob nicht durch die gleichmäßige Schwellenbreite bei gewissen Geschwindigkeiten Resonanzen auftreten.

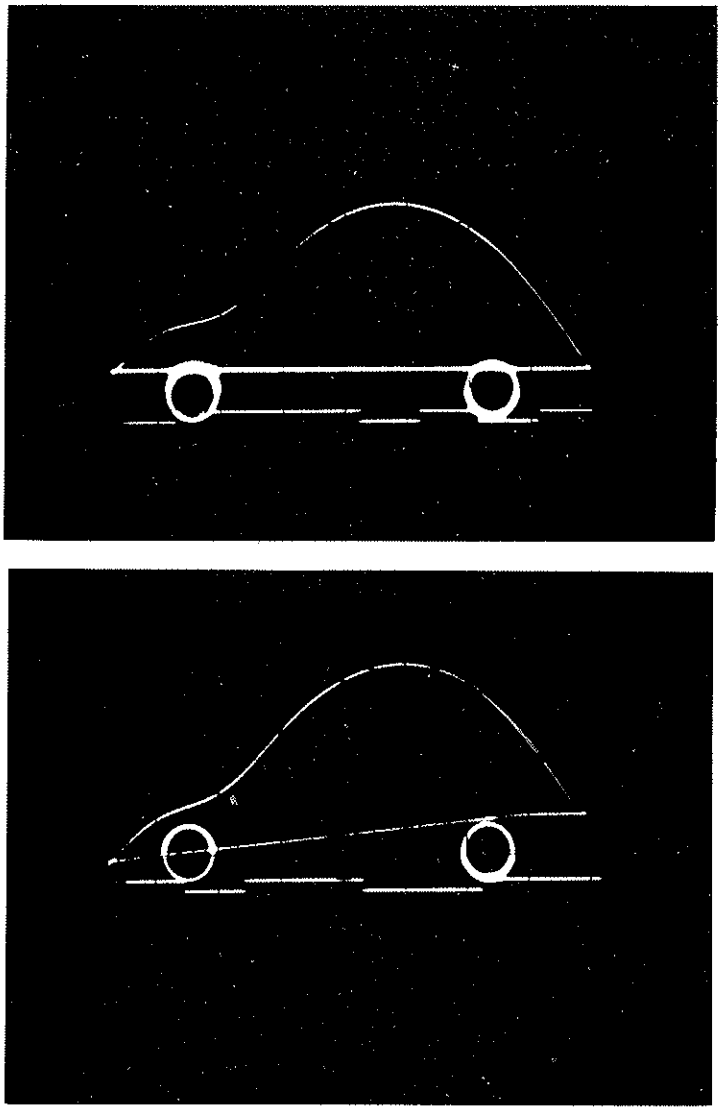


Abb. 4

Fahrzeug in Ruhe und im angeregten Zustand

Dieses Beispiel zeigt, daß die parallele Arbeitsweise des Analogrechners in Verbindung mit der vorhandenen parallelen Logik auch geeignet ist, Simulationen, die aus verschiedenen Teilaufgaben bestehen, mit Erfolg zu rechnen und bei Bedarf auch ein ansprechendes Display am Oszillographen zu erzeugen. Weitere Beispiele, die mit ähnlichem Konzept gelöst wurden, werden in späteren Ausgaben beschrieben werden.

Summary

A problem given for students training on the hybrid computation centre was an analog real-time simulation of a car moving on a test-course including a display on the oscilloscope. The patchings were developed for three connected EAI-18Ø and also for the EAI-68Ø computer. The test-course was simulated by a binary shift register. The movemement of the car was approximated by two differential equations of second order. The display on the oscilloscope was realized in 6 phases using only three integrators.

F. Rattay

benützerforum

RENDEZVOUS IM WELTALL

Beispiel aus dem Praktikum für Hybridrechner I
(Gruppe Berger E., Berger F.)
F. Breitenecker
Institut für Technische Mathematik
Technische Universität Wien

Das Rendezvous eines Satelliten mit einer Raumstation kann auf zwei verschiedene Arten erfolgen, nämlich direkt von einer Aufstiegsbahn oder aber von einer Wartebahn aus, wobei die Bedingungen für ein Rendezvous von einer Wartebahn aus wesentlich günstiger sind als von einer Aufstiegsbahn. Hierbei wird der Satellit zunächst auf eine Umlaufbahn, die innerhalb oder außerhalb der Zielbahn liegen kann, gebracht. Je nach der Wahl der Wartebahn ergibt sich früher oder später eine günstige Rendezvoussituation.

Die Durchführung des Rendezvous wird in zwei Phasen getrennt:

1. primäre Annäherung (weitgehendst vom Boden aus gesteuert)
2. Endphase (mit bordeigenen Geräten)

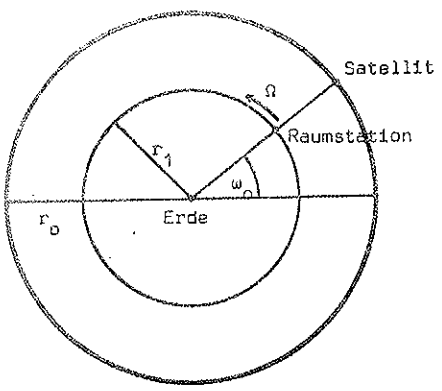
Im folgenden wird nun die Endphase (Andockmanöver) betrachtet.

Aufgabenstellung

Energieoptimale Steuerung eines Andockmanövers nach folgenden Angaben:

Die Raumstation bewege sich auf einer Kreisbahn mit dem Radius r_1 und der Winkelgeschwindigkeit Ω ($\Omega = \sqrt{\frac{\mu}{r_1^3}}$, wobei $\mu = f.M$; M = Masse des Himmelskörpers, f = Gravitationskonstante; z.B. Erde: $\mu = 3986 \cdot 10^2 km^3 sec^{-2}$).

Der Satellit bewege sich auf einer Kreisbahn mit dem Radius r_0. Das Andockmanöver soll bei einem Winkel ω_0 beginnen, soll nach T Zeiteinheiten beendet sein und soll möglichst wenig Energie (= Treibstoff) verbrauchen.



Mathematische Problembeschreibung

Die folgenden Gleichungen beschreiben in linearer Näherung die Relativbewegung des Satelliten (ξ,η) bezüglich der Raumstation.

$$\ddot{\xi} = 2\Omega\dot{\eta} + \frac{1}{m} P_\xi$$

$$\ddot{\eta} = -2\Omega\dot{\xi} + 3\Omega^2\eta + \frac{1}{m} P_\eta$$

P_ξ tangentiale
P_η radiale Schubkraftkomponente

Der Einfachheit halber wird hier nur jener Fall betrachtet, bei dem der Satellit nur tangential angetrieben wird, d.h. $P_\eta = 0$.

Mathematische Lösungsmethode

Unterwirft man das obige Differentialgleichungssystem zweiter Ordnung zwei Tranformationen, so erhält man ein Differentialgleichungssystem erster Ordnung mit doppelter Dimension:

$$\dot{x}(t) = Ax(t) + Bu(t) \qquad x(t) \in \mathbb{R}^4,\ u(t) \in \mathbb{R}$$

$$A = \begin{pmatrix} 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & -1 & 0 \end{pmatrix} \qquad B = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 1 \end{pmatrix}$$

Die Randwerte $x(0) = x_0$ und $x(T) = x_T$ ergeben sich aus der Forderung für die Position des Satelliten vor und nach dem Andocken.

Die Steuerung $u = u(t)$ (transformierte tangentiale Schubkraftkomponente) soll nun so gewählt werden, daß für das Andocken möglichst wenig Energie verbraucht wird, was folgender mathematischen Formulierung entspricht:

$$J(u(t)) = \int_0^T (\langle u(t), Ru(t)\rangle + \langle x(t), Q(t)x(t)\rangle)dt \to \text{Min}$$

$$R \equiv 1 \qquad Q = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

Nach dem Minimumprinzip von Pontryagin wird diese Funktionalminimierung zurückgeführt auf folgendes Randwertproblem:

$$\dot{x}(t) = Ax(t) + BR^{-1}B^T p(t) \qquad x(0) = x_0$$

$$\dot{p}(t) = -A^T p(t) - Qx(t) \qquad x(T) = x_T$$

Damit ist also die gesuchte Steuerung als

$$u(t) = R^{-1}B^T p(t)$$

festgelegt.

Numerische Lösungsmethode

Eine direkte Lösung dieses Randwertproblems durch das Überlagerungsverfahren oder durch iterative Methoden bringt numerische Schwierigkeiten, da in der Koeffizientenmatrix des gesamten Systems sowohl A als auch $-A^T$ vorkommen, was in jedem Fall zu Instabilität führt:

$$\begin{pmatrix} \dot{x} \\ \dot{p} \end{pmatrix} = \begin{pmatrix} A & BR^{-1}B^T \\ -Q & -A^T \end{pmatrix} \begin{pmatrix} x \\ p \end{pmatrix}$$

Zur Lösung wurde daher ein völlig anderer Weg beschritten, nämlich der des "invariant imbedding". Dabei werden drei Matrixdifferentialgleichungen D(t), E(t) und F(t) gelöst, die aus folgendem Ansatz abgeleitet werden:

$$x(t) = D(t)x(0) + E(t)p(t)$$
$$p(t) = F(t)x(0) + D^T(t)p(t)$$

$$\dot{D}(t) = -(A-E(t)BR^{-1}B^T)D(t) \qquad D(0) = I$$
$$\dot{E}(t) = -Q-A^TE(t)-E(t)A + E(t)BR^{-1}B^TE(t)$$
$$E(0) = 0$$
$$\dot{F}(t) = D^T(t)BR^{-1}B^TD(t) \qquad F(0) = 0$$

Man beachte, daß die Matrixdifferentialgleichung für E(t) vom Typ Riccati ist.

Der Vorteil dieses Verfahrens ist, daß das Randwertproblem in ein Anfangswertproblem und in Lösen von Gleichungssystemen übergeführt wird, wobei die Koeffizienten der Gleichungssysteme die Elemente der Matrizen D, E und F sind.

Vorteilhaft ist auch, daß die Matrixdifferentialgleichungen unabhängig vom Anfangswert x(0) und vom Endwert x(T) sind, sie müssen daher für verschiedene Andockmanöver nur einmal gelöst werden.

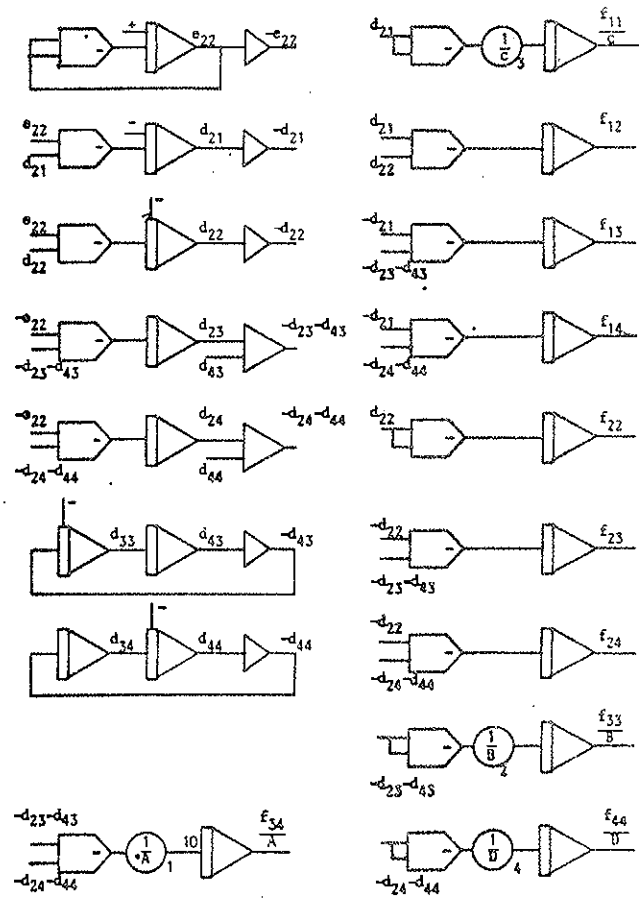
Analogprogramm

Am Analogrechner werden die drei Matrixdifferentialgleichungen im Intervall [0,T] (Es wurde T=10 gewählt) gelöst.

Potentiometerliste:

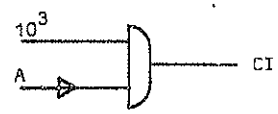
A = 0.5	1	$\frac{1}{A}$	0.2·10
B = 2.5	2	$\frac{1}{B}$	0.4
C = 10	3	$\frac{1}{C}$	0.1
D = 3	4	$\frac{1}{D}$	0.3333

Schaltplan:

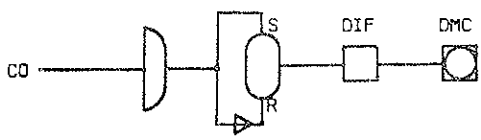


Die Werte der Matrizen wurden mittels des direkten Datenkanals am Digitalrechner abgespeichert. Die Übertragung erfolgte alle 0.5 Zeiteinheiten (d.h. es wurden 20 Werte pro Funktion überspielt).Die Steuerung der Übertragung wurde durch die digitale Clock realisiert (Anfangswert = 50).

Die folgende Schaltung sichert, daß erst zur OP-Zeit mit der Übertragung begonnen wird:



Synchronisation von CO:



Digitalprogramm

Die für die analoge Integration erforderlichen Skalierungsfaktoren wurden mittels eines CSMP-Programmes gefunden.

Die überspielten Werte wurden dann auf einen File abgespeichert.

auf dem File abgespeicherten Werte und die eingelesenen Parameter r_1, r_0, ω_0 und T können nun die Gleichungssysteme zu jedem Zeitpunkt t_i ($0 = t_1 < t_2 < \ldots < t_i < \ldots < t_n = T$; $t_i - t_{i-1} = 0.5$) gelöst werden und man erhält somit die Relativbewegung diskretisiert im Intervall [0,T]. Durch Rücktransformation kommt man zur eigentlichen Lösungskurve, der Flugbahn des Satelliten zur Raumstation, welche aufgezeichnet wurde.

Digital- und Analogprogramm wurden auch noch modifiziert, um statt 20 Stützstellen 200 übertragen zu können.

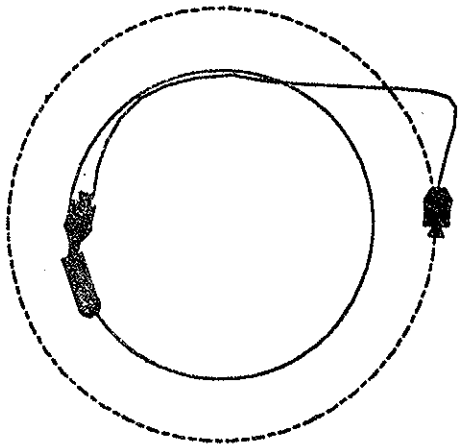


Abbildung 1

Energieoptimale Andockbahn des Satelliten von der äußeren Umlaufbahn her

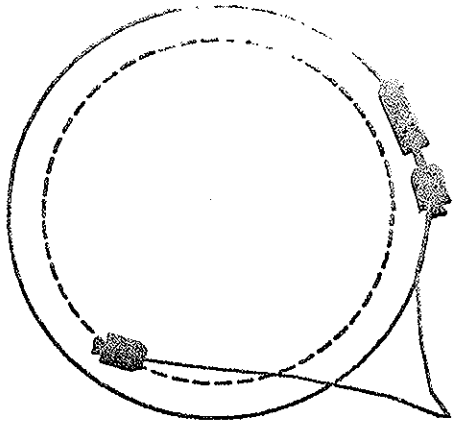


Abbildung 2

Energieoptimale Andockbahn des Satelliten von der inneren Umlaufbahn her

Summary

For the final phase of a docking manoeuvre between a satellite and a space station, both in circular orbit, the - with respect to energy - optimum control of the satellite is computed. The differential equations are transformed into matrix initial value problems and systems of linear equations. The differential equations are solved on the analog computer, with data being transferred to the digital computer for solving the linear equations, re-transformation, and plotting of the solution.

Literatur

F. Breitenecker: Zur Dualität von Riccati-Ansatz und Invariantem Einbetten bei Problemen des Optimal Control. Diplomarbeit TU Wien, Institut für Technische Mathematik, SS 1976

P. Sagirow: Satellitendynamik. BI 719