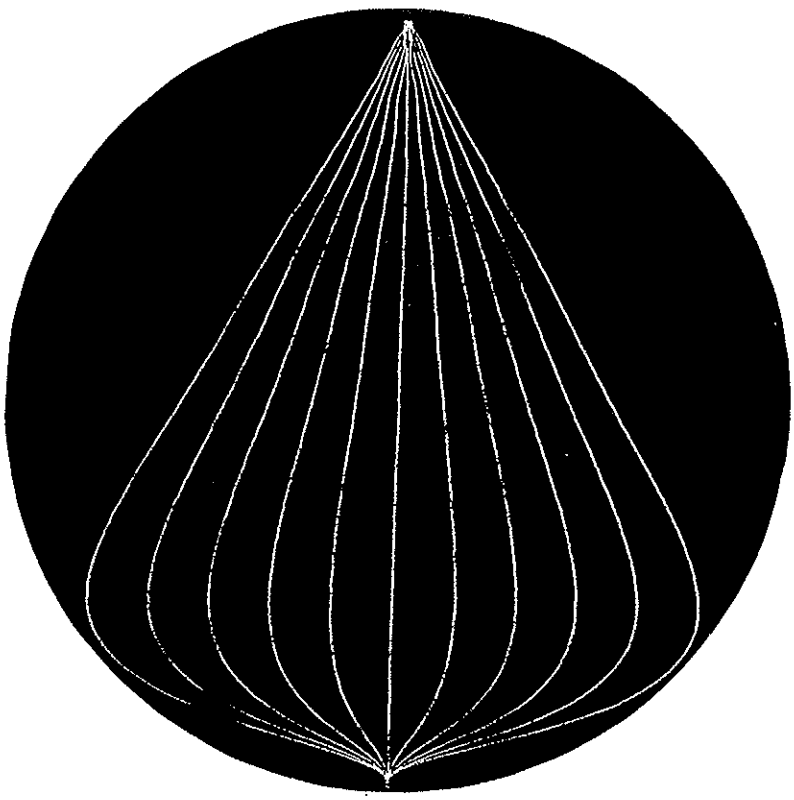
# Interface

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

Nummer 21
Juni 1984



Anflugwege unter Kontrolle des Leitsystems ILS

INHALTSVERZEICHNIS



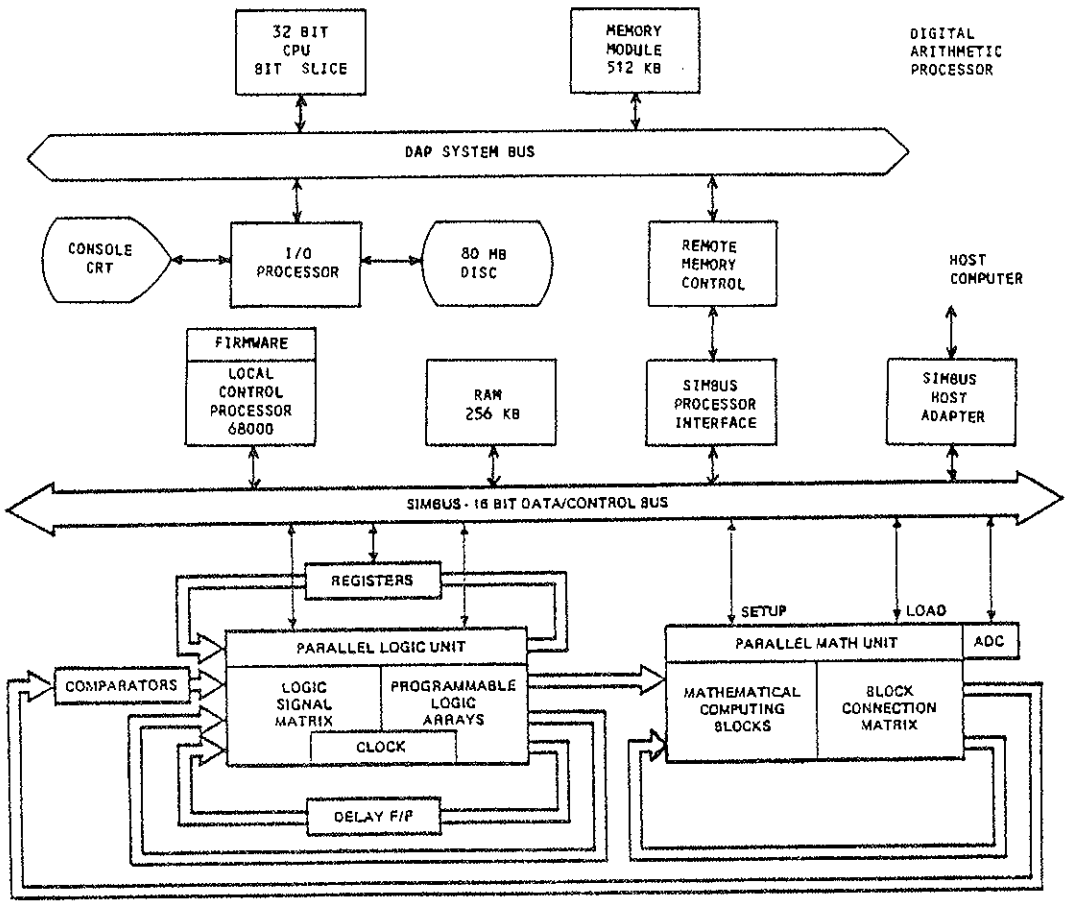

Redaktion: Irmgard Husinsky
Abt. Hybridrechenanlage des EDV-Zentrums der Technischen Universität Wien, A-1040 Wien, Gußhausstraße 27-29. Herausgeber, Verleger, Hersteller: EDV-Zentrum der Technischen Universität Wien, Abt. Hybridrechenanlage, Leitung: Dipl.Ing.Dr. W. Kleinert, A-1040 Wien, Gußhausstraße 27-29. Telephon: (0222) 56 01/3706 oder 3669. Telex: 136875 rzthw a, Modem: (0222) 65 93 17. Druck: Hochschülerschaft Technik, A-1040 Wien, Argentinierstr. 8.

# Neuer Analogrechner EAI – SIMSTAR bestellt !

W. Kleinert

Nach einer mehrjährigen Planungs- und Vorbereitungsphase konnte am 14. Mai 1984 zwischen dem Bundesministerium für Wissenschaft und Forschung und der Firma Electronic Associates Inc., New Jersey (USA), ein Vertrag über die Lieferung eines EAI-SIMSTAR Simulationssystems als Ersatz für den 16 Jahre alten Analogteil der Hybridrechenanlage der TU Wien abgeschlossen werden. Damit kann die für die ingenieurmäßige Simulation des dynamischen Verhaltens technischer Systeme so wichtige, auf der parallelen Arbeitsweise kontinuierlicher Funktionsblöcke beruhende hybride Rechentechnik an der TU Wien auf den neuesten technologischen Stand gebracht werden. Mit der für Anfang 1985 geplanten Installation des SIMSTAR-Systems wird auch die Lücke zwischen der an der Hybridrechenanlage entwickelten modernen, weltweit anerkannten Simulationssoftware und der veralteten Hardware geschlossen werden können.

EAI, mittlerweile der einzige Hersteller von leistungsfähigen Hybridrechensystemen, hat mit SIMSTAR einen neuartigen, vollständig automatisierten Simulationsrechner entwickelt, der die neueste analoge VLSI Technologie mit fortgeschrittenem CPU-, Mikroprozessor- und Controller-Design vereinigt und dadurch eine um den Faktor 20 bessere Performance als unser alter Analogrechner besitzt. Die folgende Abbildung zeigt ein Funktionsblockschaltbild von SIMSTAR.




SIMSTAR Funktionsblockschaltbild
Abbildung 1

Das Kernstück von SIMSTAR ist die analoge Parallel Mathematical Unit (PMU), bei der die mathematischen Funktionsblöcke (Hardware Makros) - ganz ähnlich unserem jetzigen Konzept - mit Hilfe einer elektronischen Schaltmatrix programmierbar gekoppelt sind. Die große Performancesteigerung resultiert u.a. aus:

- dem dicht gepackten Aufbau der Hardware Makros, bei denen sich alle für die betreffenden mathematischen Funktionen notwendigen Verstärker, Koeffizienteneinheiten, Begrenzer usw. auf einer Karte befinden,

- der Anordnung der Hardware Makros in unmittelbarer Nähe der Schaltmatrix mit optimalen Verbindungswegen,

- der ausschließlichen Verwendung von sogenannten "Compound Amplifiers", d.h. Verstärkern mit äußerst hoher Bandbreite und sehr geringem Offset bzw. Offsetdrift in Verbindung mit einer automatischen Nullpunktabgleichung für jeden Verstärker,

- den 17-Bit Koeffizienteneinheiten unter Verwendung hochgenauer multiplizierender monolithischer CMOS DACs,

- der Verwendung von VLSI CMOS Schaltern innerhalb der Schaltmatrix,

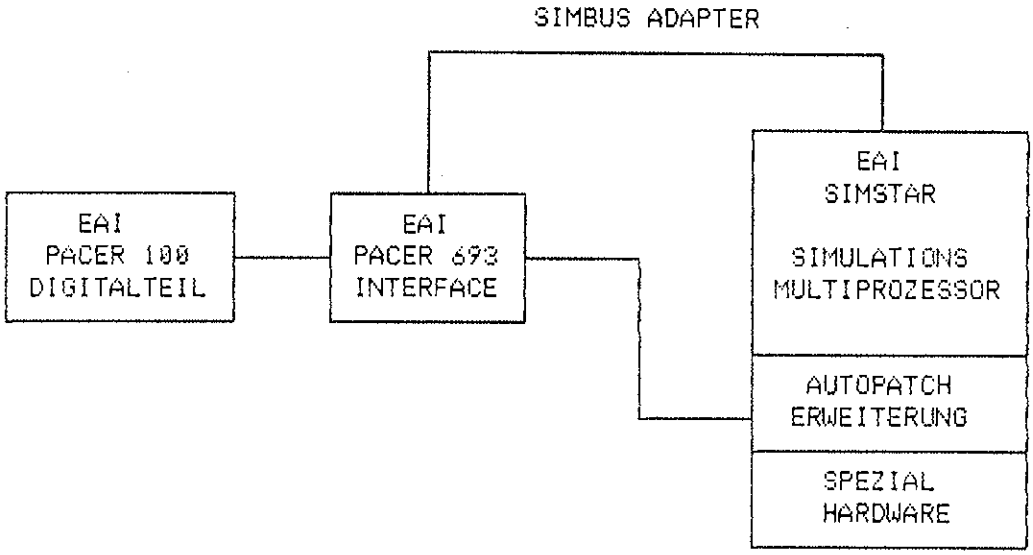
- der automatischen Bereichsumschaltung bei Multiplizierern und Koeffizienteneinheiten, die einen prozentuellen Ausgangsfehler auch bei kleinen Größen garantiert.

Darüberhinaus gestattet eine spezielle Auslese-Hardware die Überprüfung von 3000 Meßpunkten. Die Firmware im Local Control Processor (LCP), einem 68000 Mikroprozessor, kann unter Verwendung spezieller Testspannungen automatisch defekte Elemente der Schaltmatrix sowie fehlerhafte Makroboards erkennen. Der LCP verwaltet alle hardware-abhängigen Datenbasen und ermöglicht die Initialisierung eines Problemwechsels auf höherer Ebene.

Zur Implementierung logischer Bedingungen steht eine programmierbare Parallel Logic Unit (PLU) zur Verfügung, bei der die 160 logischen Eingangssignale über maximal 16 Pipelinestufen auf die 320 logischen Ausgänge gelegt werden können. Da jede Pipelinestufe durch RAMs realisiert ist, können logische Gleichungen beliebiger Komplexität praktisch ohne Einschränkungen realisiert werden. Die PLU steuert nicht nur Schalter und Track-Store-Einheiten in der PMU, sie erlaubt auch die programmierte Einzelsteuerung aller Integrierer. Die gesamte parallele Rechenkapazität von PMU und PLU wird diejenige unseres jetzigen AutoPATCH-Systems übertreffen, sodaß alle Probleme vergleichbarer Größenordnung auch auf dem SIMSTAR bearbeitet werden können.

Integrierter Bestandteil des SIMSTAR Systems ist ferner ein in Bit-Slice-Technik realisierter 32-Bit Digital Arithmetic Processor (DAP), der vor allem zur schnellen Abarbeitung der digitalen Unterprogramme eingesetzt werden soll, die zeitlich parallel zur analogen Simulation ablaufen.

Der Austausch des Analogteils der Hybridrechenanlage an der TU Wien bedingt keine wesentliche Änderung des Betriebssystems JCS/VS 8 und des hybriden Sprachprozessors HYBSYS. Da die elementaren mathematischen Funktionsblöcke des SIMSTAR Multiprozessors (Hardware Makros) äquivalente Funktionen zu denjenigen des jetzigen AutoPATCH-Systems liefern, sind auch die Softwareschnittstellen äquivalent. Auf der Hardwareseite ist lediglich die Erweiterung der PACER 693 Interface-Einheit durch einen bereits im Entwurf vorliegenden Host-Adapter für den SIMBUS, einen erweiterten INTEL-MULTIBUS, erforderlich.



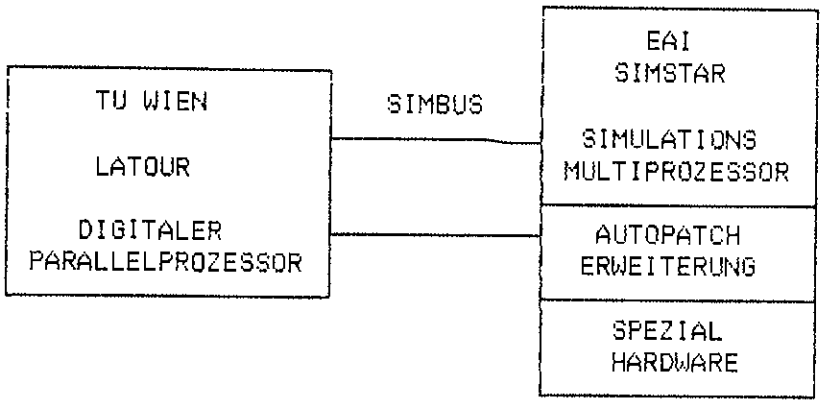

Konfiguration des Hybridrechners mit SIMSTAR
Abbildung.2

Mit dieser Schnittstelle brauchen bloß die Betriebssystem-Treiber auf unterster Ebene an das neue Datenformat und die analoge Hardware-Datenbasis auf die geplante Konfiguration angepaßt werden. Die unserem jetzigen AutoPATCH-System zugrunde liegende Schaltmatrix soll zur Ankoppelung der an der TU Wien entwickelten Spezial-Hardware (Hybrid Memories, Hybrid Functional Units, etc.) sowie von externen Trunks als Verbindung zu Online-Experimenten an den SIMSTAR weiter verwendet werden.

Zum zeitlichen Ablauf ist zu sagen, daß mit dem Installationsbeginn spätestens bis zum Frühjahr 1985 zu rechnen ist. Nach einer 3-4-monatigen Entwicklungs- und Testphase für selbstentwickelte Hard- und Software soll bis Ende 1985 ein Testbetrieb laufen, bei dem neben dem alten PACER 680 AutoPATCH-Analogteil auch der SIMSTAR (allerdings ohne Spezial-Hardware) betrieben wird. Spätestens Anfang 1986 soll die endgültige Umstellung auf den neuen Simulations-Multiprozessor SIMSTAR erfolgen. Die Programmierung wird voll softwareKompatibel von HYBSYS aus erfolgen.

Auf Dauer gesehen kommt der bisher als Digitalteil des Hybridrechners eingesetzte, 12 Jahre alte Minicomputer EAI PACER 100 als Wirtsrechner allerdings nicht mehr in Frage. Um die Kosten für den Ankauf eines eigenen digitalen Wirtsrechners als Ersatz für den PACER 100 zu sparen, und um den Aufwand bei der Übertragung der bewährten selbstentwickelten Systemsoftware (Betriebssystem und hybrider Compiler) auf einen neuen Rechner so gering

als möglich zu halten, soll im Rahmen des Projekts "Parallelrechner LATOUR" ein leistungsfähiger digitaler Multiprozessor an der TU Wien entwickelt und mit dem SIMSTAR zu einem einzigartigen Simulationssystem mit analoger und digitaler Parallelrechenkapazität verbunden werden.




Konfiguration des geplanten Hybridrechnerausbaus
Abbildung 3

---

## TU Vienna ordered EAI-SIMSTAR !

After several years of planning and preparation the Austrian Federal Ministry for Science and Research ordered a new SIMSTAR simulation multiprocessor in May 1984. This system will replace the 16 year-old analog part of the Hybrid Computation Centre of the Technical University of Vienna. It will fill the gap between the modern self-developed and worldwide acknowledged simulation software and the obsolete hardware at our computation centre.

The unique attached simulation multiprocessor SIMSTAR will be used in combination with a university designed and built digital MIMD-computer system, model LATOUR. The new SIMSTAR-LATOUR computing facility will provide the Technical University of Vienna with a powerful, modern, extremely fast multiprocessor system for real-time engineering and dynamic analysis, to be used in education, research, design and industrial applications.

The SIMSTAR system will be installed early 1985.

# Der Overlay RANDOM

F. Breitenecker
Abteilung für Regelungsmathematik, Hybridrechen- und Simulationstechnik
Institut für Analysis, Technische Mathematik und Versicherungsmathematik
Technische Universität Wien

Wichtige Aufschlüsse über die Güte eines mathematischen Modells (für einen dynamischen Prozeß) liefert die Sensitivitätsanalyse. Neben analytischen Untersuchungen wird dabei oft ein Parameter oder eine Zustandsgröße des Modells mit geeigneten Störungen überlagert und deren Auswirkungen auf andere Zustandsgrößen beobachtet (/1/).

HYBSYS bietet standardmäßig zwei Möglichkeiten, das Verhalten eines Modells bei Störungen zu untersuchen. Als einfachste Form dieser Sensitivitätsanalyse kann die Variation eines oder mehrerer Parameter innerhalb einer Schleife mit Display oder Zeichnung (auch dreidimensional) einer oder mehrerer interessierender Variablen angesehen werden. Die zweite Möglichkeit besteht darin, eine Variable mit einem Rauschgenerator (NLO, NHI, /2/) zu überlagern, wobei ein Eingriff in die Modellbeschreibung zu unternehmen ist. Ein Simulationslauf gibt in diesem Fall die Antwort des Systems auf eine statistische dynamische Störung wieder.

Neben diesen beiden Möglichkeiten - der "deterministischen Parameterstörung" und der "statistischen dynamischen Störung" - wäre für viele Zwecke eine "statistische Parameterstörung" wünschenswert und auch ausreichend. Anstrebenswert wäre auch, das Ergebnis der Störung nicht als Einzeldarstellung für jede Störung als Display oder Zeichnung zu erhalten, sondern einen Mittelwert (Mittelwertkurve).

Der Overlay RANDOM wurde nun konzipiert und implementiert, um die beschriebene Sensitivitätsanalyse mit "statistischer Parameterstörung" möglichst komfortabel durchführen zu können. Essentielles Element des Overlays ist eine statistische Verteilung, nach der ein Parameter gestört wird. Eine interessierende Variable wird unter diesen Störungen bei hinreichend vielen Simulationsläufen beobachtet, um die Mittelwertkurve dieser Variablen berechnen und ausgeben zu können.

## Aufruf des Overlays

Der im Rahmen eines Praktikums (/3/) implementierte und ausgetestete Overlay wird mit

RANDOM,METHOD(PAR1,PAR2) NDAVAR=NVAR BY NPAR

aufgerufen. Die Parameter der Befehlszeile bedeuten im einzelnen:

METHOD ... mnemotechnische Bezeichnung für die Verteilung (Gaußverteilung, Gleichverteilung, T-Verteilung,...)

PAR1,PAR2 Parameter, die die Verteilung METHOD charakterisieren*) (Typ: Integer, Real)

NDAVAR ... Name der Variablen, die nach Exekution des Overlays die Mittelwertkurve der interessierenden Variablen NVAR darstellt (Typ: DACF)

NVAR .... Name der Variablen, die unter der statistischen Störung beobachtet werden soll (Typ: beliebige dynamische (analoge) HYBSYS-Variable)

BY ..... Füllwort

NPAR .... Name des Parameters, der mit der Verteilung METHOD(PAR1,PAR2) additiv gestört werden soll.

## Arbeitsweise des Overlays

Nach syntaktischer Prüfung der Befehlszeile und überprüfung der Variablentypen wird untersucht, ob eine Verteilung METHOD existiert. (z.B. GAUSS,GLEICH) und ob zur Verteilung die richtigen Parameter angegeben wurden. Nun werden RANDNR Simulationsläufe durchgeführt. Bei jedem Lauf werden der Parameter NPAR mit einer Zufallszahl nach der angegebenen Verteilung additiv überlagert und die Variable NVAR (an den NPOINTS Stützpunkten) ausgelesen. Simultan wird die jeweilige modifizierte Mittelwertkurve $\bar{X}^{(i)}(t_K)$ nach i Läufen aus der modifizierten Mittelwertkurve nach (i-1) Läufen $\bar{X}^{(i-1)}(t_K)$ und der Variablen im i-ten Lauf $VAR^{(i)}(t_K)$ berechnet:

$$\bar{X}^{(i)}(t_K) = \bar{X}^{(i-1)}(t_K) + VAR^{(i)}(t_K)/RANDNR$$

$$\bar{X}^{(1)}(t_K) = VAR^{(1)}(t_K)/RANDNR$$

k=1,..,NPOINTS (Anzahl der Datenpunkte)

i=2,..,RANDNR (Anzahl der Simulationsläufe)

---

*) Es sind immer zwei Zahlenwerte anzugeben. Benötigt die Verteilung nur einen bestimmenden Parameter (oder keinen), so wird der erste Wert PAR1 (oder kein Wert) genommen.

Nach dem letzten Simulationslauf (RANDNR) wird der DACF mit dem Namen NDAVAR mit der Mittelwertkurve $\bar{X}^{(RANDNR)}(t_k)$ geladen.

Nach Exekution des Overlays steht daher die Mittelwertkurve auf dem DACF NDAVAR für weitere Verarbeitung (Display, PLOT,..) zur Verfügung; der Parameter NPAR hat jenen Wert, mit dem der Overlay aufgerufen wurde.

Standardmäßig gilt RANDNR=100, der Benutzer kann jedoch eingreifen, indem er im HYBSYS-Programm den Parameter RANDNR definiert und seinen Wert auf die gewünschte Probenanzahl setzt (z.B. DECLAR PAR: RANDNR=250), $5 \leq$ RANDNR $\leq 1000$; Im Overlay wird nicht überprüft, ob bei einem Simulationslauf eine übersteuerung stattfindet.

## Verwendung des Overlays

Die momentane Version des Overlays kann von jedem Benutzer vom District 19 mit

```
GET,OV,RANDOM
```

geladen werden. Derzeit ist als Methode METHOD nur die Gaußverteilung GAUSS(μ,σ) implementiert. Exekutiert wird der Overlay mit

```
RANDOM,METHOD(PAR1,PAR2) NDAVAR=NVAR BY NPAR
```

Nach Exekution steht die Mittelwertkurve auf NDAVAR zur Verfügung. Wird diese Befehlszeile durch einen Doppelpunkt beendet (":"), so wird während der Exekution die Nummer jedes Laufes und der momentane gestörte Parameterwert ausgegeben.

## Beispiel

Um in der Differentialgleichung

$$\ddot{x} = -a\dot{x} - bx + cx^2, \quad x(0)=0,\ \dot{x}(0)=1$$

die Sensitivität eines der Parameter a, b oder c zu untersuchen, ist das Modell

```
MODEL
   PAR: A=3,B,C
   INT: X,DX
   SQR: XQ
   X = DX
   DX,IC=1,-A*X,-B*DX,C*XQ
   XQ = X
END
```

zu erweitern durch

```
   DECLAR DACF: DAX
```

Für C=5 zeigt Bild 1 Originalkurve und Mittelwertkurve, das mit dem Aufruf

```
C=5; RANDOM,GAUSS(0,1) DAX=X BY C; PLOT X,DAX
```

erzeugte Ergebnis. Bild 2 zeigt dieselbe Sensitivitätsanalyse mit

```
RANDOM,GAUSS(0,2) DAX=X BY C
```

Bild 1:

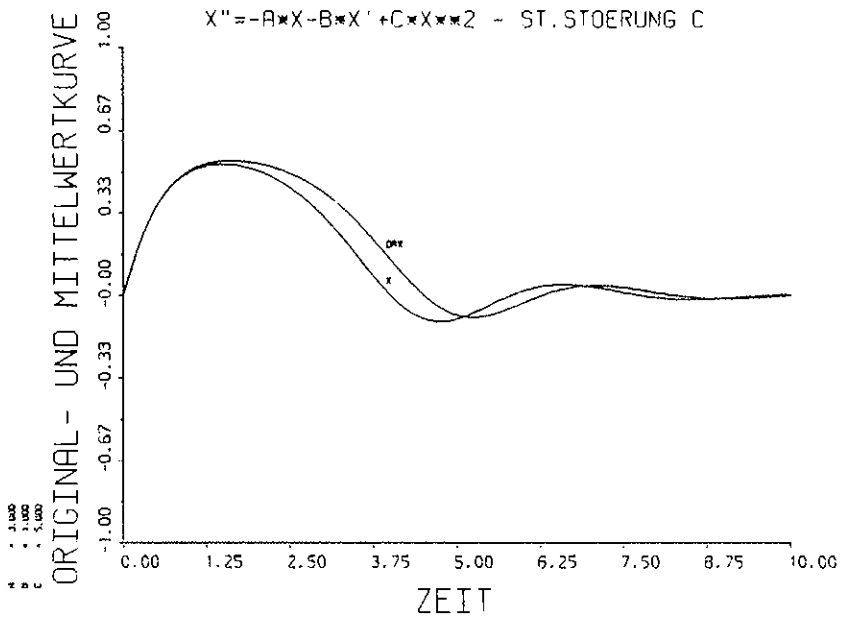
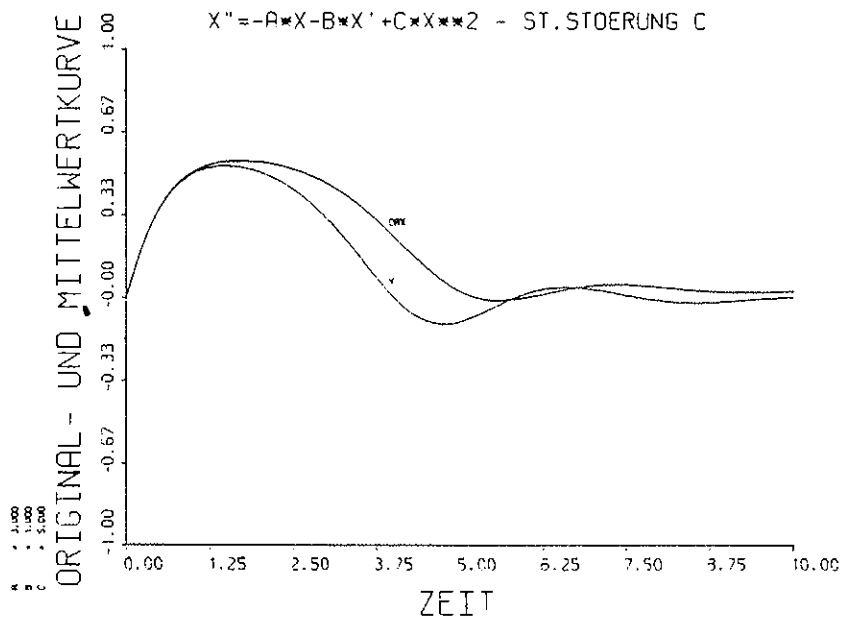



Bild 2:




Literatur

/1/ R.Tomovic, M.Vukobratovic: General Sensitivity Theory. American Elsevier Comp., New York, 1972.

/2/ D.Solar: HYBSYS User Manual, Version 5 TS. Abt. Hybridrechenanlage des EDV-Zentrums der Technischen Universität Wien, Juli 1983

/3/ I.Pseiner, M.Schiffl: Praktikumsarbeit im "Fortgeschrittenenpraktikum für hybride Analogrechner" (Prof.Dr.I.Troch), 1984

# F T U – FORTRAN 77 Compiler
# Version FTU.1

I. Husinsky

Von dem an der Hybridrechenanlage entwickelten FORTRAN 77-Compiler FTU, über den in den letzten Nummern von INTERFACE berichtet wurde, ist nun die Version FTU.1 implementiert, die gegenüber FTU.0 die Befehle CHARACTER, COMPLEX, DOUBLE PRECISION, IMPLICIT, Block IF, ELSE IF, ELSE und END IF enthält.

## Erweiterungen gegenüber FTU.0

Der Typ Character wurde implementiert. Es können Konstante, Variable, Felder und Funktionsunterprogramme vom Typ Character sein. Für Operationen zwischen Character-Größen gibt es die "Concatenation", die zwei Zeichenketten aneinanderhängt. Für Vergleichsoperationen zwischen Character-Größen können alle Vergleichsoperatoren verwendet werden. In einem READ- oder WRITE-Befehl kann anstelle einer Formatnummer auch eine Character-Konstante, Character-Variable oder ein Character-Ausdruck stehen, der das Format enthält (z.B.: WRITE (IOUT,'(1X,I5)') K). Als interne Standardfunktionen stehen die Funktionen LEN (Länge einer Zeichenkette), ICHAR und CHAR (Umwandlung Character - Integer), SUBSTR (Teil einer Zeichenkette, Substring) und INDEX (Position eines Substrings) zur Verfügung.

Ferner sind die Datentypen Complex und Double Precision implementiert.

Es können Block IF-Strukturen verwendet werden. Diese werden durch einen Block IF-Befehl eingeleitet (IF (logischer Ausdruck) THEN), enden mit einem END IF-Befehl und können einen oder mehrere ELSE IF-Blöcke sowie einen ELSE-Block enthalten. Der ELSE IF-Befehl kann anstelle von THEN auch einen beliebigen anderen exekutierbaren Befehl enthalten. Maximal 10 Ebenen von Block IF-Strukturen können ineinander verschachtelt sein.

Für symbolische Namen können ab dem zweiten Zeichen auch Kleinbuchstaben verwendet werden. Das erste Zeichen muß immer ein Großbuchstabe sein.

## Verbesserungen gegenüber FTU.0

Das Format des Source Files , den der Compiler FTU mit dem symbolischen Code generiert, wurde komprimiert, d.h. es wurden die Leerzeichen weitgehend eliminiert. Der PACER-Assembler wurde dahingehend modifiziert, daß er auch dieses Source Format verarbeiten kann. Die daraus resultierenden kürzeren Source Files (nur etwa ein Achtel der bisherigen Länge) brachten eine Verkürzung der Rechenzeit um etwa ein Drittel bei der Generierung des Source Files und der anschließenden Assemblierung des Files.

Die auf der Map-Liste angegebenen Variablen, Konstanten und externen Größen sind jetzt alphabetisch sortiert .

Ein ausführliches Manual für den FTU-Compiler mit Syntaxbeschreibung ist erhältlich.

# Neu im Betriebssystem JCS/VS 8

F. Blöser

## Umschalten auf einen neuen Input-File

Daten, die in einem Programm von der Input-Device mit der Nummer 15 (Card Reader) eingelesen werden (z.B. mittels READ-Statement), müssen unmittelbar nach der /RUN-Controlline folgen oder müssen sich auf einem separaten Source-File befinden, der mittels der INPUT-Option auf der /RUN-Controlline spezifiziert wird.

Die neue Routine CONNEC (in der System Library JCSLIB) ermöglicht es nun, programmgesteuert auf einen neuen Source-File umzuschalten, von dem dann die weiteren Daten eingelesen werden. Da die Routine die momentane Position am bisherigen Input-File liefert, ist es auch möglich, etwa nach dem Einlesen der Daten vom neuen File, den vorangegangenen Input-File an der richtigen Position durch einen Aufruf von CONNEC wieder anzusprechen.

## Neue Intrinsic Functions in FORTRAN

Im Zuge der Realisierung des neuen FORTRAN 77-Compilers FTU wurde es notwendig, zusätzliche, im FORTRAN 77-Standard enthaltene, interne Standardfunktionen (Intrinsic Functions) zu implementieren. Außerdem wurde eine Reihe von nützlichen Routinen entwickelt, die logische Operationen und Bitmanipulationen mit 1 Wort-Größen durchführen.

Es handelt sich bei den neuen Standardfunktionen um folgende vier Gruppen von Routinen:

- Trigonometrische und hyperbolische Funktionen:
  TAN, DTAN
  ASIN, DASIN, ACOS, DACOS
  SINH, DSINH, COSH, DCOSH, DTANH
- Routinen zum Runden ("nearest integer"):
  NINT, ANINT, DNINT, IDNINT
- Funktionen für den Typ Character:
  CHAR, ICHAR, LEN, INDEX, SUBSTR
  LGE, LGT, LLE, LLT
- Routinen für logische Operationen und Bitmanipulationen:
  AND, OR, XOR, EQV, NEQV
  MASK, SHIFT, COMPL

Die Routinen befinden sich auf der FTU-Library und können von allen FORTRAN-Benützern auch bei Verwendung des FORTRAN 66-Compilers JCSFOR aufgerufen werden, sofern es sich nicht um FORTRAN 77-spezifische Routinen (wie z.B. Routinen mit Typ Character) handelt. Bei Verwendung mit JCSFOR ist eine explizite Typvereinbarung erforderlich, sofern der Typ der Funktion sich von dem durch den Anfangsbuchstaben festgelegten impliziten Typ unterscheidet. Beim Binden muß die Load-Anweisung LOAD,D25 FTU/SR angegeben werden.

Die folgende Tabelle enthält alle internen Standardfunktionen. Die Typen der Parameter sind dabei wie folgt:

| | | | |
|---|---|---|---|
| i | ... Integer | c | ... Complex |
| r | ... Real | a | ... Character |
| d | ... Double Precision | x | ... beliebige 1 Wort-Größe |

## Tabelle der internen Standardfunktionen

| Aufruf | Typ | Bedeutung |
|---|---|---|
| ABS(r) | Real | Absolute Value |
| ACOS(r) | Real | Arccosine |
| AIMAG(c) | Real | Imaginary Part $r_i$ of Complex Argument $(r_r,r_i)$ |
| AINT(r) | Real | Truncation to Whole Number |
| ALOG(r) | Real | Natural Logarithm |
| ALOG10(r) | Real | Common Logarithm (Base 10) |
| AMAX0($i_1,i_2,..$) | Real | Largest Value of $i_1,i_2,..$ |
| AMAX1($r_1,r_2,..$) | Real | Largest Value of $r_1,r_2,..$ |
| AMIN0($i_1,i_2,..$) | Real | Smallest Value of $i_1,i_2,..$ |
| AMIN1($r_1,r_2,..$) | Real | Smallest Value of $r_1,r_2,..$ |
| AMOD($r_1,r_2$) | Real | Remaindering $r_1-int(r_1/r_2)*r_2$ |
| AND($x_1,x_2,..$) | Integer | Boolean Product of $x_1,x_2,..$ |
| ANINT(r) | Real | Nearest Whole Number |
| ASIN(r) | Real | Arcsine |
| ATAN(r) | Real | Arctangent |
| ATAN2($r_1,r_2$) | Real | Arctangent of $(r_1/r_2)$ |
| CABS(c) | Real | Absolute Value $(r_r^2+r_i^2)^{1/2}$ |
| CCOS(c) | Complex | Cosine |
| CEXP(c) | Complex | Exponential |
| CHAR(i) | Character | Conversion Integer to Character*1 |
| CLOG(c) | Complex | Natural Logarithm |
| CMPLX($r_1,r_2$) | Complex | Conversion to Complex $c = (r_1,r_2)$ |
| COMPL(x) | Integer | Complement of x |
| CONJG(c) | Complex | Conjugate of Complex Argument $(r_r,-r_i)$ |
| COS(r) | Real | Cosine |
| COSH(r) | Real | Hyperbolic Cosine |
| CSIN(c) | Complex | Sine |
| CSQRT(c) | Complex | Square Root |
| DABS(d) | Double | Absolute Value |
| DACOS(d) | Double | Arccosine |
| DASIN(d) | Double | Arcsine |
| DATAN(d) | Double | Arctangent |
| DATAN2($d_1,d_2$) | Double | Arctangent of $(d_1/d_2)$ |
| DBLE(r) | Double | Conversion Real to Double |
| DCOS(d) | Double | Cosine |
| DCOSH(d) | Double | Hyperbolic Cosine |
| DDIM($d_1,d_2$) | Double | Positive Difference $max(d_1-d_2,0)$ |
| DEXP(d) | Double | Exponential |
| DIM($r_1,r_2$) | Real | Positive Difference $max(r_1-r_2,0)$ |
| DINT(d) | Double | Truncation to Whole Number |
| DLOG(d) | Double | Natural Logarithm |
| DLOG10(d) | Double | Common Logarithm (Base 10) |
| DMAX1($d_1,d_2,..$) | Double | Largest Value of $d_1,d_2,..$ |
| DMIN1($d_1,d_2,..$) | Double | Smallest Value of $d_1,d_2,..$ |
| DMOD($d_1,d_2$) | Double | Remaindering $d_1-int(d_1/d_2)*d_2$ |
| DNINT(d) | Double | Nearest Whole Number |
| DPROD($r_1,r_2$) | Double | Double Precision Product $r_1*r_2$ |
| DSIGN($d_1,d_2$) | Double | Sign Transfer $abs(d_1)*sign(d_2)$ |

| Aufruf | Typ | Bedeutung |
|---|---|---|
| DSIN(d) | Double | Sine |
| DSINH(d) | Double | Hyperbolic Sine |
| DSQRT(d) | Double | Square Root |
| DTAN(d) | Double | Tangent |
| DTANH(d) | Double | Hyperbolic Tangent |
| EQV($x_1$,$x_2$,..) | Integer | Equivalence |
| EXP(r) | Real | Exponential |
| FLOAT(i) | Real | Conversion Integer to Real |
| IABS(i) | Integer | Absolute Value |
| ICHAR(a) | Integer | Conversion Character*1 to Integer |
| IDIM($i_1$,$i_2$) | Integer | Positive Differenz max($i_1$-$i_2$,0) |
| IDINT(d) | Integer | Conversion Double to Integer |
| IDNINT(d) | Integer | Nearest Integer |
| IFIX(r) | Integer | Conversion Real to Integer |
| INDEX($a_1$,$a_2$) | Integer | Location of Substring $a_2$ in String $a_1$ |
| INT(r) | Integer | Conversion Real to Integer |
| ISIGN($i_1$,$i_2$) | Integer | Sign Transfer abs($i_1$)*sign($i_2$) |
| LEN(a) | Integer | Length of Character String |
| LGE($a_1$,$a_2$) | Logical | Lexical Comparison $a_1 \geq a_2$ |
| LGT($a_1$,$a_2$) | Logical | Lexical Comparison $a_1 > a_2$ |
| LLE($a_1$,$a_2$) | Logical | Lexical Comparison $a_1 \leq a_2$ |
| LLT($a_1$,$a_2$) | Logical | Lexical Comparison $a_1 < a_2$ |
| MASK(i) | Integer | Mask of i left-justified 1-bits |
| MAX0($i_1$,$i_2$,..) | Integer | Largest Value of $i_1$,$i_2$,.. |
| MAX1($r_1$,$r_2$,..) | Integer | Largest Value of $r_1$,$r_2$,.. |
| MIN0($i_1$,$i_2$,..) | Integer | Smallest Value of $i_1$,$i_2$,.. |
| MIN1($r_1$,$r_2$,..) | Integer | Smallest Value of $r_1$,$r_2$,.. |
| MOD($i_1$,$i_2$) | Integer | Remaindering $i_1$-int($i_1$/$i_2$)*$i_2$ |
| NEQV($x_1$,$x_2$,..) | Integer | Non-Equivalence |
| NINT(r) | Integer | Nearest Integer |
| OR($x_1$,$x_2$,..) | Integer | Boolean Sum of $x_1$,$x_2$,.. |
| REAL(c) | Real | Real Part $r_r$ of Complex Argument ($r_r$,$r_i$) |
| SECOND() | Real | CP-Time in Seconds from Start of Task |
| SHIFT(x,i) | Integer | Shift x left circular, right with sign extension |
| SIGN($r_1$,$r_2$) | Real | Sign Transfer abs($r_1$)*sign($r_2$) |
| SIN(r) | Real | Sine |
| SINH(r) | Real | Hyperbolic Sine |
| SNGL(d) | Real | Conversion Double to Real |
| SQRT(r) | Real | Square Root |
| SUBSTR(a,$i_1$,$i_2$) | Character | Substring a($i_1$:$i_2$) |
| TAN(r) | Real | Tangent |
| TANH(r) | Real | Hyperbolic Tangent |
| XOR($x_1$,$x_2$,..) | Integer | Exclusive Or |

# Neues vom Metaassembler METASM und seiner Applikationssoftware

F. Berger

Völlig unterschiedliche Assembler wie zum Beispiel der PACER-, der DEC-oder der INTEL-Assembler stellen an einen generell einsetzbaren Metaassembler hohe Anforderungen, um für diese verschiedene Assembler-, Cross Assembler- beziehungsweise Metaassemblerfiles für die Emulation anderer Operationscodes erstellen zu können. Hinzu kamen durch die häufiger werdende Anwendung des Metaassemblers zusätzliche Wünsche von Benutzerseite. In den letzten Ausgaben von INTERFACE wurde der an der Hybridrechenanlage implementierte Metaassembler METASM mit einigen Erweiterungen vorgestellt, dessen Eigenschaften bis auf die Möglichkeit der Definition bedingter Felder - z.B. für variable Operationscodes oder Operationscodelänge auf Grund vorangegangener Bitmuster - fast nichts mehr zu wünschen übrig lassen:

## 26 AFFIXE

Jede Instruktion kann nicht nur definitiv 26 Parameter enthalten - referenzierte Adressen oder instruktionsspezifische Konstante Werte - sondern auch 26 Affixe, welche zum Unterschied von Parametern eine variable Interpretation dieser Instruktion erst zum Zeitpunkt der Erstellung des Assemblerprogrammes ermöglichen. Diese werden, durch Beistriche getrennt, an den symbolischen Instruktionsnamen angehängt. Die verschiedenen Affixe werden wie bei den die Operanden definierenden A- oder D-Bits durch Markieren der F-Bits durch einen Kleinbuchstaben im Definitionsstring festgelegt und unterschieden.

## FREIE DEFINITION DER KONTROLLZEICHEN

Die bisher starr vorgegebenen Kontroll- oder Trennzeichen können mit Hilfe des Befehls CHARS: vor jedem Definitionsteil oder Assemblierungsteil umdefiniert werden, wie zum Beispiel die Steuerzeichen für Kommentarzeilen, Fortsetzungszeilen, Seitenvorschübe oder die Kennzeichnung von oktalen oder hexadezimalen Konstanten oder von ASCII-Strings, oder die Trennzeichen für Operanden, Affixe oder Zonen.

## FREI DEFINIERBARE PSEUDO-CODES

Um etwaigen Loader-Programmen beziehungsweise Memory-Image-Generatoren Lade- oder Bindeinformationen übergeben zu können, wurden frei definierbare Pseudo-Codes implementiert:

```
                    DEF:        PACER
*
*       PACER-Assembler Pseudo-Codes
*
        REL         a           2=a
        ABS         a           3=a
        BSS         b           4=*+a
        NAME        z           5=*
        EXTERN      z           6=*
        COMMON      a           7=0
                    .
                    .
        END
```

Der Kleinbuchstabe in Zone 2 gibt bei Pseudo-Codes die Maximalanzahl der möglichen Operanden an. In Zone 3 wird zuerst die Nummer des Pseudo-Codes definiert. Der folgende Ausdruck liefert dem Metaassembler die Information, mit welcher Adresse er die Assemblierung fortsetzen soll. Alle diese Informationen werden auch im Objektcode übergeben.

## NEUER META-OBJEKT-CODE

Der neue Objekt-Code enthält neben den Kontrollinformationen (Start/ End of Objekt, Start/End of File, Version, Datum, Zeit, Wordcount, Fehlercount, Checksumme, etc.) und den eigentlichen Objektframes auch Informationen über etwaige Feldüberläufe bei Adreß- oder Konstanteneinsetzungen und vor allem die Informationen, die über Pseudo-Codes einem Memory-Image-Generator, PROM-Programmer oder Mikroprozessor-Loader übergeben werden sollen (Ladeadresse, Anzahl der Operanden und Anzahl der Affixe, Länge und Namen der Operanden und Affixe und für jeden Operanden ein Kontrollwort und einen Wert). Mit Hilfe dieser Informationen ist es möglich, jeden beliebigen Memory-Image-Generator, PROM-Programmer oder Mikroprozessor-Loader zu entwickeln. Ein Memory-Image-Generator auf Metaobjektbasis für den PACER-100 an der Hybridrechenanlage ist bereits fertiggestellt.

## EMULATION DES INTEL 8086 OPERATIONSCODES

Da, wie eingangs erwähnt, der implementierte Metaassembler METASM die Definition von bedingten Feldern noch nicht zuläßt, - an der Definition einer entsprechenden Struktur wird momentan gearbeitet -, stieß die Entwicklung eines Metafiles für den INTEL 8086 Operationscode auf erhebliche Schwierigkeiten, da INTEL einen Operationscodesatz variabler Länge aufweist. Aus diesem Grund mußten für viele Operationscodes anstelle eines Befehls mehrere Befehle implementiert werden, was zu einer Aufblähung des Operationscodesatzes auf 267 Befehle führte (z.B. gibt es statt des einen MOV-Befehls 15 entsprechende MOVxx-Befehle).

Ein neu überarbeitetes, ausführliches Manual für die Benutzung des Metaassemblers METASM ist an der Hybridrechenanlage erhältlich.

BEISPIEL

Beispiel einer mikroprogrammierten Steuerung einer Kaffeemaschine (unter Verwendung einer Mikroprogrammsteuereinheit AM2910), die wahlweise Kaffee, Schokolade oder Suppe austeilt. Wird Kaffee gewählt, so kann auch Zucker und Sahne gewählt werden.

Es soll eine Steuerung für eine Kaffeemaschine entworfen werden, die eine einfache fehlerfreie wahlweise Ausschank macht. Sie soll folgendermaßen arbeiten:

1. Keine Aktion bis eine Münze eingeworfen wird.

2. Wenn eine Münze eingeworfen wird, soll ein Lämpchen aufleuchten und ein Becher ausgeworfen werden.

3. Innerhalb von 1.5 Sekunden muß der Becher in Position sein.

4. Es gibt keine Möglichkeit festzustellen, ob der Becher korrekt positioniert ist oder ob er überhaupt dort steht.

5. Bevor das Pulver ausgeworfen wird, wird für 1 Sekunde heißes Wasser aufgedreht (dadurch bleibt das Pulver nicht unschön am Grund des Bechers liegen).

6. Das heiße Wasser bleibt weiter für insgesamt 10 Sekunden aufgedreht.

7. Das Lämpchen leuchtet während des ganzen Vorganges.

8. Je nach Wahl - Kaffee, Suppe oder Schokolade - wird ausgeteilt:

| | |
|---|---|
| Kaffee | 2.5 Sekunden |
| Suppe | 2.0 Sekunden |
| Schokolade | 3.5 Sekunden |

9. Falls Kaffee gewählt wurde, wird nach dem Kaffeepulver Zucker oder Sahne ausgeworfen

| | |
|---|---|
| Zucker | 1.5 Sekunden |
| Sahne | 2.0 Sekunden |

10. Falls Zucker und Sahne gewünscht wurde, kommt zuerst Zucker, dann Sahne.

11. Nachdem der Becher mit Wasser gefüllt ist, soll 3.5 Sekunden lang gewartet werden (Zeit zum Entfernen des Bechers), bevor der Einwurf einer weiteren Münze getestet wird.

12. Der Takt des internen Zeitgebers beträgt eine halbe Sekunde.

Es gibt 6 Auswahlmöglichkeiten:

- Kaffee, schwarz
- Kaffee, Sahne
- Kaffee, Zucker
- Kaffee, Sahne und Zucker
- Schokolade
- Suppe

Literatur:

Advanced Micro Devices Customer Education,
The 2900 Family Study Guide and Teachers Manual, 1981, Seite 105ff

```
METAASSEMBLER METASM 8.12.09  JCS/VS 8 * PACER 132  84/05/14 15.21  PAGE     1
DEFINITION PASS      META FILE:  COFFDF     67

 2: *
 3: ******************************************
 4: *                                        *
 5: *  THE FAMOUS COFFEE MACHINE             *
 6: *                       DEFINITIONSTEIL  *
 7: *                                        *
 8: ******************************************
 9: *
10: *
11: *
12: *  CONTROL PSEUDO-CODES
13: *
14:                       WORD:          6       Wortlaenge: 6 Halbbytes
15: *
16: *  AM2910   FRAME-DEFINITIONEN
17: *
18: <Frame>               JZ             :4L
19: <Frame>               CJP            :LLHH
20: <Frame>               RPCT           :HLLH
21: <Frame>               LOCT           :HHLL
22: <Frame>               CONT           :3HL
23: *
24: *  MULTIPLEXER
25: *
26: <Frame>               NOCOIN         :3L
27: <Frame>               NULL           :LLH
28: <Frame>               SOUPT          :LHL
29: <Frame>               CHOCT          :LHH
30: <Frame>               CREMT          :HLL
31: <Frame>               SUGRT          :HLH
32: <Frame>               CAFET          :HHL
33: <Frame>               PASS           :3H
34: *
35: *  CONTROL BITS
36: *
37: <Frame>               OFF            :8L
38: <Frame>               BUSY           :H7L
39: <Frame>               CUP            :HH6L
40: <Frame>               WATER          :HLH L
41: <Frame>               COFFEE         :HLHH4L
42: <Frame>               SUGAR          :HLHLH3L
43: <Frame>               CREAM          :HLHLLHLL
44: <Frame>               CHOCO          :HLH3LHL
45: <Frame>               SOUP           :HLH4LH
46: *
47: *  MICROINSTRUKTIONEN
48: *
49:       <META>          JUMP    a      1:(CJP)3Fa6Aa8Fb3X
50:       <META>          NOP            1:(CONT)(NULL)6X8Fa3X
51:       <META>          LOAD    a      1:(LOCT)(NULL)6Aa8Fa3X
52:       <META>          REPEAT a       1:(RPCT)(NULL)6Aa8Fa3X
53:       <META>          JUMPZ          1:(JZ)(NULL)6X8Fa3X
54:                       END
```

```
METAASSEMBLER METASM 8.12.09  JCS/VS 8 * PACER 132  84/05/14 15.21  PAGE     3
ASSEMBLER PASS       2  META FILE:  COFFDF   BINARY OBJECT FILE:  COFFOB     67

 2: *
 3: ******************************************
 4: *                                        *
 5: *  THE FAMOUS COFFEE MACHINE             *
 6: *                       ASSEMBLIERUNGSTEIL *
 7: *                                        *
 8: ******************************************
 9: *
10: 0000  3000 00          0000   ZERO   JUMP,NOCOIN,OFF      ZERO
11: 0001  E206 00                         NOP,CUP
12: 0002  E204 00                         NOP,BUSY
13: 0003  E204 00                         NOP,BUSY
14: *
15: *  DREHE WASSER AUF UND VERZWEIGE ZU GEWUENSCHTER ROUTINE
16: *
17: 0004  36D5 00          001A           JUMP,CHOCT,WATER     CHOC
18: 0005  34FD 00          001F           JUMP,SOUPT,WATER     SOUP
19: *
20: *  DIE TASTE FUER KAFFEE WURDE GEDRUECKT
21: *
22: 0006  C265 80          000C   COFFEE LOAD,COFFEE          12
23: 0007  E205 80                         NOP,COFFEE
24: 0008  E205 80                         NOP,COFFEE
25: 0009  3A70 80          000F           JUMP,SUGRT,COFFEE    SUGAR
26: 000A  38C5 90          0018           JUMP,CREMT,COFFEE    CRE2
27: *
28: *  BEENDE DAS ANFUELLEN DES BECHERS MIT WASSER
29: *
30: 0008  9253 00          0008   LOOP   REPEAT,WATER         LOOP
31: 000C  C224 00          0004          LOAD,BUSY            4
32: *
33: *  VOR NEUEM START ERLAUBE DEN BECHER ZU ENTFERNEN
34: *
35: 000D  926C 00          000D   BUSY   REPEAT,BUSY          BUSY
36: 000E  0204 00                        JUMPZ,BUSY
37: *
38: *  TEILE ZUCKER AUS
39: *
40: 000F  E205 80                 SUGAR  NOP,COFFEE
41: 0010  C240 40          0009          LOAD,SUGAR           9
42: 0011  3990 40          0013          JUMP,CREMT,SUGAR     CREAM
43: 0012  3E5D 40          0008          JUMP,PASS,SUGAR      LOOP
44: *
45: *  TEILE SAHNE AUS
46: *
47: 0013  C22D 40          0005   CREAM  LOAD,SUGAR           5
48: 0014  E205 20                        NOP,CREAM
49: 0015  E205 20                        NOP,CREAM
50: 0016  E205 20                 ENTRY  NOP,CREAM
51: 0017  3E5D 20          0008          JUMP,PASS,CREAM      LOOP
52: 0018  C245 20          0008   CRE2   LOAD,CREAM           8
53: 0019  3E85 20          0016          JUMP,PASS,CREAM      ENTRY
54: *
55: *  TEILE SCHOKOLADE AUS
56: *
57: 001A  E205 00                 CHOC   NOP,WATER
58: 001B  C22D 10          0005          LOAD,CHOCO           5
59: 001C  92E5 10          001C   CHOCLP REPEAT,CHOCO         CHOCLP
60: 001D  C245 00          0008          LOAD,WATER           8
61: 001E  3E5D 00          0008          JUMP,PASS,WATER      LOOP
62: *
63: *  TEILE SUPPE AUS
64: *
65: 001F  E205 08                 SOUP   NOP,SOUP
66: 0020  E205 08                        NOP,SOUP
67: 0021  C26D 08          000D          LOAD,SOUP            13
68: 0022  3E5D 08          0008          JUMP,PASS,SOUP       LOOP
69: 0023                                 END
```

# A C S L – gibt es Neuigkeiten ?

H. Hummer

Es gibt sie. Eine ausführliche Beschreibung der Release 8 wird allerdings erst im nächsten INTERFACE erscheinen. An der Fertigstellung des deutschsprachigen Manuals, welches Release 8 beinhaltet, wird gearbeitet. Selbstverständlich wird der letzte Informationsstand wie bisher auch telefonisch weitergegeben.

Neu ist, daß die Arbeiten an jenem Ausgabemodul von ACSL beendet wurden, welcher eine Übertragung der an der CYBER erzeugten Plotterfiles mittels Magnetband an die Hybridrechenanlage ermöglicht. In Einzelfällen wurde diese Übertragung zwar auch schon bisher durchgeführt, doch dabei mit permanenten Dateien gearbeitet.

Dieses Verfahren erweist sich, wenn es von mehreren Personen zugleich in Anspruch genommen wird, als unbrauchbar.

- Die Plotter Files überschreiten sehr schnell jene Größe, ab der direkte Files verwendet werden müssen. Das führt zu organisatorischen Problemen, da vom Digitalrechenzentrum nur eine Usernummer pro Institut für direkte Files zugelassen wird. So müßten alle Benützer eines Instituts (auch Praktikanten) das Password dieser Nummer verwenden, wobei im Regelfall gerade diese Nummer zum Archivieren großer Datenbestände dient. Damit würde ein beträchtliches Sicherheitsproblem entstehen.

- Ob direkte oder indirekte Files, es müßte ein Mechanismus entwickelt werden, der die Daten gleichzeitig arbeitender Kunden eindeutig und automatisch kennzeichnet und den Zugriff kontrolliert.

- Mehrere von einem Kunden schnell nacheinander angelegte Datenbestände müßten mit diesem Mechanismus ebenso kontrolliert werden, da zum Zeitpunkt des Anlegens eines neuen Plotterfiles die alten Daten eventuell noch nicht auf Band gerettet worden sind und überschrieben werden könnten.

Schnelleren Erfolg verspricht der Weg, während eines ACSL-Laufes ein Programm zu "Konstruieren", welches beim Beenden der Arbeit in die Eingabe-Warteschlange gestellt wird. Diesem können Plotterdaten mitgegeben werden, ohne daß sie überhaupt plattenresident gemacht werden müssen ("SUBMIT-Job").

Bei der Durchführung trat das Problem auf, die Plotter-Ausgabedaten in ein Format umzusetzen, welches als Bestandteil eines SUBMIT-Jobs zugelassen ist. Die mit HYPLOT und HYLIB erzeugten Daten sind für die Übertragung mit Magnetband optimiert und binär kodiert. Sie erfüllen die obige Bedingung nicht. Das Problem wurde durch Erzeugen eines Zwischenfiles im ASCII-Format gelöst. Daten in diesem Format dürfen Bestandteil eines SUBMIT-Jobs sein und werden jedesmal an dessen Steuerkarten durch Kopieren angehängt. Das Zwischenformat wird nachverarbeitet und Daten in "echtem HYLIB-Format" auf Band abgelegt. Der zusätzliche Zeitverbrauch ist nicht groß.

Außerdem muß man dem abzusetzenden Programm Informationen über Usernummer, Password, Charge, Projektnummer und die Nummer (VSN) des zu beschreibenden Bandes mitgeben. Diese Werte sind im Regelfall nicht die beim Starten des ACSL-Simulators verwendeten. Besonders selten ist das bei den Praktikumsnummern mit ihren allseits eng geschnürten Accounting-Limits der Fall.

Benutzen Sie bitte folgende Parameter, wenn Sie ACSL rechnen und an der Hybridrechenanlage plotten wollen:

- CN=cn Chargenummer für den SUBMIT-Job
- US=us Usernummer für den SUBMIT-Job
  Zweistelliges Anhängsel an Chargenummer
- PW=pw Password für den SUBMIT-Job
- PN=pn Projektnummer für den SUBMIT-Job
- VSN=vsn VSN Nummer des zu beschreibenden Bandes

Ein Beispiel für den praktischen Gebrauch :

```
ACSL(I=mod,IR=run,PLT=HYB,CN=E113S,US=02,PW=X,PN=1PR,VSN=AG0)
```

Die neuen Parameter werden nur benötigt, wenn PLT=HYB verwendet wird. In diesem Falle müssen sie Werte zugewiesen erhalten, sonst wird bei z.B. fehlender Charge mit der Fehlernachricht

```
SUBMIT JOB: CHARGE NUMBER IS EMPTY
```

abgebrochen. Für die anderen Parameter gibt es sinngemäß gleiche Fehlernachrichten. Werden Werte zugewiesen, erfolgt eine Eintragung der Form

```
COMMENT: SUBMIT JOB: CHARGE cn, USER us, PROJECT pn, VSN vsn
```

in das Dayfile.

Beim Eintragen des SUBMIT-Jobs in die Eingabe-Warteschlange erscheint eine Meldung der Form

```
SUBMIT COMPLETE. JOBNAME IS Job
```

am Terminal.

Die Ausgabe wird wieder in die Terminal-Warteschlange gestellt. Das ist beim interaktiven Arbeiten sinnvoll. Wird ACSL im Batch betrieben, läuft es oft unter Jobnummern, die IAF nicht verwenden dürfen (z.B. Praktikanummern). Diese Kunden könnten dann das Ergebnis ihrer Arbeit nicht kontrollieren. Deshalb wird bei ACSL-Jobs mit "BATCH Origin" ein ROUTE durchgeführt. Das Ergebnis wird dadurch automatisch am Schnelldrucker gedruckt.

Bei Angabe ungültiger Accounting-Informationen im ACSL-Aufruf führt IAF automatisch ein LOGOUT des Terminals beim SUBMIT (ROUTE) durch, um Mißbrauch des Systems zu verhindern.

Dieser Fall ist besonders unangenehm, weil RECOVER nicht mehr möglich ist (!) und damit alle Zeichnungen verloren sind.

Manchmal kommt es vor, daß ein mit PLT=HYB gestarteter Lauf wegen falscher Modelldefinition etc. abgebrochen werden muß. Bei Beendigung mit END wird für PLT=HYB aber immer die Nachverarbeitung gestartet. Deshalb hat sich für PLT=HYB bewährt, gleich am Anfang

ZZTLXP = .F.

zu setzen - das "weiche" CTL-T Trapping, welches Fehlerzustände im ACSL-Lauf behandelt und ein Abbrechen mit Rücksprung ins Betriebssystem verhindert, ist dann ausgeschaltet.

Der Kunde holt schließlich das richtig beschriebene Band (Ausdruck kontrollieren) ab und gibt es dem Operator der Hybridrechenanlage zum Plotten. Die notwendigen Steuerkarten (3 Stück) sowie weitere Hinweise sind der an der Hybridrechenanlage aufliegenden Beschreibung des Programmes CDCPLT zu entnehmen.

---

## KURSE

Folgende Kurse werden von der Hybridrechenanlage nach Vereinbarung abgehalten:

| | |
|---|---|
| RH1 | GERÄTETECHNIK EAI PACER 600A AUTOPATCH-SYSTEM |
| RH2 | BENÜTZUNG DES BETRIEBSSYSTEMS JCS/VS 8 |
| RH3 | HINWEISE FÜR FORTRAN-PROGRAMMIERER AN DER HYBRIDRECHENANLAGE |
| RH6 | ASSEMBLER-PROGRAMMIERUNG |
| RH7 | SOFTWAREUNTERSTÜTZUNG FÜR DIE BENÜTZUNG DES PACER 600 ALS PLOTTER SYSTEM |
| RH9 | EINFÜHRUNG IN DAS HYBRIDE AUTOPATCH-SYSTEM |
| RH13<br>RH14 | BEDIENUNG DES HYBRIDEN PROZESSORS HYBSYS<br>ZUR SIMULATION DYNAMISCHER SYSTEME AM AUTOPATCH-SYSTEM, TEIL 1 BZW. TEIL 2 |
| RH15 | PROGRAMMENTWICKLUNG AM TERMINAL |

Nähere Auskünfte und Anmeldungen zu den Kursen telefonisch oder persönlich bei Herrn M. Schandl (1040 Wien, Gußhausstraße 27-29, 4. Stock, Zimmer 1404/05, Tel: 5601 / 3706 oder 3669 DW).

# MODULARE SIMULATION VON INDUSTRIEROBOTERN

I. Troch
P. Kopacek
K. Desoyer
F. Breitenecker
F. Rattay

## 1. Einleitung und Problemstellung

Nach VDI-Richtlinie 2860/9 sind "Industrieroboter universell einsetzbare Bewegungsautomaten mit mehreren Achsen, deren Bewegungen hinsichtlich Bewegungsfolge und Wegen beziehungsweise Winkeln frei (d.h. ohne mechanischen Eingriff) programmierbar und gegebenenfalls sensorgeführt sind." Sie sollen den Menschen von immer wiederkehrenden Handhabungsaufgaben entlasten.

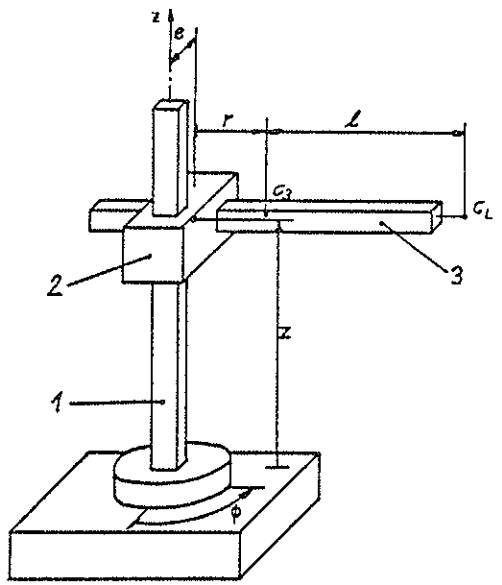
Industrieroboter werden heute in den verschiedenen Bereichen der Fertigungstechnik, besonders in der Autoindustrie, erfolgreich eingesetzt. Die Entwicklung ist aber noch lange nicht abgeschlossen. Man ist in Zukunft bestrebt, "gefühlvollere" und "intelligentere" Roboter zu entwickeln. Dafür sind neue Erkenntnisse und Verbesserungen in der Sensortechnik sowie in der Steuer- und Regelungstechnik von größter Wichtigkeit.

Die optimalen Bewegungsabläufe im Arbeitsraum eines Roboters können am fertigen Roboter selbst oder über eine Simulation am Computer untersucht werden. Thema mehrerer Arbeiten war es, zum Teil analog und zum Teil digital unter Verwendung der Sprachen HYBSYS bzw. ACSL zunächst die freie Bewegung eines Roboters zu simulieren und anschließend die gesteuerte (geregelte) Bewegung zwischen zwei vorgegebenen Punkten des Arbeitsraumes zu berechnen.

Den Untersuchungen wurde zunächst ein Roboter der verbreiteten kinematischen Struktur vom Typ DSS, wie er in Abbildung 1 schematisch dargestellt ist, zugrundegelegt, jedoch sollen die Programme in relativ einfacher Weise auf andere Typen übertragbar sein.

Die Abkürzung DSS steht dabei für eine Drehbewegung ($\phi$) und zwei Schubbewegungen (z,r). Die Antriebe der einzelnen Achsen können elektrisch, elektro-hydraulisch oder - seltener - elektro-pneumatisch erfolgen, wobei die mathematische Modellierung einerseits von der gewählten Antriebsart abhängt, andererseits aber auch unterschiedlich genau erfolgen kann.

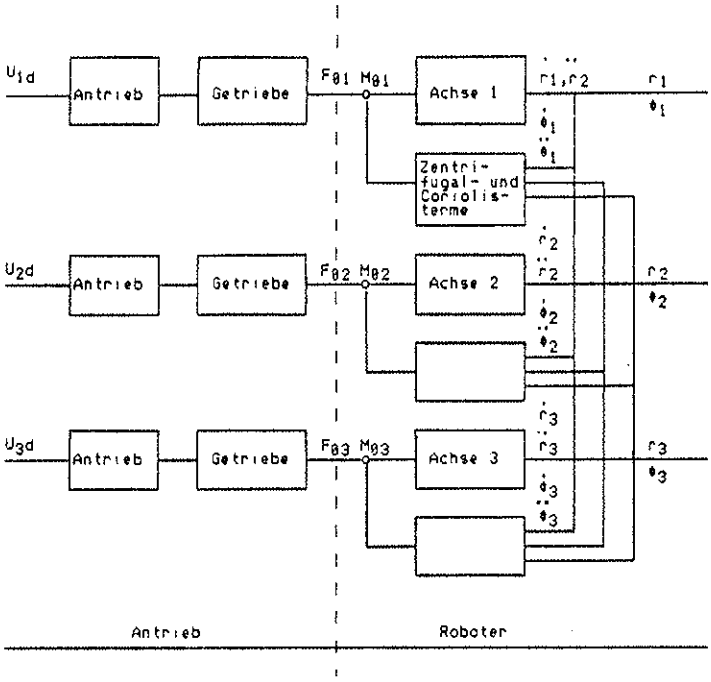
So werden z.B. elektrische Antriebe durch Verzögerungsglieder 2. Ordnung, elektrohydraulische Antriebe in erster Näherung als Integralglied mit Verzögerung 1. oder 2. Ordnung modelliert, genauere Modelle führen auf stark nichtlineare Differentialgleichungssysteme der Gesamtordnung 6.




Schematische Darstellung eines DSS-Greiferführungsgetriebes
Abbildung 1

Die Beschreibung des Getriebes hängt von der gewählten Antriebsart und der Art (Schub oder Drehung) des betreffenden Robotergliedes ab.

Will man für unterschiedliche Typen von Greiferführungsgetrieben und unterschiedliche Antriebsarten Regelalgorithmen testen und vergleichen, so ist es wünschenswert, die Simulation modular aufzubauen, um so den Programmieraufwand gering zu halten. Dies bedeutet, daß die Teile "Antrieb", "Getriebe" und "Roboterglied" bezüglich ihres Ein/Ausgangsverhaltens zu modellieren und programmieren sind, wobei die Wechselwirkungen zwischen ihnen ebenfalls zu berücksichtigen sind, vgl. Abbildung 2.



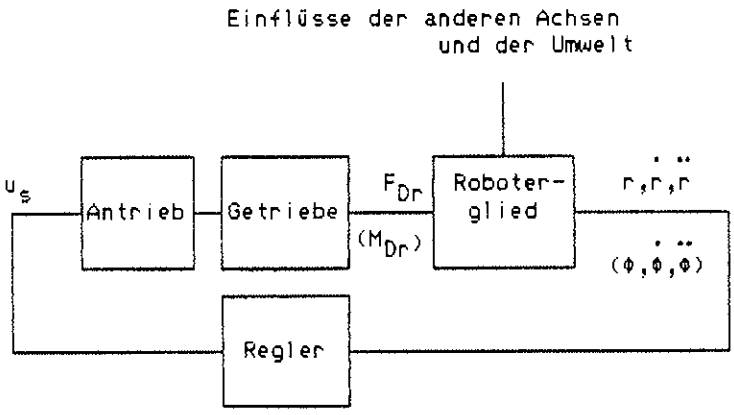

Schema eines Roboters mit Antrieben
Abbildung 2

Ein Blick auf die kinematischen Gleichungen der unterschiedlichen Arten von Greiferführungsgetrieben zeigt, daß es zwar sinnvoll ist, die Antriebe und Getriebe für jede Achse gesondert zu simulieren, jedoch die "Mechanik" des Roboters als eine Einheit gesehen und simuliert werden sollte.

Diesem Modell sind dann die Regler hinzuzufügen, deren Eingänge roboterspezifische Daten wie Längen, Winkel, Geschwindigkeiten, Beschleunigungen oder Kräfte sind und deren Ausgänge die elektrischen Steuerspannungen der Antriebe sind.

## 2. Modell

Ein solches modulares Simulationskonzept bildet demnach zwar das Greiferführungsgetriebe als Einheit nach, jedoch kann für jede Bewegungsachse die in Abbildung 3 angegebene Struktur abgelesen werden.




Prinzip der Regelung
einer Roboterachse
Abbildung 3

Für den in Abbildung 1 skizzierten Roboter vom DSS-Typ lauten mit den Bezeichnungen

| | |
|---|---|
| $\phi$, z, r | Lagekoordinaten |
| $m_1$, $m_2$, $m_3$ | Massen der Roboterglieder |
| $m_L$ | Masse Greifer + Last |
| $C_3$, $C_L$ | Schwerpunkt der Masse $m_3$ bzw. $m_L$ |
| $I_1$, $I_2$ | Trägheitsmoment des Gliedes 1 bzw. 2 um die Drehachse (z-Achse) |
| $I_3$ | Trägheitsmoment des Gliedes 3 um die Vertikale durch $C_3$ |
| $I_L$ | Trägheitsmoment der Masse $m_L$ um die Vertikale durch $C_L$ |
| e | Exzentrizität der Längsschwerachse des Gliedes 3 (mit daraufliegendem Schwerpunkt $C_L$) von der Drehachse |
| l | Abstand von $C_3$ zu $C_L$ |
| $M_\phi$, $K_z$, $K_r$ | Antriebsmoment, Antriebskräfte |
| $M_{R\phi}$, $K_{Rz}$, $K_{Rr}$ | Reibungsmoment, Reibungskräfte |
| $M_{L\phi}$, $M_{Lz}$, $M_{Lr}$ | Lastmomente |

die Bewegungsgleichungen folgendermaßen:

$$M_\phi = M_{R\phi} + \left[I_1+I_2+I_3+I_L+(m_3+m_L)(r^2+e^2)+ +m_L(2r+l)l\right]\ddot{\phi}-(m_3+m_L)e\ddot{r}+2\left[(m_3+m_L)r+m_Ll\right]\dot{r}\dot{\phi}$$

$$K_z = K_{Rz}+(m_2+m_3+m_L)(\ddot{z}+g)$$

$$K_r = K_{Rr}+(m_3+m_L)(\ddot{r}-e\ddot{\phi})-\left[(m_3+m_L)r+m_Ll\right]\dot{\phi}^2$$

Um den simulierten Bewegungsablauf eines Industrieroboters möglichst dem tatsächlichen Ablauf anzunähern, ist in den Dreh- und Schubgelenken des Roboters die Reibung zu berücksichtigen.

In der Regelungstechnik hat sich ein vereinfachter Reibungsansatz durchgesetzt. Die Reibkraft wird durch einen im Sinne eines Mittelwertes konstanten Anteil $F_o$ (Coulomb'scher Reibkraftanteil) und einen geschwindigkeitsproportionalen Anteil $f_v$ angenähert.

Linearbewegung:

$$F_R = f_v \cdot v + F_o \cdot \mathrm{sign}(v)$$

Drehbewegung:

$$M_{R\phi} = f_M \cdot \dot{\phi} + M_o \cdot \mathrm{sign}(\dot{\phi})$$

Symbolik:

| | |
|---|---|
| $F_R$ (N) | Reibkraft des Schubgliedes |
| $v$ (m/s) | Geschwindigkeit |
| $f$ (Ns/m) | Konstante des geschwindigkeitsproportionalen Anteiles der Reibkraft |
| $F_o$ (N) | Coulomb'scher Anteil der Reibkraft |
| $M_{R\phi}$ (Nm) | Reibmoment des Drehgliedes |
| $\dot{\phi}$ (rad/s) | Winkelgeschwindigkeit |
| $f_M$ (Nms/rad) | Konstante des zur Winkelgeschwindigkeit proportionalen Anteiles des Reibmomentes |
| $M_o$ (Nm) | Coulomb'scher Anteil des Reibmomentes |

Wählt man für alle Antriebe Gleichstrommotore, so kann jeder durch Gleichungen der Form, /1/,

$$\frac{L_A}{R_A}\,\dot{I}_A + I_A = \frac{1}{R_A}\,(U_A - c.\varphi.\omega)$$

$$I.\dot{\omega} = M_A - M_L$$

$$M_A = c.\varphi.I_A$$

$$T_{ST}.\dot{U}_A + U_A = K_{ST}.U_S$$

beschrieben werden. Hierin bezeichnet $U_A$ die Ankerspannung, $U_S$ die Steuerspannung, $I_A$ den Ankerstrom, $\omega$ die Winkelgeschwindigkeit der Motorwelle, $M_A$ das Antriebs- und $M_L$ das Lastmoment. $L_A$, $R_A$, $c$, $I$, $T_{ST}$ und $K_{ST}$ sind motorspezifische Konstante, die i.a. für jeden der drei Motore verschieden sind.

Die Kopplung über das Getriebe führt auf Beziehungen der Form

$$\dot{\phi} = \frac{\omega}{k}$$

$$M_L = \frac{M_{Dr}}{k\eta}$$

für ein Drehgelenk und

$$\dot{r} = \frac{\omega}{k_1}$$

$$M_L = \frac{F_{Dr}}{k_1\eta}$$

für ein Schubgelenk, worin $k$ und $k_1$ vom Übersetzungsverhältnis bestimmt sind und $\eta$ den Wirkungsgrad angibt ($0 < \eta \leq 1$).

## 3. Simulation

Durch Zusammenfassen der in Abschnitt 2 angeführten Beziehungen kann man nun entweder auf ein System von 3 Differentialgleichungen 3. Ordnung für r, $\phi$ und z kommen, das die Auswirkung der 3 angelegten Steuerspannungen $u_{sr}$, $u_{s\phi}$, $u_{sz}$ auf die drei Bewegungen $r=r(t)$, $\phi=\phi(t)$, $z=z(t)$ beschreibt. Die andere Möglichkeit besteht darin, den modularen Aufbau in das Programm zu übernehmen, also implizite Techniken zu verwenden. Die erstgenannte Vorgangsweise hat den Nachteil, daß das Programm bei einer Änderung z.B. in der Struktur der Antriebe neu geschrieben werden muß, und zwar zumindest der die betroffene Roboterachse beschreibende Teil, häufig jedoch auch mehr. Solche Änderungen sind naturgemäß bei einer modular aufgebauten Simulation einfach durchführbar.

Bei Verwendung von HYBSYS hat sich der modulare Programmaufbau sehr bewährt, insbesondere waren Genauigkeit und Stabilität der Simulation praktisch unabhängig von der Geschicklichkeit des Programmierenden, wie ein Vergleich von /2/ und /3/ zeigt.

Schwieriger ist die Realisierung des modularen Konzeptes in ACSL, da diese Sprache keine derartige implizite Modellierung (von einem sehr einfachen, hier nicht vorliegenden Spezialfall abgesehen) vorsieht. Es muß daher bei jedem Integrationsschritt ein Gleichungssystem über eine vom Benutzer bereitzustellende Routine gelöst werden, was sich naturgemäß sehr ungünstig auf die Rechenzeit auswirkt. Es ist daher in diesem Fall i.a. wohl zweckmäßiger, die Gleichungen händisch aufzulösen und hiebei, soweit möglich, auf eine "änderungsfreundliche" Struktur zu achten.

## 4. Simulationsergebnisse

In der Arbeit /3/ wurden an dem in /1/ beschriebenen Modell eines Roboters mit der kinematischen Struktur DSS mit elektrischen Antrieben konventionelle, lineare Regelalgorithmen durch Simulation am Hybridrechner verglichen.

Es wurden ein Anfangspunkt A und ein Endpunkt B in der x-y-Ebene vorgegeben, wodurch eine Regelung der z-Achse entfällt. Man benötigt daher nur einen Regler für die r-Achse und die $\phi$-Koordinate, da die z-Achse von den beiden anderen entkoppelt ist. Dies stellt keine Einschränkung der Allgemeinheit dar, sondern dient nur der Minimierung des Programmieraufwandes. Untersucht wurden folgende Regelalgorithmen:

a) Proportional-(P-) Regler

$$y = K_p x_d$$

$y$ ... Stellgröße
$x_d$... Regeldifferenz
$K_p$... Verstärkungsfaktor

b) Proportional-Differential-(PD-) Regler

$$y = K_p x_d + K_D \dot{x}_d$$

$K_D$... Differenzierbeiwert

c) Proportional-Integral-(PI-) Regler

$$y = K_p x_d + K_I \int_0^t x_d(\tau) d\tau$$

$K_I$... Integrierbeiwert

Bei den durchgeführten Untersuchungen wurden jeweils für beide Achsen die gleichen Regelalgorithmen, jedoch mit unterschiedlichen Einstellparametern ( $K_P$, $K_D$, $K_I$ ) verwendet. Der Verlauf der Bahnkurve zwischen den Punkten A und B ist im Wesentlichen durch die eingestellten Reglerparameter bestimmt. Stellvertretend seien hier einige Simulationsergebnisse der Regelung beider Achsen mit einem P-Regler wiedergegeben. Abbildung 4 und 5 zeigen den Verlauf des Radius r und des Winkels $\phi$ über der Zeit für eine Bewegung des Roboters von einem Punkt A (RA=0.05m, PHIA=0.393rad) zu einem Punkt B (RB=0.25m, PHIB=1.178rad). Die an den P-Reglern eingestellten Verstärkungen waren für die r-Achse KPR=65 und für die $\phi$-Achse KPPHI=66.2. Abbildung 6 zeigt die zugehörige Bahnkurve in der X-Y-Ebene. Der Einfluß einer Verstärkungsänderung auf die Bahnkurve ist in Abbildung 7 dargestellt. Bei gleicher Verstärkung der $\phi$-Achse KPPHI=66.2 wurde die Verstärkung des Reglers der r-Achse KPR auf 260 erhöht. Charakteristisch ist die für Regler ohne I-Anteil auftretende bleibende Regelabweichung. Die Ergebnisse für den PI- und PD-Regler sind ähnlich, jedoch tritt beim PI-Regler keine bleibende Regelabweichung auf.

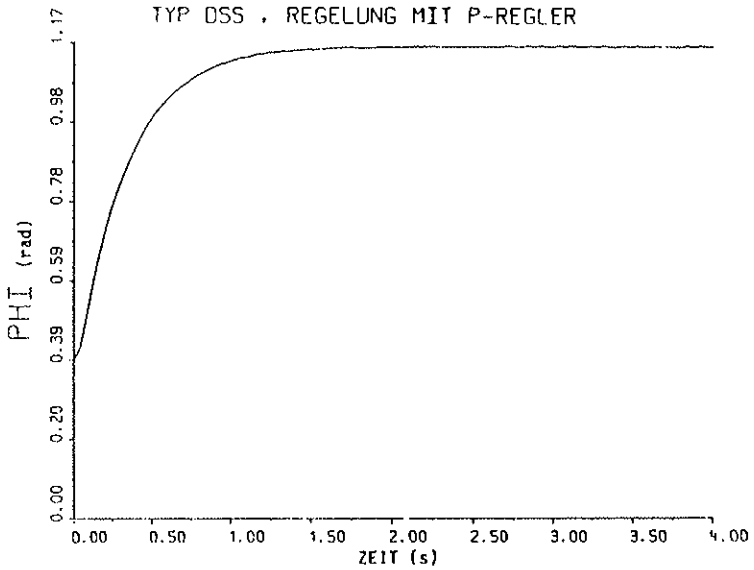



Abbildung 5

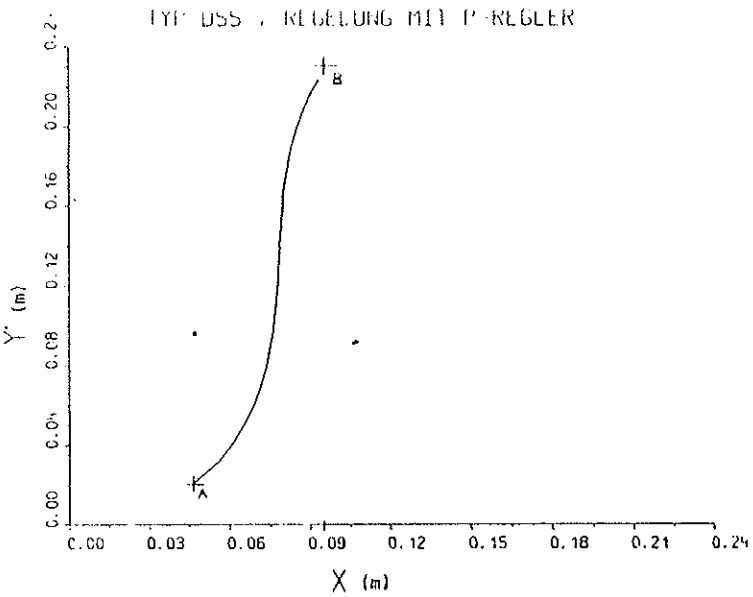



Abbildung 6

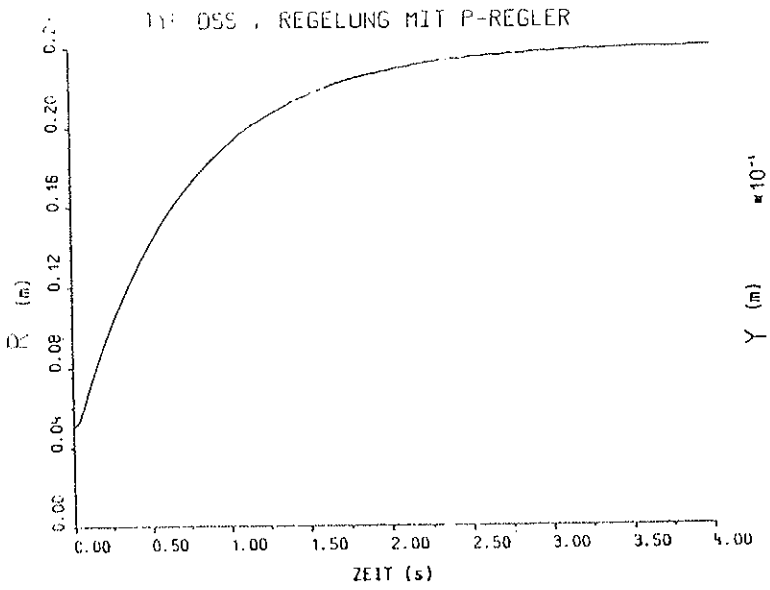



Abbildung 4

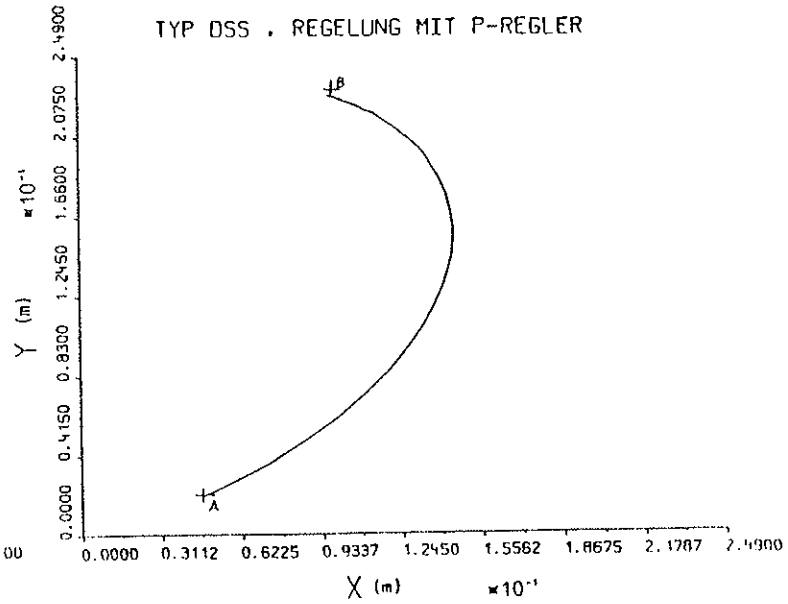



Abbildung 7

Zum Vergleich eines "konventionellen" PID-Algorithmus mit einem "fortgeschrittenen" Zustandsregler wurde der gleiche Robotertyp (DSS) mit Hilfe der Programmiersprache ACSL digital simuliert /4/. Die Ergebnisse zeigen eine annähernde Gleichwertigkeit beider Reglertypen, aber bessere Regelergebnisse als mit P-, PD- und PI- Algorithmen.


Summary

The movement of a robot is simulated and not only the kinetic of the arm but also the dynamic behaviour of the drives and controllers are modelled. Straight-forward manipulation of the resulting equations leads to a set of coupled differential equations, which changes structure not only in case the kinematic structure of the robot but also in case the type of one drive or one controller is changed. So a modular approach to simulation has been implemented on the hybrid computer using HYBSYS, allowing for easy and quick changes in structure. The results showed good accuracy and stability of the simulation. Further, a comparison with a digital simulation based on ACSL has been made, the latter has the drawback, that explicit differential equations have to be formulated. Finally, a short report on the performance of various control strategies is given.


Literatur


/1/ O.Föllinger, F.Dörrscheidt, M.Klittich: Regelungstechnik, Elitera Berlin, 1978

/2/ N.Hamata, Z.Resatko: Interne Forschungsarbeit über die Simulation von Industrierobotern (TU Wien, 1984)

/3/ R. Hittmair: Hybride Simulation des dynamischen Verhaltens von Industrieroboter, Diplomarbeit (TU Wien, 1984)

/4/ Ch. Schwamminger: Digitale Simulation fortgeschrittener Regelalgorithmen für Industrieroboter, Diplomarbeit (TU Wien, 1984)

# HEPATISCHE GLUCOSEPRODUKTION IN VITRO – MATHEMATISCHE MODELLIERUNG MITTELS REGELUNGSTECHNISCHER ÜBERTRAGUNGSGLIEDER DURCH SIMULATION AUF EINEM HYBRIDRECHNER*)

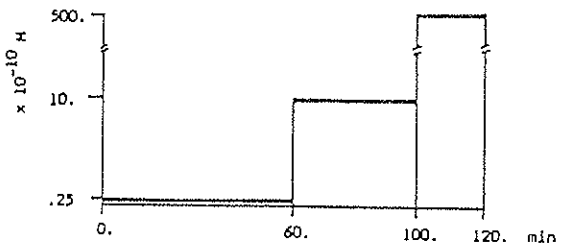
Dipl.Ing. Jutta Gampe
Kurzfassung Diplomarbeit
(Betreuer: Prof.Dr. I.Troch, DI.Dr. F.Breitenecker
Abteilung für Regelungsmathematik, Hybridrechen- und Simulationstechnik
Institut für Analysis, Technische Mathematik und Versicherungsmathematik
Technische Universität Wien)

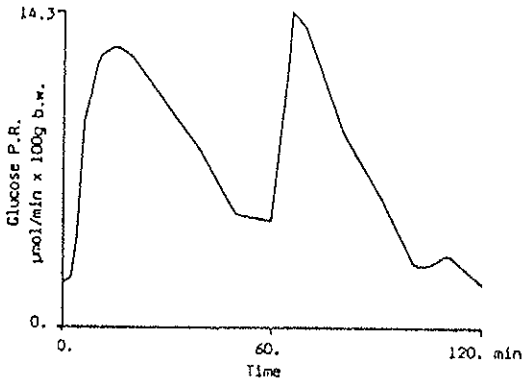
Bei der Untersuchung der hormonell gesteuerten hepatischen (= zur Leber gehörigen) Glucoseproduktion läßt sich trotz kontinuierlicher Hormonexposition ein Nachlassen der Wirkung dieses Reizes auf die Freisetzung von Glucose beobachten. Die Ursachen für diesen sogenannten "evanescent effect" sind weitgehend unbekannt, und zur näheren Erkundung des Phänomens sollte der Versuch der mathematischen Modellbildung unternommen werden.

Die auf einen Hormon- (=Glucagon-)reiz erfolgende Glucoseausschüttung läuft – schematisch reduziert auf ihre wichtigsten Teile – folgendermaßen ab: Der auf die Leberzellen auftreffende Glucagonreiz wird über eine Enzymkaskade verstärkt, von der ein daran beteiligtes Zwischenprodukt, das cyclische AMP (3',5'-Adenosin-Manophosphat), meßbar ist. Mittels dieser Enzyme wird der Abbau der in einer polymeren Form (Glycogen) gespeicherten Glucose angeregt und diese an den Blutkreislauf abgegeben.

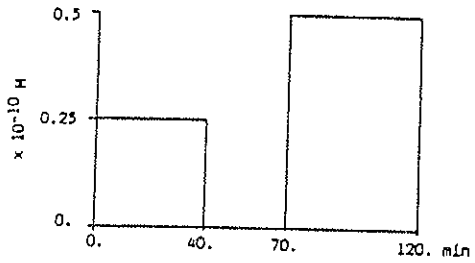
Zur Untersuchung dieser Vorgänge wurde die hepatische Glucoseproduktion in vitro an Rattenlebern künstlich stimuliert und die Produktionsraten von Glucose und cyclischem AMP aufgezeichnet. Zwei Meßreihen standen zur Verfügung, die sich in der Gestalt des den Vorgang auslösenden Glucagonreizes unterschieden. Von jeder dieser Meßreihen ist hier exemplarisch je ein Versuch herausgegriffen:

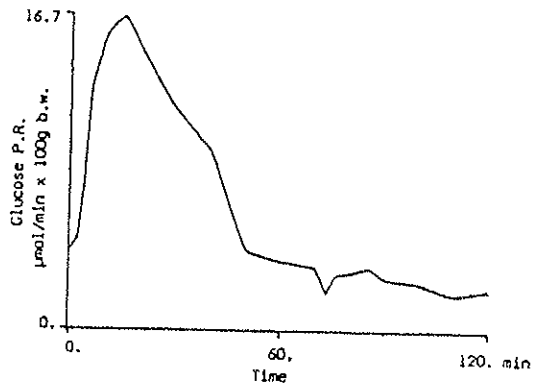
1. Meßreihe



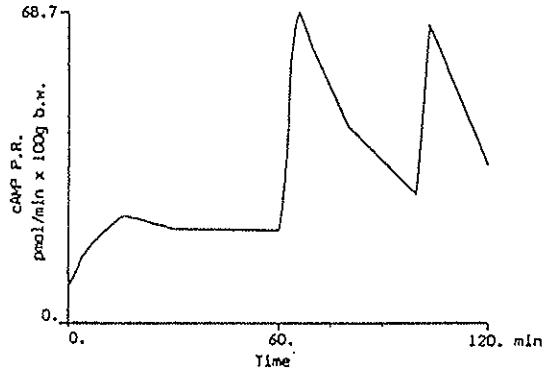

2. Meßreihe



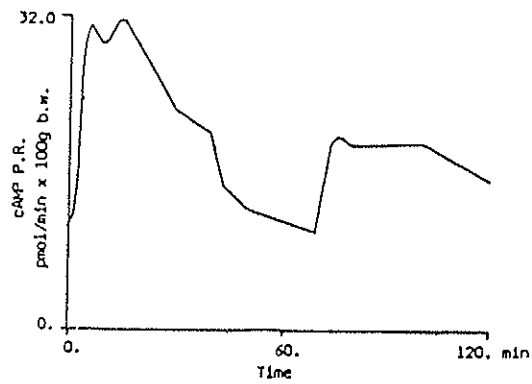




*) Dieser Beitrag stellt eine Zusammenfassung der unter demselben Titel am Institut für Analysis, Technische Mathematik und Versicherungsmathematik durchgeführten Diplomarbeit dar. Entsprechend der Themenschwerpunkte des "INTERFACE" ist hier das Hauptaugenmerk auf die praktische Durchführung der Simulation gelegt; für genauere Ausführungen hinsichtlich physiologisch- biochemischer Grundlagen und Modellbildung sei an das Original verwiesen.

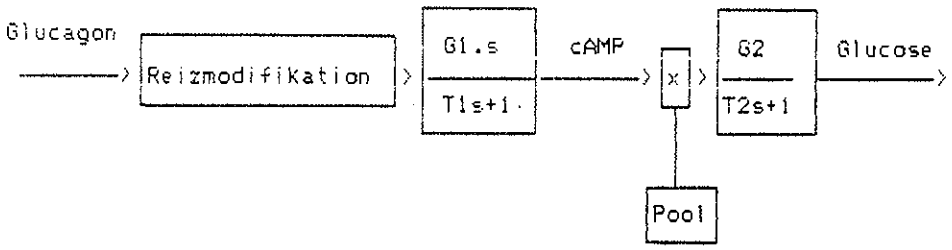
1. Meßreihe




2. Meßreihe




Aufgrund der signifikanten Gestalt der Meßkurven und gestützt durch gewisse biochemische Kenntnisse wurde folgender Modellansatz mittels regelungstechnischer Übertragungsglieder gewählt:




Hierzu ist folgendes zu bemerken:

o Reizmodifikation:

1. Meßreihe: Wie schon aus der Meßkurvengestalt ersichtlich, erfolgt die Reaktion der cAMP-Produktion auf die verschiedenen Stufen des Glucagonreizes unterschiedlich, nicht zuletzt deshalb, weil es sich um Differenzen der Dosis um das 2000-fache(!) handelt. Bei erstmaliger Reizung erfolgt eine auf die minimale Dosis vergleichsweise sehr starke Reaktion, hingegen ist der für den Abbau des cyclischen AMP verantwortliche Mechanismus noch nicht ausreichend aktiviert, sodaß der Kurvenabfall im ersten Zeitintervall signifikant abweicht.

2. Meßreihe: Ebenfalls aus der Meßkurvengestalt ersichtlich ist eine Reaktion des Systems bei der cAMP-Produktion auch auf Reize abfallender Flanke (Glucagon-Absetzung nach 40 min). Jedoch ist diese Sprungfähigkeit nicht so stark wie diejenige des Übertragungsgliedes, was bei der Modellierung berücksichtigt werden muß. Für beide Meßreihen ist die Berücksichtigung der obengenannten Phänomene unter dem Ausdruck "Reizmodifikation" zusammengefaßt.

o Pool:

Hierbei handelt es sich um den relativen Poolinhalt der Leber an Glucose, das heißt

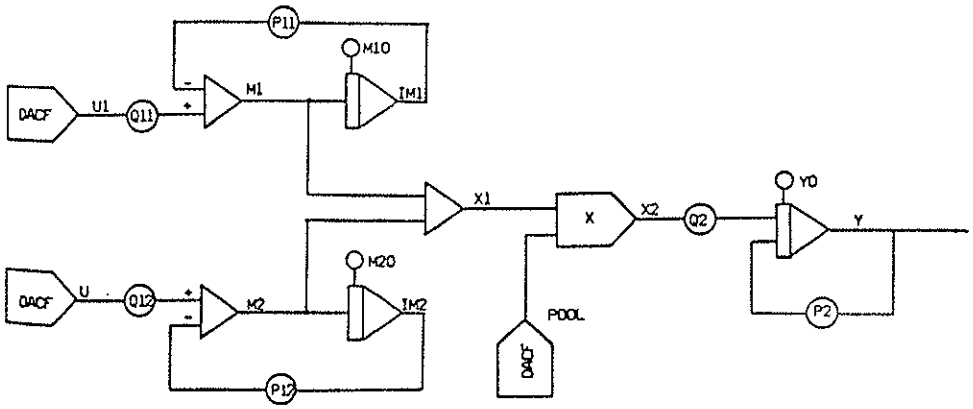
$$\text{Pool}(t) = \frac{\int_0^{12o} \text{Glucose} - \int_0^{t} \text{Glucose}}{\int_0^{12o} \text{Glucose}}$$

Messungen der Glycogenspeicher nach Versuchsende ergaben, daß im Fall der 1. Meßreihe tatsächlich eine vollkommene Entleerung der Leber stattgefunden hat, im Fall der 2. Meßreihe muß berücksichtigt werden, daß zu Versuchsende erst etwa 70% der Glycogenreserve aufgebraucht waren.

Hieraus ergeben sich folgende Schaltungen, wobei gilt:

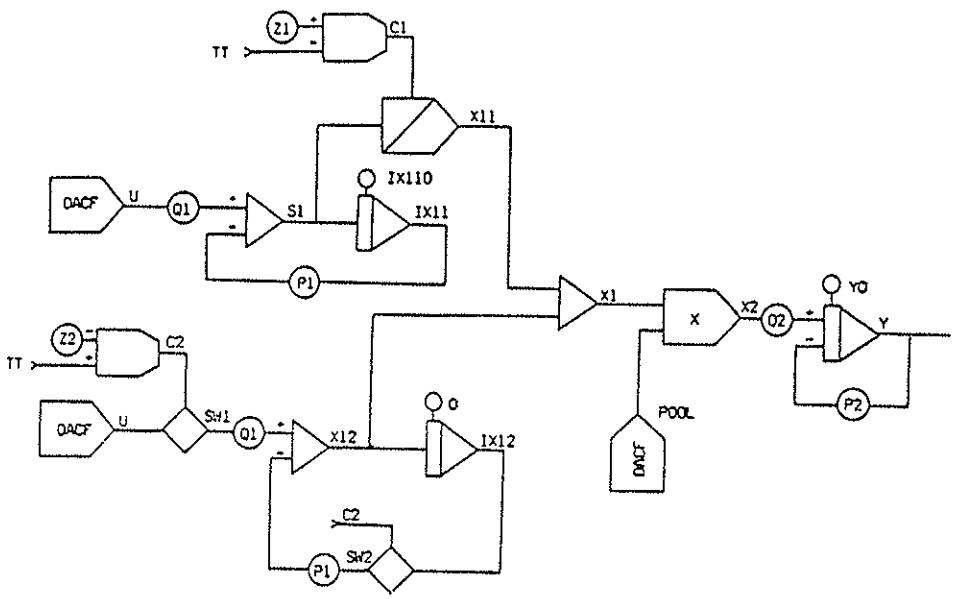
U(U1) ≙ Glucagon
X1 ≙ cAMP
Y ≙ Glucose

1.Meßreihe:




Die beiden DACFs U bzw. U1 (entsprechend gesetzt in einem HYBSYS-Overlay) repräsentieren die beiden unterschiedlichen Reaktionsformen der cAMP-Produktion auf erstmaligen bzw. wiederholten Reiz. Der DACF POOL enthält den digital über Trapezsummen (Meßkurven sind stückweise lineare Funktionen) berechneten relativen Poolinhalt, da aufgrund der Gestalt des Poolverlaufes zu dessen Berechnung jeweils zu jedem Zeitpunkt der Endwert ( $o/12o$ Glucose, siehe Formel) benötigt wird.

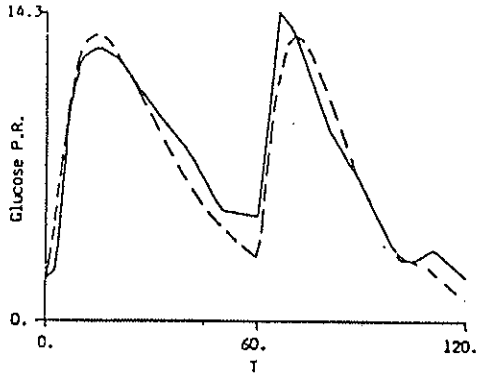
2. Meßreihe:

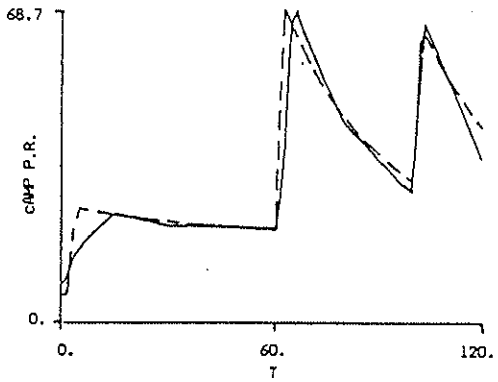



Die Komparatoren C1 und C2 sowie die Trackstore-Einheit X11 sind für die richtige Anpassung der Kurven an die verminderte Sprungfähigkeit des Systems auf die Reizabsetzung verantwortlich.

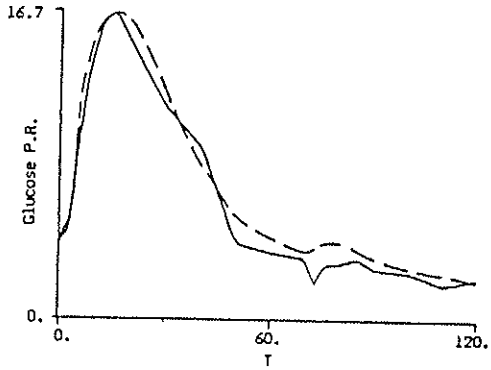
Mit der auf Basis dieser beiden Schaltungen durchgeführten Simulation ließen sich folgende Anpassungsergebnisse erzielen:

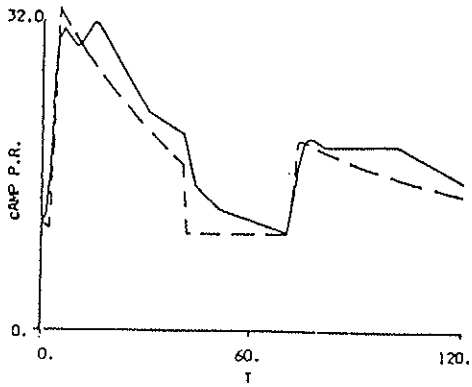
1. Meßreihe:



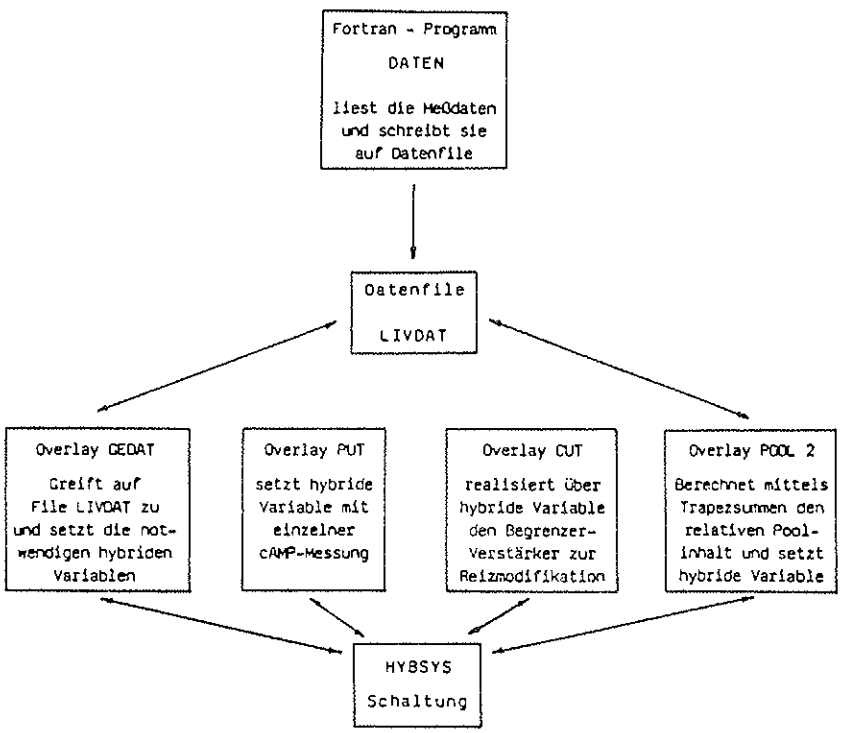

2. Meßreihe:




Zusätzlich zu den bereits oben dargestellten Schaltungen erfolgte die Simulation mittels der nach folgendem Schema zusammenwirkenden Programme:

Hinsichtlich der Ermittlung der Parameterwerte (Zeitkonstanten, Verstärkungsfaktoren) bleibt zu sagen, daß lediglich versucht wurde, das charakteristische Gesamtbild der Meßkurven möglichst gut nachzubilden, da die mit durchaus beachtlichen Meßungenauigkeiten behafteten biochemischen Meßdaten die Anwendung eines Optimierungsverfahrens kaum rechtfertigen können.



# LATERALES AUTOPILOTENSYSTEM FÜR LUFTFAHRZEUGE
# und
# INSTRUMENTENLANDESYSTEM MIT LEITSTRAHL-GESTEUERTEM LATERALEN AUTOPILOT

H. Hummer
Hybridrechenanlage
Technische Universität Wien
nach
D.J. Murray-Smith
Universität Glasgow
Dept. of Electronics and Electrical Engineering

The following two studies involve computer-based investigations of control engineering problems associated with aircraft automatic landing systems. The specific control system considered involves the lateral autopilot together with the lateral beam guidance system which aligns the aircraft automatically with the runway. This is not a study of a complete automatic landing system since only motion of the aircraft in the horizontal plane is being considered.

The system considered does, however, provide an illustration of the application of control principles. A full documentation of the studies was kindly provided by Dr. D.J. Murray-Smith, senior lecturer at the University of Glasgow. The systems have been implemented on the Hybrid Computation Centre of the Technical University of Vienna using the simulation processor HYBSYS.

As an example the effect of changing the coupler gain constant $G_c$ on the overall performance has been examined. The investigation is based upon the use of frequency response and root locus methods of control systems analysis in conjunction with computer simulations. Linearised mathematical models form the basis of the analytical part of the study and the results so obtained are interpreted using simulations involving more complex nonlinear models.

This study should provide a useful illustration of the value and practical limitations of some of the methods commonly used for linear control systems design.

## LATERALES AUTOPILOTENSYSTEM FÜR LUFTFAHRZEUGE

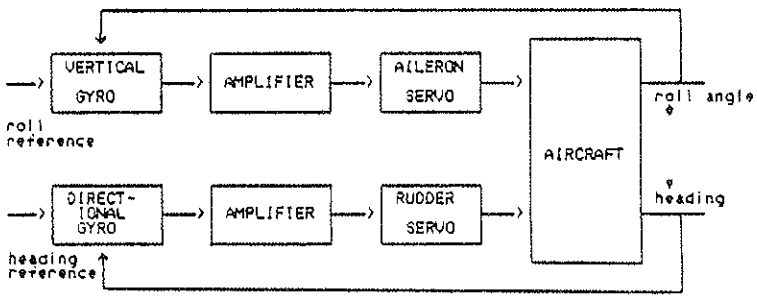
### 1. Einleitung

Die meisten Luftfahrzeuge sind nicht so konstruiert, daß sie nach einer Störung ihrer Gleichgewichtslage wieder ihre ursprüngliche Flugrichtung und ihren ursprünglichen Rollwinkel einnehmen. Zu derartigen Störungen kann es durch Betätigung der Ruder oder atmosphärische Einflüsse kommen. Der Pilot ist daher zu häufigen Kurskorrekturen gezwungen, wenn er eine bestimmte Flugrichtung einhalten will.

Die älteren Autopilot-Typen waren so konzipiert, daß sie die Fluglage der Tragflächen und das Einhalten einer bestimmten Flugrichtung steuerten.

Die Manövrierfähigkeit dieser Typen von Autopiloten ist sehr beschränkt. Sobald nämlich die einzuhaltenden Größen gewählt sind und der Autopilot in Funktion ist, können aus den im folgenden angeführten Gründen nur mehr kleine Richtungsänderungen geflogen werden. Üblicherweise wird eine Richtungsänderung durch Veränderung der Richtungsvorgabe bewirkt, was eine Betätigung des Seitenruders auslöst. Allerdings ist dieses Manöver "unkoordiniert", weil der Rollwinkel-Regelkreis versucht, die Fluglage der Tragflächen unverändert zu halten, was sich negativ auf das Flugverhalten auswirken muß. Aus aerodynamischen Gründen löst - wie bekannt - jede Seitenruder-Bewegung (und Richtungsänderung) ein Voreilen einer der beiden Tragflächen aus. Voreilen einer Fläche bedeutet aber höhere Geschwindigkeit und höheren Auftrieb gegenüber der zurückbleibenden Fläche. Das Flugzeug hebt also eine Fläche an und senkt die andere - der Rollwinkel ändert sich. Dieser natürlichen Tendenz, bei einer Höhenruderbetätigung (ohne Betätigung des Querruders) eine Querlage einzunehmen, welche extremes Schieben über die voreilende Fläche verhindert, wird damit entgegengewirkt, womit das Flugverhalten als "unkoordiniert" zu bezeichnen ist.

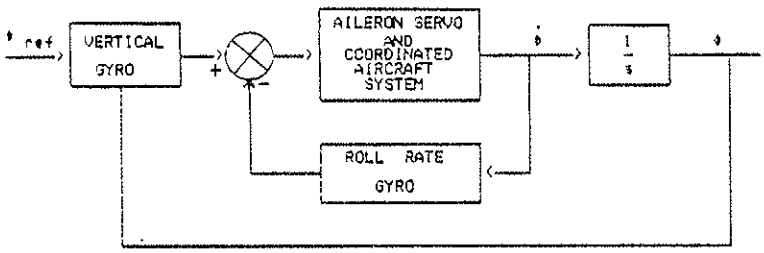
Bei modernen Autopiloten gibt es daher eine koordinierte Bewegung von Seiten- und Querruder, außerdem wird die Rollbewegung bedämpft.



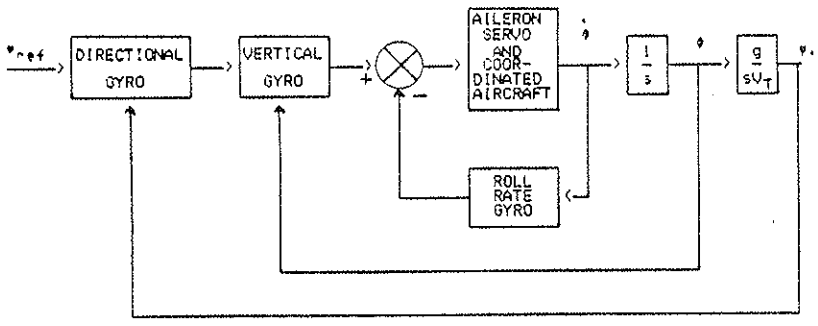

Grundkonfiguration eines lateralen Autopilotensystems
(Rudder = Seitensteuer, Aileron = Querruder)
Bild 1

## 2. Das mathematische Modell

Bild 2 zeigt ein derartiges "Koordiniertes Luftfahrzeug". Über einen weiteren Regelkreis (Bild 3) wird die Richtungsabweichung vom Soll korrigiert und in den Querlage-Steuerkreis eingegeben.




Blockdiagramm der Querlagesteuerung
Bild 2



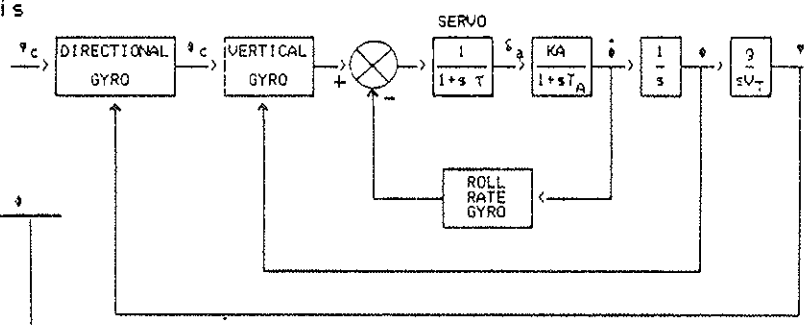

Blockschema des Autopiloten
Bild 3

Mit der gezeigten Anordnung könnte man bei der Korrektur großer Kursabweichungen extreme Rollwinkel (Querlagen) erhalten. Dies ist aus der Sicht der Passagiere unerwünscht und wegen der auf die Tragflächen wirkenden Kräfte auch nur begrenzt zulässig. Der maximale Rollwinkel wird daher eingeschränkt.

Für kleine Rollwinkel $\Phi$ bestimmt man die Winkelgeschwindigkeit des Luftfahrzeuges aus

$$\dot{\psi} = \frac{g}{V_T} \Phi$$

wobei $g$ die Schwerebeschleunigung und $V_T$ die Geschwindigkeit bedeuten. Bild 4 zeigt ein vollständiges Diagramm, welches geeignete Übergangsfunktionen für die Querlagesteuerung und das "Koordinierte Luftfahrzeug" enthält.




Lateraler Autopilot
mit Richtungsregelkreis
Bild 4

## 3. Modellparameter

Im folgenden einige realistische Parameterwerte:

$T_A$ = 2.0 sec
$K_A$ = 1.0 rad/sec/rad
$c$ = 0.1 sec
$g$ = 32.2 ft/sec$^2$ (9.81 m/sec$^2$)
$V_T$ = 200 ft/sec (60.96 m/sec)
$K_R$ = 2.5 (Querlage-Dämpfung, Empfindlichkeit)
$K_U$ = 2.5 (Rollwinkel-Steuerung, Empfindlichkeit)
$K_D$ = 2.0 (Richtungs-Steuerung, Empfindlichkeit)

## 4. Systemgleichungen

$$(1) \quad \tau \frac{d\delta_a}{dt} + \delta_a = e_s$$

$$(2) \quad T_A \frac{dp}{dt} + p = K_A \delta_a \quad \text{mit } p = \dot{\phi}$$

$$(3) \quad e_s = K_v e_v - K_R p$$

$$(4) \quad e_v = K_d e_d - \phi$$

$$(5) \quad e_d = \psi_c - \psi$$

$$\text{mit} \quad \dot{\psi} = \frac{g}{V_T} \phi$$

## 5. Durchführung des Modells

Das Modell wurde am Hybridrechner der TU Wien in HYBSYS realisiert.

Untersucht man das Verhalten des Autopiloten für verschiedene Werte des Rolldämpfungs-Parameters $K_R$, so sieht man daß für $K_R = 10$ gegenüber $K_R = 2$ die Rollbeschleunigung betragsmäßig nur einen etwa halb so großen Wert annimmt (Bild 5). Diese Größe "PHIP" ($\dot{\phi}$) nimmt zwar auf jeden Fall danach negative Werte an, bei $K_R = 10$ sind diese aber klein. Für beide $K_R$-Werte ist das Einschwenken bei $V_T = 200$ nach etwa 10000 Fuß Flugstrecke beendet. $K_R = 10$ bewirkt wesentlich weniger und wesentlich weniger heftige Rollbewegungen und damit gegenüber einem unbedämpften System erhöhten Komfort für Passagiere und Piloten, wobei die Durchführung der Korrekturen noch nicht wesentlich verschlechtert ist.

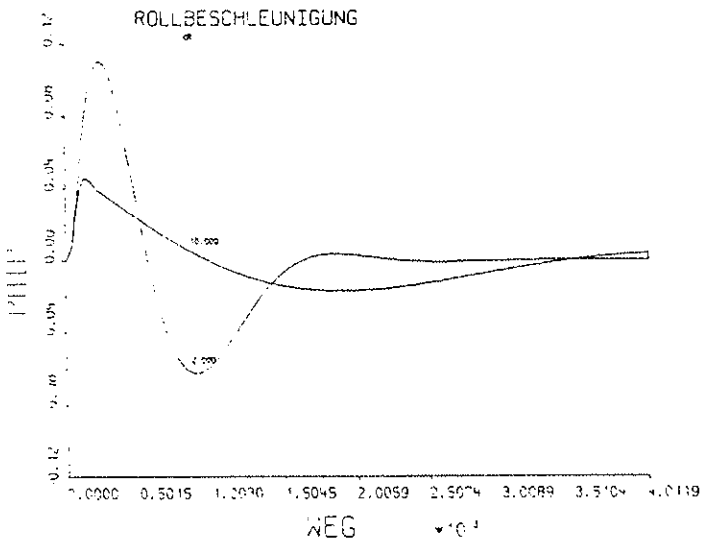



Bild 5

Untersucht man das Verhalten des Systems für verschiedene Geschwindigkeiten des Luftfahrzeuges, so kann man sehen, daß bei z.B. $V_T = 200$ ft/sec und dem angegebenen Parametersatz das Einschwenken auf die neue Route bereits nach etwa 12000 Fuß Flugstrecke abgeschlossen ist. Es kommt dabei nur zu einem einmaligen, geringfügigen Überschwingen (Bild 6). Für $V_T = 400$ ft/sec erfolgt das Einschwenken in Form einer langsamen Annäherung an den Sollkurs, und dieser ist auch nach einer Flugstrecke von 20000 Fuß noch nicht erreicht (Bild 6 und 7).

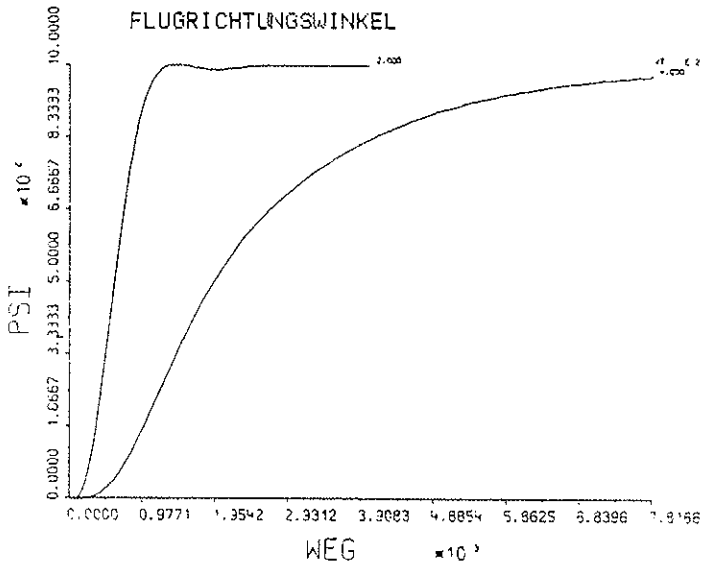



Bild 6

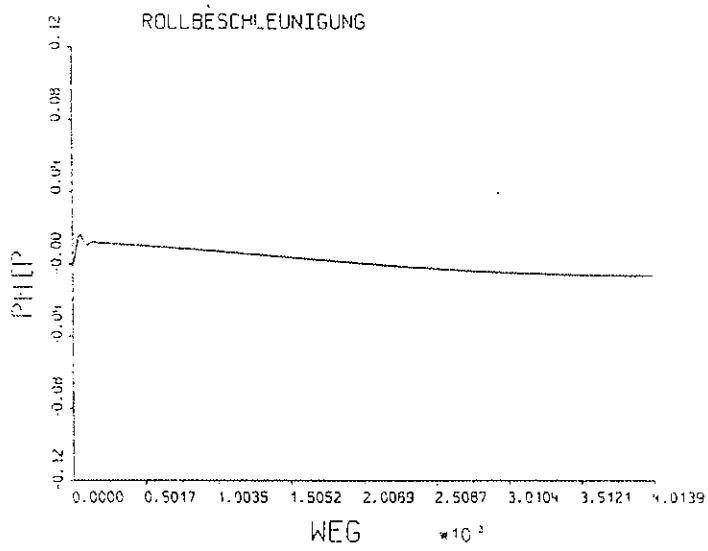



Bild 7

# INSTRUMENTENLANDESYSTEM MIT LEITSTRAHLGESTEUERTEM LATERALEN AUTOPILOT

## 1. Einführung

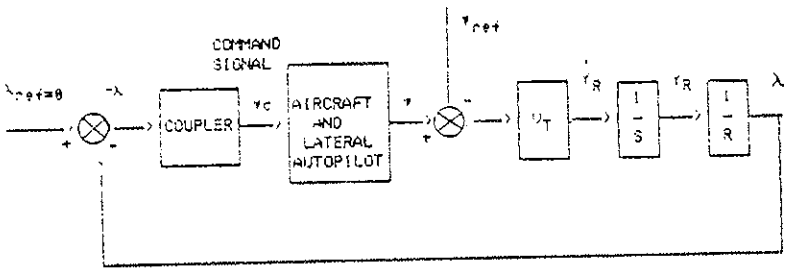
Die meisten großen Flughäfen sind mit einem Instrument Landing System (ILS) ausgerüstet, und ermöglichen damit einen automatisierten Landevorgang. Dazu müssen technische Einrichtungen vorhanden sein, welche dem Flugzeug Informationen über horizontale und vertikale Position durch Gleitpfad-Bakensender übermitteln, und das Flugzeug muß eine entsprechende Ausrüstung für die automatische Landung besitzen.

Generell wird die automatische Landung durch Einspeisung von Daten über die Richtungsabweichung in den Autopiloten ermöglicht. Die letzte Phase der Landung wird von einem automatischen Schubsteuerungs-Subsystem kontrolliert.

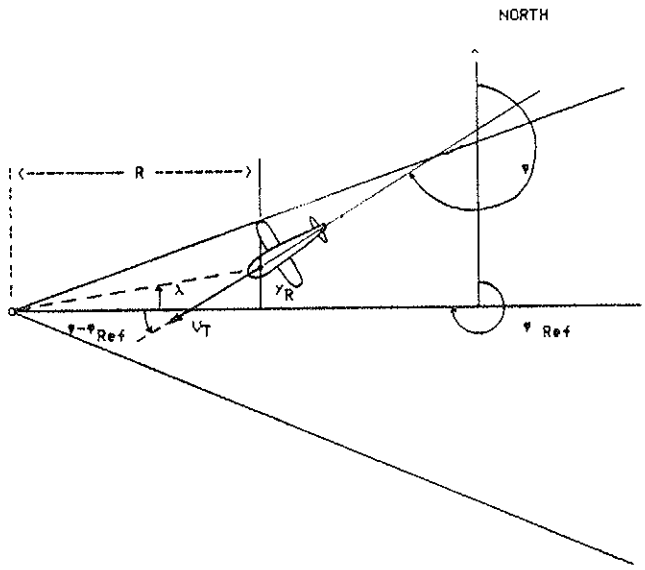
Das hier beschriebene Modell enthält nur das automatische Steuersystem für die Einhaltung der Flugrichtung. Das Flughöhen-Steuersystem ist darin nicht enthalten.

## 2. Das mathematische Modell

Bild 1 zeigt ein Blockdiagramm des betrachteten Systems. Die zum Verständnis notwendigen Geometriebeziehungen werden in Bild 2 dargestellt.




Allgemeines Blockdiagramm eines lateralen leitstrahlgesteuerten Landesystems
Bild 1




Geometrie des Landesystems (Grundriß)
Bild 2

R = Entfernung zwischen Flugzeug und Aufsetzpunkt

$Y_R$ = seitliche Abweichung des Flugzeugs vom Sollkurs

Die Variable $\psi_{Ref}$ gibt jenen Winkel an, den der Leitstrahl besitzt, der als Richtungsreferenz genommen wird. Diese Auswahl wird vom Piloten getroffen. Der Einfachheit halber wird dieser Wert in den ersten Beispielen als "0" angenommen.

Wenn $Y_R$ die horizontale Abweichung des Flugzeuges von der Soll-Gleitrichtung bedeutet und R die aktuelle Distanz zum Aufsetzpunkt, und wenn der Fehlwinkel $\lambda$ heißt, so gilt die Beziehung

$$\tan \lambda = \frac{Y_R}{R}$$

und man kann für kleine Winkel auch die Näherung

$$\lambda \approx \frac{Y_R}{R}$$

benutzen. Für die Änderung der Abweichung von der Soll-Gleitrichtung gilt

$$\dot{Y}_R = V_T \sin(\psi - \psi_{Ref})$$

Für $\psi_{Ref} = 0$ und kleine Winkel gilt somit

$$\dot{Y}_R \simeq V_T \psi$$

Für kleine Rollwinkel $\phi$ gilt die Beziehung

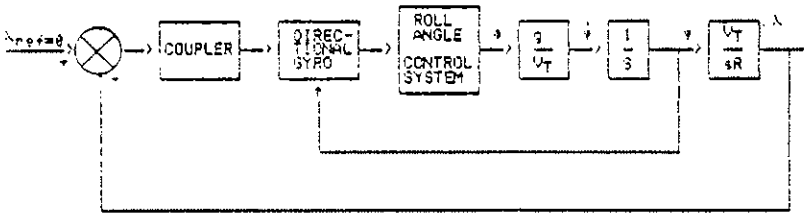
$$\dot{\psi} \simeq \frac{g}{V_T} \phi$$

wobei g die Gravitationskonstante bedeutet und $V_T$ die Geschwindigkeit des Flugzeugs.

Die Größe, welche dem Autopiloten vom Leitstrahlempfänger übermittelt wird, heißt $\psi_c$ (dies ist der Vorhaltewinkel zur Kurskorrektur). Man berechnet ihn für kleine Winkel aus

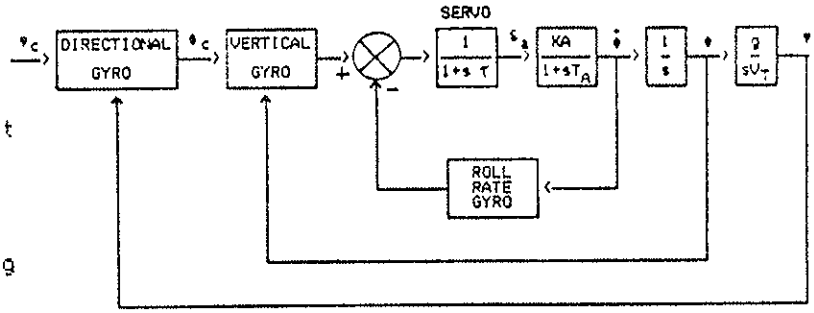
$$\psi_c = G_c \frac{Y_R}{R}$$

wobei $G_c$ der Verstärkungsfaktor der Einkoppelung des Korrektursignals ist. Bild 3 zeigt ein genaues Blockschema des Landesystems. Bild 4 zeigt das "koordinierte Luftfahrzeug", welches eine über Autopiloten gekoppelte Querlage- und Richtungssteuerung besitzt.




Blockschema des Landesystems
Bild 3

Die Ausgangsgrößen des Systems in Bild 4 werden in Lage- und Richtungssteuerung eingespeist. Die dort erhaltenen Ausgangsgrößen speisen wiederum das Lagesystem des "Koordinierten Luftfahrzeuges".




Blockschema der Querlagekontrolle
mit Richtungsregelkreis
Bild 4

## 3. Modellparameter

Im folgenden einige realistische Parameterwerte:

$T_A = 2.0$ sec
$K_A = 1.0$ rad/sec/rad
$\tau = 0.1$ sec
$g = 32.2$ ft/sec$^2$ (9.81 m/sec$^2$)
$V_T = 200$ ft/sec (60.96 m/sec)
$K_R = 2.5$
$K_V = 2.5$
$K_D = 2.0$
$G_c = 8.0$
$K_I = 0.1$

$\psi_\beta = 0.125$ rad
$Y_{R_\beta} = 200$ ft (60.96 m)

$V_T = 200$ ft/sec (60.96 m/sec)
$\phi_\beta = 0$ rad
$R_\beta = 20000$ ft (6096 m)

## 4. Systemgleichungen

$$\psi_c = G_c \lambda = G_c \frac{Y_R}{R}$$

$$\dot{Y}_R = V_T \psi$$

(1) $$\tau \frac{d\varepsilon_a}{dt} + \varepsilon_a = e_s$$

(2) $$T_A \frac{dp}{dt} + p = K_A \varepsilon_a \quad \text{mit } p = \dot{\phi}$$

(3) $$e_s = K_V e_v - K_R p$$

(4) $$e_v = K_d e_d - \phi$$

(5) $$e_d = \psi_c - \psi$$

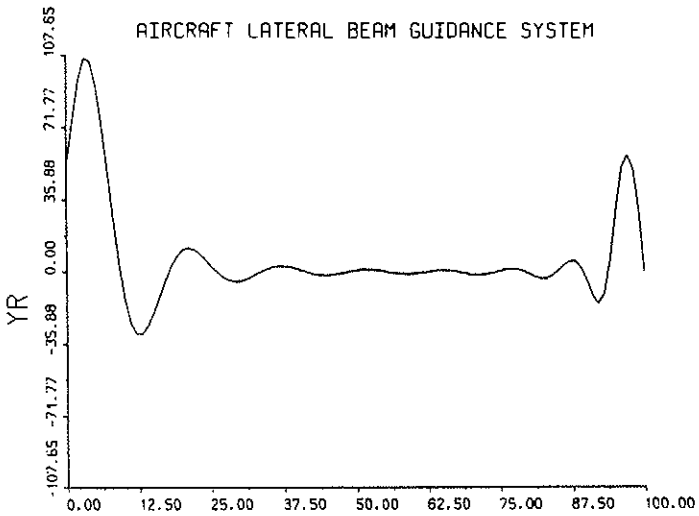
(6) $$\dot{\psi} = \frac{g}{V_T} \phi \qquad \text{mit } V_T = -\frac{dR}{dt}$$

## 5. Durchführung

Das Modell wurde am Hybridrechner der TU Wien in HYBSYS realisiert.

Die Untersuchung des Einflusses des Verstärkungsfaktors $G_c$ für die Einkopplung des Korrekturwinkels zeigt, daß Werte ab etwa 20 zwar geringfügig beschleunigtes Einschwenken auf den Soll-Gleitpfad bewirken, dortselbst jedoch ein "unruhiges" Pendeln zur Folge haben und in großer Nähe des Aufsetzpunktes sogar zu unkontrollierbaren, sich aufschaukelnden Richtungsänderungen führen (Bild 5).

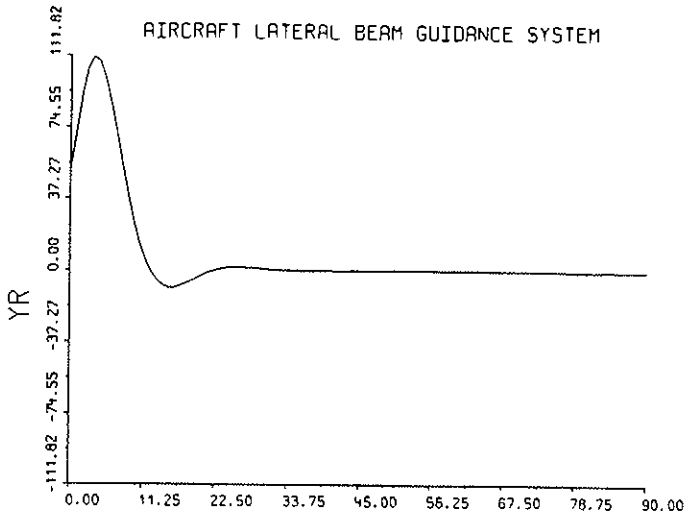
Für $G_c$=24 erreicht das Flugzeug z.B. die Sollflugrichtung nach bereits 10 Sekunden, endgültig stabilisiert wird diese jedoch erst nach 5-maligem Überschwingen und nachfolgender Rücksteuerung nach ca. 50 Sekunden (Bild 5).




$G_c$ = 24

Bild 5

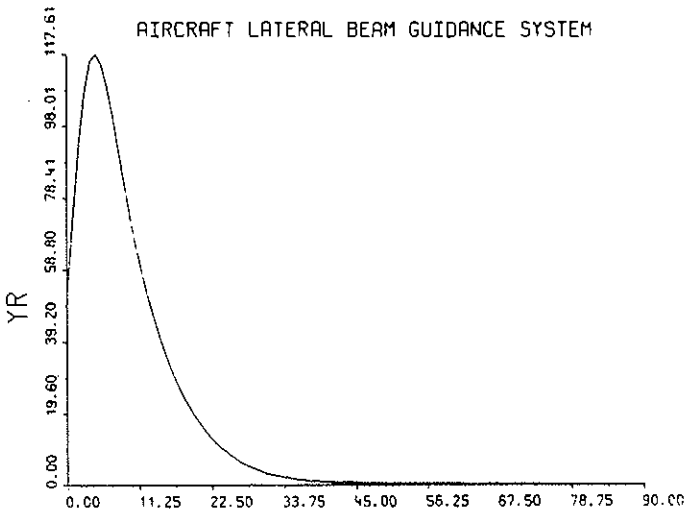
Für $G_c$=16 kommt es dagegen zu einem 2-maligen Überschwingen, und die Ruhelage tritt nach etwa 30 Sekunden ein. Das erstmalige Erreichen des Sollgleitpfades erfolgt nach 12 Sekunden (Bild 6).




$G_c$ = 16

Bild 6

Für $G_c$=8 wird der Sollgleitpfad zwar erst nach ca. 50 Sekunden erreicht, die Annäherung erfolgt jedoch ohne jegliches Überschwingen über die Sollinie (Bild 7).




$G_c$ = 8

Bild 7

Literatur:

Sewell Bill: Instrument Landing Systems, Radio Electronics, März 1984

# DER ENERGIEVERBRAUCH EINER AUTOFAHRT

F. Rattay
Abt. für Regelungsmathematik,
Hybridrechen- und Simulationstechnik
Institut für Analysis, Technische Mathematik
und Versicherungsmathematik
Technische Universität Wien

Mit einem einfachen Modell wurde versucht, den Energieaufwand eines Autos abzuschätzen. Um die Aufgabe nicht zu aufwendig werden zu lassen, wurde nicht ganz allgemein nach einer optimalen Strategie gesucht, sondern es war bloß das folgende Fahrverhalten zu untersuchen.

Eine ebene Strecke der Länge l soll so durchfahren werden, daß

zunächst für $0 \le t < t_1$ mit starker Antriebskraft $f_1$,
dann für $t_1 \le t < t_2$ mit kleiner Antriebskraft $f_2$ und
schließlich für $t_2 \le t \le T$ mit der Bremskraft $f_3$

gefahren wird, sodaß nach der Fahrtdauer T das Fahrzeug wieder zum Stillstand kommt. Der zurückgelegte Weg s(t) wird dann näherungsweise durch die Gleichung

$$m.\ddot{s} = f(t) - \mu.g.m - cw.\frac{\rho}{2}.\dot{s}^2.A \qquad (1)$$

beschrieben, wobei

$$f(t) = \begin{cases} f_1 & 0 \le t < t_1 \\ f_2 & t_1 \le t < t_2 \\ f_3 & t_2 \le t \le T \end{cases}$$

gilt.

Um das Fahrzeug zu bewegen, ist die Arbeit

$$E = \int_0^l f(s)\,ds$$

aufzubringen, zu der noch eine Leerlaufleistung L zu rechnen ist. Somit ergibt sich der Energieaufwand

$$E_{ges} = \int_0^T (f_+(t).\dot{s}(t) + L)dt \qquad (2)$$

mit

$$f_+(t) = \begin{cases} f(t) & \text{für } f(t) > 0 \\ 0 & \text{sonst} \end{cases}$$

und der Leerlaufleistung L, mit $L = L_0$ solange der Motor läuft.

Der Simulation wurden die folgenden speziellen Werte zugrunde gelegt:

| | | |
|---|---|---|
| l | Länge der Fahrstrecke | 1 km |
| $f_1, f_2$ | Antriebskräfte | 2000 N, 500 N |
| $f_3$ | Bremskraft | -3000 N |
| m | Fahrzeugmasse | 1000 kg |
| $\mu$ | Reibungskoeffizient | 0.02 |
| g | Erdbeschleunigung | 9.81 msec-2 |
| cw | Luftwiderstandsbeiwert | 0.5 |
| $\rho$ | Dichte der Luft | 1 kgm-3 |
| A | Querschnittsfläche | 2 m2 |
| $L_0$ | Leerlaufleistung | 400 Nmsec-1 |

Die Aufgabe war mit drei verschiedenen Rechnern durchzuführen, nämlich mit Hilfe

der Kleinrechner EAI-1000 rein analog,
des Hybridrechners in HYBSYS und
der Cyber mit der digitalen Simulationssprache ACSL.

Dabei wurde der erste Umschaltzeitpunkt $t_1$ verändert und in der Randwertaufgabe $t_2$ so bestimmt, daß das Fahrzeug nach der Strecke s=l zum Stehen kommt. Der Weg wird in minimaler Zeit zurückgelegt, wenn $t_1=t_2$ (T=43.2 sec). Verringern von $t_1$ führt zu den ebenfalls abgebildeten Trajektorien, die in der Anfangs- und Endphase zusammenfallen, bis schließlich für $t_1=0$ die langsamste Fahrt entsteht (T=92 sec). Da die Leerlaufleistung klein ist (sie wurde in der Abbildung vernachlässigt), ergibt sich der unterschiedliche, stückweise lineare Energieaufwand im wesentlichen dadurch, daß im ersten Abschnitt mit der vierfachen Antriebskraft gefahren wird ($f_1=4f_2$).

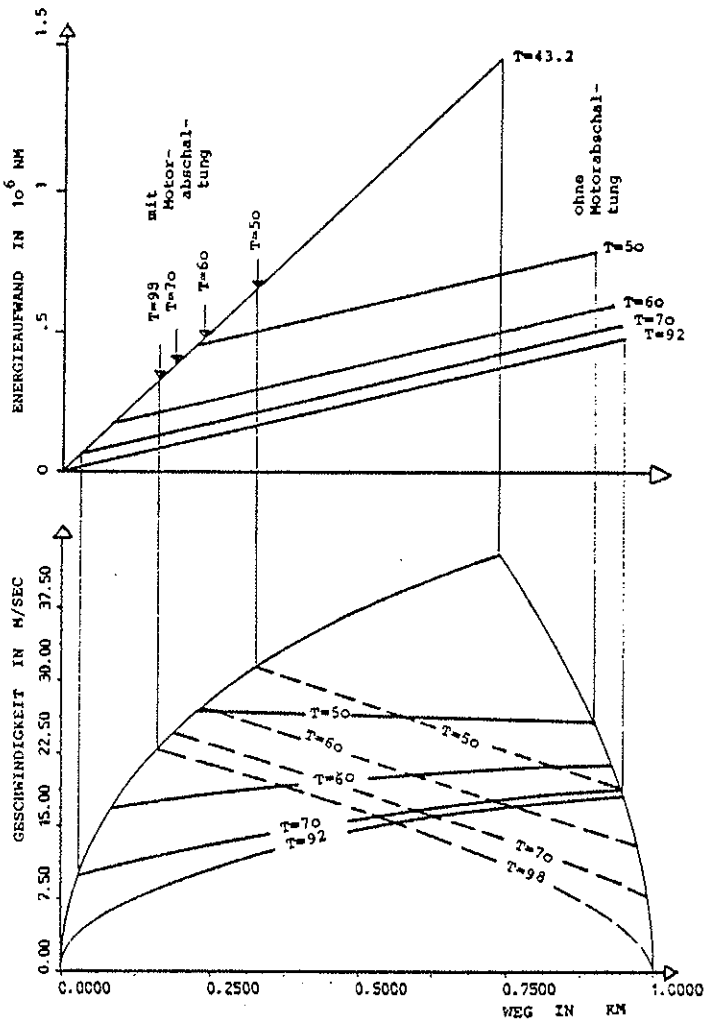
Ausserdem wurde noch in einer Variante des Modells der Fall studiert, in dem nach der ersten Beschleunigungsphase der Motor abgestellt wurde und dann natürlich auch keine Leerlaufleistung mehr braucht. Die entsprechenden Geschwindigkeiten sind in der Abbildung unterbrochen dargestellt. Die kürzeste Fahrzeit beträgt also 43.2 sec. Legt man den entsprechenden Energiebedarf bzw. Treibstoffverbrauch mit 100% fest, so ergeben sich die folgenden Prozentverbrauche:

| Fahrzeit in sec | 43.2 | 50 | 60 | 70 | 92/98 |
|---|---|---|---|---|---|
| ohne Motorabschaltung | 100 | 55 | 41 | 36 | 34 |
| mit | 100 | 48 | 34 | 28 | 23 |

Um nun die entsprechenden Verbrauchswerte als Treibstoffmenge angeben zu können, wurde zunächst jene Kraft f gesucht, mit der eine konstante Geschwindigkeit v=90 km/h gehalten werden kann. Es ergab sich f=510 N und bei einem angenommenen Normverbrauch von 8l/100km für diese Geschwindigkeit ergibt sich hier das Äquivalent von

$$0.51.10^6 \text{ Nm} = 0.08 \text{ l Benzin} \qquad (3)$$

was einem Wirkungsgrad von ca. 20% entspricht. Für die schnellste Fahrt von 43.2 sec für die 1 km lange Strecke ergibt Gleichung (2) einen Energieverbrauch von $1.5.10^6$ Nm, womit ein Schnitt von 22.5 l/100km erreicht wird. Dem steht ein Verbrauch von 5.3 l/100km bei einer Fahrtdauer von 98 sec gegenüber.




Bereits aus diesem Modell läßt sich nun ableiten, daß bei kleinen Strecken (Stadtverkehr) eine überlegte Fahrweise doch erhebliche Treibstoffeinsparung ermöglicht und daß eine Entwicklung zu Autos mit Motorabschaltung überdacht werden sollte.

# ASIM 84

Dritte Ankündigung

VORLÄUFIGES PROGRAMM

## 2. SYMPOSIUM SIMULATIONSTECHNIK

Wien, 25.-27. September 1984

**Veranstalter**

ASIM - Fachausschuß 4.5 Simulation in der Gesellschaft für Informatik (GI)
ÖCG - Österreichische Computer-Gesellschaft
TU WIEN - Hybridrechenzentrum

ASIM ist eine deutschsprachige Vereinigung, die sich der Förderung und der Weiterentwicklung der Simulation in allen Fachrichtungen widmet.

**Wissenschaftliches Programm**

- Hauptvortraege:
  * Simulation als Hilfsmittel bei der Projektierung von DV-Systemen für Handhabungsautomaten (W.Ameling, RWTH Aachen)
  * Modellprüfung - Statistische Methoden (P.Bauer, Universität Wien)
  * Computerarchitekturen der fünften Generation (W.Trattnig, Stanford University)
  * Simulationsmodelle für die Dynamik schneller Bahnsysteme - Stand und Ergebnisse (W.Kortüm, DFVLR Oberpfaffenhofen)
  * Semianalytische Methoden in der Simulationstechnik (J.Halin, ETH Zürich)
- 120 Vorträge in folgenden Gruppen:

  Simulation von Rechensystemen * Schaltkreissimulation * Simulation in Verfahrenstechnik * Fahrzeug- und Flugsimulation * Reaktor- und Kraftwerksimulation * Simulation in Biologie und Medizin * Simulation in Ökologie * Simulation in technischen Anwendungen * Simulation in Transport- und Verkehrstechnik * Simulation diskreter Prozesse * Simulationshardware * Simulationssprachen und Simulationssoftware für kontinuierliche und diskrete Systeme * Modellbildungs- und Softwaremethodik
- Podiumsdiskussion, Round-Table-Diskussionen
- Vorführungen von Simulationshardware und Simulationssoftware
- Sitzungen von ASIM-Arbeitskreisen

---

**Antwortkarte**

Ich bin einverstanden, daß die umseitig angegebenen personenbezogenen Daten in der Teilnehmerliste veröffentlicht werden:

☐ JA ☐ NEIN

ASIM -84
2. Symposium Simulationstechnik
Hybridrechenzentrum
Technische Universität Wien
Gußhausstraße 27-29
A-1040 Wien

**Vorläufiges Programm**

| | | |
|---|---|---|
| Montag, 24.9.1984 | 14-18 Uhr: | Tutorium Hybridrechnen |
| | 19 Uhr: | Tagungseröffnung, Empfangscocktail |
| Dienstag, 25.9.1984 | 9-12 Uhr: | Übersichtsvortrag, Vorträge in Parallelsitzungen |
| | 14-17 Uhr: | Übersichtsvortrag, Vorträge in Parallelsitzungen |
| | 17 Uhr: | Podiumsdiskussion "Modellvalidierung", Arbeitskreissitzungen |
| Mittwoch, 26.9.1984 | 9-12 Uhr: | Übersichtsvortrag, Vorträge in Parallelsitzungen |
| | 14-17 Uhr: | Online-Präsentation von Simulationssprachen |
| | 18 Uhr: | Heurigenabend in Gumpoldskirchen |
| Donnerstag, 27.9.1984 | 9-12 Uhr: | Übersichtsvortrag, Vorträge in Parallelsitzungen |
| | 14-17 Uhr: | Übersichtsvortrag, Vorträge in Parallelsitzungen |
| | 17 Uhr: | Round-Table-Diskussionen, ASIM-Mitgliederversammlung |

**Tagungsleitung**

F. Breitenecker, W. Kleinert, Technische Universität Wien

**Tagungsband**

Tagungssprache ist Deutsch. Der Tagungsband in der Reihe "Informatik-Fachberichte" (Springer) wird zu Beginn des Symposiums zur Verfügung stehen.

**Teilnehmerbeitrag**

| | |
|---|---|
| ÖS 1050.-/ DM 150.- | bei Bezahlung vor dem 15.7.1984 (ASIM-, GI- und ÖCG- Mitglieder ÖS 840.-/ DM 120.-) |
| ÖS 1260.-/ DM 180.- | bei Bezahlung nach dem 15.7.1984 (ASIM-, GI- und ÖCG- Mitglieder ÖS 1050.-/ DM 150.-) |

**Bankverbindung**

Konto "2.Symposium Simulationstechnik - TU Wien", Konto-Nr.: 410020-04080, Erste Österreichische Sparkasse (Bankleitzahl 20111)

Für Teilnehmer aus der BRD: "Sonderkonto ASIM 84, Dr. Ingrid Bausch-Gall, Postgirokonto 1408 24-805, Postgiroamt München

**Auskunft**

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

---

Ich melde mich an für das "2. Sympoeium Simulationstechnik" : ☐

Ich melde mich auch für das "Tutorium Hybridrechnen" an: ☐

Name: _____

Vorname: _____

Titel: _____

Institut/Firma: _____

Adresse: _____

Ort: _____

Telefon: _____

Ich interessiere mich für das Symposium. Schicken Sie mir die weiteren Aussendungen: ☐

Ich bin Mitglied von:

☐ ASIM ☐ GI ☐ ÖCG

☐ TU WIEN ☐ SONSTIGE: