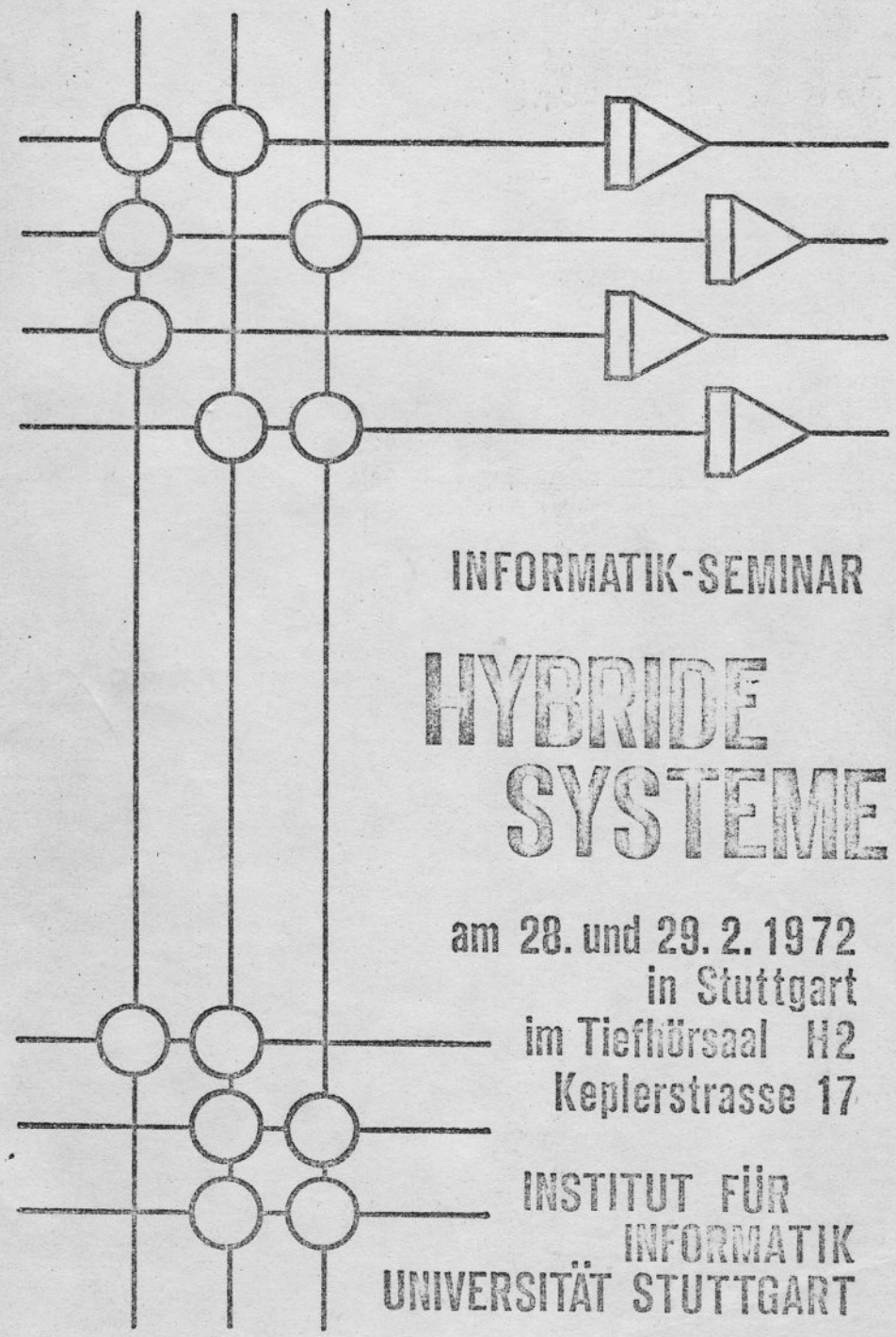


INFORMATIK-SEMINAR

HYBRIDE SYSTEME

am 28. und 29. 2. 1972
in Stuttgart
im Tieflhörsaal H2
Keplerstrasse 17

INSTITUT FÜR
INFORMATIK
UNIVERSITÄT STUTTGART

Automatisches Skalieren und statischer Test mit dem hybriden Interpreter HOI

K. Schwarze

Zusammenfassung: *In den ersten drei Abschnitten wird ein Teil der von EAI bei Hybridsystemen mitgelieferten Standardsoftware, der Hytran Operations Interpreter, ausführlich beschrieben. Am Beispiel einer inhomogenen gewöhnlichen Differentialgleichung 3. Ordnung mit nicht konstanten Koeffizienten werden in den folgenden Abschnitten die hervorstechendsten Merkmale von HOI, einer interaktiven hybriden Programmiersprache, gezeigt und diskutiert. Der statische Test der hybriden Rechenschaltung wird in diesem Programm als on-line und off-line-Check für unterschiedliche Anfangsbedingungen automatisch durchgeführt. Es wird weiterhin gezeigt, daß HOI in der Lage ist, eine automatische Normierung zu realisieren.*

Summary: *In the first three sections part of the standard EAI software package provided from EAI with Hybrid Systems, the Hytran Operations Interpreter is described in detail. At an example of a third order differential equation with variable coefficients the most significant features of HOI as an interactive hybrid programming language are shown and discussed in the following sections. In this program the on-line and off-line static check of the analog computer program and set-up is performed automatically for different initial conditions. It is further shown that HOI provides the ability of automatic scaling the analog computer program.*

1. Einleitung

Bei der Lösung eines Problems auf einem Hybridsystem kann man zwischen zwei hauptsächlich anfallenden Arbeiten unterscheiden.

Als erstes muß man sowohl auf der analogen als auch auf der digitalen Seite die verschiedenen mathematischen und logischen Ausdrücke programmieren, die das Problem beschreiben. Dazu ist es meistens erforderlich, die gegebene mathematische Beschreibung, die hier als vorhanden vorausgesetzt wird, in eine dem Rechner zugängliche Form zu bringen.

Während man als Vorarbeit den auf den Analog-/Hybrid-Rechner entfallenden Anteil der hybriden Simulation in bezug auf Amplitude und Zeit in die Größenordnung der Referenzspannung des Hybridrechners bringen muß, was man im allgemeinen als Amplituden- und Zeitnormierung oder -skalierung bezeichnet, benötigt man auf der digitalen Seite eine leistungsfähige Programmiersprache, die außer arithmetischen, algebraischen und logischen Operationen die Gesamtsteuerung der hybriden Simulation übernimmt.

Als zweites entfällt ein wesentlicher Arbeitsanteil auf die Kontrolle und Austestung des hybriden Programms.

Die Forderungen, die an eine Programmiersprache aus diesen beiden Gesichtspunkten gestellt werden müssen, unterscheiden sich ganz erheblich.

Im ersten Fall, in dem ein digitales Programm das Problem beschreibt und steuert, wird man daran interessiert sein, einen Prozessor zu benutzen, der durch eine relativ hohe Exekutionsgeschwindigkeit eine effektive Zusammenarbeit mit dem sehr schnellen, parallel orientierten Analog-/Hybrid-Rechner ermöglicht.

Für die meisten hybriden Probleme erfüllt ein Compiler wie z.B. Fortran diese Forderung am besten. Der Compiler erstellt ein relativ schnell arbeitendes Maschinenprogramm, das für die Echtzeitnatur eines hybriden Problems und die damit verbundene Kommunikation über das hybride Interface in den meisten Fällen schnell genug ist.

Demgegenüber sind die Softwareforderungen für die Kontrolle und das Austesten einer hybriden Simulation ganz anders gelagert. In diesem Falle rücken einfache Handhabung und flexible Eingriffsmöglichkeit gegenüber einer kurzen Ausführungszeit in den Vordergrund, da einerseits z.B. das Setzen von Servopotentiometern (Setzzeit 1 s) aus mechanischen Gründen ohnehin eine im Vergleich zum Gesamtsystem recht lange Zeit benötigt, andererseits beim Programmtest die Zeit im allgemeinen keine so große Rolle spielt.

Die optimale Erfüllung beider Forderungen wurde bei EAI durch die Entwicklung des „HYTRAN Operations Interpreter" (HOI) realisiert. Das Konzept der Sprache basiert also auf den Forderungen, eine möglichst leicht zu handhabende Konversationssprache zur Kontrolle hybrider Programme zu schaffen, die neben den rein digitalen Funktionen unterschiedliche Betriebsarten und flexible hybride Adressierungsmöglichkeiten enthält, um das Austesten des analogen Programms zu automatisieren.

HOI ist das Ergebnis langjähriger Erfahrung von EAI beim Bau und bei der Benutzung von Analog/Hybrid-Rechnern und Hybridsystemen in mannigfaltigen firmeneigenen Rechenzentren, das Ergebnis jahrzehntelanger Entwicklung hybrider Software.

2. Charakteristika der Programmiersprache HOI

Um den Aufwand beim Erlernen von HOI möglichst gering zu halten, ist die Syntax der Sprache der von Fortran

sehr ähnlich. Hingegen bewirken die folgenden grundlegenden Charakteristika von HOI eine wesentlich vereinfachte Handhabung und Steigerung der Leistungsfähigkeit, besonders bei Kontroll- und Testfunktionen:

1. Programmierung im Dialogverkehr,
2. wahre hybride Kommunikation,
3. besondere Betriebsarten der Sprache zum Austesten der hybriden Programme.

Jede Interpretersprache bietet die Möglichkeit der Zwiesprache mit dem Rechner im Dialogverkehr.

Mit HOI kann der Benutzer nun sowohl rein digitale als auch hybride „Statements" direkt durch Eingabe über die „Teletype" exekutieren oder auch eine Reihe von Befehlen beider Art mit Befehlsnummern versehen und speichern, um sie erst nach der Eingabe einer ganzen Befehlsreihe (= Programm) in der Reihenfolge der Befehlsnummern nacheinander abzuarbeiten.

Ein hybrides Steuerprogramm bedarf oft mannigfaltiger Änderungen während der Durchführung einer hybriden Simulation, um für richtige Initialisierung, Repetieren von „Runs" oder Wiedergabekontrolle zu sorgen. Unter Benutzung des HOI-Prozessors ist auch das leicht möglich. In einem gespeicherten Programm können Befehle abgeändert, gelöscht, aber auch an beliebiger Stelle eingefügt werden.

Die bemerkenswerteste Eigenschaft von HOI ist jedoch die wahre hybride Kommunikationsmöglichkeit. Die analogen und hybriden Komponenten werden in gleicher Weise adressiert wie die Speicherzellen des rein digitalen Prozessors. Das bedeutet, daß der Adresse einer analogen Komponente (z.B. eines Potentiometers oder Verstärkers) oder einer hybriden Komponente (z.B. Funktionsrelais) in mathematischen und logischen Ausdrücken Werte zugewiesen werden können, auf die die Komponenten automatisch gesetzt werden, wenn das gesamte Hybridsystem benutzt wird. Durch einfaches Abändern der Betriebsart der Sprache kann dasselbe Unterprogramm zum Auslesen und Verifizieren der Komponenten benutzt werden. Softwaremäßig und programmierungstechnisch sind auf diese Weise Digital- und Analog-/Hybrid-Rechner in einem Prozessor, HOI, integriert.

Die Übertragung aller mathematischen und logischen Funktionen über das hybride Standard-Interface EAI-693 kann durch diesen Aspekt der wahren hybriden Kommunikationsmöglichkeit der Sprache durchgeführt werden. Darüber hinaus gestattet eine Reihe fest verdrahteter Operationsbefehle eine flexible Überwachung der analogen Betriebsarten, der logischen Betriebsarten, der Zeitkonstanten der Integrierer usw.

Die spezielle Syntax von HOI, die zur leichten Handhabung und Ausführung dieser Übertragungs- und Kontrollfunktionen dient, wird am Beispiel weiter unten behandelt.

3. Besondere Betriebsarten der Sprache zum Austesten hybrider Programme

Um das Rechenprogramm des Analog-/Hybrid-Rechners auszutesten, bedarf es einer Vielzahl von Überprüfungen. Obwohl alle diese Tests durch IF-Abfragen in Verbindung mit dem Fortran-Compiler durchgeführt werden können, hat es sich doch im Hinblick auf Programmierungsaufwand und Flexibilität bei der Anwendung als nützlich erwiesen, durch Schaffung unterschiedlicher „Sprachenbetriebsarten" diese Testmöglichkeiten softwaremäßig in den HOI-Prozessor zu integrieren.

3.1. Digitales Austesten analoger Programme (off-line)

Eine der bedeutendsten Funktionen von HOI besteht in der Möglichkeit, ein analoges Programm durch ein digitales Prüfprogramm auszutesten.

In Bild 1 sind die primär anfallenden Arbeiten bei der Erstellung eines analogen Rechenprogramms schematisch wiedergegeben.

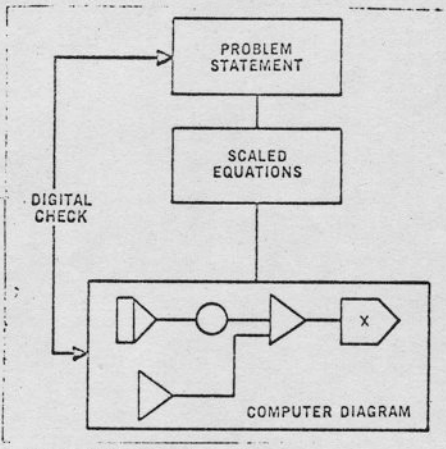


Bild 1. Die Erstellung eines analogen Rechenprogramms

Der Programmierer wird sich zuerst mit der Problembeschreibung befassen müssen. Diese besteht meistens aus einer Anzahl von Differentialgleichungen und algebraischen und logischen Beziehungen, die das Problem definieren, sowie aus Parameterwerten und Konstanten, die in der Größenordnung der physikalischen Größen in die Gleichungen eingehen. Es wird daher hier vorausgesetzt, daß bei der Problembeschreibung nur Größengleichungen und keine Zahlenwertgleichungen verwendet werden. Da der Analog-/Hybrid-Rechner eine Festkommamaschine ist, müssen die Ausgangsgleichungen so normiert oder skaliert werden, daß jede der normierten Variablen betragsmäßig $\leqslant 1$ wird.

Parallel dazu kann der Programmierer durch Umformen der gegebenen Gleichungen das Problem in eine für den Analogrechner optimale Form bringen. Dazu gehört beispielsweise eine Umformung im Hinblick auf eine minimale Komponentenzahl.

Der nächste wichtige Schritt ist die Übertragung der normierten Gleichungen in ein Computerdiagramm (eine

Rechenschaltung), das die Symbole der Rechenelemente des Analogrechners und deren Verbindungen untereinander enthält, die erforderlich sind, um die entsprechenden mathematischen Funktionen zu realisieren.

Ist das Diagramm erstellt, muß der Programmierer überprüfen, ob es tatsächlich in allen Einzelheiten die Ausgangsgleichungen der Problembeschreibung wiedergibt.

Zu diesem Zweck müssen einmal alle in den Differentialgleichungen auftretenden Größen und die damit verbundenen Spannungspegel an allen Punkten des Diagramms für den Zeitpunkt $t = 0$ berechnet werden. Ausgehend von den Anfangsbedingungen in der Rechenschaltung, müssen zum anderen nochmals alle Spannungen berechnet werden und zur Austestung der Rechenschaltung den aus den Gleichungen ermittelten gegenübergestellt werden.

Die hiermit verbundenen Berechnungen und Vergleiche werden mit HOI im digitalen Test automatisch durchgeführt.

Zur Ausführung des digitalen Tests muß die Problembeschreibung in eine für den Digitalrechner verständliche Form gebracht werden, damit dieser die Problemvariablen für den Zeitpunkt $t = 0$ auch berechnen kann. Die mathematischen und logischen Befehle des HOI-Prozessors sowie seine leichte Handhabung bei der Organisation von Programmen machen ihn zu einem idealen Hilfsmittel bei der Programmierung der Problemgleichungen.

Parallel hierzu muß aber auch die Zusammenstellung des analogen Computerdiagramms aus Komponenten der unterschiedlichsten Art in eine für den Digitalrechner verständliche Form gebracht werden. Die hybride Adressierungsmöglichkeit des Hytran Operations Interpreter ermöglicht die Aufbereitung durch das digitale Programm auf sehr einfache Weise. Die algebraische Übertragungsfunktion jeder Komponente in der Betriebsart „Anfangsbedingung" und „statischer Test" ist in HOI durch unterschiedliche Sprachenbetriebsarten enthalten. Bei richtiger Verwendung dieser verschiedenen Betriebsarten des Prozessors kann die Problembeschreibung sehr einfach mit dem Computerdiagramm verglichen werden.

Ist der digitale Test ohne Fehlermeldung durchgeführt worden, kann man sicher sein, daß sowohl das Computerdiagramm als auch dessen Gegenstück in HOI fehlerlos sind.

Während dieser digitale Test off-line, also unabhängig vom Analog-/Hybrid-Rechner durchgeführt wird, kann das gleiche digitale Programm durch einfaches Abändern der Sprachenbetriebsart zum Austesten des auf dem Analog-/Hybrid-Rechner gesteckten analogen Programms benutzt werden (on-line-Check).

3.2. Automatisches Setzen analoger Komponenten

Nachdem nun der digitale Test (off-line-Check) nach Beseitigung der in größeren Programmen fast immer auftretenden Fehler durchgeführt wurde, kann das analoge Computerdiagramm auf den Hybridrechner durch galvanisches Verbinden der entsprechenden Rechenkomponenten übertragen werden. Auch hierbei können mannigfaltige Fehler auftreten, die im „on-line-Check" eliminiert werden.

Zunächst müssen jedoch gemäß Problembeschreibung die sich aus den Anfangsbedingungen ergebenden Werte sowie Koeffizienten und logische Bedingungen gesetzt werden.

HOI unterscheidet hier zwischen drei Funktionen, die zum Setzen des Hybridrechners erforderlich sind:

a) Berechnen und Setzen von Servopotentiometern und Digital-Analog-Multiplizierern (DAMs),

b) Setzen von digital kontrollierten Funktionsgeneratoren (DCFGs),

c) Positionierung von Funktionsrelais.

Bei der Vorbereitung des digitalen Programms für den off-line-Check wurde jedem Koeffizienten durch ein „Statement" ein von den Parametern und Skalierungsfaktoren abhängiger Wert zugewiesen.

Die On-line-Berechnung und das Setzen von Potentiometern und DAMs wird nun automatisch durch Exekution dieser „Statements" in einer hybriden Sprachenbetriebsart von HOI erreicht.

Ein typisches Beispiel hierzu ist das „Statement"

1C002 = 20 x A/B .

Hierin sind A und B Parameter, die als Teil der Problembeschreibung definiert worden sind. Die Zahl 20 kann beispielsweise der Skalierungsfaktor für das Potentiometer 2 sein.

Durch die Exekution dieses Statements wird das Potentiometer 2 der Konsole 1 auf den durch die rechte Seite des Ausdrucks definierten Wert gesetzt. Die arithmetischen Operationen werden in HOI in Gleitkommaarithmetik mit einer Genauigkeit von 6 Dezimalen durchgeführt.

Für die bei EAI erhältlichen digital kontrollierten Funktionsgeneratoren (DCFGs) können beliebige Funktionen durch Setzen der Knickpunkte und Steigungen mit HOI vom Digitalrechner eingestellt werden. Auch hierbei erweist sich HOI als ideales Hilfsmittel.

Um in analogen Rechenschaltungen, bei denen sich Parameterwerte problemabhängig sprunghaft ändern, die Größen richtig einzugeben, bedient man sich der Funktionsrelais. Diese müssen der Problembeschreibung entsprechend richtig positioniert werden. Während beim EAI-690-System das Interface einen speziellen Datenkanal zur Ansteuerung der ersten 16 Relais enthält, werden in den anderen EAI-Hybrid-Systemen „Control Lines" gesetzt, die ihrerseits die Funktionsrelais richtig positionieren.

Auch hier ist HOI vornehmlich für den on-line-Check des Hybridrechners besonders nützlich. Durch planmäßiges Setzen aller dem Problem entsprechenden Relaisstellungen kann das analoge Rechenprogramm in allen seinen nur möglichen Zusammenstellungen ausgetestet werden. Wie am Beispiel weiter unten noch gezeigt wird, kann diese HOI-Funktion auch zur Automatisierung analoger Programme benutzt werden.

3.3. Digitales Austesten des gesteckten analogen Programms

Die Hauptfunktion von HOI in einem Hybridsystem ist das automatisierte Austesten des analogen Rechenprogramms auf richtige Verbindung der Rechenelemente und deren ordnungsgemäßes Arbeiten.

Zu diesem Zweck werden die auf dem Hybridrechner in Form von Spannungen anfallenden Lösungsgrößen mit dem in HOI definierten Rechendiagramm und den Ausgangsproblemgleichungen abwechselnd verglichen.

HOI isoliert auf diese Weise alle schadhaften Komponenten und auftretende Verbindungsfehler. Bemerkenswert ist dabei, daß HOI die Fehler an der Fehlerquelle aufspürt und meldet, während eventuell daraus resultierende Fehler, die z.B. von einer nicht richtig arbeitenden Komponente herrühren, nicht über den ganzen Programmabschnitt ausgedruckt werden.

Hinzu kommt noch, daß das dazu erforderliche Testprogramm das gleiche ist wie beim off-line-Check, nur daß es diesmal in einer hybriden Betriebsart der Sprache exekutiert wird.

Um die Vorteile, die HOI dem Programmierer bietet, besser beurteilen zu können, sollte man sich vor Augen halten, daß das komplette Stecken und Austesten etwa eines 100 Verstärker enthaltenden analogen Programms mit HOI in einer halben Stunde durchgeführt werden kann. Mit der älteren manuellen Technik müßte man den Rechner mehrere Tage belegen.

Insgesamt gesehen führt also HOI zu einer besseren und effektiveren Ausnutzung des Systems. Nach der Registrierung der Fehler durch HOI kann der Benutzer die Maschine wieder verlassen und seine Fehler am Problembrett bzw. im HOI-Programm in Ruhe korrigieren, während das System dem nächsten Benutzer zur Verfügung steht. Das ist ein nicht unwesentlicher Schritt vorwärts bei der Ausnutzung moderner Hybridsysteme.

Untersuchungen bei Kundensystemen haben ergeben, daß die typische Benutzungszeit weniger als zwei Stunden pro Problem beträgt, so daß in acht Stunden vier bis fünf Benutzer mit dem System arbeiten können.

Das schnelle Setzen und Austesten hybrider Programme macht sich besonders im Hochschulbereich vorteilhaft bemerkbar, wo man bestrebt ist, einer möglichst großen Zahl von Studenten den Systemzutritt zu gewähren.

In Verbindung mit einer ebenfalls mit HOI durchführbaren automatischen Programmabarbeitung und Ausgabekontrolle kann man in diesem Rahmen eine Art „batch processing" beobachten, bei der jeder Benutzer das System nur 15 bis 20 Minuten belegt.

3.4. Dynamisches Austesten analoger Rechenkomponenten

Im statischen Test werden die Verstärker, Komparatoren und Schalter sowie das IC-Widerstandsnetzwerk der Integrierer auf ihre Funktionsfähigkeit überprüft, d.h. im wesentlichen die Elemente, die keine Dynamik der Rechenschaltung bewirken. Durch eine relativ geringe Erweiterung des für den statischen Test bereits erstellten Programms bietet HOI im dynamischen Test (dynamic check) nun die Möglichkeit, die dynamischen Komponenten (also hauptsächlich Integrationsnetzwerke und Track-/Store-Einheiten) ebenfalls auf richtiges Arbeiten zu überprüfen. Da die ersten Ableitungen für die Durchführung des analogen statischen Tests ohnehin schon einzeln berechnet worden sind, ist es in den meisten Fällen nicht schwer, sie durch einen einfachen Integrationsalgorithmus nochmals zu integrieren und die dynamischen Werte der Variablen zu ermitteln.

Obwohl HOI als Interpreter nicht allzu schnell ist, nimmt man einen dynamischen Run von 5 min insbesondere bei komplexen Schaltungen für Testzwecke gern in Kauf, zumal nur ein einziger dynamischer Testrun erforderlich ist und der Hybridrechner danach Tausende solcher Runs in sehr viel kürzerer Zeit durchführen kann. Auch hier kann der Interpreter HOI durch Vergleich einen fehlerhaften Integrierer erkennen.

3.5. Automatische Steuerung hybrider Programme

Um die hohe Rechengeschwindigkeit des Hybridsystems richtig zu nutzen, damit nicht jeder Run von Hand ausgeführt werden muß und dabei kostbare Rechenzeit verlorengeht, ist es erforderlich, hybride Programme automatisch zu steuern. Es muß also ein Kontrollprogramm geschrieben werden, das die nötigen Voraussetzungen und Bedingungen für jeden Run schafft, im Fall des hybriden Problems sowohl die Betriebsarten des Hybridrechners als auch die des digitalen Programms steuert und die richtigen Informationen für eine automatisierte Wiedergabe (Display) und Ausgabe (Printer) der Lösungsgrößen liefert.

Da der Systemprogrammierer nie im voraus wissen kann, wieviel Runs er bei der Analyse eines Problems benötigt, braucht er eine Programmiersprache wie HOI, die es ihm ermöglicht, zwischendurch Runfolgen auf einfache Weise mit veränderten Parametern und anderen Problembedingungen zu programmieren. Wollte er das in einer Sprache wie z.B. Fortran durchführen, müßte er nach jeder Änderung neu kompilieren und wäre damit zumindest zeitlich stark eingeschränkt. Ein Interpreter wie HOI leistet hier die besten Dienste.

Außer Parameteränderung und Betriebsartenwahl muß die Routine aber auch die Möglichkeit bieten, in die Ausgabe und Wiedergabe entsprechend eingreifen zu können. Plottergeschwindigkeit, das Aufsetzen und Abheben der Feder, Start und Stopp sowie bei oszillographischer Anzeige die Repitierfrequenz, aber auch ein Lineprinter-Listing und die damit verbundene Reduktion von analogen Daten muß automatisierbar sein. HOI kann zur Kontrolle dieser Ausgabeeinheiten sehr vorteilhaft eingesetzt werden.

Nicht zu vergessen sei hier die leichte selektive Dokumentationsmöglichkeit mit dem Interpreter, die sich besonders im Hochschulbetrieb angenehm bemerkbar macht. Vor

dem Verlassen des Rechners gestattet HOI über die zur Verfügung stehende digitale Peripherie (Teletype oder Lineprinter) eine Zusammenstellung der analogen Ergebnisse, Potentiometereinstellungen, Verstärkerausgangswerte u.a.m..

Abgesehen von dieser automatischen Programmabarbeitung enthält HOI zur Vereinfachung zwischenzeitlicher Eingriffsmöglichkeiten in das hybride Programm einige nicht unwesentliche Aspekte. So können Parameterwerte beispielsweise unmittelbar in Maschineneinheiten eingegeben und die davon abhängigen Potentiometer automatisch gesetzt werden.

Es lassen sich weiterhin Programmteile einzeln automatisch durch eine Folge von Runs exekutieren, wobei die Kontrolle anschließend zur Eingabe neuer Parameterwerte wieder an die Teletype und damit zum Programmierer kommt. Auf diese Weise können komplexe Probleme, bei denen ein automatischer Programmablauf anfänglich nicht zu realisieren ist, Schritt für Schritt analysiert und ausgetestet werden, wobei das zeitraubende manuelle Kalkulieren von Werten und Einstellungen entfällt.

Als Nebenprodukt liefert HOI außerdem eine vollständige Dokumentation jedes Eingriffs in die Simulation.

4. Beispiel eines HOI-Programms

Die in den ersten drei Abschnitten beschriebenen Vorteile und Möglichkeiten des hybriden Interpreters HOI seien nun an einem konkreten, relativ übersichtlichen Beispiel gezeigt. Gegeben ist folgende inhomogene gewöhnliche Differentialgleichung 3. Ordnung mit nicht-konstanten Koeffizienten:

$$t^2 \dddot{y} + 5t\ddot{y} + 4\dot{y} = \ln t . \qquad (1)$$

Diese willkürlich aus der Literatur herausgegriffene Gleichung werde nun für unterschiedliche Anfangsbedingungen auf dem Hybridrechner gelöst.

4.1. Problembeschreibung

Im Hinblick auf eine echte Kontrollmöglichkeit des Rechenergebnisses wurde eine Differentialgleichung gewählt, deren allgemeine Lösung aus der Literatur bekannt ist. Es soll an dieser Stelle nicht verschwiegen werden, daß es sich bei der Problemstellung ursprünglich um einen Kundenauftrag handelte, der anhand der Ergebnisse eine Aussage über die Genauigkeit des Systems erhalten wollte. Dementsprechend ist hier die Aufgabenstellung, soweit es das hybride Steuerprogramm betrifft, komplizierter als in den meisten sonst anfallenden Simulationen.

Während man im allgemeinen davon ausgehen kann, daß die Anfangsbedingungen als gegebene Größen in die Schaltung eingehen, müssen sie im vorliegenden Fall erst berechnet und normiert werden, um dann mit HOI auf den Hybridrechner übertragen zu werden.

Da aber gerade dadurch die Flexibilität und der Anwendungsbereich des Interpreters besonders augenfällig werden, ist vom Veröffentlicher dieses etwas abstrakte Beispiel gewählt worden.

Die allgemeine Lösung der obigen Differentialgleichung lautet:

$$y = C_1 + \frac{C_2}{t} + \frac{C_3 \ln t}{t} + \frac{t}{4}(\ln t - 2) . \qquad (2)$$

Hieraus ergeben sich durch Differentiation folgende Ableitungen:

$$\dot{y} = -\frac{C_2}{t^2} + \frac{C_3 (1 - \ln t)}{t^2} + \frac{1}{4}(\ln t - 1) , \qquad (3)$$

$$\ddot{y} = \frac{2C_2 - C_3 [1 + 2(1 - \ln t)]}{t^3} + \frac{1}{4t} , \qquad (4)$$

$$\dddot{y} = \frac{C_3 \{2 + 3 [1 + 2(1 - \ln t)]\} - 6C_2}{t^4} - \frac{1}{4t^2} \qquad (5)$$

4.2. Normierung

Bezieht man in Gleichung (1) alle physikalischen Größen G_γ auf den Normierungswert $G_{\gamma m}$ mit

$G_{\gamma m} \geq G_{\gamma max}$,

so folgt:

$$t_m^2 \dddot{y}_m \left(\frac{t}{t_m}\right)^2 \left(\frac{\dddot{y}}{\dddot{y}_m}\right) + 5 t_m \ddot{y}_m \left(\frac{t}{t_m}\right)\left(\frac{\ddot{y}}{\ddot{y}_m}\right)$$

$$+ 4 \dot{y}_m \left(\frac{\dot{y}}{\dot{y}_m}\right) - \ln \left(\frac{t}{t_m}\right) - \ln t_m = 0 ; \qquad (6)$$

setzt man weiterhin

$$\frac{t}{t_m} = [t] \qquad \frac{y}{y_m} = [y] \qquad \frac{\dot{y}}{\dot{y}_m} = [\dot{y}]$$
$$\frac{\ddot{y}}{\ddot{y}_m} = [\ddot{y}] \qquad \frac{\dddot{y}}{\dddot{y}_m} = [\dddot{y}] \qquad (7)$$

und löst die Gleichung nach $[\dddot{y}]$ auf, erhält man:

$$[\dddot{y}] = -\frac{5 \ddot{y}_m}{t_m \dddot{y}_m} \frac{[\ddot{y}]}{[t]} - \frac{4 \dot{y}_m}{t_m^2 \dddot{y}_m} \frac{[\dot{y}]}{[t]^2}$$

$$+ \frac{\ln [t] + \ln t_m}{t_m^2 \dddot{y}_m [t]^2} . \qquad (8)$$

Kann auf die Größe $[\dddot{y}]$ in der Schaltung verzichtet werden, normiert man vorteilhafter wie folgt:

$$\frac{\dddot{y}}{\dddot{y}_m} = -\frac{5}{t_m} \frac{(\ddot{y}/\ddot{y}_m)}{(t/t_m)} - \frac{4}{t_m^2} \frac{\dot{y}_m}{\dddot{y}_m} \frac{(\dot{y}/\dot{y}_m)}{(t/t_m)^2}$$

$$+ \frac{\ln (t/t_m) + \ln t_m}{(t/t_m)^2 \dddot{y}_m t_m^2} ; \qquad (9)$$

mit Gleichung (7) wird daraus:

$$\frac{\dddot{y}}{\dddot{y}_m} = -\frac{5}{t_m} \frac{[\ddot{y}]}{[t]} - \frac{4}{t_m^2} \frac{\dot{y}_m}{\dddot{y}_m} \frac{[\dot{y}]}{[t]^2}$$

$$+ \frac{\ln [t] + \ln t_m}{[t]^2 \dddot{y}_m t_m^2} . \qquad (10)$$

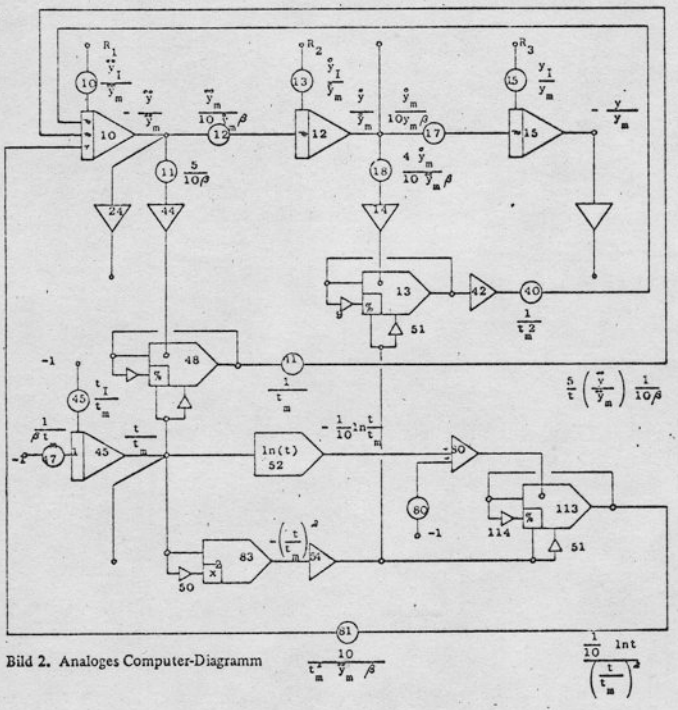


Bild 2. Analoges Computer-Diagramm

4.3. Analoge Rechenschaltung

Das sich aus Gleichung (10) ergebende analoge Computerdiagramm zeigt Bild 2. Das Beispiel wurde für das EAI-690-System programmiert, das aus dem Analog-/Hybrid-Rechner EAI-680 und dem Digitalrechner EAI-640 besteht. Alle im Diagramm auftretenden Normierungsgrößen werden in einem HOI-Unterprogramm berechnet.

Das Programm ist mit dem Faktor β zeitnormiert, der für jeden Run erneut gewählt werden kann. Da, wie schon erwähnt, die Anfangsbedingungen für die Integrierer erst berechnet werden müssen, d.h. Größe und Vorzeichen zunächst nicht bekannt sind, wurden nach Bild 3 die Referenzspannungen über Funktionsrelais an die Integrierer gelegt. Hierbei handelt es sich um Relais, die über das EAI-693-Interface vom überwachenden HOI-Programm gesteuert werden können.

Die gewählte Timerschaltung ist ebenfalls in Bild 3 wiedergegeben. Sie gewährleistet, daß der gesamte mit HOI berechnete Zeitbereich im Repetierbetrieb wiedergegeben wird, wenn die Rechenzeiteinstellung XXX für OP nur groß genug gewählt wurde. Komparator 19 sorgt durch Verbindung mit dem Skipeingang am Rechner für Betriebsartenwechsel von OP nach IC bei

$$[t] = \frac{t}{t_m} > 1 .$$

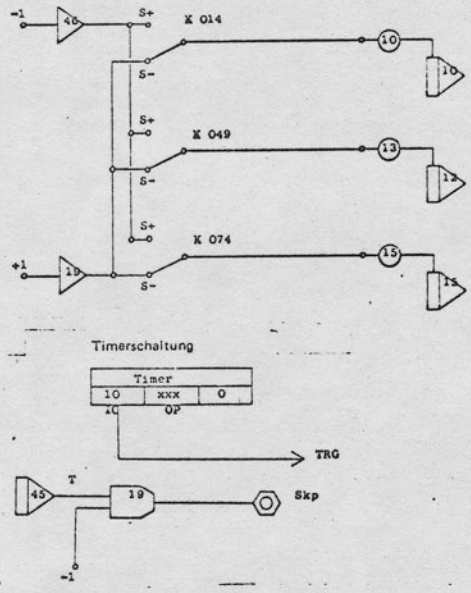


Bild 3. Aufbringen der berechneten Anfangsbedingungen

4.4. Digitales HOI-Rechenprogramm zur Ermittlung der Anfangsbedingungen

In den Bildern 4 und 5 ist das Flußdiagramm des HOI-Rechenprogramms dargestellt, das zur Ermittlung der Anfangsbedingungen und Maximalwerte dient. Im Vergleich dazu ist in Bild 6 der Steuerteil des zugehörigen HOI-Programms abgedruckt. Die Statementnummern haben die Form

XX.YYY ,

wobei XX den Programmteil und YYY den Programmschritt bezeichnen. In den ersten drei Programmschritten wird nur Text ausgedruckt. Das Exekutionszeichen dafür ist der Doppelpunkt.

Nachdem die HOI-Betriebsart Indirect Execution Mode, No Analog Normal im Programmschritt 1.04 und 1.05 für ein off-line-Arbeiten mit dem Interpreter gewählt worden ist, wird in 1.051 der Programmteil 12 und 14 exekutiert (s. Bild 8).

Im Programmteil 12 müssen die Parameter C_1, C_2, C_3, TM, TDEC und BETA über die Teletype eingegeben werden.

TM ist hierbei die Zeit, bis zu der man die Lösungsgrößen auf dem Hybridrechner berechnet haben möchte.

In analogen Rechenschaltungen erweisen sich die Dividierer in bezug auf eine sinnvolle Normierung als besonders kritisch, da immer gewährleistet sein muß, daß das Ergebnis betragsmäßig nicht größer als 1 wird. Die Kontrolle darüber kann das HOI-Programm in einfacher Weise durch Abfragen der Verstärkerausgänge durchführen.

Der Programmteil 11, der unmittelbar nach 12 exekutiert wird, benutzt nun diese Abfragemöglichkeit, um, ausgehend von T = TM, in zeitlichen Dekrementen von TDEC den Zeitpunkt T = TI zu ermitteln, bei dem die drei Dividierer 1A113, 1A048 und 1A013 gerade noch nicht übersteuern. Dazu müssen die Verstärkerausgänge für den Zeitbereich $TI \leqslant T \leqslant TM$ berechnet werden, was in der Schleife 11.01 bis 11.04 durch jeweilige Exekution der Programmteile 13 und 23 erreicht wird.

Im Programmteil 13 werden außer Y und seinen Ableitungen YP, Y2P und Y3P auch die Maximalwerte dieser Größen für $TI \leqslant T \leqslant TM$ kalkuliert, die für die automatische Normierung benötigt werden.

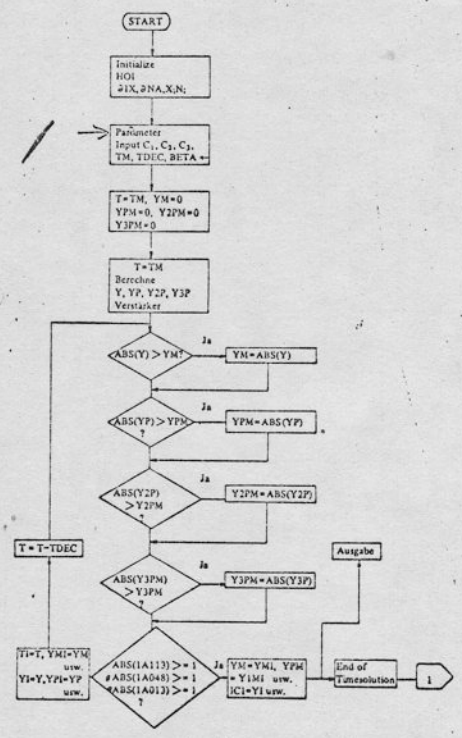


Bild 4. Flußdiagramm

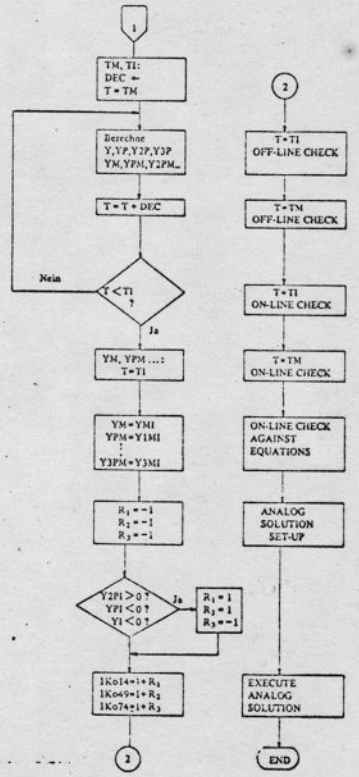


Bild 5. Flußdiagramm (Fortsetzung)

```
1.010   "LOESUNG EINER INHOMOGENEN GEWOEHNLICHEN":
1.020   "DIFFERENTIALGLEICHUNG 3. ORDNUNG MIT":
1.021   "NICHT KONSTANTEN KOEFFIZIENTEN AUF DEM":
1.022   "ANALOG/HYBRIDRECHNER EAI 680":
1.023   "FUER TI<T<TM": : ;
1.030   "TIMESOLUTION":
1.040   @IX, @NA, X;
1.050   N;
1.051   12; 14;
1.060   "OFF-LINE CHECK": : :
1.061   T=TI
1.062   17.011; 13; 11.022; 40;
1.063   "T=TI": :
1.070   "COEF": 21;
1.080   "DERIV": 22;
1.090   "AMPL": 23;
1.100   "POT OUT": 31;
1.110   2;

2.020   .001, V;
2.030   "DERIV CHK": 32;
2.040   "AMPL CHK": 33;
2.041   : : :
2.042   "T=TM": :
2.050   N;
2.060   T=TM, 13; 11.022; 40;
2.061   "COEF": 21;
2.062   "DERIV": 22;
2.063   "AMPL": 23;
2.064   "POT OUT": 31;
2.070   .001, V;
2.080   "DERIV CHK": 32;
2.081   "AMPL CHK": 33;
2.090   "OFF-LINE CHECK COMPLETE":
2.100   HALT; : : :
2.110   3;

3.010   680, C;
3.020   "ON LINE SETUP AND CHECKOUT":
3.030   @IX, @WA, X;
3.040   N;
3.041   TI=IC4
3.050   T=TI, 13; 11.022; 40; 41;
3.060   "IS ANALOG COMPUTER ON-LINE? ":
3.070   HALT;
3.071   : : "T=TI": :
3.080   1, USE;
3.090   1, C;
3.100   @SP, M; @RU, M;
3.110   "SET POTS": 21;
3.120   4;

4.010   @ST, M;
4.030   .0003, V;
4.040   "COEF CHK": 21;
4.041   H;
4.050   "POT OUTPUT CHK": 31;
4.051   .001, V;
4.060   "AMPL CHK": 33;
4.070   "DERIV CHK": 32;
4.071   :
4.080   N;
4.090   @SP, M;
4.091   : : "T=TM": : "COEF":
4.100   T=TM, 13; 11.022; 40; 41; 21;
4.110   @ST, M;
4.120   .001, V;
4.129   "POT OUTP CHK": 31;
4.130   "DERIV CHK": 32;
4.131   "AMPL CHK": 33;
4.140   "ON-LINE CHECK COMPLETE":
4.150   HALT; : : :
4.160   5;

5.010   "ON-LINE CHECK AGAINST EQUATIONS": : :
5.012   "T=TM": :
5.020   N;
5.030   T=TM, 13; 11.022; 40; TI=T, 21; 41;
5.040   .001, V;
5.050   "AMPS": 23;
5.060   "DERIVS": 22;
5.070   "EQUATION CHECK COMPLETE":
5.071   : : "ANALOG SOLUTION SET-UP": :
5.072   "TI<= T <= TM": :
5.079   N;
5.080   YI=IC1, YPI=IC2, Y2PI=IC3, TI=IC4
5.081   @SP, M;
5.082   "COEF": 21; 40; 41;
5.083   @ST, M; .0003, V;
5.084   "COEF CHK": 21; H;
5.085   "POT OUTP CHK": 31;
5.090   @PP, M; E;
```

Bild 6. HOI-Programm, Steuerteil

Übersteuert einer der Dividierer, wird in 11.02 der Programmteil 17 abgearbeitet. Er sorgt im wesentlichen für eine Zusammenstellung und das Ausdrucken der interessierenden Werte:

Anfangszeit	TI
Endzeit	TM
Maximalwerte	YM, YPM, Y2PM, Y3PM und
Anfangsbedingungen	YI, YPI, Y2PI, Y3PI.

Die sich aus den Anfangsbedingungen ergebenden richtigen Vorzeichen für die Referenzspannungen R_1, R_2 und R_3 werden im Unterprogramm 40 ermittelt.

Nach der Abarbeitung von 17 kehrt die Kontrolle in den Programmschritt 11.02 zurück, wo der Sprungbefehl 11.05. nach 11.05 ausgeführt wird. „END OF TIME-SOLUTION" erscheint jetzt auf der Teletype, und damit ist der Exekutionsbefehl in 1.051, nämlich 12; erfüllt worden.

In Programmteil oder Unterprogramm 14 werden nach Eingabe von DEC durch den Iterationsbefehl in

14.020 T = TM, DEC, TI! 13; 15;

ähnlich einem DO-LOOP, in Fortran die Zeit T in Schritten von DEC bis TI variiert und die zugehörigen Werte für Y, YP, Y2P und Y3P zum späteren Vergleich mit dem Hybridrechner rein digital berechnet und in Tabellenform ausgegeben (s. Bild 7).

Damit sind die für die Durchführung des statischen Tests erforderlichen Größen bekannt. Zu jedem gewählten TM berechnet HOI ein für die Rechenschaltung noch zulässiges TI sowie die Anfangsbedingungen und Maximalwerte automatisch. Wie weiter unten noch gezeigt wird, ist HOI in der Lage, den Hybridrechner so zu steuern, daß er genau in diesem Zeitbereich repitierend arbeitet und die Rechenschaltung auf die berechneten Maximalwerte normiert.

```
LOESUNG EINER INHOMOGENEN GEWOEHNLICHEN
DIFFERENTIALGLEICHUNG 3. ORDNUNG MIT
NICHT KONSTANTEN KOEFFIZIENTEN AUF DEM
ANALOG/HYBRIDRECHNER EAI 680
FUER TI<T<TM

TIMESOLUTION
C1 ←1
C2 ←2
C3 ←3
TM ←1
TDEC ←.01
BETA ←1
TI = .509995, TM = 1.00000
YM = 2.50000, YPM = 11.1931, Y2PM = 67.6617, Y3PM = 488.629
YI = .619806, YPI = 11.1931, Y2PI =-67.6617, Y3PI = 488.629
A013 = .261525, A048 = 1.00001, A052 = .069315, A113 = -.277269
END OF TIMESOLUTION

DIGITALE BERECHNUNG DER LOESUNGSGROESSEN
IN ZEITSCHRITTEN VON DEC FUER:
C1 = 1.00000, C2 = 2.00000, C3 = 3.00000
TM = 1.00000, TI = .509995
DEC ← -.1

T = 1.00000
Y = 2.500000, YP = .750000, Y2P = -4.75000, Y3P = 20.7500
T = .900000
Y = 2.39731, YP = 1.34845, Y2P = -7.44810, Y3P = 34.5892
T = .800000
Y = 2.21858, YP = 2.30270, Y2P = -12.0681, Y3P = 60.6850
T = .700000
Y = 1.91612, YP = 3.88537, Y2P = -20.4593, Y3P = 113.693
T = .600000
Y = 1.40258, YP = 6.65695, Y2P = -36.2911, Y3P = 232.290
YM = 2.50000, YPM = 11.1931, Y2PM = 67.6617, Y3PM = 488.629

OFF-LINE CHECK
T = TI
COEF
DERIV
AMPL
POT OUT
DERIV CHK
AMPL CHK

T = TM
COEF
DERIV
AMPL
POT OUT
DERIV CHK
AMPL CHK
OFF-LINE CHECK COMPLETE
HALT IN 2.100
←G;
ON LINE SETUP AND CHECKOUT
IS ANALOG COMPUTER ON-LINE?
HALT IN 3.070
←G;
T = TI
SET POTS
COEF CHK
 = .317800 NOT .318167 IN 21.012
 = .277400 NOT .277944 IN 21.017
HALT IN 4.041
←G;
POT OUTPUT CHK
 = -.999300 NOT -.999900 IN 31.010
 = .530300 NOT .530905 IN 31.041
AMPL CHK
DERIV CHK
 = -.296000 NOT -.294600 IN 32.045

T = TM
COEF
POT OUTP CHK
DERIV CHK
AMPL CHK
ON-LINE CHECK COMPLETE
HALT IN 4.150
←G;
ON LINE CHECK AGAINST EQUATIONS

T = TM
AMPS
 = -1.00180 NOT -1.00000 IN 23.051
 = 1.00190 NOT 1.00000 IN 23.054
 = -1.00200 NOT -1.00000 IN 23.083
DERIVS
 = -.275900 NOT -.294118 IN 22.045
EQUATION CHECK COMPLETE

ANALOG SOLUTION SET-UP

TI<=T<=TM
COEF
COEF CHK
 = .317600 NOT .318167 IN 21.012
 = .277500 NOT .277944 IN 21.017
 = .276100 NOT .276458 IN 21.045
HALT IN 5.084
←G;
POT OUTP CHK
 = -.999300 NOT -.999900 IN 31.010
 = -.999500 NOT -.999900 IN 31.013
 = .530700 NOT .531317 IN 31.041
←
```

Bild 7. Teletype-Listing

4.5. Digitales HOI-Test-Programm (off-line-Check)

Das HOI-Steuerprogramm zur Durchführung des statischen off-line-Tests beginnt in 1.061 und endet bei 2.1. Zur Exekutionskontrolle wird der in Gänsefüßchen stehende Text ausgegeben. Wie aus dem Programm zu ersehen ist, wird der Test für zwei unterschiedliche Zeitpunkte TI und TM durchgeführt, um eine zusätzliche Kontrollmöglichkeit durch Vergleich der Testergebnisse zu erhalten.

4.5.1. Problembeschreibung

Der Programmteil 13, der schon bei der Bestimmung der Anfangszeit TI benutzt wurde, enthält die Problemgleichungen. Daneben bewirkten die in ihm enthaltenen logischen Abfragen die Ermittlung der Maximalwerte der Lösungsgrößen innerhalb des betrachteten Zeitbereiches. Nach Abspeicherung dieser Maximalwerte durch Exekution von 17.011 in 1.062 wird derselbe Programmteil 13 in Verbindung mit Schritt 11.022 zur Ermittlung der Anfangsbedingungen für T = TI benutzt. Wiederum wird durch Programmteil 40 den Referenzspannungen das richtige Vorzeichen zugeordnet.

```
11.010  13: 23;
11.020  (ABS(1A113) >= 1) # (ABS(1A048) >= 1) # (ABS(1A013) >= 1)? 17; 11.05.
11.021  TI=T, YMI=YM, Y1MI=YPM, Y2MI=Y2PM, Y3MI=Y3PM
11.022  YI=Y, YPI=YP, Y2PI=Y2P, Y3PI=Y3P
11.030  T=T-TDEC
11.040  11.01.
11.050  "END OF TIMESOLUTION" : : :

12.010  C1, C2, C3, TM, TDEC, BETA ←
12.011  YM=0, YPM=0, Y2PM=0, Y3PM=0
12.020  T=TM
12.021  11;

13.010  "EQUATIONS"
13.020  Y=C1+C2/T+C3*LN(T)/T+T/4*(LN(T)-2)
13.022  ABS(Y) > YM?  YM=ABS(Y)
13.030  YP=-C2/T↑2+C3*(1-LN(T))/T↑2+.25*(LN(T)-1)
13.032  ABS(YP) > YPM?  YPM=ABS(YP)
13.040  Y2P=2*C2/T↑3-C3*(1+2*(1-LN(T)))/T↑3+1/4/T
13.042  ABS(Y2P) > Y2PM?  Y2PM=ABS(Y2P)
13.050  D=1+2*(1-LN(T))
13.060  A=-6*C2/T↑4
13.070  B=C3*(2+3*D)/T↑4
13.080  C=-1/4/T↑2, Y3P=A+B+C
13.082  ABS(Y3P) > Y3PM?  Y3PM=ABS(Y3P)

14.010  "DIGITALE BERECHNUNG DER LOESUNGSGROESSEN":
14.011  "IN ZEITSCHRITTEN VON DEC FUER: ": : :
14.012  C1, C2, C3:
14.013  TM, TI: DEC ← : : :
14.020  T=TM, DEC, TI! 13; 15;
14.030  YM, YPM, Y2PM, Y3PM: : :
14.040  E;

15.020  T: Y, YP, Y2P, Y3P:

17.010  TI, TM:
17.011  YM=YMI, YPM=Y1MI, Y2PM=Y2MI, Y3PM=Y3MI
17.012  40;
17.013  IC1=YI, IC2=YPI, IC3=Y2PI, IC4=TI
17.020  YM, YPM, Y2PM, Y3PM:
17.030  YI, YPI, Y2PI, Y3PI:
17.040  A013, A048, A052, A113:

21.010  1C010=ABS(Y2PI)/ Y2PM
21.011  1C011=5/(BETA*10)
21.012  1C012=Y2PM/(YPM*BETA*10)
21.013  1C013=ABS(YPI)/YPM
21.015  1C015=ABS(YI)/YM
21.017  1C017=YPM/(YM*BETA*10)
21.018  1C018=4*YPM/(Y2PM*BETA*10)
21.040  1C040=1/TM↑2
21.041  1C041=1/TM
21.045  1C045=TI/TM
21.047  1C047=1/(TM*BETA)
21.080  1C080=1/10*LN(TM)
21.081  1C081=10/(TM↑2*Y2PM*BETA)

22.010  1S010=Y3P/(Y2PM*BETA*10)
22.012  1S012=-Y2P/(YPM*BETA*10)
22.015  1S015=YP/(YM*BETA*10)
22.045  1D045=-1/(TM*BETA)

23.001  E=LN(T/TM)+LN(TM)
23.002  F=(T/TM)**2*10
23.009  1A009=-4*YPM*TM↑2*YP/(T↑2*Y2PM*YPM*BETA*10)
23.010  1A010=-Y2P/Y2PM
23.012  1A012=YP/YPM
23.013  1A013=4*YPM*TM↑2*YP/(T↑2*Y2PM*YPM*BETA*10)
23.014  1A014=-4*YP/(Y2PM*BETA*10)
23.015  1A015=-Y/YM
23.021  1A021=Y/YM
23.024  1A024=Y2P/Y2PM
23.042  1A042=-4*YPM*TM↑2*YP/(T↑2*Y2PM*YPM*BETA*10)
23.044  1A044=5*Y2P/(Y2PM*BETA*10)
23.045  1A045=T/TM
23.048  1A048=-5*Y2P*TM/(T*Y2PM*BETA*10)
23.049  1A049=5*Y2P*TM/(T*Y2PM*BETA*10)
23.050  1A050=-T/TM
23.051  1A051=-(T↑2)/TM↑2
23.052  1A052=-LN(T/TM)/10
23.054  1A054=T↑2/TM↑2
23.080  1A08=LN(T/TM)/10+LN(TM)/10
23.083  1A083=-(T↑2)/TM↑2
23.113  1A113=E/F
23.114  1A114=-E/F

31.010  1P010=1C010*R1
31.011  1P011=1C011*1A010
31.012  1P012=1C012*1A010
31.013  1P013=1C013*R2
31.015  1P015=1C015*R3
31.017  1P017=1C017*1A012
31.018  1P018=1C018*1A012
31.040  1P040=1C040*1A042
31.041  1P041=1C041*1A048
31.045  1P045=1C045*(-1)
31.047  1P047=1C047*(-1)
31.080  1P080=1C080*(-1)
31.081  1P081=1C081*1A113

32.010  1S010=((1P040+1P041)*10+1P081)/10
32.012  1S012=1P012
32.015  1S015=1P017
32.045  1D045=1P047

33.009  1A009=-(1A013)
33.014  1A014=-(1P018)
33.021  1A021=-(1A015)
33.024  1A024=-(1A010)
33.042  1A042=-(1A013)
33.044  1A044=-(1P011)
33.049  1A049=-(1A048)
33.050  1A050=-(1A045)
33.051  1A051=-(1A054)
33.054  1A054=-(1A083)
33.114  1A114=-(1A113)

40.014  R1=-1, Y2PI > 0?  R1=1
40.044  R2=-1, YPI < 0?  R2=1
40.074  R3=1, YI < 0?  R3=-1

41.014  1K014=1+R1
41.049  1K049=1+R2
41.074  1K074=1+R3
```

Bild 8. HOI-Programm (Fortsetzung)

4.5.2. Potentiometerwerte

Die sich mit dem Computerdiagramm aus den skalierten Problemgleichungen ergebenden mathematischen Ausdrücke für die Einstellung der Potentiometer zeigt Programmteil 21. Die Gleichungen sind in allgemeiner Form gehalten, so daß sich auch für unterschiedliche Anfangsbedingungen und Maximalwerte bei der Exekution immer die richtigen Potentiometerwerte ergeben.

Diese Statements zeigen in augenfälliger Weise die einfache, hybride Kontroll- und Adressierungsmöglichkeit. Die Potentiometer werden fast genauso gesetzt wie die Zellen im Digitalrechner.

Die erforderliche Änderung der Betriebsarten des Hybridrechners, die Umwandlung der Datenformate und die hybride Kommunikation und Datenübertragung vermag der Interpreter HOI automatisch durchzuführen.

4.5.3. Theoretische Werte der Integriereingänge und Verstärkerausgänge

Programmteil 22 und 23 berechnet die normierten Werte für die Summe der Eingangsgrößen der Integrierer und Ausgangsgrößen der Verstärker. Da bei den Integrierern die Möglichkeit besteht, daß die Summe der Eingangsspannungen größer als eine Maschineneinheit wird, ordnet man in solchen Fällen nur den zehnten Teil davon den mit 1 S XXX gekennzeichneten Variablen zu. Auf diese Weise ist gewährleistet, daß HOI die Größen fast immer im „Scaled Fraction"-Format abspeichern kann und keine zeit- und platzaufwendigen Formattransformationen durchführen muß.

Man erkennt, daß bei der Berechnung dieser Größen auf die in Programmteil 13 ermittelten Werte zurückgegriffen wird.

4.5.4. Analogon des Computer-Diagramms

Die Verbindungen der Komponentensymbole des Computer-Diagramms werden in den Programmteilen 31 bis 33 wiedergegeben. Jedes dieser einfachen Statements beinhaltet die algebraische Übertragungsfunktion der einzelnen analogen Komponenten.

4.5.5. Das HOI-Steuerprogramm zur Durchführung des off-line-Checks

Obwohl die in den Abschnitten 4.5.1 bis 4.5.4 beschriebenen Programmteile durch on-line-Eingabe von Kontroll-Statements exekutiert werden können, ist es effektiver, ein kleines Steuerprogramm in HOI zu schreiben. Die Programmteile 1 und 2 führen nach der Eingabe der Parameter und Berechnung der Anfangszeit, Anfangsbedingungen und Maximalwerte den kompletten digitalen off-line-Check des analogen Computerdiagramms einschließlich der erforderlichen Betriebsartenwahl und Textausgabe während der Exekution automatisch durch.

Im off-line-Check werden die Adressen der analogen Komponenten als digitale Variablen aufgefaßt, denen durch mathematische Ausdrücke (expressions) der anderen Variablen Werte zugewiesen werden. Die Ausdrücke können ihrerseits wieder vorher berechnete Variablen enthalten. Da die Werte nach der Exekution im Speicher zur Verfügung stehen, können sie miteinander verglichen werden.

So vergleicht man beispielsweise in den Programmteilen 32 und 33 in der Sprachenbetriebsart Varify die Variablenwerte, die durch die Verbindungsstatements definiert sind, mit den theoretisch aus den Problemgleichungen in 22 und 23 ermittelten.

4.6. Automatisches Skalieren, Setzen und Austesten des analogen Rechenprogramms

Die Operationen, die zum Setzen und Austesten von Potentiometern, Relais und Verbindungen auf dem Analogrechner durchzuführen sind, werden durch die Befehle in den Programmteilen 3 und 4 dargestellt.

Zunächst werden nach der Wahl der Konsole in 3.01 und der Sprachenbetriebsart in 3.03 und 3.04 für T = TI in den Schritten 3.041 und 3.05 nochmals die Anfangsbedingungen berechnet.

Ein automatisches Halt in 3.07 nach dem Ausdrucken einer Mitteilung an den Operateur stellt sicher, daß alle manuellen Handgriffe zum Betrieb des Hybridsystems ordnungsgemäß ausgeführt worden sind, bevor das HOI-Programm über das hybride Interface den Analogrechner anspricht.

Zum Setzen der Potentiometer wird in 3.08 und 3.09 die Konsole 1 gewählt und in 3.1 in die analoge Betriebsart Potentiometersetzen (SP) und logische Betriebsart RUN (RU) geschaltet. Nun können in 3.11 die Potentiometer gesetzt werden. Das geschieht durch Exekution des Programmteils 21, nachdem der Kommentar: „SET POTS" ausgedruckt wurde.

Die richtige Einstellung der Potentiometer wird nun in den Programmschritten 4.01 bis 4.04 überprüft. Durch einfaches Umschalten der Betriebsarten von Hybridrechner und Interpreter in 4.01 und 4.03 wird dazu in 4.04 wieder das Unterprogramm 21 verwendet. Das Protokoll zeigt beim „COEF CHK" zwei Potentiometerwerte, die außerhalb der gewählten Einstelltoleranz liegen:

```
COEF CHK
=.317800     NOT .318167   IN   21.012
=.277400     NOT .277944   IN   21.017
HALT  IN  4.041
```

Es handelt sich dabei, wie man aus den angegebenen Programmschritten ersehen kann, um die Potentiometer 12 und 17. Das HALT am Ende des Checks ermöglicht eine Ursachenanalyse und -korrektur, so daß eventuelle Setzfehler eliminiert werden können.

Durch Eingabe eines GO-Befehls (G;) über die Teletype kann das Programm wieder gestartet werden.

Es folgt der „POT OUTPUT CHECK" in 4.05, bei dem die am Schleifer liegende Spannung der Potentiometer mit der sich theoretisch aus Eingangsspannung und Potentiometerstellung ergebenden verglichen wird. Auch hierbei werden, wie das Protokoll zeigt, alle nicht innerhalb der Toleranz liegenden Potentiometer ausgedruckt.

In gleicher Weise überprüft HOI in 4.06 die Übertragungsfunktion und programmgemäße Verschaltung der Verstärker und in 4.07 die Summe der Eingangsspannungen der Integrationsnetzwerke, so daß Steckfehler immer gefunden werden.

Die restlichen Programmschritte des Programmteils 4 führen für den Zeitpunkt T = TM die gleichen Testfunktionen durch und überprüfen damit einmal die Gangbarkeit der Potentiometer, zum anderen aber das einwandfreie Umschalten der in Unterprogramm 41 angesprochenen Funktionsrelais.

Auch hierbei wird durch ein Halt in 4.15 eine Korrekturmöglichkeit geboten.

In den Programmschritten 5.01 bis 5.07 wird der on-line-Check dahingehend vervollkommnet, daß das einwandfreie Arbeiten der analogen Rechenschaltung mit den in den Programmteilen 22 und 23 aus den Gleichungen definierten theoretischen Werten verglichen wird. Die Gleichungen enthalten alle für eine Amplituden und Zeitnormierung erforderlichen Skalierungswerte in allgemeiner Form, so daß ihnen je nach der Wahl von TM und BETA die in der TIMESOLUTION ermittelten optimalen Werte zugeordnet werden. Programmschritt 5.03 läßt erkennen, daß der Test nur für den Zeitpunkt T = TM durchgeführt zu werden braucht. Ergibt der Test keine Fehlermeldungen, ist gewährleistet, daß die analoge Rechenschaltung die normierten Problemgleichungen richtig beschreibt.

4.7. Dynamischer Test des analogen Programms

Wie unter Punkt 3.4 schon beschrieben, läßt sich HOI auch für den dynamischen Test vorteilhaft einsetzen. Im vorliegenden Beispiel, bei dem die allgemeine Lösung der Differentialgleichung durch die Ausdrücke (2) bis (5) gegeben ist, erübrigt sich die Aufstellung eines Integrationsalgorithmus. Hier bietet sich vielmehr die Möglichkeit, die Lösungskurven mit den digital berechneten Lösungsgrößen, die in Tabellenform bereits vorliegen, direkt zu vergleichen. Größere Abweichungen würden in diesem Fall auf einen schadhaften Integrierer hinweisen.

Bei komplexeren Programmen mit mehr als 10 Integrierern würde man die analoge Rechenschaltung eine definierte Zeitspanne freigeben, die Ergebnisse über Track-/Store-Einheiten dem Digitalrechner zuführen und im Varify-Mode des Interpreters die Integrierer-Ausgänge mit den digital berechneten, theoretischen Werten vergleichen. Hierbei können wieder die bereits aufgestellten Programmteile 22 und 32 in unveränderter Form benutzt werden. Auf diese Weise wird dann auch ein defekter Integrierer sicher gefunden.

4.8. Automatische Durchführung des hybriden Programms

Nachdem im Programmteil 5 der „EQUATION CHECK" beendet und damit gewährleistet ist, daß im analogen und digitalen Programm keinerlei Fehler mehr vorhanden sind, kann in 5.071 der Analogrechner in den für das interessierende Zeitintervall $TI \leqslant T \leqslant TM$ erforderlichen Betriebszustand gesetzt werden. Die bereits digital berechneten und in den Zellen IC1 bis IC4 abgelegten Anfangsbedingungen werden in 5.08 den entsprechenden Variablen wieder zugeordnet. Potentiometer und Relais werden nach richtiger Betriebsartenwahl gesetzt und ein letztes Mal mit den theoretischen Werten verglichen. Eine letzte Nacheichungsmöglichkeit wird durch das HALT in 5.084 geboten, deren Erfolg in 5.085 durch Verifikation der Potentiometerschleiferspannungen beobachtet werden kann. Das letzte Statement in 5.09 setzt den Hybridrechner in Repitierbetrieb, und die Lösungskurven Y bis $\bar{Y}$ erscheinen als stehendes Bild optimal normiert auf dem Vierkanalsichtgerät. Da der Zeitbereich und die Maximalwerte, auf die die einzelnen Kurven bezogen sind, bereits im Protokoll stehen, können die Ergebnisse unmittelbar und mit maximaler Genauigkeit abgelesen werden.

5. Ausblick

Die Ausführungen haben gezeigt, daß der hybride Interpreter HOI bei der Lösung komplexer Probleme und im besonderen bei umfangreichen Simulationen eine wesentliche Hilfe für den Benutzer eines Hybridsystems darstellt. HOI macht auch das umfangreichste und komplizierteste Problem transparent und verzichtet trotz bzw. gerade wegen des hohen Niveaus dieser Sprache *nicht* auf die Möglichkeit der Zwiesprache mit dem Rechner. Gerade diese Eigenschaft ist es aber, die das Hybridsystem für die Simulation so geeignet und vorteilhaft macht. Gleichzeitig bleibt die für die Simulation so notwendige Eingriffsmöglichkeit nicht nur erhalten, sondern vereinfacht und erhöht sich noch durch die besonderen Fähigkeiten der Sprache (Iterationsbefehle). Bleiben also nur noch zwei Punkte, die automatisiert werden können:

a) das Aufstellen der skalierten Problemgleichungen,
b) das manuelle Stecken des Programms.

Auch diese beiden Punkte sind für den finanziell starken Benutzer bei EAI bereits gelöst. Eine spezielle, rein digitale Software erstellt mit dem Programm APSE aus den physikalischen Ausgangsgleichungen nach deren automatischer Skalierungsumwandlung die analoge Rechenschaltung, die beim neuesten EAI-690-Hybridsystem mit „automatic patching"-Zusatz über eine Schaltmatrix auf den Analogrechner übertragen wird.

Ausgewählte Anwendungsprobleme von Hybridrechnern als repräsentative Beispiele für die Informatikausbildung

Dr. Gerhard Schweizer

1.) Einführung

Wir untersuchen anhand von Beispielen Probleme bei der Anwendung des Hybridrechners zur Echtzeitsimulation. Die zu simulierenden Systeme werden mathematisch durch Differentialgleichungen beschrieben. In der Praxis treten bei Echtzeitsimulationen im allgemeinen stückweise stetige Systeme auf, deren Größen sich kontinuierlich mit der Zeit ändern.

Stetige Systeme wurden bislang vorwiegend auf Analogrechnern simuliert, die durch ihr Arbeitsprinzip (stetige Integration u.a.) und ihre für Ingenieure leicht erlernbare Sprache dazu prädestiniert sind.

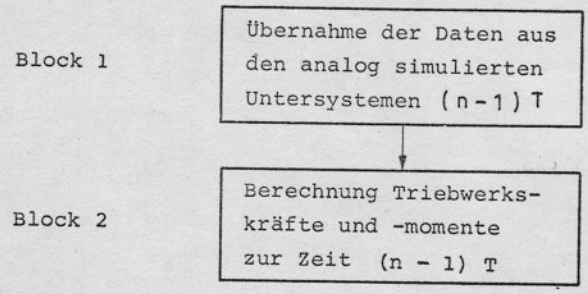
Die Nachteile und Beschränkungen des Analogrechners, vor allem die oft geringe Genauigkeit und der sehr große rechentechnische Aufwand infolge der parallelen Implementierung jeder Rechenoperation auf einem getrennten Rechenelement, führen zum Einsatz von Digital- und Hybridrechnern zur Simulation. Bei der Echtzeitsimulation ist im allgemeinen nur die Verwendung eines Hybridrechners sinnvoll, weil im Laufe der Durchführung Echtteile einzubeziehen sind, deren Anschluß an den Digitalrechner einen Analogrechner als Zwischenglied erfordert.

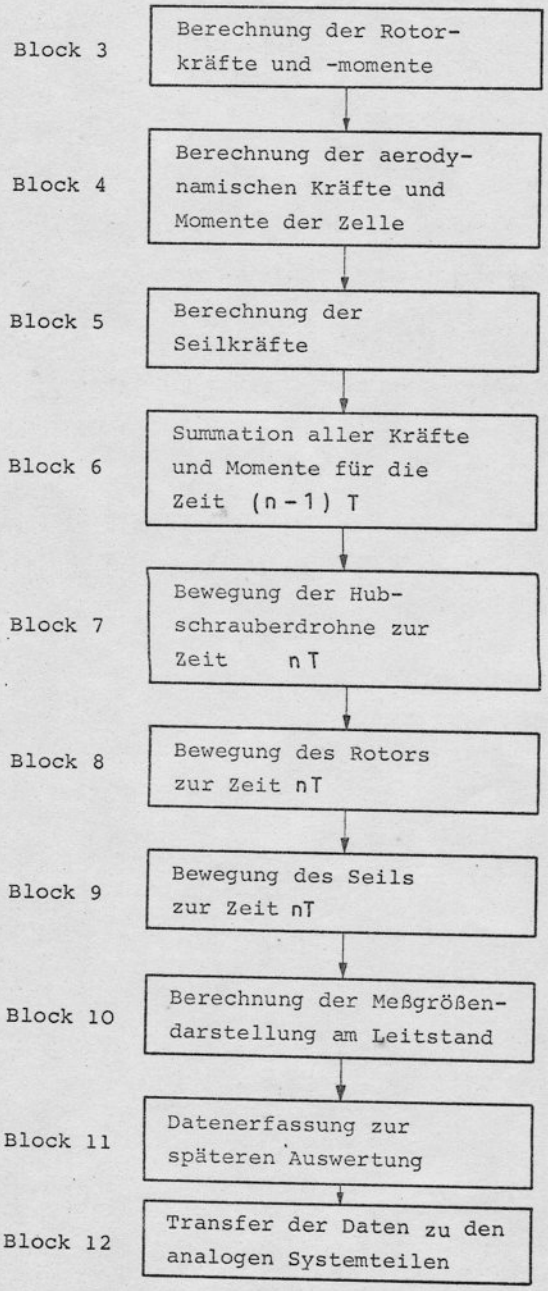
Der Einsatz von Hybridrechnern kann nur dann Erfolg haben, wenn für seine Bedienung genügend programmtechnische Mittel zur Verfügung stehen und wenn über mögliche Modellbildungen hinreichend ge-

Anforderungen her nur auf Hybridrechenanlagen durchgeführt werden können. Anhand dieses ersten Beispiels werden Besonderheiten des digitalen Teils der hybriden Simulation gezeigt. Die Problemstellung zeigt Bild 1. Bei der Gesamtsimulation müssen eine Anzahl von Untersystemen betrachtet werden. Echtzeitsimulation ist erforderlich, weil Echtteile,z.B. der Flugregler und der bedienende Mensch mit dem Lenkstand in die Simulation einbezogen werden müssen. Bei der hybriden Simulation muß man auf eine möglichst große Übersichtlichkeit des Simulationsmodelles achten, um eine gute Variabilität bzw. eine einfache Austauschbarkeit von Programmteilen zu gewährleisten. Deshalb muß bei der hybriden Simulation das Gesamtsystem in Teilsysteme zerlegt werden, wobei die Koppelkräfte und -momente, bzw. sonstige Anschlußbedingungen berücksichtigt werden müssen. Die Erfahrung zeigt, daß bei anspruchsvollen Simulationen es vorteilhaft ist, möglichst viele Teilsysteme auf dem digitalen Teil nachzubilden.

Im Falle der gefesselten Hubschrauberdrohne von Bild 1 würden in einem Simulationstakt folgende Blöcke durchlaufen.

Ablaufschema für einen Rechenzyklus vom Zeitpunkt (n-1)T bis nT




Block 3
Berechnung der Rotorkräfte und -momente
Block 4
Berechnung der aerodynamischen Kräfte und Momente der Zelle
Block 5
Berechnung der Seilkräfte
Block 6
Summation aller Kräfte und Momente für die Zeit (n-1) T
Block 7
Bewegung der Hubschrauberdrohne zur Zeit nT
Block 8
Bewegung des Rotors zur Zeit nT
Block 9
Bewegung des Seils zur Zeit nT
Block 10
Berechnung der Meßgrößendarstellung am Leitstand
Block 11
Datenerfassung zur späteren Auswertung
Block 12
Transfer der Daten zu den analogen Systemteilen

Dieses Schema für den Rechenablauf zeigt die zeitliche Folge der digitalen Rechenoperationen und die Abgrenzung der einzelnen prinzipiellen Rechenabläufe. Die Formulierung der Gleichungen für einzelne Blöcke ist hier noch offen. Unsere spezielle Erfahrung ergibt, daß es oft nicht geschickt ist, die Aussage der Gleichungen in der Form eines Blockschaltbildes darzustellen, wobei jeder Block eine mathematische Operation, wie Summierung, Multiplikation u.a. darstellt.

Die Vorteile der Blockdiagrammdarstellung sind gegenüber der Differentialgleichung

- ein anschauliches Bild der Problemstruktur, die den Wirkungszusammenhang aller physikalischen Größen leicht erkennen läßt;

- es lassen sich leicht Blöcke darstellen, die analytisch nicht ohne weiteres erfaßbar sind. Beispiel unstetige Funktion u.a.;

- die Darstellung lehnt sich völlig an die bisher gewohnte Analogrechnersimulation an.

Diesen Vorteilen stehen Nachteile gegenüber:

- bei komplexen Simulationsaufgaben wird das Problem durch komplizierte mathematische Formulierungen beschrieben, bei denen eine Darstellung als Blockdiagramm mit Blöcken für jede einzelne arithmetische Operation viel zu aufwendig und damit unübersichtlich wird. Beispiele sind drei dimensionale Koordinatentransformationen;

- die Darstellung als Blockdiagramm ist nur dann sinnvoll, wenn eine darauf abgestimmte Simulationssprache zur Verfügung steht, die darauf abgestimmt ist, daß die Eingangs- und Ausgangsgrößen in Form von Gleichheitszeichen und Klammern miteinander verbunden werden.

  Beispiel:

  ausgang = Operationstyp (eingang 1, eingang 2 ...)

Die heute zur Verfügung stehenden Simulationssprachen sind großenteils wenig zeiteffizient, so daß bei Echtzeitsimulationen Schwierigkeiten auftreten können.

Bei großen komplexen Simulationsproblemen ist es oft am vorteilhaftesten,mit einer Unterprogrammtechnik zu arbeiten, wobei einzelne Systemteile in der Maschinensprache oder in einer problemorientierten Sprache, z.B. Fortran oder Algol geschrieben werden können. Wichtig ist, daß diese Unterprogramme durch ein leistungsfähiges, leicht zu handhabendes Verteilerprogramm miteinander verbunden werden können, das gleichbleibende, immer wiederkehrende Programmteile enthält, z.B. die Betriebsartensteuerung des Analogrechners.

3.) Die Modellbildung für das Untersystem Seil

Nach den allgemeinen Erläuterungen des letzten Abschnittes soll die Modellbildung für das Untersystem Seil des in Bild 1 dargestellten Problems diskutiert werden. Das Seil hat drei Aufgaben. Es dient zur Fesselung der Hubschrauberdrohne, zum Hochpumpen des Treibstoffes für die Turbine und zur Übertragung elektrischer Signale. Bei der modellmäßigen Nachbildung müssen wir darauf achten, daß sich sowohl der

Fesselpunkt an der Drohne als auch der Aufhängepunkt an dem Lastwagen in zeitlich nicht voraussagbarer Weise bewegen. Die Ermittlung der durch das Seil am Fesselpunkt der Drohne angreifenden Kräfte und Momente ist das primäre Ziel bei der Nachbildung des Seils im Rahmen der Simulation.

Infolge des beweglichen Fessel- und Aufhängepunktes scheidet die Lösung der Seilbewegung durch die Differentialgleichung der schwingenden Seile aus. Zur Ermittlung der am Fesselpunkt angreifenden Kräfte und Momente müssen wir die longitudinalen und die transversalen Schwingungen des Seils betrachten.

Für die Nachbildung des Seils bietet sich zunächst ein Modell nach Bild 2 an. Dabei wird das Seil in Teilmassen 1 bis $i_{max + 1}$ aufgeteilt, die reibungslos an den Gelenken miteinander verbunden sind. Die einzelnen Teilmassen werden durch Punktmassen, die durch Federn mit der Steifigkeit C_{F1} und der Dämpfung K_{D1} miteinander verbunden sind, ersetzt. Unter der Annahme, daß das Seil durch insgesamt nur eine Masse dargestellt wird und die Drohnenmasse groß gegenüber der Seilmasse ist, ergeben sich die resultierenden Frequenzen der longitudinalen Schwingungen näherungsweise zu

$$\omega_1 = \sqrt{\frac{2c}{m}} \quad , \quad \omega_2 = \sqrt{\frac{c}{m_F}}$$

c = Seilsteifigkeit
m_F = Drohnenmasse
m = gesamte Seilmasse

Da in Wirklichkeit ein annähernd starres Seil vorliegt, muß eine hohe Steifigkeit vorgesehen werden.

Fall nicht brauchbar sind.

Zur Festlegung eines anderen brauchbaren Modells machen wir einige physikalische Überlegungen. Die durch die Masse des Seils hervorgerufenen longitudinalen Schwingungen beeinflussen die Bewegung der Drohne infolge der vergleichsweisen großen Masse praktisch wenig. Bei weit ausgefahrenem Seil sind dagegen die durch die tansversalen Schwingungen des massebehafteten Seils auftretenden Kräfte am Fesselpunkt nicht vernachlässigbar für die Drohnenbewegung. Wir brauchen deshalb ein Modell für das Seil, dessen Massenverteilung im wesentlichen nur die Translationsschwingung beeinflußt.

Bild 3 zeigt das verwendete Modell. Das Seil wird durch eine masselos gedachte Feder hoher Steifigkeit C_{F1} ersetzt, die den Flugkörper mit dem Bodenpunkt verbindet. Die Steifigkeit C_{F1} wird gleich der Steifigkeit des in Wirklichkeit annähernd starren Seils gemacht. Um dieses innere Seil wird ein Mantel gedacht, der durch mehrere Teilmassen im Modell dargestellt wird. Diese Teilmassen, die jeweils dem Seilgewicht - dividiert durch die Zahl der Einzelmassen - entsprechen, sind untereinander und mit dem Fessel- bzw. Aufhängepunkt durch Federn kleiner Steifigkeit C_{F2} verbunden, welche die Seilteilstücke des Mantels auf die Ausgangslängen zurückstellen. Zur Dämpfung des Seils sind parallel zu jeder Feder Dämpfer angeordnet. Das innere masselos gedachte Seil hoher Steifigkeit C_{F2} gleitet reibungsfrei in dem Seilmantel.

Durch diese Anordnung wird erreicht, daß für die Rückstellung der Hubschrauberdrohne in longitudinaler Richtung, sowie der Seilmassen bei der Transversalschwingung im wesentlichen das innere Seil hoher Steifigkeit maßgebend ist. Die Seilmasse wirkt sich dagegen auf die am Fesselpunkt infolge der Transversalschwingung auftretenden Kräfte voll aus.

Die Auslegung der im Seilsystem nach Bild 3 angenommenen Federn und Dämpfer wurde nach zwei Gesichtspunkten wahrgenommen. Erstens soll das Seil möglichst systemnahe nachgebildet werden und zweitens muß gewährleistet sein, daß das Seilmodell unter Echtzeitbedingungen numerisch berechnet werden kann. Dazu müssen in Abhängigkeit der Seillänge sowohl die Anzahl der verwendeten Teilmassen als auch die beschreibenden Parameter so gesteuert werden, daß keine der auftretenden Seilsystemeigenfrequenzen größer als der zehnte Teil der Tastfrequenz wird.

Berechnung der Seilkräfte

Wir berechnen die Seilkräfte in einem erdfesten System mit dem Bezugspunkt O. Nach Bild 3 ergibt die Seillänge der i-ten Teilfeder

$$\Delta \bar{r}_i = \bar{r}_{i-1} - \bar{r}_i$$

Daraus resultiert die Änderungsgeschwindigkeit der Federlänge

$$\Delta \dot{\bar{r}}_i = \bar{v}_{i-1} - \bar{v}_i$$

und die Verlängerung der i-ten Feder

$$\Delta L_i = | \Delta \bar{r}_i | - L_{0i}$$

Aus physikalischen Gründen muß $\Delta L_i \geq 0$ sein. Für die Änderung der Verlängerung erhält man

$$\Delta \dot{L}_i = \frac{\Delta \bar{r}_i \cdot \Delta \dot{\bar{r}}_i}{| \Delta \bar{r}_i |} - \dot{L}_{0i}$$

Eine Änderungsgeschwindigkeit der Grundlänge L_{01} ist während des Aus- und Einfahrvorgangs der Drohne möglich.

Die Gesamtverlängerung der durchgehenden inneren Feder bis zur Teilmasse i ist

$$\Delta L_{ges} = \sum_{i=1}^{i_{max}+1} \Delta L_i$$

$$\Delta \dot{L}_{ges} = \sum_{i=1}^{i_{max}+1} \Delta \dot{L}_i$$

Die resultierende Seilzugkraft vor dem i-ten Massenteil wird somit

$$|S_{L_i}| = \Delta L_i \cdot C_{F_2} + \sum_{i=1}^{i_{max}+1} \Delta L_i \cdot C_{F_1} + \Delta \dot{L}_i K_{D_2} + \sum_{i=1}^{i_{max}+1} \Delta \dot{L}_i K_{D_1}$$

In Vektorform können wir die Seilzugkraft wie folgt definieren

$$\bar{S}_{L_i} = \frac{\bar{r}_{i-1} - \bar{r}_i}{|\bar{r}_{i-1} - \bar{r}_i|} \left\{ \left[|\bar{r}_{i-1} - \bar{r}_i| - L_{0i} \right] C_{F_2} + \sum_{i=1}^{i_{max}+1} \left[|\bar{r}_{i-1} - \bar{r}_i| - L_{0i} \right] \cdot C_{F_1} + \right.$$

$$\left. + \left[\frac{\Delta \bar{r}_i \cdot \Delta \bar{\dot{r}}_i}{|\Delta \bar{r}_i|} - \dot{L}_{0i} \right] K_{D2} + \sum_{i=1}^{i_{max}+1} \left[\frac{\Delta \bar{r} \cdot \Delta \bar{\dot{r}}_i}{|\Delta \bar{r}|} - \dot{L}_{01} \right] K_{D1} \right\}$$

Die Seilbewegung ergibt unter Berücksichtigung von Windeinfluß, Erdbeschleunigung und den oben berechneten Seilkräften

$$\ddot{\bar{r}}_i = \bar{g} + \left[\bar{S}_{L_i} - \bar{S}_{L_{i+1}} \right] \frac{1}{m_s} + \frac{\rho}{2} \frac{D}{\rho_L} c_w \left| \dot{\bar{r}}_i - \bar{v}_w \right| \cdot (\dot{\bar{r}}_i - \bar{v}_w)$$

$\bar{g}$ ist der Erdbeschleunigungsvektor, m_s die Teilmasse des Seils, ρ die Luftdichte, D der Seildurchmesser, $\rho_L = m_{s_{ges}} | L_0$ die Massenbelegung des Seils, c_w der Widerstandsbeiwert des Seils und $\bar{v}_w$ die Windgeschwindigkeit.

4.) Die Hybridsimulation einer speziellen adaptiven Regelung

4.1 Ein Prozeß sei mathematisch durch eine nichtlineare Vektordifferentialgleichung

$$\dot{\bar{x}} = \bar{f}\ [\bar{x}\ (t)\ ,\ \bar{u}\ (t)\ ]$$

beschrieben. $\underline{x}(t)$ ist ein Zustandsvektor, $\underline{u}(t)$ der Steuervektor. In der Praxis der Prozeßregelung, insbesondere in der Flug- und Fahrzeugführung tritt das Problem auf, daß der bedienende Mensch nur eine beschränkte Anzahl von Komponenten des Steuervektors u(t) aus anthropotechnischen Gründen benützen kann. Dies kann zur Folge haben, daß das Übergangsverhalten des Prozesses $\underline{f}$ unbefriedigend wird. Durch ein adaptives Regelsystem kann ein zufriedenstellendes dynamisches Verhalten erreicht werden.

Wir verlangen dabei, daß ein Basissystem, dynamisch beschrieben durch die Zustandsgleichung

$$\dot{\bar{x}}_B\ (t) = \bar{f}_B\ \left[\bar{x}_B\ (t),\ \bar{u}_B\ (t)\right]$$

das gleiche dynamische Verhalten wie ein gewünschtes Nominalsystem

$$\dot{\bar{x}}_N\ (t) = \bar{f}_N\ \left[\bar{x}_N\ (t),\ \bar{u}_N\ (t)\right]$$

hat. Damit die Trajektorien gleich sind, muß zu jedem Zeitpunkt

$$\bar{x}_B\ (t) = \bar{x}_N(t)$$

erfüllt sein. Damit gilt auch

$$\dot{\bar{x}}_B\ (t) = \dot{\bar{x}}_N(t)$$

In jedem Zeitpunkt muß also

$$\bar{f}_B \left[\bar{x}_B(t), \bar{u}_B(t) \right] = \bar{f}_N \left[\bar{x}_N(t), \bar{u}_N(t) \right]$$

erfüllt sein.

Der bedienende Mensch soll das Gefühl haben, ein Nominalsystem zu bedienen. Er wird deshalb mit seinem gewohnten Steuervektor $\underline{U}_N(t)$ in das Basissystem eingreifen. Damit obige Gleichung erfüllt ist, lösen wir zu jedem Zeitpunkt nach $\underline{U}_B(t)$ auf. $\underline{U}_B(t)$ ist der modifizierte Eingriff aufgrund des vom bedienenden Menschen eingegebenen Steuersignals $\underline{U}_N(t)$, der das gewünschte dynamische Verhalten erzwingt.

Die eindeutige Auflösung nach $\underline{U}_B(t)$ verlangt, daß das System total steuerbar ist. Physikalisch sind Basis- und Nominalsystem gleich. $\underline{X}_N(t)$ ist eine Untermenge von Lösungen von $\underline{X}_B(t)$. Damit ist die oben geforderte Auflösung nach einem Steuervektor $\underline{U}_B(t)$ immer möglich.

Bei einer Echtzeitsimulation müssen wir das physikalische System $\underline{f}_B$ simulieren und durch numerische Operationen obiges nichtlineares Gleichungssystem zu jedem Zeitpunkt nach $\underline{U}_B(t)$ auflösen. Außerdem muß ein der Realität angepaßter Lenkstand vorhanden sein, mit dem der bedienende Mensch über den Steuervektor $\underline{U}_N(t)$ eingreigen kann.

Aufgabe bei der hybriden Simulation ist u.a., einen geeigneten Algorithmus zu erstellen für die Auflösung eines nichtlinearen Gleichungssystems in Echtzeit. Wir diskutieren zwei untersuchte Lösungsverfahren.

Im ersten Fall nehmen wir an, daß Basis- und Nominalsystem zum Zeitpunkt t_o übereinstimmen

$$\bar{x}_B(t_0) = \bar{x}_N(t_0), \quad \bar{x}_B(t_0) = \bar{x}_N(t_0) = \bar{x}(t_0)$$

In der Nachbarschaft von $\underline{X}(t_o)$ stellen wir die Vektorfunktionen $\underline{f}_B$ und $\underline{f}_N$ durch die ersten Glieder einer Taylorreihe dar und erhalten

$$\bar{x} + \Delta\bar{x} = \bar{f}_B(\bar{x}_o, \bar{u}_{oB}) + \left.\frac{\partial \bar{f}_B}{\partial \bar{x}}\right|_o \Delta\bar{x} + \left.\frac{\partial \bar{f}_B}{\partial \bar{u}}\right|_o \Delta\bar{u}_B$$

$$\bar{x}_o + \Delta\bar{x} = \bar{f}_N(\bar{x}_o, \bar{u}_{oN}) + \left.\frac{\partial \bar{f}_N}{\partial \bar{x}}\right|_o \Delta\bar{x} + \left.\frac{\partial \bar{f}_N}{\partial \bar{u}}\right|_o \Delta\bar{u}_N$$

Wir verlangen, daß die Beziehungen nicht nur für to sondern auch in der Umgebung $t_o + \Delta t$ für den Zustand $\underline{X}_o + \Delta\underline{X}$ übereinstimmen. Deshalb muß außer

$$\bar{f}_B(\bar{x}_o; \bar{u}_{oB}) = \bar{f}_N(\bar{x}_o; \bar{u}_{oN})$$

auch

$$\left.\frac{\partial \bar{f}_B}{\partial \bar{x}}\right|_o \Delta\bar{x} + \left.\frac{\partial \bar{f}_B}{\partial \bar{u}}\right|_o \Delta\bar{u}_B = \left.\frac{\partial \bar{f}_N}{\partial \bar{x}}\right|_o \Delta\bar{x} + \left.\frac{\partial \bar{f}_N}{\partial \bar{u}}\right|_o \Delta\bar{u}_N$$

gelten.

Wir können nach dem Zuwachs $\Delta \bar{U}_B$ des Steuervektors auflösen und erhalten

$$\Delta \bar{u}_B = \left(\frac{\partial \bar{f}_B}{\partial \bar{u}} \right)^{-1} \Bigg|_o \left[\left(\frac{\partial \bar{f}_N}{\partial \bar{x}} \Bigg|_o - \frac{\partial \bar{f}_B}{\partial \bar{x}} \Bigg|_o \right) \Delta \bar{x} + \frac{\partial \bar{f}_N}{\partial \bar{u}} \Bigg|_o \Delta \bar{u}_N \right]$$

Diese Lösung erhält man auch bei dem Versuc h, das nichtlineare Gleichungssystem über die zugehörige Differentialgleichung mittels eines Differenzenverfahrens zu lösen. Die Differentialgleichung lautet

$$\frac{\partial \bar{f}_B}{\partial \bar{x}} \cdot \frac{d\bar{x}(t)}{dt} + \frac{\partial \bar{f}_B}{\partial \bar{u}_B} \cdot \frac{d\bar{u}_B}{dt} = \frac{\partial \bar{f}_N}{\partial \bar{x}} \cdot \frac{d\bar{x}(t)}{dt} + \frac{\partial \bar{f}_N}{\partial \bar{u}_N} \cdot \frac{d\bar{u}_N(t)}{dt}$$

und nach Auflösen der zugehörigen Differenzengleichung

$$\Delta \bar{u}_B = \left(\frac{\partial \bar{f}_B}{\partial \bar{u}_B} \right)^{-1} \left[\left(\frac{\partial \bar{f}_N}{\partial \bar{x}} - \frac{\partial \bar{f}_B}{\partial \bar{x}} \right) \dot{\bar{x}}(t)\, dt + \frac{\partial \bar{f}_N}{\partial \bar{u}_N} \cdot \dot{\bar{u}}_N(t) \cdot dt \right]$$

Im zweiten Fall berechnen wir den Steuervektor $\underline{U}_B(t)$ direkt aus dem nichtlinearen Gleichungssystem iterativ mit dem Newton-Verfahren. Wir schreiben

$$\bar{P}\left[\bar{x}(t), \bar{u}_B(t), \bar{u}_N(t)\right] = \bar{f}_B\left[\bar{x}(t), \bar{u}_B(t)\right] - \bar{f}_N\left[\bar{x}(t), \bar{u}_N(t)\right]$$

Das Newton-Verfahren liefert dazu die Iterationsformel

$$\bar{u}_B^{\,1}(t_n) = \bar{u}_B^{\circ}(t_n) - \left(\frac{\partial \bar{P}(\bar{x}, \bar{u}_B, \bar{u}_N)}{\partial \bar{u}_B} \right)^{-1} \cdot \bar{P}(\bar{x}, \bar{u}_N, \bar{u}_B) \Bigg|_{\substack{\bar{x}(t_n) \\ \bar{u}_N(t_n) \\ \bar{u}_B^{\circ}(t_n)}}$$

Da $\underline{f}_N$ nicht von $\underline{U}_B$ abhängig ist, gilt

$$\frac{\partial \bar{p}}{\partial \bar{u}_B} = \frac{\partial \bar{f}_B}{\partial \bar{u}_B}$$

Wir haben damit die Iterationsgleichung

$$\bar{u}_B(t_n) = \bar{u}_B^{\circ}(t_n) - \left(\frac{\partial \bar{f}_B}{\partial \bar{u}_B}\right)^{-1} \left[\bar{f}_B\left[\bar{x}(t), \bar{u}_B(t)\right] - \bar{f}_N\left[\bar{x}(t), \bar{u}_N(t)\right]\right]_{t_n}$$

Bei kleinen Tastintervallen kann der errechnete Wert als erste Näherung für die iterative Lösung zum Zeitpunkt $\underline{U}_B(t_{n+1})$ verwendet werden.

Wenn mehrere Iterationen erforderlich sind, empfiehlt sich zur Ersparnis von Rechenzeit ein modifiziertes Verfahren.

Da die Invertierung und Ermittlung der Matrix $\partial \bar{f} / \partial \bar{u}_B$ die meiste Rechenzeit benötigt, wird sie nur für die Anfangsnäherung $\underline{U}_B^{\circ}(t)$ berechnet und für alle weiteren Iterationen benützt.

$$\bar{u}^{k+1}(t_n) = \bar{u}_B^{\,k}(t_n) + \left(\frac{\partial \bar{f}_B}{\partial \bar{u}_B}\right)^{-1}\Bigg|_{t_n} \left\{\bar{f}_B\left[\bar{x}(t_n)\ \bar{u}_B^{k}(t_n)\right] - \bar{f}_N\left[\bar{x}(t_n), \bar{u}_N(t_n)\right]\right\}$$

Dadurch reduziert sich der Rechenaufwand bei den dem ersten Näherungsschritt nachfolgenden Iterationen auf die wiederholte Berechnung des Vektors $\underline{f}_B$, eine Vektoraddition sowie die Multiplikation eines Vektors mit

einer Matrix.

Das Verfahren wurde anhand einer Flugzeugregelung untersucht. Bild 4 zeigt den Einfluß der Tastzeit.

5.) Abschließende Bemerkungen

Hybride Echtzeitsimulation hat in der Praxis große Bedeutung. Die diskutierten Beispiele zeigen, daß zur Modellbildung infolge der Echtzeitforderungen eingehende physikalische und numerische mathematische Kenntnisse erforderlich sind. Darüber hinaus wird man aber noch auf lange Zeit hinaus bei der Modellbildung ohne ingenieurmäßige Improvisation nicht auskommen. Es ist zu hoffen, daß durch die fortlaufende Beschäftigung mit der Echtzeitsimulation in späterer Zeit genügend gesichertes Wissen den Anteil der Improvisation immer kleiner macht.

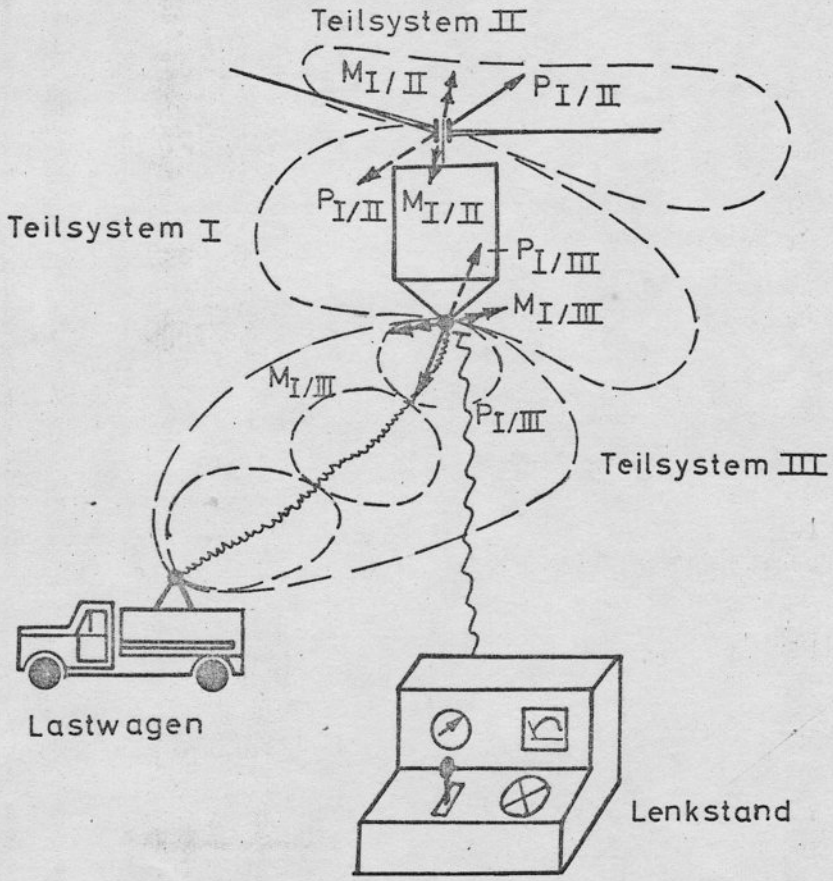


BILD 1 : AUFTEILUNG DES GESAMTEN FLUGKÖRPERSYSTEMS IN TEILSYSTEME

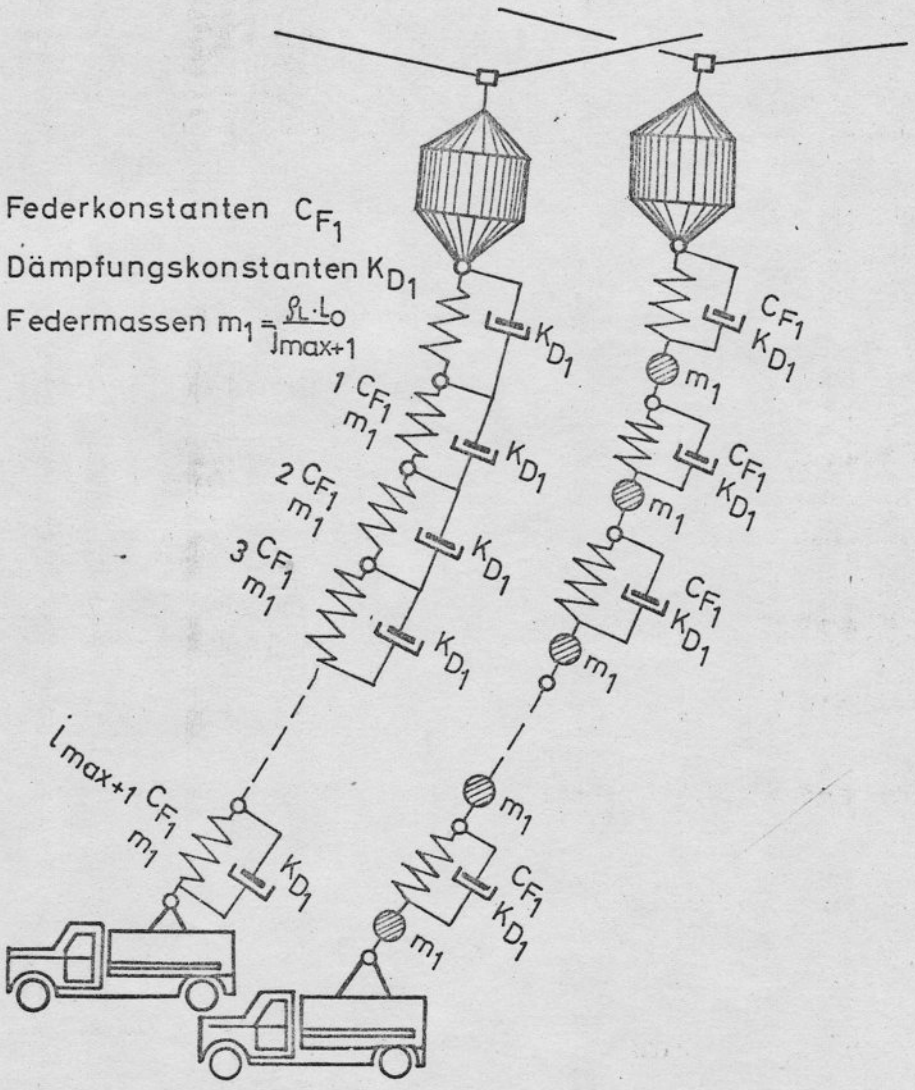


BILD 2 : SEILMODELL MIT GLEICHMÄSSIG VERTEILTEN MASSEN.

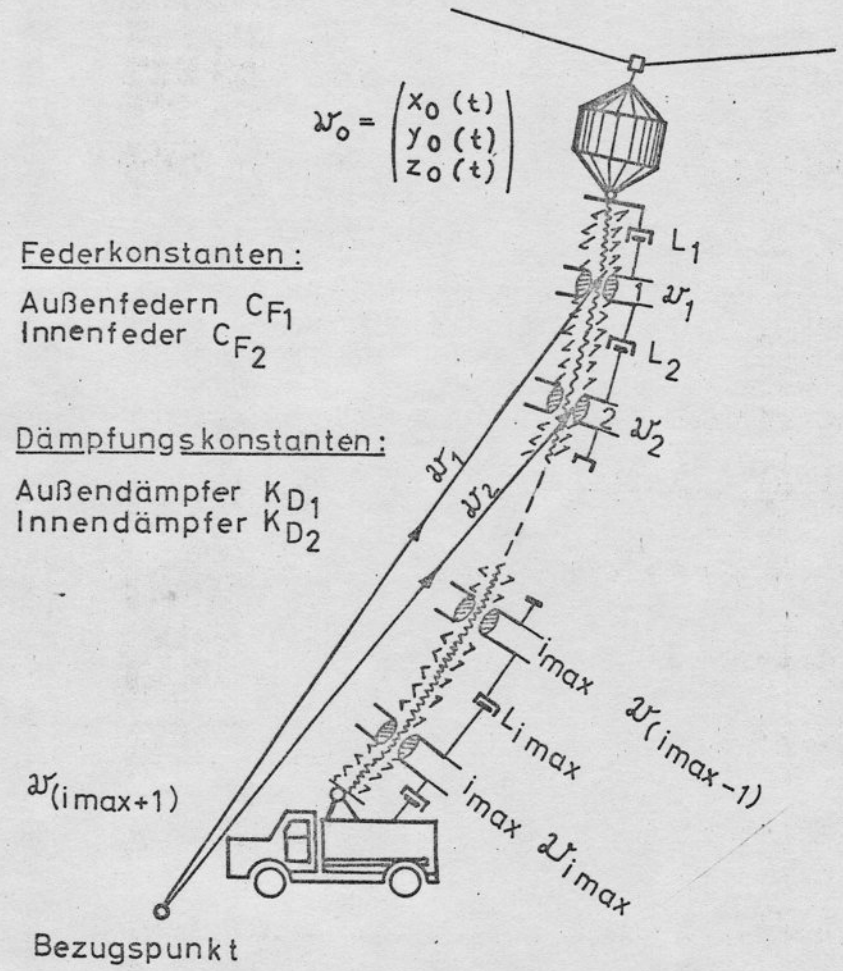


BILD 3: SEILMODELL MIT MASSELOSEM INNEN-SEIL UND MASSEBEHAFTETEM SEIL-MANTEL.

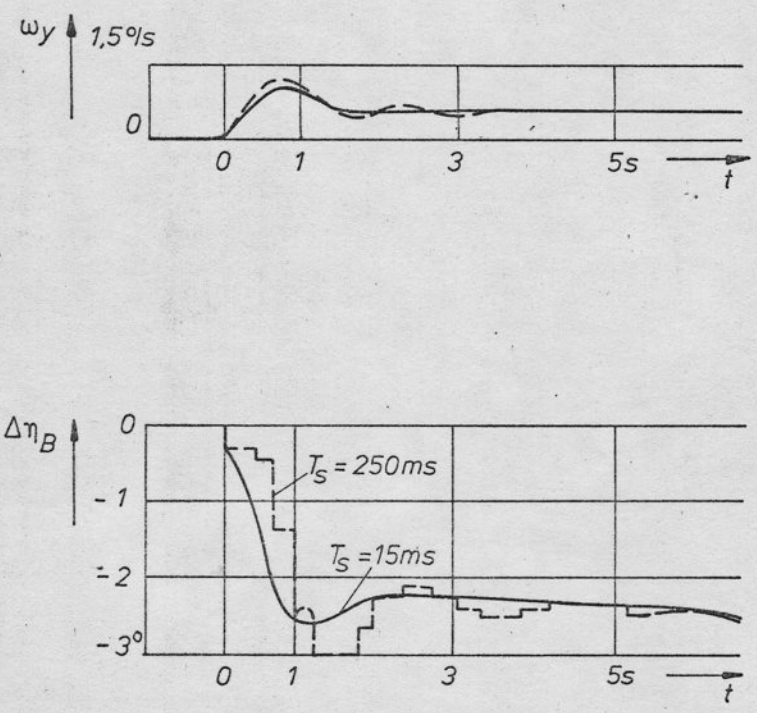


Bild 4. Einfluß der Tastzeit auf die Stabilität