MANUSKRIPT ZUR VORLESUNG

ANALOG- UND HYBRIDRECHNER

SOMMERSEMESTER 1973,
INSTITUT FÜR INFORMATIK DER UNIVERSITÄT STUTTGART,
7 STUTTGART 1, HERDWEG 51

AUTOREN: H.BANNAS, V. HECHT, W. REUSS, H. RZEHAK

I n h a l t s v e r z e i c h n i s :

§ 1 Grundgedanke der analogen Informationsverarbeitung

1.1 Analoges Modell

Die Verwendung des Analogrechners beruht auf der Tatsache, dass es für bestimmte physikalische Vorgänge eine mathematische Beschreibung gibt, die als Grundlage für die Erstellung eines elektrischen Netzwerkes, dem analogen Modell, dient.

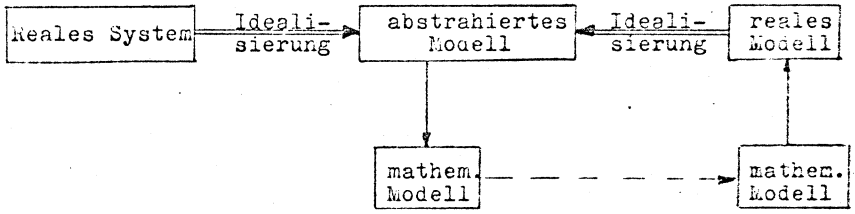


Bild 1.1: Modellgedanke

Im allgemeinen ist man daran interessiert, ein reales System nur im Rahmen einer gewünschten Zielsetzung zu untersuchen. Dazu wird ein komplexes reales System so reduziert, dass das Systemverhalten für den gewünschten Zweck noch hinreichend genau ist. Diese Idealisierung bedingt aber auch, dass das derart abstrahierte Modell dann bei anderen Problemstellungen versagt. Analytisch lässt sich dieses abstrahierte Modell in Form von mathematischen Gleichungen beschreiben. Diese mathematische Beschreibung kann nun - u.U. nach einer weiteren Bearbeitung - als Grundlage für ein analoges, reales Modell dienen. Bei der analogen Informationsverarbeitung kann dieses reale Modell ein elektrisches Netzwerk sein, das die Eigenschaften des abstrahierten Modelles idealisiert besitzt. Eine solche Idealisierung ist z.B. die Voraussetzung einer unendlich grossen Verstärkung beim Operationsverstärker.

1.2 Grundlagen des Analogrechners

Die analoge Informationsverarbeitung weist wesentliche Unterschiede gegenüber der digitalen Verarbeitung auf. (Siehe auch LVA) Beim analogen Rechnen verknüpft man also entweder die gegebenen physikalischen Problemgrössen selbst, oder physikalische Modellgrössen, die den Problemgrössen analog sind. Ein einfaches Beispiel für analoges Rechnen ist auch der Rechenstab, bei dem z.B. die Multiplikation von zwei Ziffern auf die Addition von zwei analogen Strecken zurückgeführt ist.

Bei dem heute üblichen Gleichspannungs-Analogrechner werden die Rechengrössen im realen Modell durch die Amplituden von Spannungen repräsentiert. Wichtig dabei ist, dass diese Spannungen immer kontinuierliche Funktionen der Zeit sind. Der Analogrechner eignet sich deshalb gut zur Lösung bzw. Simulation von zeitkontinuierlichen Systemen, i. a. kurz dynamische System genannt, die durch Differentialgleichungssysteme beschreibbar sind. Nachteilig aber ist, dass nur mit einer einzigen unabhängigen Variablen, nämlich der Zeit, gearbeitet werden kann.

Der Analogrechner besitzt Rechenelemente für viele vorkommende Verknüpfungen, von den 4 Grundrechenarten bis zur Funktionsbildung und Integration. Im Allgemeinen ist die Basis dieser Rechenelemente der Operationsverstärker in Verbindung mit passiven Schaltelementen. Das einfachste analoge Rechenelement für die Verknüpfung "Addition" kann aber auch durch die nebenstehende Schaltung realisiert werden.

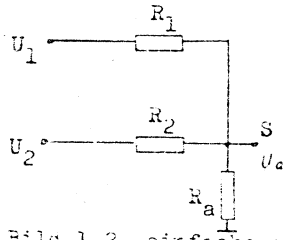


Bild 1.2 einfache analoge Summenbildung.

Es gilt:

$$U_a = \frac{R_a}{R_1+R_a} U_1 + \frac{R_a}{R_2+R_a} U_2 \qquad (1.1)$$

$$= \frac{1}{1+\frac{R_1}{R_a}} U_1 + \frac{1}{1+\frac{R_2}{R_a}} U_2$$

Ist die Beziehung

$$R_a \gg R_1, R_2 \qquad (1.2)$$

erfüllt, dann kann das Ergebnis recht genau sein. Allerdings ergeben sich beim Zusammenfügen mehrerer solcher Rechenelemente Schwierigkeiten, da dann die Belastung durch die nachfolgenden Stufen zu berücksichtigen ist.

Will man beim digitalen Rechnen nur 2 Dualstellen miteinander addieren, so benötigt man einen Volladdierer, der z.B. aus 2 Halbaddierern gemäss Bild 1.3 aufgebaut sein kann. Dabei ist aber nur eine Unterscheidung zwischen zwei Werten möglich. Eine solche Genauigkeit lässt sich auch mit der Schaltung nach Bild 1.2 ohne jede Schwierigkeit erreichen.

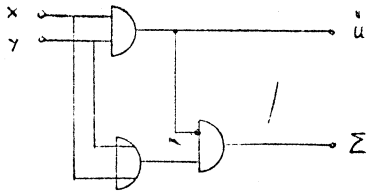


Bild 1.3 Halbaddierer

Die Genauigkeit des Analogrechners ist durch die Rechenkomponenten auf etwa 10^{-3} ... 10^{-4} beschränkt. Außerdem ist der beim analogen Rechnen ausnützbare Wertebereich durch physikalische Erscheinungen wie Drift und Rauschen auf etwa 6 Zehnerpotenzen begrenzt. Beim Digitalrechner dagegen hängt die Genauigkeit von der Wortlänge und dem numerischen Verfahren ab, und der Wertebereich kann durch die Gleitkomma-Darstellung praktisch unbegrenzt gross gemacht werden.

Da beim Analogrechner für jede notwendige Verknüpfung ein Rechenelement zur Verfügung steht, können alle Operationen gleichzeitig ausgeführt werden. Die zur Lösung eines Problems benötigte Zeit ist auf Grund dieser parallelen Arbeitsweise

i.a. wesentlich kürzer als bei einer digitalen Lösung, da hier die Operationen nacheinander ausgeführt werden (serielle oder sequentielle Arbeitsweise).

Die Zeitvorteile der parallelen Arbeitsweise werden z.T. in bescheidenem Umfang auch schon für Digitalrechner ausgenützt, dann nämlich, wenn mehrere parallel arbeitende Rechenwerke vorhanden sind. Diese Überlegungen gewinnen immer mehr an Bedeutung, allerdings treten dann ähnliche Probleme auf, wie sie z.B. von der einheitlichen Programmierung von Hybrid-Systemen her unter dem Begriff Systemaufteilung bekannt sind.

Unter der Programmierung eines Analogrechners versteht man das Zusammenfügen der Rechenelemente in der gewünschten Art und Weise, wobei für jede auszuführende Verknüpfung ein Rechenelement zur Verfügung stehen muss. Die Vorschrift für das Aufstellen des "Programms" resultiert direkt aus der mathematischen Problembeschreibung. Die Verbindung der analogen Rechenelemente erfolgt meistens noch manuell auf einem Steckbrett, doch sind auch verschiedene Verfahren zur automatischen Erstellung der Rechenschaltung (Automatic Patching) in der Implementierungsphase.

In Tabelle 1 sind die wesentlichen Gesichtspunkte noch einmal zusammengestellt.

Tabelle 1. Vergleich von Digital- und Analogrechner

Merkmal		Digitalrechner	Analogrechner
Variablendarstellung	Verfügbarkeit	diskontinuierlich	kontinuierlich
	Wertevorrat	diskret	analog
Signalparameter		Impulsanzahl	Spannungsverlauf
Signalauswertung		Zählung	Messung
Arbeitsweise (Rechenablauf)		seriell in 1 Rechenwerk	parallel in n Rechenelementen
Grundrechenoperationen		$+ - \times$: keine Integration	$+ - \times$: und Integration: $\int dt$
Echtzeitsimulation bzw. Zeitraffung		seltener möglich	leicht möglich
theoretische Grundlagen		Numerische Mathematik und Logik	physikalische Gesetze (Analogiebeziehungen)
		Zusatzkenntnisse erforderlich	normale Fachkenntnisse
Problemabstraktion		sehr stark	unbedeutend
Programmierung		Algorithmus	Koppelplan (Signalflußdiagramm)
Programmänd.		vorher geplant	leicht möglich
Mensch-Maschine-Kontakt		unzureichend	gut
Genauigkeit		beliebig	10^{-3} (10^{-4})
		verfahrensabhängig	bauelementeabhängig
Zahlbereich		$\sim 10^{\pm 20}$	$(10^{-4}) \ldots 1$
Reproduzierbarkeit		eindeutig	umweltabhängig
Anzahl der Unabhängigen		beliebig	eins (Zeit)
Normierung		meist überflüssig	Kenntnisse erforderlich
		Gleitkommarechner	Festkommarechner
Aufgabenbeschränkung		kaum	durch Rechnerbestückung
Programm- und Funktionsspeicher		billig	teuer
Sonderfunktionen		beliebig	wenige, ungenau
		Software	Hardware
automatische Dokumentation		gut	keine

1.3 Gesichtspunkte beim Einsatz eines Analogrechners

Auf Grund seiner Eigenschaften ist der Analogrechner mehr als nur ein blosses Werkzeug zur Lösung von Differentialgleichungssystemen. Insbesondere lässt sich zwischen analoger Rechenschaltung und dem physikalischen Modell eine Eins-zu-Eins-Zuordnung herstellen. Dadurch werden interne Systemvorgänge besonders transparent. Auch bei den Simulationssprachen für die digitale Simulation kontinuierlicher Systeme wurde versucht, diese Eins-zu-Eins-Zuordnung beizubehalten, indem man i. a. eine blockorientierte Struktur zugrunde legte. Wegen der besonders engen Mensch-Maschine-Beziehung ist es möglich, direkt Systemparameter z.B. durch Veränderung von Koeffizienten zu beeinflussen. Gekoppelt mit der anschaulichen Darstellung auf dem Oszillograph wird dadurch ein echter Dialogverkehr ermöglicht.

Der Analogrechner zeichnet sich besonders dadurch aus, dass er auch ein Rechenelement für die Verknüpfung "Integration" besitzt. Diese echte Integration in Verbindung mit der parallelen Arbeitsweise befähigen den Analogrechner auch zur Simulation von dynamischen Systemen in Echtheit mit extremen Zeitbedingungen. Die Modellvorgänge laufen dann genauso schnell ab wie die entsprechenden Vorgänge am echten System. Dadurch ist auch möglich, sogenannte Echtteile, z.B. das Steuerwerk eines Flugzeuges einschliesslich dem Piloten, in die Simulation mit einzubeziehen.

§ 2. Die gemischte analoge/digitale Informationsverarbeitung

2.1 Bekannte Anwendungen gemischter Informationsverarbeitung

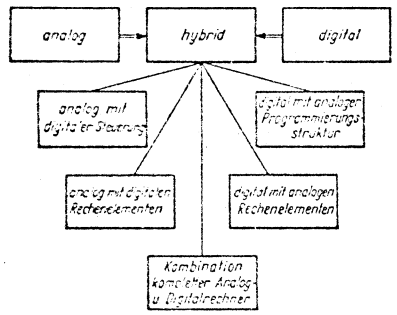


Bild 2.1 Das Spektrum elektronischer Rechenanlagen

Bei vielen Problemen ist von vornherein klar, ob der Einsatz eines Analog- oder eines Digitalrechners zweckmässig ist. In manchen Fällen erfordert die zu lösende Aufgabe aber solche Rechnereigenschaften, dass ein Rechnertyp allein nur unzureichend imstande ist, diese Bedingungen zu erfüllen. Ein Beispiel dafür kann eine Simulation in Echtzeit sein, wenn die Geschwindigkeit des Digitalrechners und die Genauigkeit des Analogrechners allein nicht ausreichen. Vor rund 15 Jahren begann sich die Lücke im Spektrum zwischen den Analogrechnern und Digitalrechnern zu schliessen (Bild2.1), indem man von dem Bemühen ausging, Eigenschaften des Analogrechners mit denen des Digitalrechners zu koppeln.

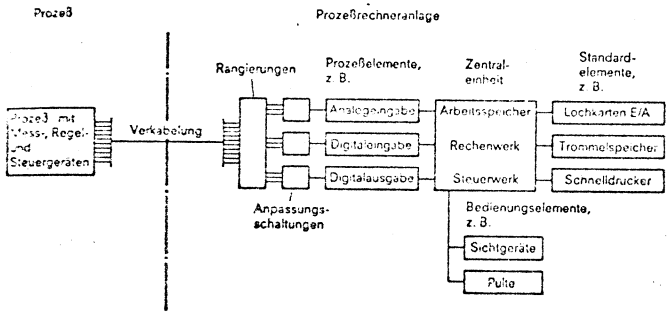


Bild 2.2 Aufbau eines Prozessrechnersystems

Ein anderes Anwendungsfeld, wo eine gemischte Informationsverarbeitung vorkommt, ist die Prozeßrechentechnik.

Dabei treten schon bei der Messwerterfassung analoge und digitale Werte gemischt auf. In der Zentraleinheit der Prozessrechneranlage erfolgt dann eine digitale Verarbeitung, deren Resultate wiederum in Form von analogen und digitalen Daten an den Prozess übermittelt werden müssen.

2.2 Überlegungen zur Arbeitsteilung zwischen Analog- und Digitalrechner

Am Beispiel eines Hybrid-Systems, bei dem komplette Analog- und Digitalrechner miteinander kombiniert werden, lassen sich die Überlegungen zeigen, die vor einer sinnvollen Aufteilung eines gegebenen Problems auf die beiden Rechner angestellt werden müssen. Nur wenn die Eigenschaften beider Rechner, ihre Vor- sowie ihre Nachteile, berücksichtigt werden, dann wird eine Problemaufteilung so ausfallen, dass die Vorteile eines Hybrid-Systems die Nachteile überwiegen, die in jedem Fall in Kauf genommen werden müssen. Zu den Vorteilen, die man durch Kopplung mit einem Analogrechner zusätzlich ausnützen kann gehören z.B.

- parallele Arbeitsweise (u.U. Voraussetzung für Echtzeitsimulation)
- echte Integration
- Austauschbarkeit mit Echtteilen und
- enge Mensch-Maschine-Beziehung.

Daneben sind aber auch Nachteile zu berücksichtigen, wie

- Erstellen der Rechenschaltung und
- Skalierung (da der Analogrechner als Festkommarechner aufgefasst werden kann).

In den Kapiteln §§ 4, 5 und 8 werden diese Punkte erläutert.

Bei vielen Problemen zeigt es sich, dass die Bearbeitung von Teilsystemen mit hohen Eigenfrequenzen keine hohe Lösungsgenauigkeit erfordert, während Teilsysteme mit hoher Rechengenauigkeit keine hohe Rechengeschwindigkeit verlangen. In einem solchen Fall ist die Aufteilung auf die beiden Rechner eigentlich schon vorgegeben. Aber auch die benötigte Rechengeschwindigkeit allein kann eine entsprechende Aufteilung erzwingen, denn Teilsysteme mit Eigenfrequenzen über rund 1 HZ können i.a. nicht mehr auf dem Digitalrechner bearbeitet werden, während Teilsysteme mit Eigenfrequenzen unter 1 HZ schlecht auf dem Analogrechner zu lösen sind.

Ein weiteres Kriterium kann sein, dass Integrationen soweit wie möglich auf dem Analogrechner durchzuführen sind. In manchen Fällen lasst sich aber eine numerische Integration auf dem Digitalrechner nicht umgehen. Dann ist ein geeignetes Integrationsverfahren auszusuchen, bei dem zwischen benötigter Rechenzeit und Genauigkeit der Lösung i.a. ein Kompromiss gebildet werden muss. Auch das Problem der numerischen Instabilität eines numerischen Verfahrens muss u.U. berücksichtigt werden, wenn sich durch akkumulierte Rundungsfehler die Lösungsfunktion des Verfahrens immer weiter von der exakten Lösungsfunktion entfernt. Sind bei einem gegebenen Problem nichtlineare Verknüpfungen herzustellen, dann können diese Verknüpfungen auf dem Analogrechner häufig gar nicht oder nur mit komplizierten Schaltungen realisiert werden, die man oft nur mit elektrotechnischen Vorkenntnissen findet bzw. versteht. Bei einer Aufteilung kann auch dieser Punkt entscheidend sein, da diese nichtlinearen Zusammenhänge auf dem Digitalrechner viel leichter zu programmieren sind. Auch logische Entscheidungen lassen sich auf dem Analogrechner nur in beschränktem Umfang durch Erstellen eines logischen Schaltnetzes auf dem Digitalzusatz programmieren, während eine Formulierung auf dem Digitalrechner kaum Schwierigkeiten macht. Ähnlich verhält es sich mit der Datenspeicherung, die auf dem Analogrechner eine komplizierte Integrierersteuerung erfordert und auf dem Digitalrechner problemlos zu handhaben ist.

Aus diesen Merkmalen lassen sich die typischen Anwendungen ableiten. Der Analogrechner ist in erster Linie für das Studium dynamischer Systeme, d.h. für die Lösung von gewöhnlichen Differentialgleichungen, geeignet. Demgegenüber bietet der Digitalrechner die Möglichkeit, umfangreiche algebraische Operationen durchzuführen, eine grosse Zahl von Werten und Zwischenergebnissen zu speichern, komplexe Unterprogramme, bedingte Anweisungen, logische Entscheidungen und Iterationsverfahren zu programmieren.

2.3 Organisation des Rechenablaufes bei einem Hybrid-System

Während in 2.2 die Gesichtspunkte für eine Organisation der Aufgabenteilung bei einem Hybrid-System dargestellt wurden, handelt es sich hier um die zeitliche Organisation der Abstimmung der beiden Rechner aufeinander.

Unter diesem Gesichtspunkt lassen sich zwei Hauptanwendungsklassen einteilen, wobei es zweckmässig sein kann, die zweite Klasse nochmals zu unterteilen.

Tabelle : Einteilung nach Anwendungsklassen hybrider Rechensysteme unter dem Gesichtspunkt der Organisation des Rechenablaufs

Klasse		Merkmal	Funktion des Analogrechners	Funktion des Digitalrechners
A		alternierende Arbeitsweise beider Rechner, nicht zeitkritisch	Lösung der Zustandsgleichungen mit hoher Rechengeschwindigkeit	Hilfsalgorithmen und Speicherung
S	S_1	simultane Arbeitsweise beider Rechner	Integration und andere lineare Operationen, einfachere nichtlineare Operationen	kompliziertere nichtlineare Operationen und Funktionsbildung, Speicherung
	S_2	zeitkritisch	abgeschlossene Teilsysteme, deren Lösung hohe Rechengeschwindigkeit und niedrige Lösungsgenauigkeit erfordern	abgeschlossene Teilsysteme, deren Lösung niedrige Rechengeschwindigkeit und hohe Lösungsgenauigkeit erfordern

In die Klasse A - alternierende Arbeitsweise - fallen alle Anwendungsfälle, die im Sinne von Echtzeitbedingungen überhaupt nicht zeitkritisch sind, da jeder Rechner seine Teilaufgaben erst dann übernimmt, wenn der andere Rechner seine Aufgaben abgeschlossen hat.

Optimierungsaufgaben fallen i.a. in diese Klasse A. Die Grundaufgabe besteht darin, freie Parameter eines Systems so zu bestimmen, dass eine bestimmte Funktion in ein Optimum überführt wird. Dabei wird das dynamische System auf dem Analogrechner simuliert, dies bedeutet, Integration der Zustandsgleichungen über dem Intervall T. Auf dem Digitalrechner müssen dann die Suchstrategie programmiert sein, nach der bei der Parameteränderung verfahren wird, und Abfragen formuliert werden, ob das gewünschte Optimum genügend genau erreicht wurde.

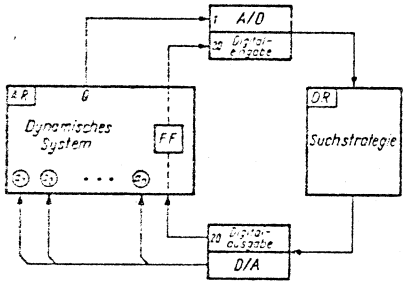


Bild 2.3 Blockschaltbild für eine automatische Optimierung

In die Klasse S - simultane Arbeitsweise - fallen alle Anwendungsfälle die zeitkritisch sind. Die Ergebnisse jedes Rechners müssen innerhalb bestimmter Antwortzeiten verfügbar sein, weil sie der andere Rechner zur Ausführung seiner Operationen benötigt und Zeitbedingungen einzuhalten sind. In die Unterklasse S_1 sind jene Anwendungsfälle eingereiht, bei denen Integrationen nur auf dem Analogrechner ausgeführt werden. Ein Beispiel dafür kann die Simulation von nichtlinearen Regelkreisen nach Bild 2.4 sein. Dabei wird der lineare Systemteil L, der durch ein lineares Differentialgleichungssystem beschreibbar ist, auf dem Analogrechner simuliert, der damit alle Integrationen ausführt. Der nichtlineare Regler NL dagegen wird gemäss den Ausführungen von 2.2 auf dem Digitalrechner simuliert.

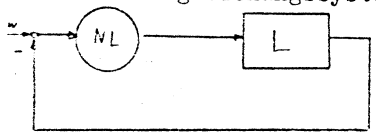


Bild 2.4: nichtlinearer Regelkreis

In der Klasse S 2 sind dann solche Fälle zu finden, bei der jeder Rechner abgeschlossene Teilsysteme einschliesslich Integrationen ausführt. Dies ist der eigentliche Fall hybrider Simulation. Dabei müssen wieder die Gesichtspunkte von 2.2 berücksichtigt werden.
Bild 2.5 zeigt als Beispiel das Blockschema der Simulation eines Führungssystems und die Aufteilung der Aufgaben auf die beiden Rechner.

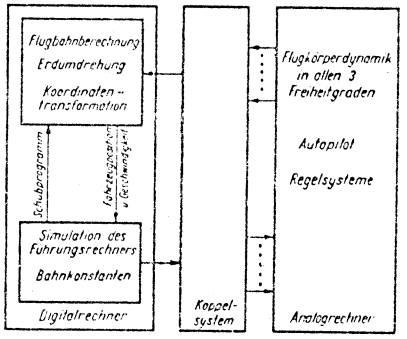


Bild 2.5: Simulation eines gelenkten Flugkörpers auf einem hybriden Rechnersystem

3.1 Das Potentiometer

Für die Multiplikation mit einer Konstanten K, für die $0 < K < 1$ gilt, verwendet man im allgemeinen Potentiometer. Ein Potentiometer ist ein Widerstand mit veränderlichem Abgriff. (Bild 3.1.0) Der Abgriff läßt sich sehr fein verstellen, und es besteht ein linearer Zusammenhang zwischen Schleiferstellung und abgegriffenen Widerstand. Als Symbol für ein Potentiometer verwendet man das in Abb. 3.1.1 dargestellte Zeichen.

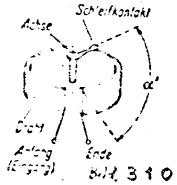


Bild 3.1.0

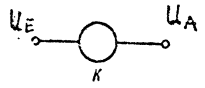


Bild 3.1.1
Symbol für ein Potentiometer

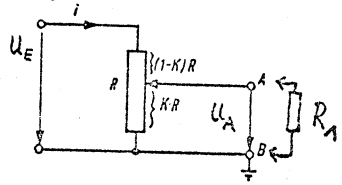


Bild 3.1.2
Unbelasteter Spannungsteiler

Potentiometer sind Spannungsteiler. Für Abb. 3.1.2 gilt:

$$U_A = U_E \cdot \frac{K \cdot R}{R} \quad \text{bzw.} \quad U_A = K \cdot U_E \qquad (3.1.1)$$

Die Eingangsspannung U_E wird an das obere Ende des Potentiometers gelegt und die Ausgangsspannung am Mittelabgriff, bei geerdetem unteren Potentiometerende, abgegriffen. Man verwendet bei Präzessions-Analogrechnern Wendelpotentiometer mit bis zu 20 Gängen und gut angenäherter Linearität, um eine Einstellgenauigkeit bis auf 3 - 4 Stellen genau zu erhalten. Der Widerstand liegt in der Größenordnung von 10-100 KΩ. Gl. (3.1.1) gilt jedoch nur für den Fall, daß keine Belastung vorliegt. In einer Analogrechenschaltung ist der Ausgang eines Potentiometers immer mit einem oder mehreren Eingängen anderer Rechenelemente verbunden, und es tritt eine Belastung R_1 des Potentiometers auf. Das Verhältnis der Ausgangs- zur Eingangsspannung ist nun nicht mehr proportional zu K. Es gilt vielmehr

$$\frac{U_A}{U_E} = K \cdot \frac{1}{1+K(1-K)\frac{R}{R_1}} \qquad (3.1.2)$$

d.h. die Skala zeigt im Belastungsfall nur den ungefähren Wert des tatsächlichen Koeffizienten an.

Da $\frac{U_A}{U_E} = K$ in sollte,

definieren wir den Fehler F zu

$$F = K - \frac{U_A}{U_E} = K - \frac{K}{1+K(1-K)\frac{R}{R_1}} \qquad (3.1.3)$$

Aus Gl. (3.1.3) entnehmen wir, daß der Fehler nur für den Fall

$$K(1+K)\frac{R}{R_1} \ll 1$$

zu vernachlässigen ist. (Bild 3.1.3) Der Belastungsfehler ist also sowohl von der Potentiometerstellung, als auch vom Belastungswiderstand abhängig. In der Praxis haben sich im wesentlichen zwei Verfahren durchgesetzt, um den Einstellfehler zu eleminieren.

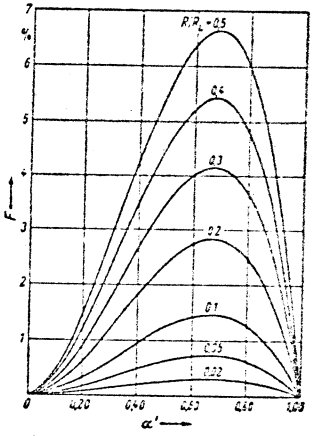
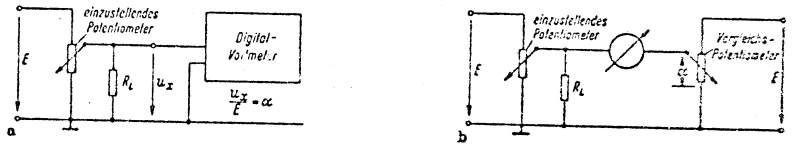


Abb. 3.1.3 Belastungsfehler eines Potentiometers als Funktion des Drehwinkels

1. Methode: Man mißt die Ausgangsspannung eines belasteten Potentiometers und schaltet zum Einstellen den Eingang auf eine Bezugsspannung (1 ME). Ein Digitalvoltmeter, das auf die Bezugsspannung normiert ist, zeigt dann direkt den Koeffizienten K an.



Einstellung eines Potentiometers
a) Mit dem Digitalvoltmeter, b) mit Nullinstrument und Vergleichspotentiometer

2. Methode: Man mißt das Teilerverhältnis des belasteten Potentiometers durch Vergleich mit einem Vergleichspotentiometer. Dabei wird der gewünschte Koeffizient am Vergleichspotentiometer eingestellt, und der Schleifer desselben mit dem Schleifer des einzu-

stellenden Potentiometers durch ein Nullinstrument verbunden. Jetzt wird der Schleifer des belasteten Potentiometers so verstellt, daß das Nullinstrument keinen Ausgleichsstrom mehr anzeigt und damit beide Schleiferpotentiale gleich sind.

3.2 Der Operationsverstärker

Der wichtigste Baustein des elektronischen Analogrechners ist der Operationsverstärker. Man versteht darunter einen speziellen Gleichstromverstärker, der sehr weitgehende Forderungen erfüllen muß:

1. Der Betrag der Verstärkung soll theoretisch unendlich groß sein.
2. Der Phasenwinkel zwischen Ein- und Ausgang soll für alle Frequenzen genau 180° betragen.
3. Der Strom am Eingang des Verstärkers soll Null sein.
4. Im Verstärker dürfen keine unkontrollierbaren Spannungsschwankungen und Störspannungen (Rauschen, Drift) auftreten.

Diese Forderungen lassen sich aus physikalischen Gründen nicht streng erfüllen.

zu 1. Der Betrag der Verstärkung erreicht in praktischen Rechenverstärkern (bis ca. 100 Hz) $V = 10^{8} - 10^{12}$

zu 2. Sehr kleiner Phasenfehler bis ca. 5 kHz (ca. 0.1° bei 100 Hz), wichtig wegen Schwingneigung.

zu 3. 10^{-10}A bzw. noch kleiner durch spezielle Schaltmaßnahmen (Chopperverstärker).

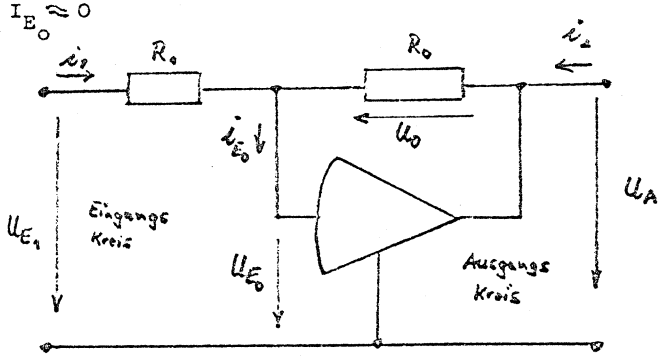
zu 4. Größenordnung 0.5 m V/h besonders Drift sehr unangenehm bei langen Rechenzeiten.

Die Verstärkung (V) eines Verstärkers ist definiert, als das Verhältnis der Ausgangsspannung U_A zur Eingangsspannung U_{E_o}.
Es gilt für

$$U_{E_o} = -\frac{U_A}{V} \qquad (3.2.1)$$

und damit für $V \to \infty$, $U_{E_o} \to 0$

Nach Forderung 3 an den Operationsverstärker ist $I_{E_o} \approx 0$



Die wichtigste Aufgabe des Operationsverstärkers ist es zu verhindern, daß Rückwirkungen zwischen dem Ausg. skreis und dem Eingangskreis entstehen.
Für den Eingangskreis gilt, daß die Abhängigkeit des Stromes i_1 von der Spannung U_{E_1} lediglich durch das Element R_1 der äußeren Beschaltung des Operationsverstärkers bestimmt wird.
Für den Ausgangskreis gilt mit $U_{E_o} = 0$

$$U_o = U_A \qquad (3.2.2)$$

und damit hängt die Beziehung zwischen U_o und i_2 wiederum von der Art der Beschaltung R_o ab.
Die einzige Kopplung zwischen Ein- und Ausgangskreis stellt die Beziehung

$$i_1 = -i_2 \qquad (3.2.3)$$

dar. Darüberhinaus besteht im Idealfall keine Rückwirkung.
Für den Widerstand im Eingangskreis wählt man einen linearen ohmschen Widerstand, sodaß im Eingangskreis Strom und Spannung proportional sind. Den 2. Widerstand wählt man so, daß sich die gewünschte Beziehung zwischen Eingangs- und Ausgangsspannung ergibt.
In dieser prinzipiellen Anordnung werden wir dem Operationsverstärker immer wieder begegnen, sei es als Summierer, Integrierer, Umkehrer oder in einem Funktionsgeber.

Die Multiplikation mit einem konstanten Faktor kann man auch durch einen beschalteten Rechenverstärker, anstelle eines Potentiometers, realisieren. Wir betrachten im folgenden diesen Fall mit einem realen Verstärker, d.h. $V \neq \infty$, damit $U_{E_o} \ll 1$ aber nicht 0, während wir die Idealisierung $i_{E_o} = 0$ beibehalten, da $i_{E_o} \ll i_1, i_2$ ist.

Aus Gl. (3.2.3) erhalten wir

$$\frac{U_{E_1} - U_{E_o}}{R_1} = \frac{U_{E_o} - U_A}{R_o} \qquad (3.2.4)$$

und mit Gl. (3.2.1)

$$\frac{U_{E_1} + \frac{U_A}{V}}{R_1} = \frac{-\frac{U_A}{V} - U_A}{R_o} \qquad (3.2.5)$$

durch Umformung

$$\frac{U_{E_1}}{U_A} = -\frac{R_1}{R_o} \cdot \left[1 + \frac{1}{V}\left(1 + \frac{R_o}{R_1}\right)\right] \qquad (3.2.6)$$

Unter der Voraussetzung $V \to \infty$ ergibt sich ein Übertragungsfaktor K zu

$$\frac{U_A}{U_E} = \frac{R_o}{R_1} = K \qquad (3.2.7)$$

der je nach Wahl der Widerstände R_1, R_o kleiner, gleich oder größer 1 sein kann

$$0 < K < \infty$$

Mit Gl. (3.2.7) haben wir bis auf die Möglichkeit, daß $K > 1$ werden kann, den Fall des unbelasteten Potentiometers. Zum Unterschied dazu können wir die Ausgangsspannung U_A jedoch beliebig (im zulässigen Leistungsbereich des Operationsverstärkers) belasten, ohne daß sich der Faktor ändert.

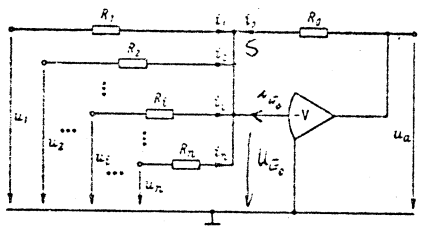
Der Fehler F beträgt

$$F = \frac{1}{V}(1 + K) \qquad (3.2.8)$$

Für einen Beispielfall von K = 10 und einen zugelassenen Fehler von 10^{-4} ergibt sich die zu fordernde Minimalverstärkung $V_{min} = 1{,}1 \cdot 10^5$

3.3 Umkehrer, Summierer

Werden die beiden Widerstände R_o, R_1 in Gl. 3.2.7 gleich groß gewählt, so daß K = 1 ist, erhält man eine Multiplikation mit -1. Es wird also lediglich das Vorzeichen umgekehrt (invertiert - daher der Name Umkehrer, Inverter). Da jedes aktive Rechenelement des Analogrechners neben seiner eigentlichen Aufgabe das Vorzeichen umkehrt, wird der Inverteroft dazu benutzt, diese manchmal unerwünschte Vorzeichenumkehr zu korrigieren. Erweitert man den Eingang durch weitere Eingänge mit den Widerständen $R_2 \ldots R_n$, erhält man einen Summierer.



Prinzipschaltung des Summierers

Wendet man die Kirchhoff'sche Knotenregel auf den Punkt S an (mit $i_{e_o} = 0$)

$$i_1 + i_2 + i_i \ldots i_n + i_o = 0 \qquad (3.3.1)$$

und das Ohmsche Gesetz für die Eingangsströme (mit $U_{E_o} = 0$)

$$i_1 = \frac{U_1}{R_1} \; ; \; i_2 = \frac{U_2}{R_2} \; ; \; i_n = \frac{U_n}{R_n} \qquad (3.3.2)$$

sowie für

$$i_o = \frac{U_A}{R_o} \qquad (3.3.3)$$

Durch Einsetzen der Beziehungen (3.3.2) und (3.3.3) in (3.3.1)

$$U_A = -\left(\frac{R_o}{R_1} U_1 + \frac{R_o}{R_2} U_2 + \ldots \frac{R_o}{R_n} U_n\right) \qquad (3.3.4)$$

bzw. abgekürzt

$$U_a = \sum_i C_i U_i \text{ mit } C_i = \frac{R_o}{R_i} \qquad (3.3.5)$$

Die Schaltung summiert also verschiedene Eingangsvariablen U_i mit den Bewertungsfaktoren C_i und bildet nachfolgend die Summe über die so bewerteten Eingangsgrößen.

3.4 Der Integrierer

Wird die Rückführung R_o durch einen Kondensator C_o ersetzt, erhalten wir einen Integrierer. Die Gl. für den Eingangskreis (3.3.1) und (3.3.2) bleiben unverändert. Für den Ausgangskreis gilt jetzt statt (3.3.3)

$$i_o = C_o \frac{du_c}{dt} \qquad (3.4.1)$$

(Das Ladungsgesetz, das aus der Tatsache folgt, daß die gespeicherte Ladungsmenge in einem Kondensator gleich dem Integral seines Zuflußes sein muß).
Mit (3.3.1), (3.3.2) und (3.4.1)

$$\frac{du_c}{dt} = -\frac{1}{R_1 C_o} U_1 - \frac{1}{R_2 C_o} U_2 \cdots - \frac{1}{R_u C_o} U_n \qquad (3.4.2)$$

und da $U_a = U_c$ (mit $U_{E_o} = 0$) ist, erhalten wir nach Integration

$$U_a = -\left[\frac{1}{R_1 C_o} \int U_a \, dt + \frac{1}{R_2 C_o} \int U_2 \, dt \ldots + \frac{1}{R_n C_o} \int U_n \, dt\right]$$

bzw. kürzer mit $\frac{1}{R_i C} = K_i$

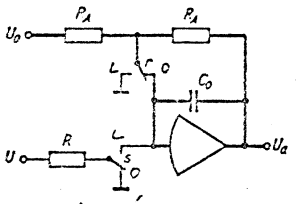
$$U_a = -\sum_i K_i \int U_i \, dt \qquad (3.4.3)$$

K_i ist eine Konstante, die angibt mit welcher Geschwindigkeit die Integration verläuft (Dimension S^{-1}).

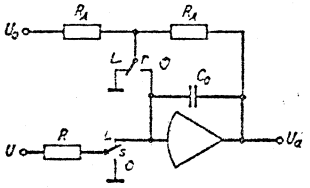
Der Integrierer vereinigt in sich die Rechenfunktion des Summierens und des Integrierens. Durch eine Umschaltvorrichtung wird dafür gesorgt, daß eine Integration mit Anfangswerten (Anfangsladung des Kondensators C_o) möglich ist.

	R	S	
Rechnen	L	L	b)
Pause	0	0	a)
Halt	L	0	c)

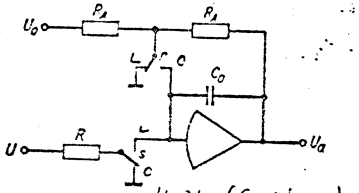
Betriebszustände des Integrierers



a) Pause



b) Rechnen



c) Halt (Speichern)

Diese Anfangsbedingungen U_o werden vor Beginn jeder Rechnung auf die entsprechenden Integrierer geschaltet. Dies geschieht in der Rechenphase Pause, die als erste Phase in einem Rechenzyklus durchlaufen wird. Der Integrierer wird durch die Schalter r,s, wie ein Umkehrer mit Eingangsspannung U_o und Ausgangsspannung U_a geschaltet, wobei der Ausgang mit der Kapazität C_o belastet ist (Bild a). Für die Kondensatorspannung U_a ergibt sich

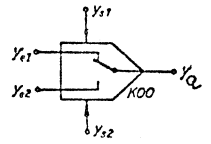
$$U_a = U_o \left(1 - e^{-\frac{t}{R_a \cdot C_o}} \right) \quad (3.4.4).$$

Entsprechend Gl. (3.4.4) erreicht die Kondensatorspannung U_a nur für $t \longrightarrow \infty$ den exakten Anfangswert U_o.

Im allgemeinen begnügt man sich mit einer Näherung für die Pausenzeit $T = R_a \cdot C_o$. In der Haltphase besitzt der Integrierer eine Analogwertspeicherfunktion, da am Eingung nur der Integrierkondensator angeschaltet ist, der durch die Rückführung dafür sorgt, daß eine Änderung am Ausgang durch eine Änderung am Eingang kompensiert wird.

3.5 Komparator

Komparatoren dienen der Ausführung logischer Entscheidungen in Abhängigkeit von den Rechenspannungen und besitzen deshalb sowohl analogen als auch digitalen Carakter. Sie arbeiten nach dem Prinzip eines gesteuerten Schalters. In Abhängigkeit einer Steuerspannung (meist der Summe od. Differenz zweier Steuerspannungen) ist am Ausgang eine von zwei weiteren Eingangsspannungen verfügbar.



$$Y_a = \begin{cases} Y_{e_1} & Y_{s_1} \geq Y_{s_2} \\ \text{für} & \\ Y_{e_2} & Y_{s_1} \leq Y_{s_2} \end{cases}$$

In der Regel werden sie durcheinen offenen Operationsverstärker, an dessen Ausgang ein Relais angeschlossen ist, realisiert. Damit der Operationsverstärker nicht übersteuert wird, befinden sich im Rückführungszweig Begrenzerelemente (z.B. Zenerdioden) für beide Polaritäten.

3.6 Funktionsgeber

Funktionsgeber dienen dazu eine bestimmte (nichtlineare) Funktion einer willkürlichen Variablen zu bilden. D.h. ein Funktionsgeber gestattet einen in weiten Grenzen beliebigen Zusammenhang zwischen Eingangs- und Ausgangsgröße herzustellen.

$$Y_a\ (t) = f\left[Y_E\ (t)\right]$$

Zur technischen Realisierung ist man die verschiedensten Wege gegangen z.B.: Diodenstrecken, angezapfte bzw. besonders geformte Potentiometer, Kathodenstrahlrohren mit Funktionsmasken, optische und magnetische Kurvenabtastung.
Als Beispiel seien hir nur Diodenfunktionsgeber erwähnt, da bei neueren Hybridrechnern die Funktionen digital erzeugt werden. In den Diodenfunktionsgebern werden die Funktionen mittels Geradenabschnitten durch interwallweise Linearisierung approximiert (Bild 3.6.0).

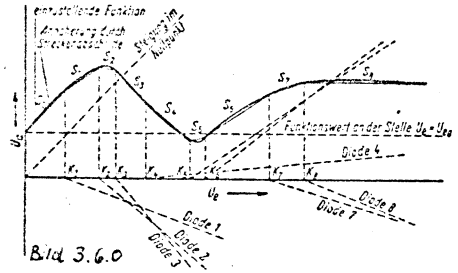


Bild 3.6.0
Arbeitsweise des Diodenfunktionsgebers

Ob dabei die Intervallbreite fest vorgegeben oder variabel ist, hängt vom jeweiligen Gerätetyp ab. Zur Funktionserzeugung werden die Schalter bzw. Ventileigenschaften von Dioden ausgenutzt. Ideale Dioden weisen für negative Spannungen einen unendlichen Widerstand auf und für positive Spannungen den Durchlaßwiderstand

$$R_d = 0.$$

Bei technischen Dioden werden diese Eigenschaften nur angenähert erfüllt (Bild 3.6.1).

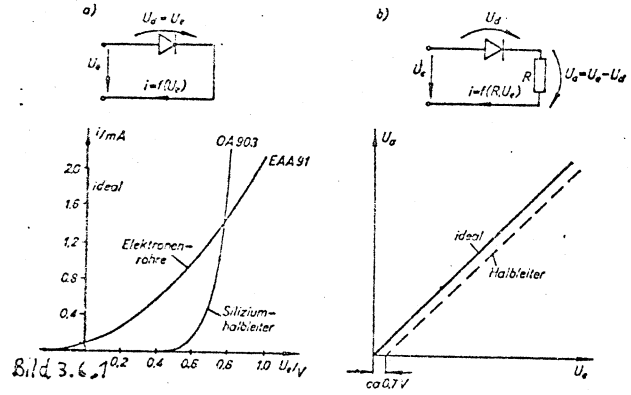


Bild 3.6.1

Wird eine Diode an den Spannungsteiler P_1 (Bild 3.6.2) geschaltet, an dessen beiden Enden die Eingangsspannung U_e und eine konst. Vorspannung $-U_v$ angeschaltet ist, so ist ihr Knickpunkt durch Veränderung von P_1 verschiebbar. Die Steigung des Geradenabschnittes wird schließlich mit dem Spannungsverteiler P_2 eingestellt. Durch Umpolen der Diode (Schalterstellung A,B) wird der Quadrant bestimmt Verwendet man einen solchen Zweig ohne Diode für die Nullpunktsgerade und mehrere Zweige mit Dioden für weitere Geradeabschnitte als Eingangswerk eines Operationsverstärkers (Bild 3.6.3) mit ohmscher Rückkopplung, so werden Teilströme wie beim Summierer aufsummiert. Man erhält so den gewünschten Polygonzug.

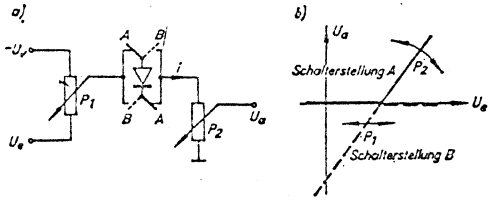


Bild 3.6.2 Ideale Diode mit Spannungsteiler P_1 für Knickpunkt- und P_2 für Anstiegseinstellung sowie mit Schalter für den Quadranten
a) Schaltung; b) Kennlinien

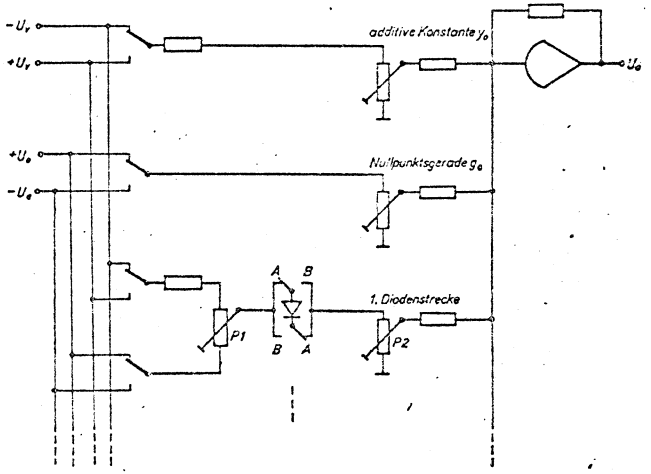


Bild 3.6.3 Prinzipschaltung eines Funktionsgenerators vom Summierertyp

3.7 Multiplizierer

Die technische Realisierung bereitet bei vernünftigem Aufwand noch heute Schwierigkeiten, wenn die Genauigkeit mit der von linearen Rechenelementen vergleichbar sein soll. Es wurden die verschiedensten Verfahren entwickelt, die man grob in zwei Klassen einteilen kann:

1.) direkte Multiplizierverfahren, die bestimmte technische Prinzipien oder physikalische Effekte ausnutzen. Beispiele dafür sind steuerbare Widerstände (Servomultiplizierer) oder Mudulationsmultiplizierer, die die Technik der Mudulation benutzen.

2.) indirekte Verfahren, bei denen das Produkt durch mathematisch begründete Umformungen mit speziellen Funktionsgeneratoren gebildet wird.
Als Beispiel sei hier die Vierte lquadratmethode angeführt, die die Beziehung

$$Y = X_1 \cdot X_2 = \frac{1}{4}(X_1 + X_2)^2 - (X_1 - X_2)^2 \qquad (3.7.1)$$

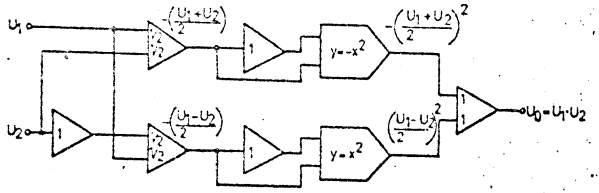
benutzt. Auf dieses, bei Parabelmultiplizierern benutzte Verfahren, soll im Folgenden näher eingegangen werden.

Gl. 3.7.1 kann man noch vereinfachen zu

$$U_1 \cdot U_2 = \left(\frac{U_1 + U_2}{2}\right)^2 - \left(\frac{U_1 - U_2}{2}\right)^2$$

In der einfachsten Form werden zunächst die Ausdrücke $(U_1 + U_2)/2$ und $(U_1 - U_2)/2$ mit Hilfe von Summierern und Umkehrern gebildet. Diese gelangen anschließend getrennt an zwei Funktionserzeuger, die die vorzeichenrichtigen Quadrate der Ausdrücke bilden (Bild 3.7.1). Für die Bereitstellung von $(U_1 + (\pm U_2))/2$ nutzt man die Tatsache aus, daß in den Schaltungen (Bild 3.7.2) sich der gleiche Strom i und die gleiche Spannung U_E einstellen.

Bild 3.7.1



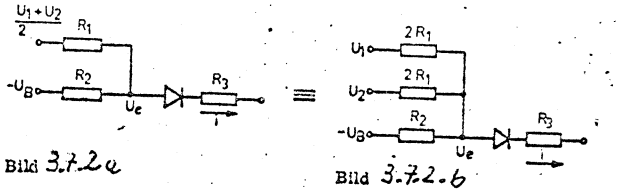


Bild 3.7.2a Bild 3.7.2b

Dieser Spannungsteiler wird von jeder Diodenstrecke des Quadratbildners gesondert verwendet, so daß sich Bild 3.7.3 als Blockschaldbild für einen kompletten Multiplizierer ergibt.

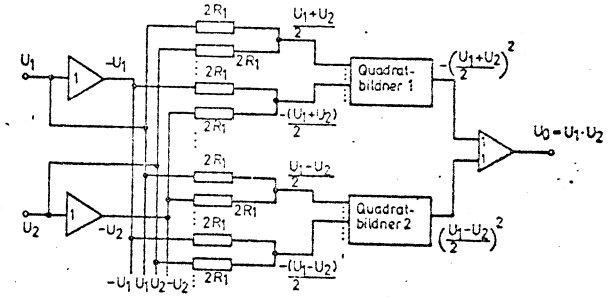
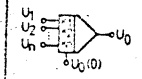
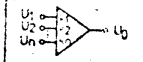
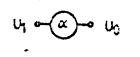
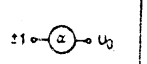
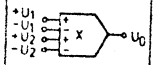
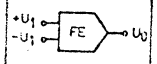
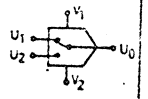


Bild 3.7.3

Bei den meisten serienmäßig hergestellten Analogrechnern fehlen die beiden Inverter zur zur Vorzeichenumkehr von U_1 und U_2, so daß beide Multiplikanden mit beiden Vorzeichen als Eingangsgrößen des Multiplizieres vorhanden sein müssen.

Tabelle der Rechenelemente

Recheneinheit	Programmier-symbol	Ausgangsspannung	Bemerkungen
Integrator		$U_o = -\left(\sum_{j=1}^{n} k_j \int_0^t U_j dt + U_o(0)\right)$	$k_j = \frac{1}{R_j C}$
Summator		$U_o = -\sum_{j=1}^{n} k_j U_j$	$k_j = \frac{R_o}{R_j}$
Inverter		$U_o = -U_1$	$R_o = R_1$
Potentio-meter		$U_o = \alpha U_1$	$0 \le \alpha \le 1$
Festwert-geber		$U_o = \alpha U_1$	$0 \le \alpha \le 1$ $U_1 = +1$ oder $U_1 = -1$
Multiplika-tor		$U_o = U_1 \cdot U_2$	Eingangs-variable: $\pm U_1$, $\pm U_2$
Funktions-erzeuger		$U_o = f(U_1)$	Eingangs-variable: $\pm U_1$
Komparator		$U_o = U_1$, falls $V_1 > V_2$ $U_o = U_2$, falls $V_1 < V_2$	auch: $U_o = U_1$ falls $V_1 + V_2 > 0$ $U_o = U_2$ falls $V_1 + V_2 < 0$

§ 4 Zusammmenfügen der analogen Rechenelemente zur Rechenschaltung

4.1 Lösung einer gewöhnlichen Differentialgleichung n-ter Ordnung

Im vorigen Abschnitt wurden die einzelnen Rechenelemente vorgeführt, die auf einem Analogrechner i.a. zur Verfügung stehen. Soll nun ein gegebenes mathematisches System gelöst werden, dann müssen die Rechenelemente zu einer Rechenschaltung zusammengefügt werden, die unter bestimmten Voraussetzungen das reale Modell des analytisch beschriebenen physikalischen Systems ist. (Siehe § 1) Da der Analogrechner hauptsächlich bei der Berechnung bzw. Untersuchung von dynamischen Systemen eingesetzt wird, sei hier nur das grundsätzliche Vorgehen beim Programmieren eines Analogrechners anhand von Differentialgleichungen erläutert. Hierfür wurde zunächst eine lineare Differentialgleichung beliebiger Ordnung mit konstanten Koeffizienten

$$x^{(n)} + a_{n-1}x^{(n-1)} + \ldots + a_m x^{(m)} + \ldots + a_1 x' + a_o x + b = 0 \qquad (4.1)$$

mit b=const und positiven Anfangsbedingungen

ausgewählt, um bezüglich der Ordnung der Gleichung die Aufgabenstellung allgemein zu halten.
Diese Gleichung löst man zunächst nach dem Glied mit der höchsten Ableitung auf und erhält so die Gleichung

$$x^{(n)} = -a_{n-1}x^{(n-1)} - \ldots - a_1 x' - a_o x - b \qquad (4.2)$$

Nimmt man nun an, am Eingang eines Integrators sei die Funktion $x^{(n)}(t)$ vorhanden, dann ist eine n-malige Integration bis zur Funktion x(t) nötig. Dabei erscheinen an den einzelnen Ausgängen der Integrierverstärker nacheinander jeweils auch die (n-1) Ableitungen der gesuchten Funktion $x'(t)$ bis $x^{(n-1)}(t)$, aus denen sich nach obiger Beziehung die n-te Ableitung von x(t) zusammensetzt. Die einzelnen Ableitungen müssen also nur noch innerhalb der Integrationskette abgegriffen, mittels eines Koeffizientenpotentiometers mit den jeweiligen Konstanten a_o, a_1, a_2, ... multipliziert und an den Eingang des ersten Integrators zurückgeführt werden, wo zu Beginn die n-te Ableitung angenommen wurde (Bild 4.1).

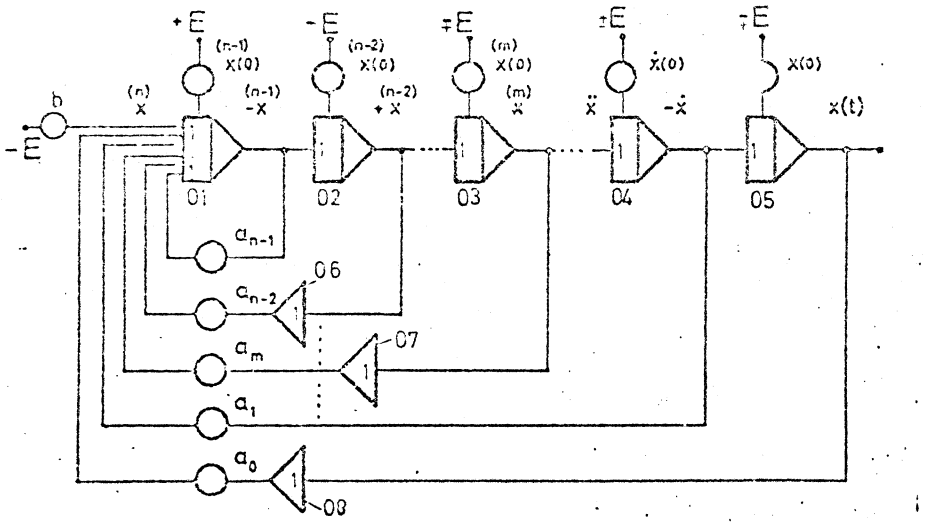


Bild 4.1

Rechenschaltung zur Lösung einer gewöhnlichen linearen Differentialgleichung n-ter Ordnung (E: maximale Maschinenspannung)

Hierbei ist darauf zu achten, daß bei der Integration jeweils auch eine Vorzeichenumkehr erfolgt; d.h., daß in der Rückführung zur Vorzeichenkorrektur bisweilen eine neue Umkehrung stattfinden muß (Umkehrverstärker 06, 07, 08). Die Konstante b wird in Form einer konstanten Spannung auch an den Eingang von Verstärker 01 angelegt.

Die Lösung einer gewöhnlichen Differentialgleichung n-ter Ordnung erfordert n Anfangsbedingungen. Diese werden in Form von entsprechenden Spannungen an die Integratoren angelegt. Somit ist ein fertiges Rechenschaltbild entstanden. Die gesuchte Funktion wird am Ausgang des Verstärkers 05 abgegriffen.

Beim Entwerfen des Rechenschaltbildes ging man davon aus, daß es sich bei der vorliegenden Differentialgleichung bereits um eine normierte Differentialgleichung handelte. Was hierunter zu verstehen ist, wird im Abschnitt 5 erläutert. Wäre die vorliegende Differentialgleichung noch nicht normiert, dann wäre das entworfene Schaltbild lediglich als Prinzipschaltbild zu werten, das zwar sämtliche für die Rechnung nötigen Rechenelemente in der richtigen Anordnung enthielte, deren Potentiometereinstellungen aber noch entsprechend der Normierung zu ermitteln wären. Dies gilt auch für die nun folgenden Beispiele.

Das Verfahr soll an 2 Beispielen vorgeführt werden. Zu lösen sei die gewöhnliche lineare Differentialgleichung 2. Ordnung mit linearen Koeffizienten.

$$\frac{d^2x}{dt^2} + \omega^2 x = 0 \qquad (4.3)$$

mit den Anfangsbedingungen

$$x(0) = \hat{x} \; ; \quad \dot{x}(0) = 0 \quad .$$

Nach der höchsten Ableitung aufgelöst, ergibt sich die Differentialgleichung in der Form

$$\ddot{x} = -\omega^2 x \quad . \qquad (4.4)$$

Hieraus läßt sich das Prinzipschaltbild nach der bereits erläuterten Methode entwickeln (Bild 4.2).

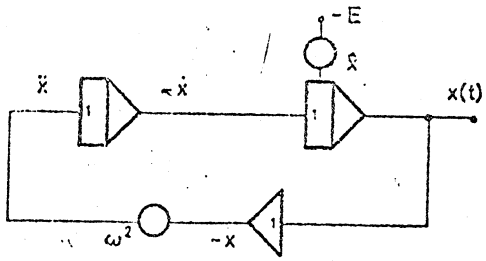


Bild 4.2
Rechenschaltbild einer linearen, homogenen Differentialgleichung 2. Ordnung

Durch zweimalige Integration von $\ddot{x}(t)$ erhält man $x(t)$. $\ddot{x}$ ist nach obiger Beziehung aber $-\omega^2 x$. Deshalb wird $x(t)$ abgegriffen, im Vorzeichen umgekehrt und mit Hilfe eines Potentiometers mit ω^2 multipliziert und an den Eingang des ersten Verstärkers zurückgeführt. Jetzt werden nur noch die Anfangsbedingungen $\dot{x}(0) = 0$ und $x(0) = \hat{x}$ in Form von analogen Spannungen an die jeweiligen Verstärker angelegt. $\dot{x}(0) = 0$ bedeutet, daß die Anfangsspannung gleich Null wird, d.h. wegzulassen ist.

Die Differentialgleichung hat die Lösung:

$$x = \hat{x} \cdot \cos\omega t$$

Wählt man andere Anfangsbedingungen, wie z.B. $x(0) = 0$ und $\dot{x}(0) = a$, dann ergibt sich als Lösung eine Sinusschwingung. Deshalb wird diese Schaltung am Analogrechner häufig als Sinusgeber verwendet.

Für das Problem einer gedämpften Schwingung soll ein Rechenschaltbild zur Lösung der Differentialgleichung entworfen werden. Die Differentialgleichung lautet für diesen Fall

$$\ddot{x} + a_1\dot{x} + a_o x = f(t) \qquad (4.5)$$

mit den Randbedingungen $x(0) = 0$; $\dot{x}(0) = s_o$.

Die Auflösung nach der höchsten Ableitung liefert

$$x = f(t) - a_1\dot{x} - a_o x \quad . \qquad (4.6)$$

Daraus ergibt sich die im Bild 4.3 dargestellte Rechenschaltung.

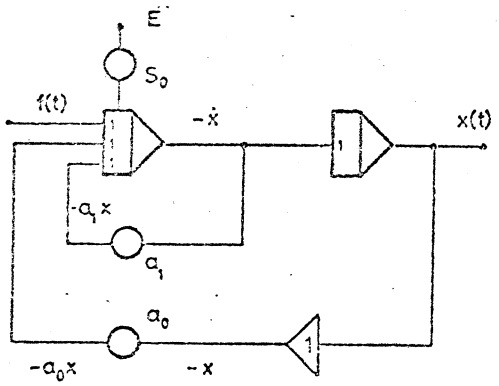


Bild 4.3
Rechenschaltbild einer linearen, inhomogenen Differentialgleichung 2. Ordnung

4.2 Lösung von linearen Differentialgleichungssystemen mit konstanten Koeffizienten

Zahlreiche Regelungssysteme, mechanische Schwingungssysteme oder elektrische Netzwerke stellen typische Beispiele für miteinander gekoppelte einfachere Übertragungssysteme dar. Die mathematische Behandlung solcher Übertragungssysteme führt im allgemeinen auf die Lösung eines Systems simultaner Differantialgleichungen 1. Ordnung.

Auch andere technische Probleme, die zu einer Differentialgleichung höherer Ordnung führen, lassen sich wiederum in solche Systeme von Differentialgleichungen 1. Ordnung aufspalten.

Die Normalform für Differentialgleichungssysteme mit konstanten Koeffizienten lautet:

$$\begin{aligned}
\frac{dx_1}{dt} &= a_{11}x_1 + a_{12}x_2 + \ldots\ldots + a_{1n}x_n + b_1f_1(t)\\
\frac{dx_2}{dt} &= a_{21}x_1 + a_{22}x_2 + \ldots\ldots + a_{2n}x_n + b_2f_2(t)\\
&\vdots\\
\frac{dx_n}{dt} &= a_{n1}x_1 + a_{n2}x_2 + \ldots\ldots + a_{nn}x_n + b_nf_n(t)
\end{aligned} \qquad (4.7)$$

Bild 4.4 zeigt die dazugehörige Rechenschaltung

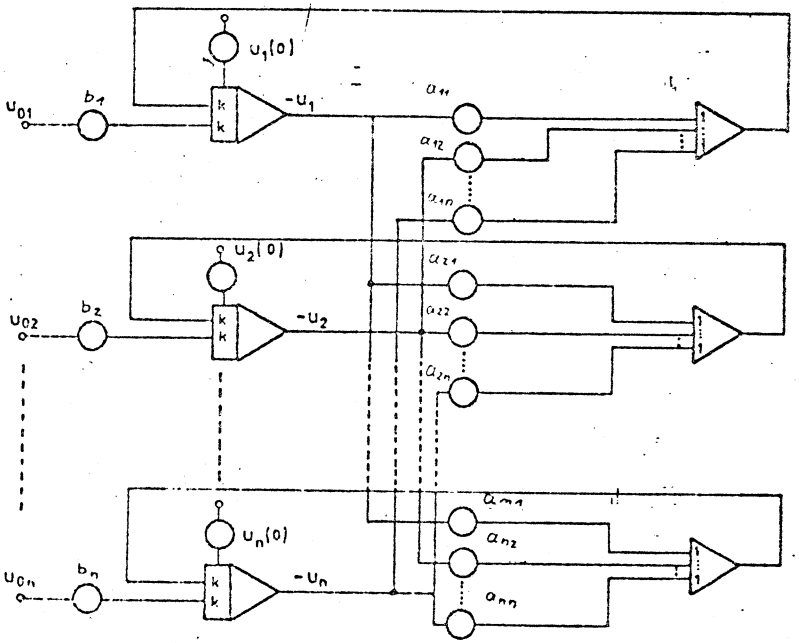


Bild 4.4 Rechenschaltung zur Lösung von Gl. (4.7)

Für den homogenen Fall - $f_i(t)=0$ - kann die Gleichung (4.7) auch in Form einer Koeffizientenmatrix angeschrieben werden.

$$\begin{pmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & & \\ a_{n1} & \cdots & a_{nn} \end{pmatrix} \qquad (4.8)$$

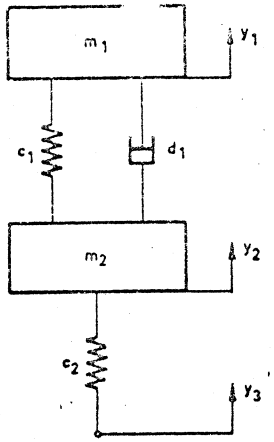
Wird die Differentialgleichung n-ter Ordnung (4.1) in ein Differentialgleichungssystem n-ten Grades durch die Substitution

$$\begin{aligned} x_1 &= x \\ &\vdots \\ x_i &= x^{(i-1)} \\ &\vdots \\ x_n &= x^{(n-1)} \end{aligned} \qquad (4.9)$$

dann hat die Koeffizientenmatrix nach (4.8) folgende Gestalt

$$\begin{pmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & & & & \\ 0 & 0 & 0 & \cdots & 1 \\ -a_0 & -a_1 & \cdots & & -a_{n-1} \end{pmatrix} \qquad (4.10)$$

Als Beispiel sollen bei einem Zwei-Massensystem nach Bild (4.5) die Auslenkungen $y_1(t)$ und $y_2(t)$ bei sprungförmiger Störung $y_3(t)$ ermittelt werden. Für die Dämpfer- und Federkennlinien gelten lineare Zusammenhänge.



m_1, m_2 : Massen
c_1, c_2 : Federkonstanten
d_1 : Dämpfungskonstante

Bild 4.5: Zwei-Massen-System

Als Ergebnis erhält man ein gekoppeltes Differentialgleichungssystem.

$$\frac{d^2y_1}{dt^2} + \frac{d_1}{m_1}\left(\frac{dy_1}{dt} - \frac{dy_2}{dt}\right) + \frac{c_1}{m_1}(y_1 - y_2) = 0$$

$$\frac{d^2y_2}{dt^2} + \frac{d_1}{m_2}\left(\frac{dy_2}{dt} - \frac{dy_1}{dt}\right) + \frac{c_1}{m_2}(y_2 - y_1) + \frac{c_2}{m_2}(y_2 - y_3) = 0 \qquad (4.11)$$

Nach der Substitution

$$\begin{aligned} x_1 &= y_1 \quad \text{und} \quad x_3 = y_2 \\ x_2 &= y_1' \qquad\quad x_4 = y_2' \end{aligned} \qquad (4.12)$$

enthält das neue Differentialgleichungssystem als höchste Ableitung nur noch die Ableitung x_i'.

Es gilt:

$$
\begin{aligned}
x_1' &= x_2 \\
x_2' &= -a_{21}x_1 - a_{22}x_2 + a_{23}x_3 + a_{24}x_4 \\
x_3' &= x_4 \\
x_4' &= a_{41}x_1 + a_{42}x_2 - a_{43}x_3 - a_{44}x_4 - b_4 \cdot f_4(t)
\end{aligned}
\qquad (4.13)
$$

mit $a_{ij} > 0$

Setzt man für alle Anfangswerte den Wert 0 ein, dann ergibt sich für die Rechenschaltung die Struktur vom Bild 4.6.

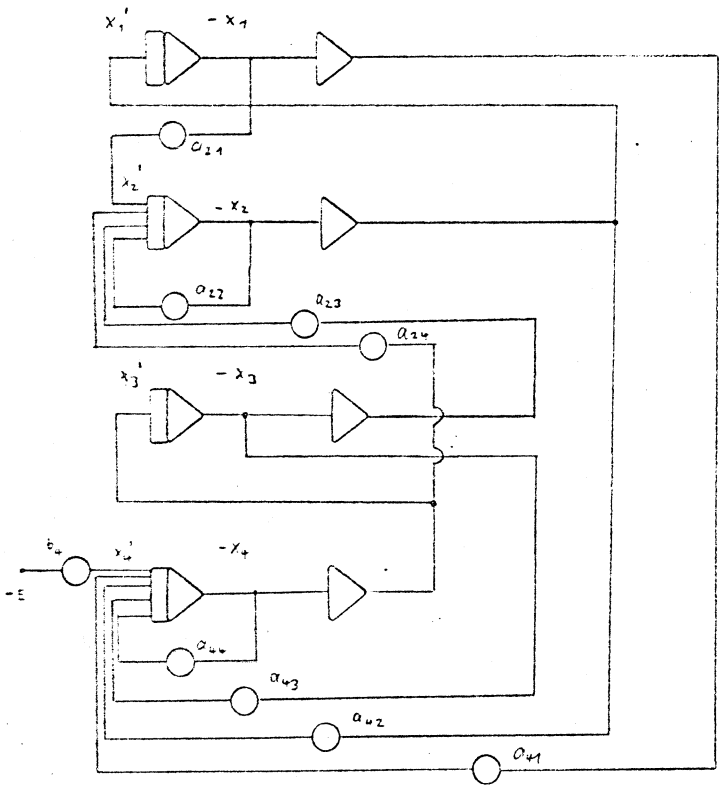


Bild 4.6: Rechenschaltung zur Lösung von Gleichung (4.13)

4.3 Nachbildung allgemeiner Übertragungssysteme

Bei vielen technischen Problemen wird ein physikalisches System nicht durch ein Differentialgleichungssystem beschrieben, sondern man verwendet die Übertragungsfunktion bzw. stilisierte Blockschaltbilder. Sowohl die Beschreibung durch ein Differentialgleichungssystem als auch diejenige durch die Übertragungsfunktion hängen eng miteinander zusammen und beinhalten dieselbe Information über das System. Jedoch ist die Beschreibung durch ein Differentialgleichungssystem allgemeiner, da eine Übertragungsfunktion nur für lineare Systeme aufgestellt werden kann, und außerdem noch alle Anfangsbedingungen des Systems gleich null sein müssen. Der Zusammenhang zwischen Übertragungsfunktion und Differentialgleichung ist über die Laplace-Transformation gegeben, mit deren Hilfe man ähnlich dem Logarithmenrechnen die Ordnung von Rechenoperationen an Zahlen um eine Stufe erniedrigen kann. Differentialgleichungen können in algebraische Gleichungen überführt werden, die i.a. leichter zu lösen sind.
Grundlage ist das Laplace-Integral

$$F(s) = \int_0^\infty e^{-st} f(t)\,dt \equiv \mathfrak{L}\{f(t)\} \qquad (4.14)$$

mit dem komplexen Parameter

$$s = \delta + j\omega \qquad (4.15)$$

Hierbei soll $f(t)$ eine Einschaltfunktion sein mit

$$f(t) \equiv 0 \text{ für } t<0 \qquad (4.16)$$

Zu (4.14) gibt es die Rücktransformation

$$f(t) = \frac{1}{2\pi j}\int_{\delta-j\infty}^{\delta+j\infty} e^{st} F(s)\,ds \equiv \mathfrak{L}^{-1}\{F(s)\}. \qquad (4.17)$$

Das Funktionenpaar

$$F(s) \circ\!\!-\!\!\bullet f(t) \qquad (4.18)$$

stellt eine Korrespondenz dar. Einige wichtige Funktionen sind nocheinmal in Bild 4.7 zusammengestellt.

f(t)	F(s)
Sprungfunktion 1(t)	$\frac{1}{s}$
Nadelimpuls $\delta(t)$	1
t	$\frac{1}{s^2}$
t^n	$\frac{n!}{s^{n+1}}$
sin ωt	$\frac{\omega}{s^2+\omega^2}$
cos ωt	$\frac{s}{s^2+\omega^2}$
e^{kt}	$\frac{1}{s-k}$
$t\,e^{kt}$	$\frac{1}{(s-k)^2}$

Bild 4.7

Zusammengehörige Transformationspaare der Laplace-Transformation

Damit lassen sich lineare Systeme in einem Unterbereich - dem Zeitbereich - und einem Bildbereich darstellen.

$$X_e(s) \longrightarrow \boxed{G(s)} \longrightarrow X_a(s)$$

$$x_e(t) \longrightarrow \boxed{g(t)} \longrightarrow x_a(t)$$

Bild 4.7: Darstellungsmöglichkeit linearer Systeme

Die Übertragungsfunktion des Systems lautet nun:

$$G(s) = \frac{\mathcal{L}\{x_e(t)\}}{\mathcal{L}\{x_a(t)\}} = \frac{x_e(s)}{x_a(s)} \qquad (4.19)$$

Setzt man in (4.19) die komplexe Variable $p=j\omega$, dann geht die Übertragungsfunktion G(s) in den Frequenzgang F(jω) über.

Die Sprungantwort eines Systems mit

$$x_e(t) = 1(t) \circ\!\!-\!\!\bullet X_e(s) = \frac{1}{s}$$

$$x_e = 1(t) \longrightarrow \boxed{\begin{matrix}\text{lineares}\\ \text{System}\end{matrix}} \longrightarrow x_a = h(t) \qquad (4.20)$$

Bild 4.8: Übergangsfunktion als Sprungantwort

heißt Übergangsfunktion (Bild 4.8):

$$h(t) = \mathcal{L}^{-1}\left\{\frac{G(s)}{s}\right\} = \int_0^t g(\tau)\,d\tau \qquad (4.21)$$

Die graphische Darstellung dieser Übergangsfunktion wird i.a. verwendet, um technische Systeme in Form von Blockschaltbildern zu beschreiben.

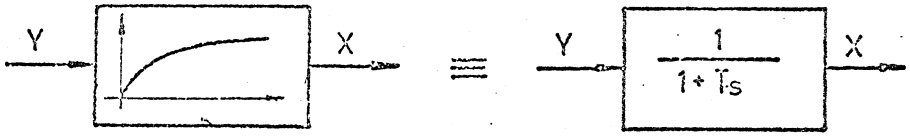


Bild 4.8 Systembeschreibung durch das Blockschaltbild

Wird ein System vorausgesetzt, das im Einschaltzeitpunkt energielos war, also

$$x_a(+0) \equiv 0 \qquad (4.22)$$

dann kann nach dem Differentiationssatz

$$\frac{d}{dt} f(t) \circ\!\!-\!\!\bullet s \cdot F(s) - f(+0) \qquad (4.23)$$

die komplexe Variable s rein formal durch den Operator $\frac{d}{dt}$ ersetzt werden und erhält so aus der Übertragungsfunktion die Differentialgleichung, die das gegebene System entsprechend beschreibt. Bei der Aufstellung der dazugehörigen Rechenschaltung ist jetzt aber häufig ein Problem zu lösen, das auch bei linearen Differentialgleichungen auftritt, sobald die Störfunktion nicht mehr direkt auf dem Analogrechner verfügbar ist.

Bei einer gebrochenen rationalen Übertragungsfunktion

$$G(s)=\frac{Z(s)}{N(s)}=\frac{s^m+\ldots+b_1 s+b_o}{s^n+\ldots+a_1 s+a_o} \qquad (4.24)$$

läßt sich eine Differentialgleichung der Form

$$\frac{dx_a^n}{dt^n}+\ldots+a_1\frac{dx_a}{dt}+a_o x_a(t)=b_m\frac{d^m x_e}{dt^m}+\ldots+b_1\frac{dx_e}{dt}+b_o x_e(t) \qquad (4.25)$$

aufstellen. Jetzt kann nicht mehr nach dem beschriebenen Verfahren (Auflösung nach der höchsten Ableitung) vorgegangen werden, da sonst eine mehrfache Differentiation der Eingangsgröße $x_e(t)$ nötig ist. Deshalb wird die Gleichung (4.25) so lange integriert, bis auf der rechten Gleichungsseite keine Ableitungen mehr auftreten. Für ein Beispiel

$$G(s)=\frac{s+b_o}{s^2+a_1 s+a_o} \qquad (4.26)$$

lautet die Differentialgleichung:

$$\frac{d^2x_a}{dt^2}+a_1\frac{dx_a}{dt}+a_o x_a=\frac{dx_e}{dt}+b_o x_e \qquad (4.27)$$

Aus Gleichung (4.27) folgt durch einmalige Integration:

$$\dot{x}_a=-a_1 x_a+x_e+\int_o^t(b_o x_e-a_o x_a)\,dt \qquad (4.28)$$

Bild 4.9 zeigt die dazugehörige Rechenschaltung.

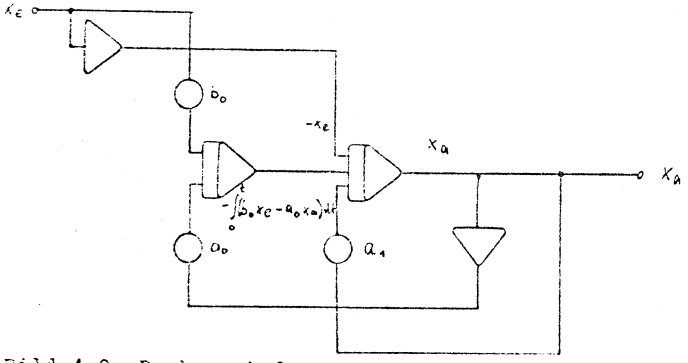


Bild 4.9: Rechenschaltung zur Gleichung (4.26)

Häufig werden technische Systeme durch einzelne Standard-Übertragungsglieder nach Bild 4.8 beschrieben. Kennt man die Gesetzmäßigkeiten der Übertragungsfunktion, dann läßt sich oft eine gegebene Übertragungsfunktion in solche Standardübertragungsglieder aufspalten.
Bild 4.11 zeigt die Rechenschaltungen solcher Standard-Übertragungssysteme.

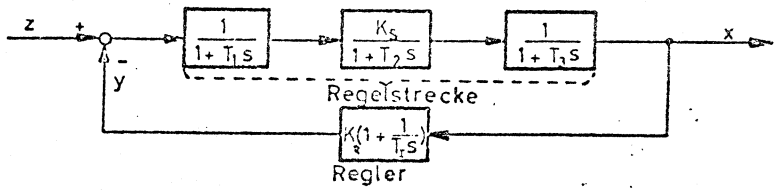
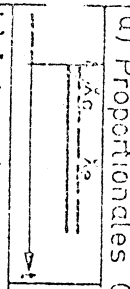
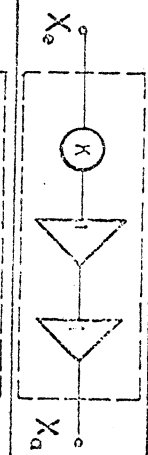
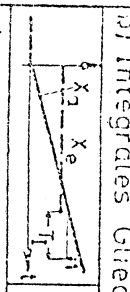
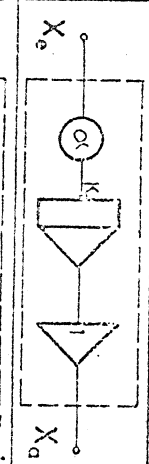
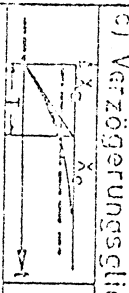
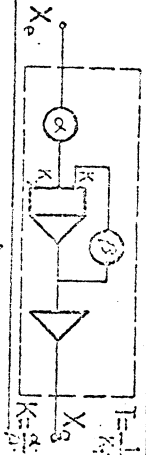
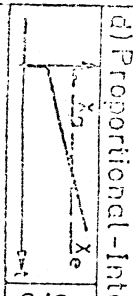
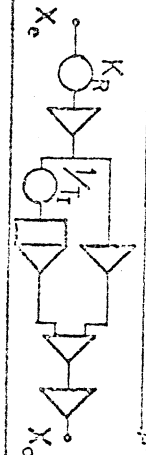
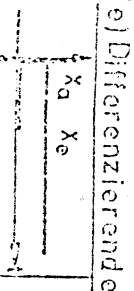
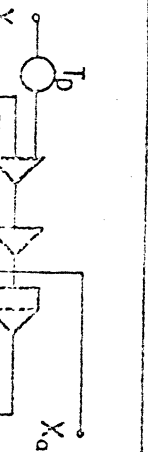


Bild 4.10

Blockschaltbild des zu untersuchenden Regelkreises

Ein System, wie z.B. der Regelkreis vom Bild 4.10, kann nun mit den verfügbaren Standard-Übertragungsgliedern bausteinartig aufgebaut werden. Natürlich kann auch die Übertragungsfunktion dieses Systems z.B. bezüglich der Führungsgröße Z ermittelt werden, und diese dann direkt in eine Rechenschaltung überführt werden. Obwohl beim ersten Verfahren u.U. mehr Rechenelemente benötigt werden, hat es jedoch den Vorteil, daß die Rechenschaltung durchsichtiger ist, und eine 1:1-Zuordnung zwischen realem System und Rechenschaltung möglich ist.

TABELLE 2 STANDARD - Übertragungssysteme

Übergangsfunktion	Differentialgleichung	Übertragungsfunktion	Rechenschaltung
a) Proportionales Glied (P-Verhalten)			
	$K \cdot X_e = X_a$	$G(s) = K$	
b) Integrales Glied (I-Verhalten)			
	$T_I \cdot \frac{dx_a}{dt} = X_e$	$G(s) = \frac{1}{T_I s}$	
c) Verzögerungsglied 1.Ordnung (PT_1-Glied)			
	$X_a + T \frac{dx_a}{dt} = K X_e$	$G(s) = \frac{K}{1+Ts}$	
d) Proportional-Integralwirkendes Glied (PI-Glied)			
	$\frac{dx_a}{dt} = K_R \left(\frac{dx_e}{dt} + \frac{1}{T_I} X_e\right)$	$G(s) = K_R \left(1 + \frac{1}{T_I s}\right)$	
e) Differenzierendes Glied (D-Verhalten)			
	$X_a = \frac{dx_e}{dt} \cdot T_D$	$G(s) = T_D \cdot s$	

5 Die Normierung analoger Rechenschaltungen

5.1 Das Problem der Normierung

5.1.1 Die qualitative Programmierung

Ehe man darangehen kann, am Analogrechner Rechenschaltungen zu erstellen, muß man ein zu bearbeitendes Problem eindeutig und vollständig in Form von Differentialgleichungen bzw Differentialgleichungssystemen und dazugehörigen Anfangswerten formulieren. Es ist eine Liste der im Problem auftretenden Konstanten und Parameter aufzustellen. Handelt es sich um ein physikalisches Problem, so ist die mathematische Formulierung unter Umständen ein wesentlicher Teil der Programmierung.

Für die Nachbildung eines technischen Systems auf dem Analogrechner gibt es zwei grundsätzliche Möglichkeiten. Man versucht entweder, das Problem durch möglichst wenige Differentialgleichungen zu beschreiben und diese zu programmieren. Dann benötigt man weniger Rechenelemente erhält aber ein recht unübersichtliches Programm, in dem alle Koeffizienten in komplizierter und vielfältiger Weise von den Systemparametern abhängen.

Der andere Weg besteht darin, alle einzelnen Gleichungen eines Systems unmittelbar zu programmieren. Dann entsprechen die Koeffizienten der Rechenschaltung noch unmittelbar den Systemparametern. Allerdings benötigt man mehr Rechenelemente (z. B. Potentiometer und Summierer). Solche Systeme lassen sich besonders leicht normieren, da das Verhalten der im System auftretenden Größen leichter überschaubar ist, und da diese unmittelbar verändert werden können, wenn am Rechner Übersteuerungen auftreten. Nicht zuletzt sind auf diese Weise die Rechenergebnisse oft leichter zu interpretieren.

Die Darstellung von Lösungen auf dem Analogrechner hat den Nachteil, daß man nur jeweils eine partikuläre Lösung eines Problems erhält, die abhängt von den gegebenen Anfangsbedingungen. Dagegen erhält man bei einer Lösung nach herkömmlichen mathematischen Lösungsmethoden einen analytischen Ausdruck für die Lösung. Erst durch das Einsetzen spezieller Anfangsbedingungen erhält man daraus eine bestimmte konkrete Lösung. Da bei Differentialgleichungen mit variablen Koeffizienten, bei Randwertproblemen und bei nichtlinearen Differentialgleichungen meist keine Lösungen in geschlossener Form existieren besitzt hier der Analogrechner Vorzüge.

Normalerweise sollte versucht werden, die zu bearbeitenden Probleme so zu formulieren, daß möglichst wenige Anfangsbedingungen ungleich Null vorliegen. Handelt es sich dabei um ein physikalisches-technisches Problem, so ist z. B. zu fragen, ob das vorliegende System im Ruhezustand oder in einem besonders interessierenden Bewegungszustand stehen soll, wenn die Beobachtung beginnt. Im allgemeinen ist es dann empfehlenswert mit dem Ruhezustand zu beginnen, da hier bei zweckmäßiger Formulierung der Aufgabe keine Spannungen zur Einstellung von Störgrößen oder Anfangsbedingungen benötigt werden. Die natürlichen Anfangswerte heben sich in ihrer Wirkung gegen die wirksamen Störfunktionen gerade weg. Bei dieser Programmierungsmethode erscheinen am Ausgang der Rechenelemente nur die veränderlichen Teile der beobachteten Größen. Ein weiterer Vorteil ist dann gegeben, wenn die Abweichungen der veränderlichen Größen gegenüber bestimmten Festwerten relativ klein sind (siehe "Normierung nach Abweichungen von Festwerten").
An die mathematische Vorbereitung schließt sich der Entwurf eines qualitativen Programmbildes an. Dabei zeichnet man im Blockschaltbild ohne Berücksichtigung von Normierungsbedingungen die den Rechenelementen entsprechenden Symbole ein, verbindet deren Ein- und Ausgänge in der Weise, wie sie durch die Problemstellung vorgeschrieben wird und bezeichnet alle Ein- und Ausgangsgrößen der Rechenelemente (Namen der Variablen) sowie alle Parameter- und Koeffizientenwerte. Die qualitative Programmierung ist normalerweise ohne Schwierigkeiten durchzuführen. Im Gegensatz dazu muß bei der quantitativen Programmierung streng darauf geachtet werden, daß alle auftretenden Rechengrößen innerhalb erlaubter Größenordnungen bleiben, was unter Umständen beträchtliche Schwierigkeiten mit sich bringen kann.
Ein besonderes Problem bei der analogen Programmierung ist die Frage der Stabilität der Rechenschaltung. Dazu gibt es mathematische Methoden, um festzustellen, ob Stabilität vorliegt. Meist wird diese Frage experimentell zu untersuchen sein. An dieser Stelle kann auf Stabilitätsfragen nicht weiter eingegangen werden.

5.1.2 Die Skalierung von Rechengrößen

Die Probleme, welche am Analogrechner bearbeitet werden sollen, liegen normalerweise vor als Differentialgleichungen oder Systeme von Differentialgleichungen, d. h. in rein mathematischer Form. Im allgemeinen liegen diesen Gleichungen physikalische Systeme zugrunde. Der Analogrechner selbst kann ebenfalls als physikalisches System aufgefaßt werden, dessen Signale sich analog zu denen des zu betrachtenden Systems verhalten. Allerdings liegen meist die Problemgleichungen nicht sofort in einer Form vor, die den Bedingungen am Analogrechner entspricht, da die zu erwartenden Bezugs- und Einstellgrößen noch nicht berücksichtigt sind, die am Analogrechner auftreten können.
Außerdem führt die mathematische Formulierung eines technischen Problems zunächst immer auf Gleichungen, in denen die einzelnen Größen noch mit Dimensionen behaftet sind. Ebenso sind die am Analogrechner auftretenden Größen mit Dimensionen behaftet, die normalerweise nicht mit denen der Problemgrößen identisch sind.
Am Analogrechner ist die unabhängige Problemgröße immer die Zeit, und es gibt auch nur diese eine unabhängige Größe. Alle abhängigen Problemgrößen am Analogrechner sind Spannungen, die an bestimmten Stellen (z. B. Verstärkerausgänge, Potentiometerabgriffe) gegen Rechenerde abgegriffen werden können.
Voraussetzung für eine sinnvolle Benutzung des Analogrechners ist es, daß eine eindeutige Zuordnung zwischen den sogenannten Problemgrößen (abhängig vom Problem) und zwischen den Maschinengrößen (Zeit und Spannungen) geschaffen werden kann. Erst dann ist es möglich, die Rechenschaltung geeignet zu programmieren und die Rechenergebnisse richtig zu deuten.Es ist deshalb nötig sowohl für die unabhängige als auch für die abhängigen Variablen Maßstabsfaktoren einzuführen, die Verknüpfungen einerseits zwischen den Betragsgrößen und andererseits zwischen den Dimensionen der betrachteten Größen herstellen. Dabei ist es eine weitere Nebenbedingung, daß alle im Problem auftretenden Koeffizienten und Parameterwerte in den am Analogrechner zulässigen Bereich transformiert werden müssen.

Beispiele, in denen sowohl die Problemzeit als auch die sonstigen Problemgrößen mit den Maschinengrößen größenordnungsmäßig nicht zur Deckung kommen, sind z. B. die Bewegung eines Elektrons um

einen Atomkern oder Planetenbewegungen, sowie die F kurven von Erdsatelliten, die Halbwertszeit beim Atomzerfall, der Ladestrom am Kondensator, Wärmeleitungsprozesse, chemische Prozesse usw. Eine besondere Art der Zeitnormierung wird dann notwendig, wenn bei physikalischen oder mathematischen Problemen nicht die Zeit sondern ein Winkel, ein Weg oder andere Größen als unabhängige Variable auftreten. Das kann z. B. auftreten bei der barometrischen Höhenformel, bei der Stabknickung oder anderen Belastungsproblemen, bei der Streuung von Kernteilchen beim Atomzerfall usw.

Es kann häufig geschehen, daß verschiedene Prozesse auf denselben Gleichungstyp führen, z. B. führen viele Schwingungsprobleme auf den Typ der gedämpften Schwingung mit konstanten Koeffizienten. Zwei physikalische Systeme sind einander analog, d. h. sie besitzen dasselbe mathematische Modell, wenn sie - abgesehen von den Dimensionen der betrachteten Größen und den Amplituden- und Zeitmaßstabsfaktoren - durch dieselben Gleichungen beschrieben werden.

Unter Normierung oder Skalierung versteht man nun die wohldefinierte und nach einem geeignet zu wählenden Maßstab vorgenommene Zuordnung der Größen eines betrachteten Systems zu den Rechengrößen des Analogrechners (Spannung und Zeit). Durch geschicktes Vorgehen erhält man aus der ursprünglichen Differentialgleichung bzw dem Differentialgleichungssystem eine dimensionslose Gleichung, die sich in eine am Analogrechner herstellbare Rechenschaltung abbilden läßt. Die Wahl der Maßstabsfaktoren ist von entscheidender Bedeutung für die Genauigkeit der zu erwartenden Rechenergebnisse. Allerdings hat die Normierung als solche grundsätzlich noch nichts mit einer Lösung eines Problems am Analogrechner zu tun, da hierdurch die mathematischen Zusammenhänge nicht verändert werden, sondern nur die konkreten Zahlenverhältnisse eines Problems in einen dem Analogrechner und der menschlichen Anschauung (Beobachtung am Bildschirm und manuelles Eingreifen durch den Programmierer) entsprechenden Zahlenbereich transformiert werden.

Das Hauptproblem bei der Normierung einer Rechenschaltung besteht in der Amplitudennormierung, vor allem im Beschaffen der richtigen Maßstabsfaktoren, die stark von den auftretenden Rechengrößen abhängen. Die Zeitnormierung sollte grundsätzlich nach der Amplitudennormierung ausgeführt werden, da sie normalerweise keine großen Schwierigkeiten bereitet und dazu benützt werden kann, die nach der Amplitudennormierung auftretenden Koeffizientenwerte in den Systemgleichungen in einstellbare Wertebereiche zu transformieren.

5.2 Die Normierung der abhängigen Variablen (Amplitudennormierung)

5.2.1 Die Notwendigkeit der Amplitudennormierung

Am Analogrechner werden alle abhängigen Variablen durch Spannungen dargestellt. Den mit irgendeiner Dimension behafteten Problemgrößen entsprechen als analoge Rechengrößen also immer Spannungen. Voraussetzung für eine sinnvolle Benutzung des Analogrechners ist somit, daß zwischen den Größen des Problems und den Rechengrößen des Analogrechners ein eindeutig definierter Zusammenhang hergestellt werden kann. Dazu wird zunächst ein Faktor für die Dimensionierung benötigt, d. h. ein Umrechnungsfaktor, der aus den als Spannungen abgegriffenen Rechengrößen solche mit der Dimension der Problemgröße macht. Außerdem muß dafür gesorgt werden, daß die analogen Rechengrößen, den am Analogrechner zulässigen Bereich der Spannungen nicht überschreiten. Dieser Rechenbereich ist bei verschiedenen Rechnertypen unterschiedlich, meist 10 V oder 100 V Arbeitsspannung maximal (am hybriden Präzisionsrechner HRS 860 wird mit 10 V gearbeitet). Diese Grenzen dürfen nach positiver (+10 V) und nach negativer (-10 V) Richtung nicht überschritten werden, da sonst einerseits die betroffenen Verstärker überlastet werden und andererseits die Rechenergebnisse verfälscht werden. Bild 5.1 zeigt den Zusammenhang zwischen der auf einen Verstärkereingang aufgegebenen, d. h. aufgrund der Rechnung anstehenden Wert U_E der Rechenspannung und dem tatsächlich am Ausgang erscheinenden Wert der Spannung U_A. U_A folgt U_E genau linear, solange U_E sich im erlaubten Spannungsbereich $-10\ V = U_E = +10\ V$ bewegt. Ist das nicht mehr der Fall, so weicht U_A immer stärker von U_E ab. Aus Sicherheitsgründen ist eine geringfügige Übersteuerung der Rechenelemente zulässig, so daß U_A im Maximalfall bis zu etwa ± 12 V betragen kann.

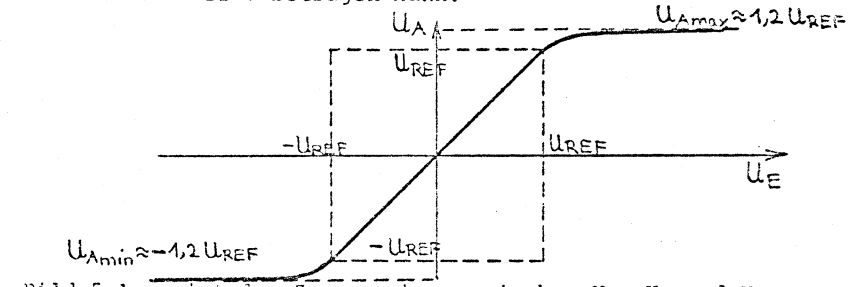


Bild 5.1 zeigt den Zusammenhang zwischen U_A, U_E und U_{REF}

Daraus ergibt sich, daß die Zuordnung einer Problemgröße zu einer Rechengröße am Analogrechner so geschehen muß, daß die Rechengröße den zulässigen Spannungsbereich nicht überschreitet, d. h. im allgemeinen Falle muß nicht nur nach der Dimension sondern ebenfalls nach der Größenordnung der Variablen ein Maßstabsfaktor eingeführt werden. Dabei ist vor allem darauf zu achten, daß der mögliche Rechenbereich gut ausgenützt wird, da sonst die Ergebnisse ungenauer werden. Es gibt nämlich eine untere Genauigkeitsgrenze ab der die Rechnung wegen der zu schlechten Aussteuerung der Rechenelemente mit größeren Fehlern behaftet ist. Bei den (10 V)-Analogrechner liegt dieser Grenzbereich bei etwa 10 mV. Deshalb dürfen sich die Variablen nur in einem Bereich von höchstens vier Dezimalen ändern, wenn sie genau bleiben sollen. Das bringt oft empfindliche Beschränkungen in der Verwendung von Analogrechnern mit sich.

5.2.2 Die Normierung nach Höchstwerten

Um am Rechner dimensionslose Vergleichsgrößen zu haben, bezieht man zur Vereinfachung alle Rechenspannungen auf die Maschinenspannung 10 V (bzw 100 V), indem man statt der jeweiligen Rechenspannung $u(t^*)$ die normierte Größe (E = Maschinenspannung = 10 V) $U(t^*)$ einführt, für die gilt

$$U(t^*) = \frac{u(t^*)}{E} \qquad (5.1)$$

Dann bewegt sich diese normierte Rechengröße nur noch im Bereich

$$-1 \leqq U(t^*) \leqq +1$$

Dabei ist t^* die sogenannte Maschinenzeit (s. u.).
Nach dieser Normierung kann man den Analogrechner mit einem Digitalrechner vergleichen, der nur mit Festkommazahlen (hier echten Brüchen) arbeiten kann, d. h. die normierte Rechengröße $U(t^*)$ ist einer Festkommazahl vergleichbar. Übersteuerungen bei Rechengrößen entsprechen dann einer Bereichsüberschreitung (BÜ-Alarm) bei Digitalrechnern. Die Aufgabe der Normierung der Problemgrößen besteht nun darin, um den Vergleich weiterzuführen, die Problemgrößen, den Gleitkommaoperationen bei Digitalrechnern ähnlich, in einen Rechenbereich zu transformieren, der am Analogrechner darstellbar ist, d. h. in den normierten, dimensionslosen Rechenbereich $-1 \leqq$ (Rechengröße) $\leqq +1$. Es hat sich dabei eingebürgert, die unnormierten Größen mit kleinen Buchstaben und die normierten Größen mit großen Buchstaben zu benennen.

Bei der Normierung nach Höchstwerten bzw nach Festwerten wird eine Problemgröße ersetzt durch die folgende dimensionslose Rechengröße

$$X(t) = \frac{x(t)}{|x|_{max}} \qquad (5.2)$$

für die wie verlangt gilt

$$-1 \leqq X(t) \leqq +1 \qquad (t \text{ heißt Problemzeit}).$$

Statt $|x|_{max}$ kann ein beliebiger anderer Festwert als Bezugsgröße gewählt werden, z. B. dann, wenn $|x|_{max}$ nicht bekannt ist.

Will man nun der Problemgröße $x(t)$ die Analogrechnergröße $u(t^*)$, die man Maschinengröße nennt, zuordnen, so kann dies z. B. dadurch geschehen, daß man der maximalen Problemgröße $|x|_{max}$ die Maschinenspannung E als entsprechende Vergleichsgröße zur Ermittlung des Maßstabsfaktors für die Umrechnung zuweist. Durch das Gleichsetzen der normierten Größen $X(t)$ und $U(t^*)$ - dabei wird vorausgesetzt, daß t und t^* durch die Zeitnormierung auf entsprechende Weise verknüpft sind (s. u.) - erhält man den Zusammenhang zwischen Maschinen- und Problemgröße

$$\frac{x(t)}{|x|_{max}} = \frac{u(t^*)}{E} \qquad (5.3)$$

oder

$$x(t) = \frac{|x|_{max}}{E}\, u(t^*) \qquad (5.4)$$

Als Dimensionen sind für die Problemgrößen die im Problem auftretenden Dimensionen einzusetzen, während die Maschinengrößen immer die Dimensionen [V] und [sec] besitzen.

In derselben Weise sind die Ableitungen $\dot{x}(t), \ddot{x}(t), \ldots$ nach Fest- bzw Höchstwerten zu normieren. Da die zu erwartenden Problemgrößen nur in den seltensten Fällen mit den Maschinengrößen nach der Dimension und nach den Zahlengrößen direkt zusammenfallen, ist die Amplitudennormierung immer auszuführen. Die Amplitudennormierung bringt für die Programmierung und die Verfolgung der Ergebnisse, (z. B. auf dem Oszillographen oder X-Y-Schreiber) bedeutende Vorteile. Man braucht nur noch zu wissen, daß sich alle Rechengrößen zwischen -1 und +1 bewegen, und alle Anzeigegeräte und Einstellgeräte - vor allem die Potentiometerwerte - können einheitlich darauf normiert bzw geeicht werden. An Potentiometern treten demgemäß nur noch Koeffizienten zwischen 0 und +1 auf. Ein weiterer beträchtlicher Vorteil besteht darin, daß mit den auf den Bereich $-1 \leqq$ Rechengröße $\leqq +1$ normierten Rechengrößen in Hybridrechner-Systemen sofort der Anschluß an die Festkommazahlendarstellung ermöglicht wird. Das ist z. B. auch beim System HRS 860 der Fall.

5.2.3 Ermittlung der Bezugsgrößen

Bei der Amplitudennormierung besteht die eigentliche Schwierigkeit darin, geeignete Maßstabsfaktoren, d. h. geeignete Maximalwerte zu finden, so daß die normierten Größen betragsmäßig nicht größer als 1 werden und andererseits auch nicht zu klein geraten. Die Maximalgrößen lassen sich in vielen Fällen mit hinreichender Genauigkeit abschätzen, entweder anhand der bekannten Eigenschaften und Gegebenheiten eines technischen Problems oder aufgrund mathematischer Abschätzungen über die Extremwerte der Problemgrößen. Unter Umständen müssen die ersten Abschätzungen durch Experimente am betrachteten System selbst erst gewonnen werden. Es ist darauf zu achten, daß die eingesetzten Maximalwerte immer nach der sicheren Seite hin abgeschätzt werden, d. h. sie sollen lieber etwas zu groß sein, da dann zwar die Rechengrößen kleiner werden aber immerhin keine Übersteuerung von Rechenelementen auftreten kann. Andernfalls kann es zu Übersteuerungen kommen, die sich auf weitere Rechenelemente fortpflanzen, so daß oft nicht mehr festgestellt werden kann, an welchem Rechenelement die ursprüngliche Übersteuerung eingesetzt hatte. Bild 5.2 zeigt anschaulich, daß für die Abschätzung der Maximalwerte nicht die maximale Auslenkung einer Problemgröße entscheidend ist ($|y_{max}|$) sondern die betragsmäßig maximale Auslenkung ($|y|_{max}$). Dabei kann es z. B. auch geschehen, daß - evtl. aufgrund bestimmter Anfangswerte der Rechenschaltung - alle auftretenden Werte in der Rechenzeit entweder negativ oder positiv ausfallen (siehe Normierung nach Abweichungen von Festwerten).

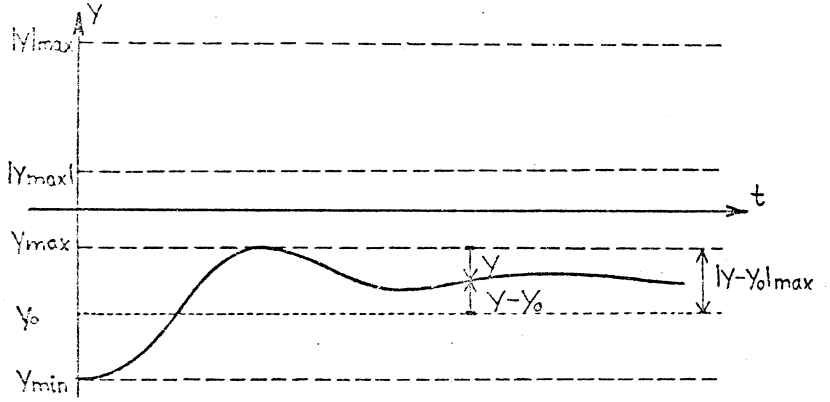


Bild 5.2 zeigt die Verhältnisse bei der Bestimmung des Maximalwerts

Es ist immer anzustreben, die notwendige Amplitudennormierung optimal vorzunehmen, d. h. so daß keine Übersteuerung eintritt daß aber auch alle Rechenelemente möglichst gut ausgenützt werden, also die Rechengrößen den ganzen Bereich zwischen -1 und +1 möglichst weitgehend durchlaufen. Eine solche optimale Programmierung liefert Lösungen mit höchstmöglicher Genauigkeit. In den Fällen, in denen keine Anhaltspunkte für die zu erwartenden Maximalwerte vorliegen, muß man von willkürlich zu wählenden Maximalwerten bzw Schätzungen dafür ausgehen. Dabei ist wieder darauf zu achten, daß im Zweifelsfalle besser zu große Werte genommen werden. Es kann notwendig sein, mit einer Ausgangsprogrammierung durch Testrechnungen die Amplitudenwerte zu ermitteln und die Normierungsfaktoren zu verbessern. Diese Korrektur ist bei linearen Differentialgleichungen besonders einfach, weil dabei die Maßstabsfaktoren nur als Konstanten bei den Störfunktionen oder bei den Anfangswerten auftreten (siehe die Erzeugung der Sinusfunktion).

Die Amplitudennormierung ist nicht nur für die Variablen eines Problems selbst, sondern - soweit sie auftreten - auch für deren Ableitungen durchzuführen. Ebenso sind alle Anfangswerte zu normieren. Diese können durch die Zeitnormierung nicht beeinflußt werden! Tritt auch die unabhängige Problemvariable explizit in den Problemgleichungen auf, so muß auch für sie eine Bezugsgröße zur Maschinenzeit hergestellt werden (siehe Zeitnormierung).

Es ist manchmal zweckmäßig für gleiche physikalische Größen, die man untereinander vergleichen will - z. B. räumliche Koordinaten x, y, z - , verschiedene Maßstabsfaktoren bei der Normierung zu verwenden, z. B. wenn diese betragsmäßig in sehr unterschiedlichen Größenordnungen liegen. Will man das vermeiden, so muß man sich nach dem Höchstwert unter den auftretenden Maximalwerten für die verschiedenen (dimensionsgleichen) Problemgrößen richten.

Ehe ein Programm endgültig festgelegt wird, sollte eine Aussteuerkontrolle erfolgen. Die schrittweise Verbesserung einer eventuell zu schlechten Aussteuerung von einzelnen Rechengrößen muß auf jeden Fall erfolgen, ehe ein Ergebnis der Rechnung festgehalten wird. Soweit es möglich ist, sollten als Maßstabsfaktoren immer ganze Zahlen gewählt werden, damit die spätere Auswertung und Umrechnung

leichter wird. In besonders günstigen Fällen können die verschiedenen Maßstabsfaktoren so gewählt werden, daß sie sich nur durch Zehnerpotenzen unterscheiden.

Nach der beendeten Normierung empfiehlt es sich eine Dimensionskontrolle durchzuführen. Alle Koeffizienten von Potentiometern vor Summierern und die Anfangswertpotetiometer müssen dimensionslos sein. Dagegen weisen die Koeffizienten an den Eingängen der Integrierer die reziproke Einheit der unabhängigen Problemvariablen auf.

In Bild 5.2 ist angedeutet, daß es sich nicht immer empfiehlt, die Amplitudennormierung nach Festwerten vorzunehmen, d. h. nach der Abweichung einer Rechengröße von Null. Bei vielen Problemen ist ein stationärer Anfangswert gegeben, der im Laufe der Rechnung den veränderlichen Anteilen überlagert bleibt, so daß eigentlich bei der Lösung nicht der Gesamtwert interessiert, sondern die Abweichung von diesem als Anfangswert vorgegeben Festwert. Allerdings kann es möglich sein, durch eine geeignete Änderung am Problem, solche Festwerte auf den Wert Null zurückzuführen, z. B. wenn man bei physikalischen Systemen mit dem System in Ruhelage die Rechnung beginnt (s. o.).
Bei der Normierung nach Abweichungen von einem Festwert lautet die entsprechenden Transformationsgleichung zu Gleichung (5.2)

$$X(t) = \frac{x(t) - x_o}{|x(t) - x_o|_{max}} \qquad (5.5)$$

mit x_o als stationärem Ausgangswert. Damit wird $X(t) = 0$ für $x = x_o$. Da x_o ein konstanter Summand ist, bleiben die Ableitungen davon unberührt.
Weiter erhält man statt Gleichung (5.4) die neue Gleichung

$$x(t) = \frac{|x(t) - x_o|_{max}}{E} u(t^*) \qquad (5.6)$$

Eine weitere Möglichkeit der Normierung ist die Normierung mit Dimensionen. Dabei behalten die Spannungen am Rechner die Dimension V , die unabhängige Variable - die Maschinenzeit - besitzt die Dimension sec und die Problemvariablen behalten ihre Dimensionen. Diese Normierungsmethode enthält dimensionsbehaftete Maßstabsfaktoren,und sie ist wesentlich schwieriger als die oben beschriebene Art. Sie besitzt keine zwingenden Vorteile, allerdings erlaubt sie es, die ursprünglichen Bezeichnungen der Problemvari-

ablen im Schaltplan beizubehalten.

5.2.4 Amplitudennormierung nichtlinearer Differentialgleichungen

Bei linearen Differentialgleichungen ist die volle Ausnützung des ganzen Arbeitsbereiches der Maschinenspannungen bzw Rechengrößen weniger entscheidend als bei nichtlinearen Differentialgleichungen, da die linearen Rechenelemente des Analogrechners (Summierer, Integrierer, Potentiometer) sehr genau arbeiten. Nichtlineare Rechenelemente (Multiplizierer, Funktionsgeber usw) arbeiten viel ungenauer, deshalb ist hier die Normierung wesentlich schwieriger. Für die Normierung nichtlinearer Differentialgleichungen gibt es keine allgemeinen Anleitungen, jedoch kann man eine gewisse Systematik dadurch erreichen, daß man versucht ein vorliegendes Problem in einen linearen und einen nichtlinearen Teil aufzuspalten, die man dann getrennt normieren kann. Ein besonders häufiger Spezialfall sind nichtlineare Differentialgleichungen, in denen nur Produkte oder Potenzen von Variablen auftreten.
Die Normierung von Produkten ist zunächst einfach; denn das Produkt zweier Größen, die beide betragsmäßig kleiner als 1 sind, kann selbstverständlich betragsmäßig auch nicht größer als 1 werden. Das Problem besteht allerdings darin, daß das Produkt zweier normierter Größen nicht immer optimal normiert ist, da das Produkt unter Umständen sehr klein werden kann. Multipliziert man zwei Größen 0.01, so liegt das Ergebnis 0.0001 am Rande des Rechenbereichs und eine geringfügige Schwankung verfälscht es beträchtlich. Deshalb ist oft eine zusätzliche interne Verstärkung erwünscht. Es ist also darauf zu achten, daß durch Abspaltung von Faktoren in den auftretenden Konstanten und durch Einschalten von Zwischenverstärkungen der Arbeitsbereich der Multiplizierer so günstig wie möglich gewählt wird.
Für zwei normierte Größen $Y(t)$, $X(t)$ gilt dann z. B.

$$Y(t) = \frac{|y(t)|_{max}}{E} u_1(t^*); \quad X(t) = \frac{|x(t)|_{max}}{E} u_2(t^*)$$

und für das Produkt

$$Y(t) * X(t) = \frac{|y(t)|_{max} * |x(t)|_{max}}{E^2} * u_1(t^*) * u_2(t^*) \qquad (5.7)$$

Bei nichtlinearen Differentialgleichungen,in denen weitere Nichtlinearitäten auftreten, gibt es keine allgemeine Regel mehr. Es muß dabei versucht werden,jede nichtlineare Funktion auf ihren Maximalwert hin zusätzlich zu normieren.

5.3 Normierung der unabhängigen Variablen (Zeitnormierung)

5.3.1 Das Problem der Zeitnormierung

In vielen Problemen der Technik tritt die Zeit als unabhängige Variablen auf. Das entspricht zunächst den Verhältnissen am Analogrechner, wo die Zeit ebenfalls die unabhängige Variable ist. Allerdings variieren die Zeiten bei den verschiedenen anfallenden Problemen ganz beträchtlich. Gitterschwingungen von Molekülen, Atomzerfallsprozesse usw sind viel zu schnell, als daß sie in "Echtzeit" verfolgt und berechnet werden könnten. Andererseits dauern Probleme wie die Verfolgung einer Satellitenbahn oder das Verhalten chemischer Prozesse u. ä. viel zu lange. Das bedeutet, daß für den Analogrechner sowohl zu kurze wie auch zu lange Rechenabläufe ungeeignet sind und deshalb durch eine Zeittransformation die Problemzeit in einen vernünftigen Zeitbereich der Maschinenzeit geschafft werden muß. Das ist auch deshalb notwendig, weil bei zu kurzen Rechenzeiten das Frequenzverhalten der Rechenelemente und vor allem das der Registriergeräte (Oszillograph, X-Y-Schreiber) berücksichtigt werden muß und in die Rechenergebnisse eingeht. Bei zu langen Zeiten gehen z. B. an den Integrierern alle Fehlerspannungen - multipliziert mit der Integrationszeit! - in die Rechenergebnisse ein.
Die Zeitnormierung bestimmt die Geschwindigkeit, mit der ein Vorgang am Analogrechner abläuft, dadurch wird das Problem dem Rechner dynamisch angepasst. Andererseits läßt sich mit Hilfe der Zeitnormierung auch oft die Forderung verwirklichen, daß die Koeffizienten, die in der Rechenschaltung auftreten innerhalt des zulässigen Bereichs liegen, in dem sie sinnvoll und einstellbar sind, das ist $0.01 \leqq \text{Koeffizient} \leqq 10$. Vor allem auch diese Möglichkeit, die Koeffizienten in den günstigen Bereich zu bringen, wird dazu führen, daß meist nicht nur eine Amplituden- sondern auch eine Zeitnormierung durchgeführt wird.

Die im Problem vorliegende Zeit wird normalerweise als Problemzeit t bezeichnet, die am Rechner wirkende Maschinenzeit heißt t^*, die normierte Zeit wird mit τ bezeichnet. Diese Bezeichnungen wurden bereits in den Gleichungen (5.1) bis (5.7) verwendet. Die Zeitnormierung stellt nun zwischen der Problemzeit t und der Maschinenzeit t^* einen Zusammenhang her. Durch die Normierung wird ein Zeitintervall $t_1 \leqq t \leqq t_2$ der Problemzeit

in ein Intervall $0 \leqq t^* \leqq T$ der Maschinenzeit transformiert, wobei T die Gesamtrechenzeit ist. Es gilt dann allgemein die Beziehung

$$t^* = \beta(t-t_1) \qquad \text{oder, wenn } t_1 = 0 \text{ gewählt wird}$$

$$t^* = \beta \cdot t \qquad (5.8) \quad .$$

Für einen Maßstabsfaktor $\beta=1$ verläuft der Rechenvorgang in Echtzeit, d. h. die Problemzeit ist gleich der Maschinenzeit am Rechner und die Rechengeschwindigkeit ist gleich der Geschwindigkeit des wirklichen Prozesses. Wählt man β größer als 1, dann verläuft der Rechenvorgang am Analogrechner mit einer Zeitdehnung; verwendet man für β einen kleineren Wert als 1 - β ist grundsätzlich positiv zu wählen! - , so verläuft der Rechenvorgang mit einer Zeitraffung, d. h. also schneller als beim wirklichen Prozeß.
Bei der Durchführung der Zeitnormierung führt man zunächst eine normierte Zeit τ ein, die eine Art Hilfsgröße ist, da sie am Rechner selbst nicht direkt auftritt. Normiert man sowohl die Problemzeit t als auch die Maschinenzeit t^* bezogen auf diese normierte Zeit τ, so erhält man für die Zeitnormierung zwei Freiheitsgrade. Der erste ist die Einführung der normierten Problemzeit in der folgenden Form

$$\tau = \lambda \cdot t \qquad (5.9).$$

Dabei hat λ die reziproke Dimension der Zeit t bzw der unabhängigen Problemvariablen (falls diese nicht die Zeit ist).
λ muß so gewählt werden, daß die dadurch veränderbaren Koeffizienten und Anfangsbedingungen der zu lösenden Differentialgleichungen mit Hilfe von Potentiometern am Analogrechner eingestellt werden können, ohne daß eine Übersteuerung eintritt (s. u.).

$$y = A\int_0^t dt \quad \xrightarrow{d\tau = \lambda \cdot dt} \quad y = A\int_0^{\tau/\lambda} \frac{d\tau}{\lambda} = \frac{A}{\lambda}\int_0^{\tau/\lambda} d\tau$$

Bild 5.3 zeigt die Integration nach Problemzeit t bzw normierter Zeit τ

Der zweite Freiheitsgrad besteht in der Einführung der normierten Maschinenzeit in der folgenden Form

$$\tau = k_o \cdot t^* \qquad (5.10)$$

Der Integrationsfaktor k_o ist gegeben als der Einfluß verschiedener zur Auswahl stehender Integrationskondensatoren an den Integrierern. k_o gibt an, in welchem Zeitmaßstab ein Integrierer arbeitet. Der Integrationsfaktor besitzt die Dimension [1/sec] .

Aus den beiden Gleichungen (5.9) und (5.10) erhält man entsprechend wie bei der Amplitudennormierung durch Gleichsetzen den Maßstabsfaktor für die Abbildung der Problemzeit in die Maschinenzeit

$$\lambda \cdot t = k_o \cdot t^*$$

oder

$$t^* = \frac{\lambda}{k_o} \cdot t \qquad (5.11) \quad .$$

Demgemäß legt der Faktor k_o/λ den Maßstab fest, in dem man die Lösung erhält.

Die damit festgelegten Zeittransformationen müssen auch auf die zeitlichen Ableitungen (bzw auf die Ableitungen nach der unabhängigen Variablen), soweit sie im Differentialgleichungssystem enthalten sind, angewendet werden.
Allgemein gilt für die Ableitung einer Variablen x nach der Zeit

$$\frac{d\,x(t^*)}{dt^*} = \frac{d\,x(t)}{\frac{\lambda}{k_o} \cdot dt} = \frac{k_o}{\lambda} \cdot \frac{d\,x(t)}{dt} \qquad (5.12)$$

Das bedeutet, daß bei einer Zeittransformation nach den Gleichungen (5.9) - (5.11) die Zeitnormierung vor jedem Integrierer ausgeführt werden muß und in der Differentialgleichung bei der n-ten Ableitung nach der Problemzeit t stehen muß

$$\frac{d^n x(t)}{dt} = \frac{\lambda^n}{k_o^{\,n}} \cdot \frac{d^n x(t^*)}{dt^{*n}} \qquad (5.13) \quad .$$

Ist der Faktor $\beta = \frac{\lambda}{k_o}$ größer als 1, so müssen also auch die Ableitungen nach der Maschinenzeit mit Faktoren größer 1 multipliziert werden. Ist β kleiner als 1, so werden alle Ableitungen nach t^* mit Faktoren kleiner 1 multipliziert. Bei der Auswertung der Rechenergebnisse sind diese Faktoren zu berücksichtigen, z. B. wenn eine abgeleitete Größe wie Geschwindigkeit oder Beschleunigung interessiert. Es darf durch die Zeitnormierung bei der Auswertung keine Verfälschung der Amplituden eintreten.

5.3.2 Eigenschaften des Zeitmaßstabes

5.3.2.1 Der Integrationsfaktor (k_o)

Wie schon erwähnt, läßt sich der Integrationsfaktor k_o ändern, wenn man den Eingang der Integrierer ändert. Wird k_o vergrößert, so wird die Rechengeschwindigkeit im selben Maß erhöht. Soll eine Schaltung z. B. 10-mal bzw 100-mal schneller rechnen, so genügt es, die Wertigkeit aller Integriereingänge mit dem Faktor k_o=10 bzw k_o=100 zu erhöhen. Die Gesamtrechenzeit bleibt dabei gleich im Unterschied dazu, wenn man am Analogrechner die Taste 10X drückt, womit die Rechnung 10-mal schneller gemacht wird, aber auch die Rechenzeit 10-mal kürzer. Normalerweise sind als k_o-Faktoren die Werte 1 (keine besondere Schaltung notwendig), 10 und 100 vorhanden (letztere müssen durch einen Kurzschlußstecker gesteckt werden). Sind in einer Rechenschaltung mehrere Integrierer enthalten, so kann der Zeitmaßstab der gesamten Schaltung geändert werden, indem alle Integrierer mit demselben k_o-Faktor beschaltet werden. Dabei darf die gesamte Rechenschaltung sonst völlig beliebig sein, also auch beliebig kompliziert und umfangreich, d.h. die Rechengeschwindigkeit eines ganzen Programms kann ohne Änderung der Programmierung erhöht werden. Im Gegensatz zum Faktor λ (s. u.) kann dadurch keine Übersteuerung am Rechner auftreten.
Es soll deshalb möglichst so programmiert werden, daß die Wahl des Zeitmaßstabs nur auf die Beschaltung der Integrierer Einfluß hat. Dann kann z. B. nach einem unbefriedigenden Probelauf leicht der Zeitmaßstab geändert werden, ohne daß die restliche Rechenschaltung davon berührt wird.
Wenn die Zeittransformation alle Koeffizienten in denselben Zahlenbereich bringen soll, so muß für zwei ursprüngliche Koeffizienten, die vor der i-ten bzw(i+1)-ten Ableitung einer bestimmten Rechengröße X(t) stehen

$$a_i \approx k_o \cdot a_{i+1} \qquad (5.14)$$

bzw

$$a_i \approx k_o^{\,i} \cdot a_o \qquad (5.15) \quad ,$$

wenn gilt a_i = Koeffizient vor der i-ten Ableitung von X(t).

Diese Formel kann nur dann verwendet werden, wenn das Verhältnis der aufeinanderfolgenden Koeffizienten annähernd konstant und etwa gleich einem möglichen Faktor k_o ist. Das ist meist nicht

für alle Koeffizientenwerte erfüllt. Jedoch läßt sich immer ein Teil davon durch Wahl eines geeigneten Faktors k_o in den am Analogrechner einstellbaren Bereich transformieren.

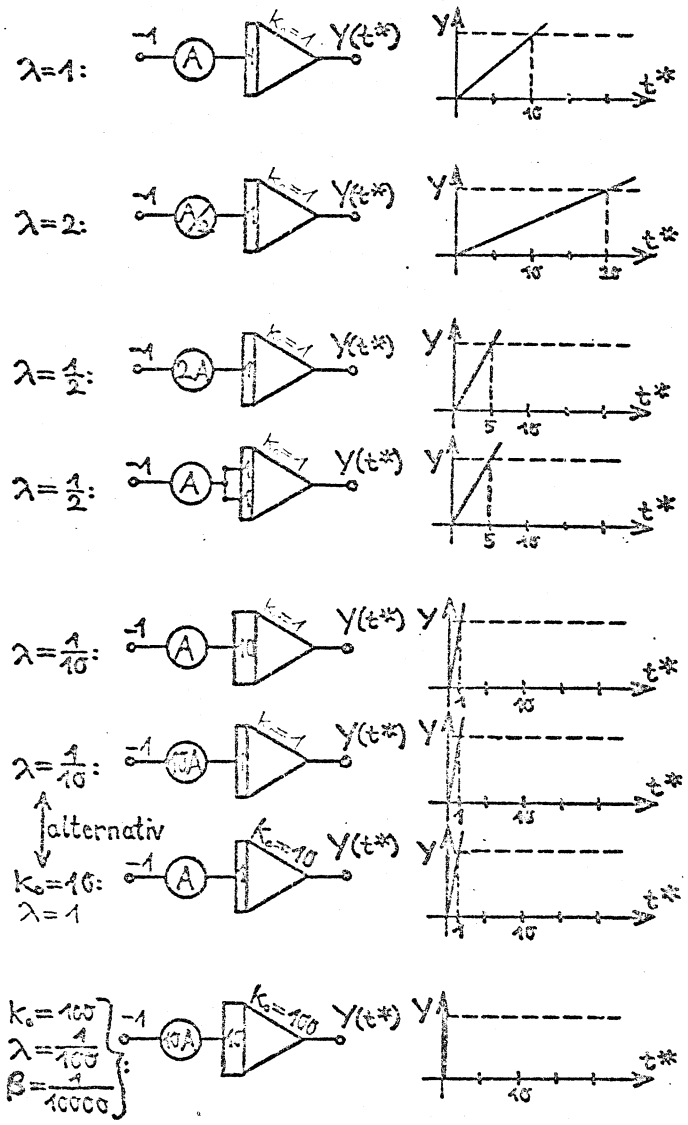


Bild 5.4 zeigt den Zusammenhang zwischen Problemzeit t und Maschinenzeit t^* gemäß der Normierungsformel $t^* = \frac{\lambda}{k_o} \cdot t$

Eine höhere Rechengeschwindigkeit kann auch erreicht werden, wenn bei allen Integrierern die Wertigkeiten der Eingänge um denselben Faktor erhöht werden. Diese Möglichkeit der Veränderung der Geschwindigkeit ist vor allem bei Echtzeitrechnung von Bedeutung. Die Wirkung ist dieselbe wie wenn der k_o-Faktor um denselben Bruchteil erhöht wird. Sei die Eingangsbewertung eines Eingangs c_i, so kann dafür normalerweise nur c_i=1 oder c_i=10 gesetzt werden, so daß durch den Einfluß der Eingangsbewertung ein zusätzlicher Faktor 10 in die Normierung kommt, der allerdings im Faktor λ eingeht. Es gilt $\lambda \sim \frac{1}{c_i}$.
Man kann so die Zeitnormierung durch k_o und λ auffassen als eine Veränderung der Eingangswertigkeit aller Integrierer. Das kommt daher, daß die Rechengeschwindigkeit der Integrierer und damit auch die Rechergeschwindigkeit der ganzen Analogschaltung alleine von der Wertigkeit der Eingänge und der Kapazität des Integriererkondensators abhängt. Beim Integrierer bewirkt eine höhere Eingangswertigkeit nur eine Erhöhung der Rechengeschwindigkeit, während z. B. beim Summierer die eingegebene Rechengröße über den Eingang unmittelbar verstärkt wird und die entsprechend verstärkte Größe am Ausgang erscheint. Im Gegensatz zum Summierer, wo auf diese Weise unter Umständen sofort eine Übersteuerung auftreten kann, ist also beim Integrierer keine unmittelbare Übersteuerung zu befürchten, außer sie wäre aufgrund der Rechenschaltung sowieso aufgetreten. Es verschiebt sich dann nur der Zeitpunkt, wann übersteuert wird.

5.3.2.2 Der Zeitnormierungsfaktor

Die Einstellung der Koeffizienten in der Rechenschaltung kann durch den Faktor λ ermöglicht werden, der es erlaubt, jeden einzelnen Koeffizienten - innerhalb gewisser Grenzen - in den erlaubten Einstellbereich des Analogrechners zu transformieren. Er geht außerdem in die Anfangswerte der Ableitungen und in die Störfunktionen ein. Der Zeitmaßstab für die gesamte Zeitnormierung ergibt sich, wenn man durch geeignete Wahl von λ dafür sorgt, daß alle Koeffizientenwerte in den einstellbaren Bereich (möglichst Werte zwischen 1 und 0.1) transformiert werden, und durch die Wahl eines erwünschten k_o-Faktors. Da die Wahl von λ durch die Problemkoeffizienten weitgehend festgelegt ist und für k_o nur die Faktoren 1, 10 und 100 zur Verfügung stehen, ergibt sich meist von selbst der Zeitmaßstab.

Bei vorgegebenen Faktoren k_o ist die Einführung eines Maßstabsfaktors λ zur Erzielung einstellbarer Koeffizienten gleichbedeutend mit einer Angleichung der Lösungszeit des Problems an die mögliche Rechengeschwindigkeit des Analogrechners.
Für die Einstellung der Potentiometer gelten die beiden Beziehungen

$$a_i = \frac{A_i}{\lambda^n} \qquad (5.16)$$

und

$$0.1 \leqq \frac{A_i}{\lambda^n} \leqq 1 \qquad (5.17)$$

wenn gilt a_i= einzustellender Potentiometerwert,
A_i= zugehöriger Koeffizient im Problem,
n : A_i steht bei der n-ten Ableitung einer Problemgröße als Koeffizient .

Beispiel: Es sei gegeben eine einfache Integration $x(t) = A\int_0^t dt$.
Die zugehörige Rechenschaltung entspricht Bild 5.3 .

Sei $A=5\cdot10^6$: man kann jetzt z. B. wählen $\lambda = 10^7$, damit sich ein einstellbarer Koeffizientenwert ergibt, sowie $k_o=10$.
Dann folgt

$$t^* = \frac{10^7}{10}\, t = 10^6 t$$

, d. h. das Problem wird auf dem Analogrechner 10^6-mal langsamer gelöst bzw gerechnet als beim wirklichen Prozeß.

Sei $A=4\cdot10^{-3}$: hier wählt man z. B. $\lambda = 10^{-7}$ und $k_o=100$. Dann ergibt sich

$$t^* = \frac{10^{-7}}{10^2}\, t = 10^{-9} t$$

, d. h. das Problem wird 10^{-9}-mal schneller gerechnet als beim wirklichen Prozeß.

Der Spielraum für λ ist groß genug, daß in den meisten Fällen die Wahl nicht kritisch ist, sondern sogar ein ganzzahliger Wert (evtl. eine Zehnerpotenz) gewählt werden kann. Der freibleibende Spielraum kann zur Optimierung der Normierung verwendet werden, um Koeffizientenwerte zwischen 0.1 und 1 zu erhalten. In schwierigen Fällen muß man mehrmals probeweise verschiedene Werte für λ einsetzen, ehe ein Wert gefunden wird, der alle geforderten Bedingungen erfüllt. Nur in Sonderfällen wird die Normierung ganz mißlingen, weil ein oder mehrere Werte zu klein geraten oder so groß bleiben, daß eine Übersteuerung eintritt. Die untere Grenze für die Einstellung von Potentiometern hängt von der Genauigkeit ab, in der die Koeffizienten gegeben sind.

Die obere Grenze für die Koeffizienten kann mit Hilfe von Summiereingängen mit der Wertigkeit 10 bis zum Wert 10 gestreckt werden, jedoch ist dabei die Gefahr von Übersteuerungen dann groß. Ergeben sich im unvollständig normierten Schaltplan nicht einstellbare Potentiometerwerte, so kann man diese in einen einstellbaren Wert und eine Zehnerpotenz aufspalten, die als Vergrößerung des k_o-Faktors in die Schaltung eingeht. Dieser zusätzliche Faktor kann auch auf verschiedene k_o-Werte verteilt werden. In einem solchen Fall wird dann trotzdem in Echtzeit gerechnet. Formelmäßig ergibt sich dafür folgender Zusammenhang. Man kann damit ein unterschiedliches λ an verschiedenen Integrierern wählen, wenn gilt

$$\frac{\lambda_1}{k_{o_1}} = \frac{\lambda_2}{k_{o_2}} = \text{konstant für alle Integrierer} \qquad (5.18)$$

Das bedeutet, daß man λ und k_o für einen einzelnen Integrierer ändern darf, ohne die Rechenschaltung und die Normierung inhaltlich zu verändern, wenn dadurch das Verhältnis von λ zu k_o nicht verändert wird.

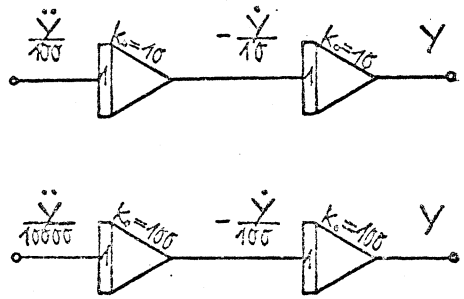


Bild 5.5 zeigt die Verwendung von Gleichung (5.18)

Wird eine Übersteuerung angezeigt, so kann man versuchen den dafür verantwortlichen Koeffizienten z. B. zu halbieren und entsprechend überall, wo er weiter auftaucht, den Wert zu korrigieren.

Mit Hilfe der Eingangsbewertungen lassen sich auch Faktoren 0.1 herstellen, wenn die ganze Rechenschaltung so ausgelegt ist, daß sonst nur 10-er-Eingänge bei Integrierern benutzt werden. Man kann dann z. B. statt eines 10-er-Eingangs einen 1-er-Eingang benutzen und stattdessen den einzustellenden Potentiometerwert vor dem betreffenden Integrierer um den Faktor 10 größer wählen.

Für eine sinnvolle Lösung gilt die notwendige Bedingung, daß alle Einzelgleichungen und alle Zeitmaßstäbe der Integrierer mit demselben Verhältnis $\beta = \frac{\lambda}{k_o}$ bemessen werden, wobei allerdings λ und k_o gemäß Gleichung (5.18) variabel sind.

Bei Problemen, die die Zeit als unabhängige Variable besitzen, deuten sehr große Koeffizienten bei Integrierern auf schnelle und kleine Koeffizienten auf langsame Zustandsänderungen hin. Als Korrektur ist dann im ersten Fall eine Zeitdehnung durch Verkleinerung der Koeffizienten, im zweiten Fall eine Zeitraffung durch Vergrößern der Koeffizienten notwendig.

Soll eine Lösung am Analogrechner in Echtzeit bearbeitet werden, so besteht die notwendige Bedingung, daß die Normierungsfaktoren λ und k_o einander gleich sein sollen. Da für k_o nur die Werte 1, 10 und 100 zur Verfügung stehen, kann für die Echtzeit-Verarbeitung auch nur ein Wert 1, 10 oder 100 für λ gewählt werden. Probleme, bei denen sich die Bedingung $\lambda = k_o$ nicht erfüllen läßt, kann nicht in Echtzeit gerechnet werden.

Nach der Durchführung der Amplituden- und Zeitnormierung müssen alle Koeffizienten vor Potentiometern und Summierern, Multiplizierern usw dimensionslos sein, während bei den Koeffizienten, die vor Integriereingängen auftreten, die reziproke Dimension der unabhängigen Problemvariablen stehen muß (z. B. $[1/sec]$).

Nach der Normierung läßt sich in der normierten Rechenschaltung der Einfluß einzelner Glieder auf die Ergebnisse abschätzen. Ausdrücke mit sehr kleinen Koeffizienten üben normalerweise einen geringen Einfluß aus. Oft ist dabei die Dauer der Rechnung an den größten Koeffizienten der ersten Ableitungen ablesbar.

Nach der Normierung und Rechnung müssen die Ergebnisse wieder entnormiert werden, d.h. mit Hilfe der Maßstabsfaktoren der Normierung müssen die ursprünglichen Größenverhältnisse wieder hergestellt werden.

Eine besondere Art der Zeitnormierung muß dann vorgenommen werden, wenn im Problem die unabhängige Variable nicht die Zeit sondern eine andere Größe ist. Die Integration nach einer anderen Größe als der Zeit ist leicht auszuführen, solange zwischen der unabhängigen Variablen und der (Maschinen-) Zeit ein linearer Zusammenhang besteht und gilt - mit x als unabhängiger Variablen -

$$t^* = \beta \cdot x \qquad (5.19)$$

Dann erhält man für die Integration den Zusammenhang

$$\int y \cdot dx = 1/\beta \int y \cdot dt^* \qquad (5.20)$$

Die Zuordnung zwischen x und t^* gemäß Gleichung (5.19) erfolgt in derselben Weise, wie bei der Umnormierung der Problemzeit t in die Maschinenzeit t^*. Dabei ist der Normierungsfaktor β mit einer Dimension behaftet. β muß so gewählt werden, daß der ganze Rechenbereich der unabhängigen Problemvariablen genau in das gesamte Rechenzeitintervall T am Analogrechner abgebildet wird.

Ebenso geht der Faktor β in die Ableitungen ein, so daß gilt

$$\frac{d^n y(x)}{dx^n} = \beta^n \frac{d^n y(t^*)}{dt^{*n}} \qquad (5.21)$$

Eine Normierung nach Abweichungen der unabhängigen bzw der abhängigen Variablen von bestimmten Festwerten kann entsprechend zu den Gleichungen (5.5), (5.6) bzw (5.8) vorgenommen werden.

Gilt statt Gleichung (5.19), daß die unabhängige Variable x nicht die Größe ist, nach der integriert werden soll, sondern soll nach einer Größe $z = f(x)$ integriert werden, so kann keine sinnvolle und ausführbare Normierung mehr vorgenommen werden. Eventuell ist es dann möglich durch Diskretisierungsmaßnahmen eine Näherungslösung zu erhalten.

5.4 Einige Beispiele zur Normierung

5.4.1 Die ungedämpfte harmonische Schwingung

Eine ungedämpfte harmonische Schwingung liegt in allgemeinster Form vor durch die Gleichung

$$x(t) = a\cdot\sin(\omega\cdot t+\varphi) \qquad (5.22)$$

Es sind dabei

a = Amplitude der Schwingung
ω = Frequenz
$T = \frac{2\pi}{\omega}$ = Schwingungsperiode
φ = Phasenverschiebung

Außerdem existieren zwei Anfangsbedingungen der Form

$$x(0) = x_o = a\cdot\sin\varphi$$
$$\dot{x}(0) = \dot{x}_o = a\omega\cdot\cos\varphi \qquad (5.23)$$

Insgesamt erhält man nach dem Ableiten von x die Differentialgleichung

$$\ddot{x} + \omega^2\cdot x = 0 \qquad (5.24)$$

Man kann zu diesem System bestehend aus der Gleichung (5.24) und den Anfangsbedingungen (5.23) eine qualitative Rechenschaltung herstellen wie sie Bild (5.6) zeigt. Darin wurde der Faktor ω^2 auf zwei Potentiometer aufgeteilt. Auf diese Weise erhält man am Ausgang des ersten Integrierers $-\dot{x}/\omega = -a\cos(\omega\cdot t+\varphi)$ und nach der Multiplikation mit ω am Eingang des zweiten Integrierers $-\dot{x}$. Am Ausgang des zweiten Integrierers steht x. Multipliziert mit ω erhält man $-\ddot{x}/\omega$ am Eingang des Umkehrers und $\ddot{x}/\omega$ am Eingang des ersten Integrierers. Der Anfangswert am ersten Integrierer ist in dieser Schaltung nicht $\dot{x}_o$ sondern $\dot{x}_o/\omega = a\cdot\cos\varphi$.

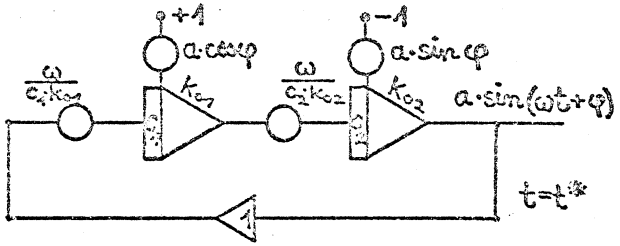


Bild 5.7 zeigt die qualitative Rechenschaltung für den Sinus

In Bild 5.7 enthalten die Koeffizienten der Potentiometer vor den Integriereingängen noch im Nenner Faktoren c_i und k_{o_i} . Trotzdem entsteht ein Sinus in Echtzeit, da c_i und k_{o_i} auch am Integrierer selbst jeweils als Faktor auftreten, so daß insgesamt gilt $\lambda/k_o = 1$.

Als konkretes Beispiel einer ungedämpften Schwingung sei nun die Schwingung eines Feder-Masse-Systems betrachtet.

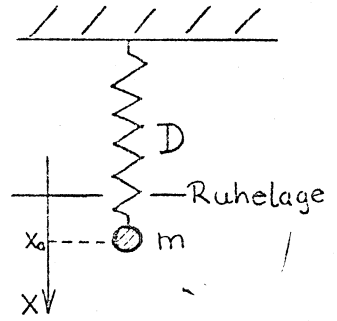


Bild 5.8 Ungedämpfte Feder-Masse-Schwingung

Es gelte:

$$m = 0.05\ [kg]$$
$$D = 45\ [kg/sec^2]$$

Das System wird beschrieben durch die Differentialgleichung für die wirkenden Kräfte

$$m\ddot{x} + Dx = 0 \quad \text{bzw} \quad \ddot{x} = -900x \qquad (5.25)$$

und die Anfangsbedingungen des ausgelenkten Systems

$$\dot{x}(0) = \dot{x}_o = 0;$$
$$x(0) = x_o = 0.1\ [m]. \qquad (5.26)$$

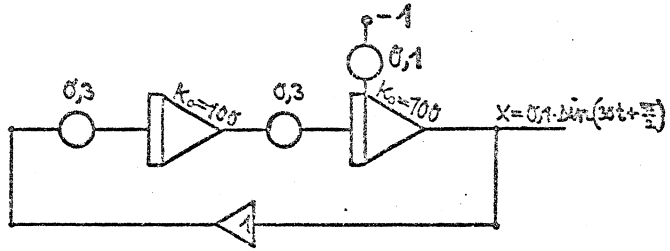


Bild 5.9 zeigt den qualitativen unnormierten Schaltplan der ungedämpften Feder-Masse-Schwingung

Als Lösung der Gleichungen (5.24) und (5.25) erhält man $x = x_o\cdot\sin(\omega\cdot t+\frac{\pi}{2})$. Diese Funktion besitzt die Frequenz $\omega = \sqrt{\frac{D}{m}}$. Man erhält aus den gegebenen Größen $\omega = 30\ [1/sec]$ und damit die Schwingungszeit $T = \frac{2\pi}{\omega} \approx 0.2\ [sec]$. Damit lautet die Lösung $x = x_o\cdot\sin(30t+\frac{\pi}{2})$.

Amplitudennormierung:

Für die maximalen Amplituden gelten folgende Abschätzungen:

$|x|_{max} = x_o = 0.1;\ |\dot{x}|_{max} = x_o\omega = 3;\ |\ddot{x}|_{max} = x_o\omega^2 = 90.$ (5.27)

Die Normierungsgleichung zu (5.25) lautet allgemein geschrieben

$$\ddot{X}\,\frac{|\ddot{x}|_{max}}{|\ddot{x}|_{max}} = -900\,\frac{|x|_{max}}{|\ddot{x}|_{max}}\,x \quad \text{bzw} \quad \text{mit den normierten Größen } \ddot{X}, X$$

$$\ddot{X} = -900\,\frac{|x|_{max}}{|\ddot{x}|_{max}}\,X \qquad (5.28)$$

Setzt man die gegebenen Werte ein, so folgt daraus

$$\ddot{X} = -X \qquad (5.29)$$

mit den normierten Anfangswerten

$$X_o = x_o/x_o = 1;\quad \dot{X}_o = 0\ . \qquad (5.30)$$

Dadurch fallen im qualitativen Bild der Rechenschaltung die beiden Koeffizientenpotentiometer mit den Werten ω heraus. Andererseits müssen zwei neue Potentiometer vor den Integriereringängen angebracht werden, die so bemessen sind, daß am Eingang jedes Integrierers jeweils eine der normierten Größen $\ddot{X}$, $\dot{X}$ erscheint. Dann ist nämlich gesichert, daß es in der Schaltung keine Übersteuerungen gibt. Entsprechendes gilt dann, wenn ein Summierer in der Schaltung enthalten ist, für dessen Eingangswerte. In der amplitudennormierten Schaltung erscheinen in diesem Fall - mit derselben Amplitude vom Betrag 1 und derselben Frequenz - alle drei Größen $\ddot{X}$, $\dot{X}$ und X mit identischen Schaubildern am Oszillograph.

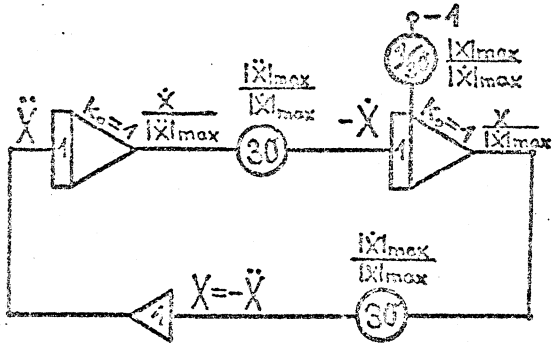


Bild 5.10 zeigt die amplitudennormierte Rechenschaltung

Die Koeffizienten der beiden Potentiometer haben zwar nun eine andere Interpretation gewonnen, aber sie sind von derselben Größe geblieben wie vor der Amplitudennormierung. Das ist deshalb der Fall, weil bei allen normierten Rechengrößen die exakten Amplitudenwerte als Maximalwerte verwendet wurden. Dagegen hat sich der Wert des Anfangswertpotentiometers verändert, da am Ausgang des zweiten Integrierers nicht mehr x erscheint sondern der normierte Wert $\frac{x}{|\ddot{x}|_{max}} = X\frac{|\ddot{x}|_{max}}{|\ddot{x}|_{max}} = X/30$. Entsprechend heißt der neue "Anfangswert" $x_o/30 = 1/30$.

Zeitnormierung:

Die Koeffizientenwerte 30 sind nicht direkt einstellbar, deshalb muß eine Zeitnormierung vorgenommen werden. Dafür empfiehlt sich z. B. eine Zeitdehnung um den Faktor ß=10, d. h. es soll gelten

$t^* = 10\cdot t$ bzw $\lambda = 10\cdot k_o$ (5.30)

Man kann also wählen $\lambda = 10$, $k_o = 1$. Dann erhält man

$X(t)=X(t^*);\ \dot{X}(t)=10\dot{X}(t^*);\ \ddot{X}(t)=100\ddot{X}(t^*)$ und

$|x(t^*)|_{max}=x_o;\ |\dot{x}(t^*)|_{max}=x_o\omega^*;\ |\ddot{x}(t^*)|_{max}=x_o\omega^{*2}$

bzw als neue zeit- und amplitudennormierte Differentialgleichung

$$100\ddot{X}(t^*) = -900\,\frac{|x(t^*)|_{max}}{|\ddot{x}(t^*)|_{max}}\,X(t^*) \quad \text{oder} \quad \ddot{X}(t^*) = -X(t^*) \qquad (5.31)$$

Die neuen Anfangsbedingungen lauten $\dot{X}_o = 0;\ X_o = 1$ (5.32)

und als Einstellwert des Anfangswertpotentiometers

$$X_o\,\frac{|x(t^*)|_{max}}{|\dot{x}(t^*)|_{max}} = 1/3.$$

Damit werden zwei neue Potentiometer benötigt, die mit je einem Wert $\frac{1}{10} = \frac{1}{\sqrt{100}}$ beschaltet werden müssen und jeweils vor einen der beiden Integrierer gelegt werden müssen, damit der Faktor ß=10 entsprechend der Zahl der Ableitungen in die Rechenschaltung eingeht. Da nun je ein Potentiometer mit dem nicht einstellbaren Wert 30 und mit den Wert 0.1 hintereinander in der Schaltung liegen würden, faßt man sie zusammen zum Faktor 3 und verwirklicht das Ganze, indem man den Potentiometer auf den Wert 0.3 einstellt und auf einen 10-er-Eingang führt.

Als fertignormierte Lösung erhält man die Funktion $X=\sin(3t^*+\frac{\pi}{2})$.

Die Schwingungsdauer beträgt dann $T = \frac{2\pi}{3} \approx 2$ [sec].

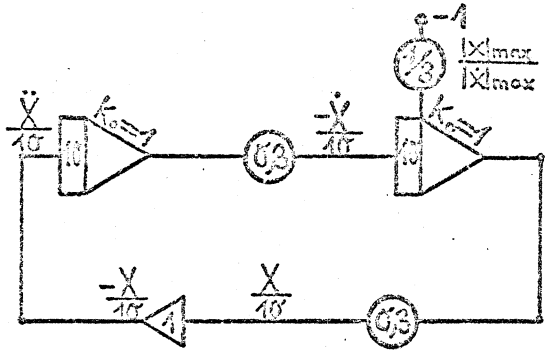
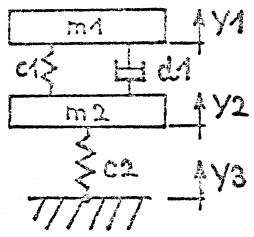


Bild 5.11 zeigt den vollständig normierten Schaltplan für die Feder-Masse-Schwingung gemäß Gleichung (5.25), (5.26)

5.4.2 Beispiel eines gekoppelten Differentialgleichungssystems

Am Beispiel der Automobilfederung soll ein gekoppeltes DGL-system vorgeführt werden. Die Schwingungsberechnung bei Kraftfahrzeugen bereitet große Schwierigkeiten. Man kann das Kraftfahrzeug als ein Gebilde betrachten, das aus drei Massen besteht (Automasse, Masse der Vorderräder + Vorderachse und Masse der Hinterräder + Hinterachse), die durch drei Feder- und zwei Dämpfungsglieder miteinander verknüpft sind. Werden die Nichtlinearitäten der Dämpfungscharakteristiken berücksichtigt, so erhält man insgesamt ein System nichtlinearer DGL's 8-ter Ordnung. Das ist für unsere Verhältnisse noch zu kompliziert, so daß es sich empfiehlt, ein nur grob angenähertes System der Automobilfederung zu verwenden, das die wirklichen Verhältnisse zwar im Prinzip widerspiegelt, aber nur mit ungenauen Ergebnissen.



Die Automobilfederung wird nun beschrieben als Gebilde bestehend aus den Massen m_1 (=Automasse) und m_2 (=Masse der vier Räder + Achsen) und den sie verbindenden Federungsgliedern c_1 (Auto gegen Räder) und c_2 (Räder gegen Boden) sowie dem Dämpfungsglied d_1 (Auto gegen Räder).

Bild 5.12 zeigt schematisch die Automobilfederung

Die Störung: Größe y_3 kann z. B. als sprungartige Störung (Stoß durch Bodenunebenheit!) oder als sinusförmige Störung auftreten. Zu Beginn sei das ganze System in Ruhe, d. h. alle Anfangswerte seien Null. Das bedeutet unter anderem, daß bei allen Normierungsschritten die Anfangswerte unberücksichtigt bleiben können. Andererseits ist es vernünftig, das System in Ruhelage zu setzen, ehe die zu beobachtenden Störungen wirksam werden, da dann nur der Einfluß der Störglieder für sich untersucht werden kann. Die Beschreibung dieser Automobilfederung ist gegeben durch Gleichung (4.11).

Durch Messung und Abschätzung seien die folgenden Zahlenwerte gegeben (sie stammen aus der Literatur)

$$m_1 = 500\ [kg]\ ;\ m_2 = 40\ [kg]\ ;\ c_1 = 80000\ [kg/sec^2];$$
$$d_1 = 1100\ [kg/sec]\ ;\ c_2 = 400000\ [kg/sec^2]. \qquad (5.33)$$

Diese Werte werden in Gleichung (4.11) eingesetzt, dann erhält man folgendes Differentialgleichungssystem

$$\ddot{y}_1 = -2{,}2(\dot{y}_1-\dot{y}_2)\ -160(y_1-y_2)$$
$$\ddot{y}_2 = -27{,}5(\dot{y}_2-\dot{y}_1)\ -2000(y_2-y_1)\ -10000(y_2-y_3) \qquad (5.34)$$

Zu diesem Differentialgleichungssystem kann man sofort den qualitativen unnormierten Schaltplan herstellen. Bild 5.13 zeigt den unnormierten Schaltplan, der zunächst keinen einzigen einstellbaren Koeffizientenwert bei einem Potentiometer enthält, d. h. die Normierung ist auf jeden Fall notwendig. Die Funktion y_3 ist durch einen Potentiometer symbolisiert; über ihren Wert ist damit noch nichts ausgesagt.

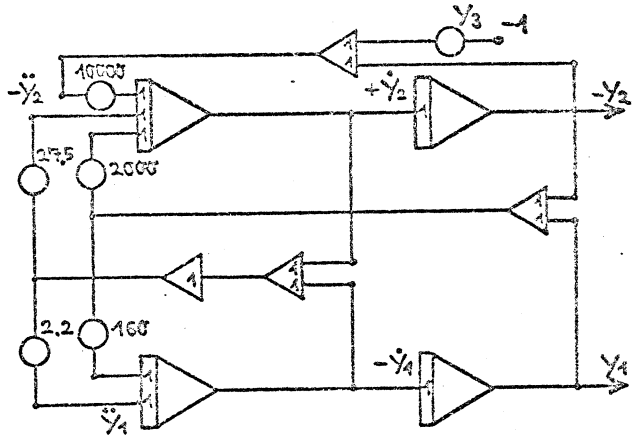


Bild 5.13 zeigt den qualitativen Schaltplan für die Autofederung

Amplitudennormierung:

Die Abschätzung der Maximalwerte ist nicht ganz einfach. Deshalb empfiehlt es sich - zumindest für den Anfang, d. h. evtl. für einen Probelauf -, alle Maximalwerte vorsichtig und lieber zu groß zu wählen und diese dann eventuell später noch zu verbessern. Geht man davon aus, daß das betrachtete Automobil erträgliche Federungseigenschaften aufweisen soll, so kann man z. B. folgende Abschätzung treffen : alle Größen y_1, y_2 und y_3 sowie auftretende Ableitungen, das sind also die Ausschläge in senkrechter Richtung, die Geschwindigkeiten in senkrechter Richtung und die Beschleunigungen, die am Auto auftreten können, ebenfalls in senkrechter Richtung, sollen betragsmäßig nicht größer als 0.8 werden. Damit kann man zunächst einmal setzen

$$|y_1|_{max} = |y_2|_{max} = |y_3|_{max} = 0.8\,[m];$$
$$|\dot{y}_1|_{max} = |\dot{y}_2|_{max} = 0.8\,[m/sec];$$
$$|\ddot{y}_1|_{max} = |\ddot{y}_2|_{max} = 0.8\,[m/sec^2]\,. \qquad (5.35)$$

Nun heben sich alle Normierungsfaktoren für die Amplitudennormierung gegenseitig heraus und ihr Einfluß auf das System (5.34) ist insgesamt Null. Stellt sich bei der Lösung heraus, daß eine Korrektur der Maximalwerte nicht unbedingt notwendig ist, oder wieder in der Weise erfolgen kann, daß alle Maximalwerte vom selben Betrag gewählt werden können, so ist dies von großem Vorteil, da dann alle Problem- bzw Rechengrößen unmittelbar in ihren gegenseitigen Größenverhältnissen im Schaltplan erscheinen und deshalb auch keine Umrechnungen der Koeffizientenwerte der Potentiometer nötig werden.

Zeitnormierung:

Wie bereits erwähnt, können die vorliegenden Potentiometerwerte nicht eingestellt werden. Der größte in der Rechenschaltung vorkommende Koeffizientenwert ist 10000. Es liegt nahe,den Potentiometerwert 1 abzuspalten und zu setzen $k_o^2/\lambda^2 = 10000$. Dann erhält man $k_o/\lambda = 100$. Dieser Quotientenwert läßt sich einstellen durch Werte $k_o = 10$ und $\lambda = \frac{1}{10}$. Es ist allerdings günstiger, in Echtzeit zu rechnen und z. B. zu wählen $k_o = 10$ und $\lambda = 10$ (oder $k_o = 100$ und $\lambda = 100$). Dann erscheinen in der Rechenschaltung an allen Integrierern k_o-Faktoren 10 und die Eingänge können z. B. mit 10 bewertet werden (es gilt $\lambda = 10 = \frac{100}{c}$, wenn c die 10-er-Eingangsbewertung ist). Es muß in diesem Zusammenhang daran erinnert werden, daß der Faktor k_o/λ bei jeder Ableitung einer Rechengröße einmal im Ableitungsausdruck auftritt, und daß daher die Verteilung der beiden Größen auf jeden Integrierer herkommt. Nun werden folgende Koeffizientenwerte im Schaltplan auftreten: 0.022; 0.016; 0.275; 0.2; 1. Der Potentiometer mit dem Wert 1 kann aus der Schaltung gestrichen werden, dagegen sind die beiden Potentiometerwerte 0.022 und 0.016 kleiner als es erwünscht ist. Um die Einstellwerte zu verbessern, gibt es mehrere Wege. Es ist z. B. möglich, zwei weitere Potentiometer in die Rechenschaltung einzubauen, die den Wert 0.1 tragen und je einen davon vor die betrachteten Koeffizientenpotentiometer zu legen, die dann auf die Werte 0.22 bzw 0.16 eingestellt werden. Einfacher ist es, die beiden Potentiometer auf die Werte 0.22 bzw 0.16 einzustellen und nicht auf Zehner- sondern auf Einer-Eingänge des Integrierers zu führen. Dadurch wird λ nicht verändert (siehe z. B. Bild 5.7 und den Einfluß der Faktoren c_i, die in λ enthalten sind). Würde man stattdessen am Ausgang des Integrierers ein Potentiometer mit dem Wert 0.1 einfügen und Zehnereingänge verwenden, so würde der Integrierer in einem anderen Zeitmaßstab arbeiten; das Ergebnis wäre verfälscht.

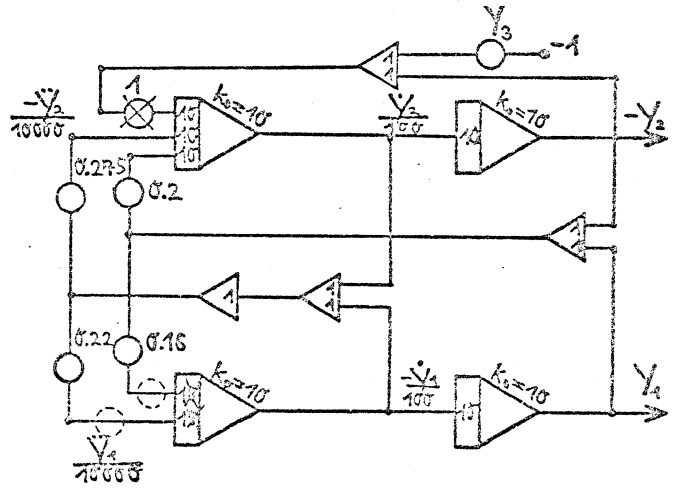


Bild 5.14 zeigt den vollständig normierten Schaltplan für das gekoppelte System bei der Automobilfederung

Bei schwierigeren Aufgaben wird auch die Normierung unter Umständen wesentlich schwieriger. Genauere Ergebnisse der Rechnung werden also mit größerem Aufwand bei der Normierung erkauft.

6 Gesamtaufbau des Analogrechners

Der Analogrechner besteht aus den Rechenelementen, die am Analogprogrammierfeld zu beliebigen Rechenschaltungen verknüpft werden können und den Ausgabegeräten, auf denen die Lösungsfunktionen dargestellt werden können. Ausgabegeräte sind Koordinatenschreiber, Oszillograph und Digitalvoltmeter. Gesteuert wird der Analogrechner von einem zentralen Bediengerät, oder von einem Digitalzusatz auf dem sich spezielle Programmabläufe programmieren lassen, oder völlig extern von einem Digitalrechner (in einer Hybridrechenanlage).

6.1 Das Bediengerät des Analogrechners

Die Funktionen des Bediengerätes teilen sich in 3 Gruppen auf: 1. Anwahloperationen, 2. Zeitwahl, 3. Betriebsartensteuerung.

Anwahloperationen

Alle Rechenelemente haben eine Adresse. Sie besteht erstens aus einer einstelligen Nummer des Rechenelements, zweitens aus der Feldnummer des Analogprogrammierfeldes in den das Rechenelement liegt und drittens aus der Rechnernummer, wo sich das Analogprogrammierfeld befindet, sowie viertens aus der Art des Rechenelements.
Durch die An-wahl wird der Ausgang des Rechenelements auf eine zentrale Meßleitung geschaltet und dadurch mit dem Digitalvoltmeter verbunden. Als zusätzliche Kontrolle erscheint am Digitalvoltmeter ein Kennbuchstabe für die Art der Verwendung (S=Summierer, I = Inte—grierer, M = Multiplizierer, usw). Insgesamt können 600 Adressen angewählt werden und damit an allen Rechenelementen Messungen zur Kontrolle der Rechenschaltung durchgeführt werden.

Zeitwahl

Ein zentraler Taktgenerator (100 kHz) steuert alle Abläufe. Aus diesem Takt werden durch Frequenzteilung die übrigenZeiten erzeugt. Die Dauer der einzelnen Rechenphasen: Pause, Rechnen, Halt können am Bediengerät digital eingestellt werden. Dazu ist ein Grundtakt einstellbar (entspricht gleichzeitig,wenn nicht anders programmiert, der Dauer der Haltphase) und vielfache des Grundtaktes GT als T_R (Dauer der Rechenphase) und als T_p (Dauer der Pause).

Betriebsartensteuerung.

Grundsätzlich besteht jede Rechnung aus den 3 Phasen: 1.Pause in der die Integrierer die Anfangswerte übernehmen(muß mindestens $T_p \geq R \cdot C$ sein), 2. Rechnen, in der, der eigentliche Rechengang erfolgt und 3. Halt, eine Rechenphase in der, der augenblickliche Zustand z.B. für Meßzwecke erhalten bleibt (Integrierer gehen in den Zustand Speichern), sodaß aus der Phase Halt wieder in die Phase Rechnen übergegangen werden kann.

Zusätzlich zu diesen Möglichkeiten ist am Analogprogrammierfeld vorgesehen, Integrierer in einer inversen Phase zu betreiben. Zu diesem Zweck gibt es einen 2. Grundtakt GT2 von dem sich die einstellbaren Zeiten Rechen 2 und Pause 2 ableiten.

Dadurch ist es möglich einen Gesamtablauf in der Weise: Pause 1, Rechnen 1, Halt 1, Pause 2, Rechnen 2, Halt 2 zu erzeugen und in der zweiten Rechenphase Rechnen 2 z.B. Werte zu speichern, Kriterien zu prüfen oder neue Anfangswerte für eine weitere Rechenphase zu ermitteln.

Die Betriebsartensteuerung läßt sich nochmals unterteilen in Vorbereitungsbefehle und Ausführungsbefehle. Vorbereitungsbefehle sind: Dauerrechnen, Rechnen mit Halt, repetierend Rechnen, iterativ automatisch. Diese Befehle werden durch den Ausführungsbefehl Rechnen gestartet und durch Halt oder Pause abgebrochen sowie durch Weiter fortgestartet.

§ 7 Parameteroptimierung

7.1 Das Problem der Optimierung

Die Optimierung von Lösungen und Einstellwerten ist eines der wichtigsten Probleme und eine Notwendigkeit bei der Bearbeitung mathematischer oder technischer Systeme und Prozesse. Die Aufgabe der Optimierung besteht, allgemein ausgedrückt, darin, die Parameter $x_1, x_2, \ldots, x_n$, welche ein Objekt und dessen Ausprägung festlegen, so zu bestimmen, daß die Ausgangsgröße $Q(x_1,\ldots,x_n)$ des Objekts einen Extremwert, also ein Minimum oder ein Maximum, annimmt. Das verlangt die Wahl der Systemparameter in einer Weise, daß die Leistung des beschriebenen Systems dem möglichen Optimum so nahe als möglich kommt. Das Optimum wird mit Hilfe der Ausgangsgröße Q ermittelt, d.h. Q muß so gewählt werden, daß es einen Extremwert annimmt, wenn das zu optimierende System optimal arbeitet. Die Funktion $Q(x_1,\ldots,x_n)$ muß demgemäß von allen n Parametern des Systems abhängen und kann als Erfolgsfunktion oder Verhaltens-Kriteriumsfunktion bezeichnet werden.

Ist Q nicht zeitabhängig, handelt es sich bei dem beschriebenen System also um ein statisches Objekt, so spricht man von statischen Optimierungsaufgaben. Dabei kann es sich z. B. um algebraische Gleichungen oder Ungleichungen, Anfangswertprobleme usw. handeln.

Ist die Funktion Q zeitabhängig, d. h. handelt es sich um ein zu optimierendes dynamisches System, so wird Q durch einen zeitlichen Integrationsprozeß aus einer anderen Funktion $z(x_1,\ldots,x_n)$ gewonnen und es gilt

$$Q(x_1, \ldots, x_n) = \int_0^T z(x_1, \ldots, x_n)\, dt \qquad (7.1)$$

Dann spricht man von dynamischen, d. h. durch Differentialgleichungen beschriebenen Optimierungsaufgaben. Die meisten Steuerungs- und Leitsystemaufgaben, die Modellbildung dynamischer Systeme usw fallen unter die dynamische Optimierung.

Es kann z. B. verlangt sein, daß die Kosten eines Prozesses verringert werden, daß der Energieaufwand vermindert, eine zurückgelegte Strecke über bestimmte Punkte optimal geleitet oder ein Gewinn erhöht wird. Ebenso kann es notwendig sein, eine Satellitenbewegung oder anderes in einer Analog- oder Digitalrechnersimulation zu optimieren.

Die wichtigen Anwendungsgebiete für Optimierungsaufgaben sind:

1. mathematische Optimierungsaufgaben

Dabei handelt es sich um statische Optimierung. Es ist z. B. möglich, algebraische Gleichungen nach ihren Wurzeln aufzulösen, wenn man eine Gleichung in Real- und Imaginärteile der Wurzeln zerlegt und die dann vorliegenden Gleichungen durch Parametervariation zu Null macht. Randwertaufgaben können gelöst werden, indem zunächst grobe Näherungswerte als Randwerte eingesetzt werden und durch Veränderung dieser Parameter schließlich die gesuchte oder eine gesuchte Lösung angenähert wird. Ebenso können Probleme der Variationsrechnung und anderes bearbeitet werden.

2. Modellbildung und Modelloptimierung

Oft werden Systeme, z. B. technische Prozesse, zuerst simuliert, ehe man darangeht, sie in die Wirklichkeit umzusetzen, etwa dann wenn neue Flugzeugprofile, chemische Prozesse usw getestet werden sollen oder z.B. auch bestimmte neuentwickelte oder neu zu entwickelnde Eigenschaften von Fahrzeugen, Energiesystemen usw. Vorgegeben sind dann meist bestimmte technisch-industrielle Randbedingungen, denen ein Prozeß durch optimale Wahl der Parameter angepaßt werden soll. Es können auch Prozesse, in allen verschiedenartigen Ausprägungen beobachtet und optimale Lösungen bestimmt werden, deren Auswirkungen in der Realität noch nicht bekannt sind, wie z. B. bei der Verkehrssimulation, die oft ehr dazu dient, bestimmte unerwünschte Erscheinungen zu verhindern.

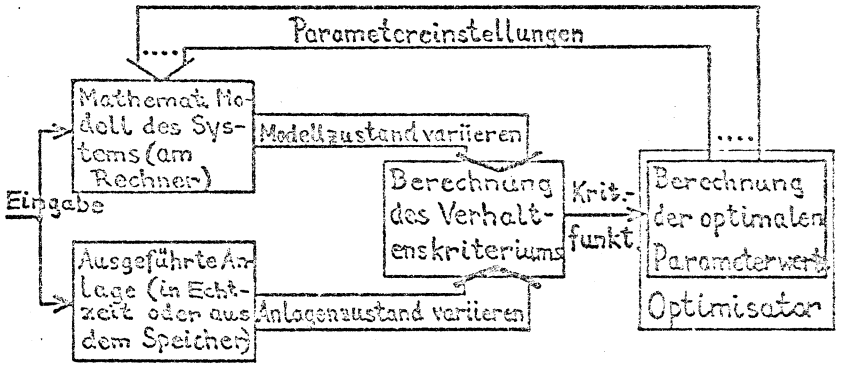


Bild 7.1 zeigt das Schema der Modelloptimierung einer dynamischen Anlage (unterer Teil darf fehlen)

3. Prozeßoptimierung

Die Optimierung laufender Prozesse bedeutet, daß eine zeitabhängig arbeitende (z. B. technische) Anlage, während sie in Betrieb ist, in einen optimalen Zustand gebracht und gehalten wird. Das ist z. B. bei nahezu allen Prozessen der Industriellen Fertigung notwendig, etwa bei der Metallverarbeitung in kontinuierlichen Walzstraßen, bei der Hochofenregulierung, bei Mischungsprozessen der chemischen Industrie (Stickstoffsynthese usw), Energiesystemen (Kraftwerke, Atommeiler usw) und anderen.

Prozeßoptimierungssysteme können häufig vor der wirklichen Ausführung erst durch eine Modellbildung und Modelloptimierung vorgetestet werden, so daß z. B. bei der Modellbildung die optimalen Parameterwerte und bei der Prozeßoptimierung mehr das Einhalten bestimmter optimaler Parameterwerte die Aufgabe ist.

Ein besonders wichtiger Spezialfall der Prozeßoptimierung sind Regelkreise. Dabei sind bestimmte Sollwerte bereits fest vorgegeben, und optimiert werden soll das Verhalten eines Systems in der Weise, daß die Abweichung von den Sollwerten (über der Zeit integriert) minimal werden soll.

Bei der Prozeßoptimierung mit Hilfe von Hybridrechnern oder hybriden Analogrechnern ist es besonders wichtig, daß alle Meßwerte der zu optimierenden Prozesse in normierter Form ins Rechnersystem eingegeben und dort verarbeitet werden müssen. Ebenso sind die Ergebniswerte, ehe sie an ein dynamisches System zurückgeliefert werden, wieder zu entnormieren und in den ursprünglichen Größenbereich der Problemgrößen zurückzutransformieren.

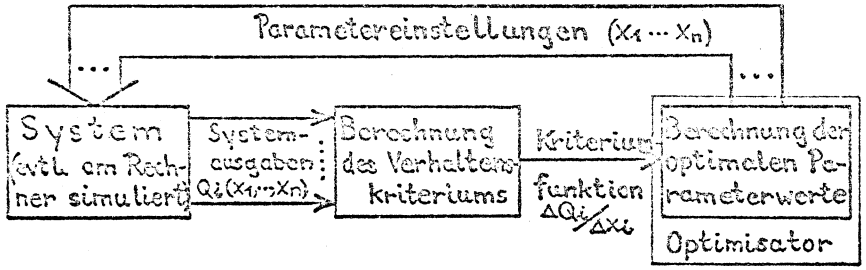


Bild 7.2 zeigt das Schema einer Prozeßoptimierungsaufgabe

Im Gegensatz zu mathematischen Optimierungsproblemen und zur Modelloptimierung darf bei Prozeßoptimierungsaufgaben nur eine Normierung nach Amplitudenwerten vorgenommen werden, da es sich dabei immer um die Bearbeitung bzw Optimierung echter Prozesse handelt, also um Echtzeitverarbeitung. Das gilt auch dann, wenn die Daten des Prozesses z. B. auf Band festgehalten sind. Allerdings ist am Analogrechner eine Normierung in Echtzeit erlaubt, d. h. wenn gilt $\beta = \frac{\lambda}{k_o} = 1$ (siehe § 5 Normierung von Rechenschaltungen).

Die verschiedenen Teile einer Optimierungsaufgabe sind:

die Entscheidung über eine geeignete Kriteriumsfunktion zur Ermittlung des bzw der Extremwerte

die Isolierung der optimalen Parameterwerte (Parameteridentifikation)

die Wahl und Mechanisierung einer Parametereinstellmethode für eine automatische Optimierung der Kriteriumsfunktion.

Die einfachste Art der Optimierung von Systemen besteht in der Regelung durch eine Person, z. B. durch einen Operateur am Rechner, der durch Bedienung eines Hebels oder Druckknopfes oder einer Fußtaste usw den Prozeß regelt bzw optimiert. Einfache Prozesse dieser Art liegen vor bei der Einstellung eines Radiogerätes, in der Optimierung des Verkehrsflusses durch Rundfunkdurchsagen, Beschilderung und Reaktion der Verkehrsteilnehmer, die evtl. eine Nebenstrecke befahren, in der Gestaltung von Zeitplänen (Eintragen, wer wann am Analogrechner arbeitet), in der Betätigung von Hebeln oder Druckknöpfen im Maschinenleitstand nach dem Ablesen von gewissen Einstellwerten (Druck, Temperatur usw) an den Meßgeräten.

Auch am Analogrechner oder am Terminal eines Digitalrechners kann ein Operateur versuchen, durch systematisches Testen der Systemparameter eine optimale Lösung für das Optimierungsproblem zu finden. Es sind oft sogar noch Systeme mit zwei Parametern per Hand lösbar, indem man iterativ die Lösung annähert. Soll jeder Parameter nur 10 verschiedene Werte annehmen dürfen, so ist allerdings bei zwei Parametern bereits eine 10·10-Werte-Matrix abzuarbeiten, d. h. es sind 100 Prozeßabläufe bzw Rechnerläufe notwendig, um das Optimum zu bestimmen. Es ist klar, daß spätestens für Systeme mit mehr als zwei Parametern die Lösung

nur noch mit Hilfe automatischer Verfahren bestimmt werden kann.

Es existieren eine Reihe von Optimierungsmethoden,die eine erfolgreiche Behandlung von Optimierungsaufgaben auf dem Digitalrechner ermöglichen. Falls eine große Stellengenauigkeit verlangt wird und die Zahl der Variablen und der Nebenbedingungen sehr groß ist, dann kommt zur Lösung nur ein Digitalrechner in Frage. Braucht die Lösung dagegen nicht genauer als auf 2 oder 3 Stellen vorzuliegen und soll unmittelbar zu jeder Parameteränderung die Lösung geliefert werden, so wird ein Analogrechner benötigt. Auf einem Analogrechner können Parameteränderungen leicht durch Änderungen der Potentiometereinstellungen eingestellt werden, die Wirkungen sind bei repetierender Rechenweise sofort feststellbar und am Oszillograph sichtbar. Mit dem Analogrechner lassen sich ohne Schwierigkeiten zeitabhängige Probleme behandeln, da er auch bei relativ raschen Parameteränderungen sofort die neue Lösung liefert.

Geräte, die die Optimierung automatisch durchführen, heißen Optimisatoren. Das zu optimierende Objekt und der Optimisator bilden ein rückgekoppeltes System. Die Bilder 7.1 und 7.2 zeigen schematisch den Ort und die Aufgabe des Optimisators im Gefüge einer Optimierungsaufgabe. Entscheidend für die Arbeitsweise eines Optimisators ist das von ihm verwendete Suchverfahren, mit dem die gewünschte optimale Lösung (meist ein Extremwert) ermittelt wird.

Bild 7.3 veranschaulicht die drei wichtigsten Suchverfahren, die bei der Parameteroptimierung am häufigsten verwendet werden. Es gibt

1. Verschiedene Gradientenverfahren:

Bei diesen Methoden wird von einem zu wählenden Anfangspunkt für das Verfahren in Richtung des steilsten Anstiegs bzw Abstiegs zu einem Extremwert des bearbeiteten Problems fortgeschritten. Das Verfahren kann bei statischen Optimierungsaufgaben als kontinuierlicher Weg,bei dynamischen Optimierungsaufgaben angenähert kontinuierlich oder durch diskrete Schritte durchgeführt werden. Notwendige Voraussetzung für die Verwendung solcher Verfahren ist die Existenz der partiellen Ableitungen der gewählten Kriteriumsfunktion nach allen Systemparametern. In Bild 7.3 ist das Gradientenverfahren als knicklose durchgezogene Linie symbolisiert.

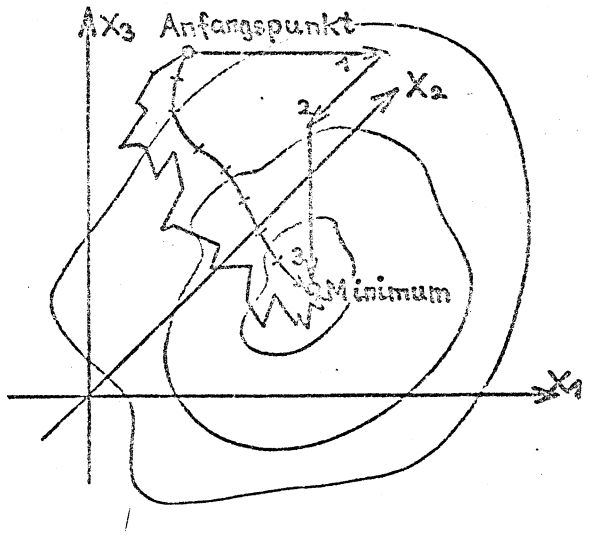


Bild 7.3 zeigt mögliche Wege im Parameterraum zur Annäherung eines Extremwertes der Zielfunktion; im Bild sind drei Parameterrichtungen dargestellt (im allgemeinen Fall n)

2. Sequentielle Einstellung der Parameter:

Dieses Verfahren wird auch Relaxationsverfahren genannt. Jeweils nur eine einzige Variable wird solange verändert, bis ein relatives Maximum bzw Minimum erreicht ist. Dabei bleiben alle anderen Parameter fest. Danach wird ein anderer Parameter systematisch solange verändert, bis auch für diesen ein relativer Extremwert des zu optimierenden Objekts erreicht ist.Diese Methode wird solange wiederholt, bis alle Parameter auf diese Weise behandelt worden sind, d. h. der gesamte Lösungsraum wird systematisch abgesucht. Bei diesem Verfahren ist die wichtigste Frage das Konvergenzverhalten, d. h. ob und wie gut das Verfahren konvergiert. In Bild 7.3 ist das Relaxationsverfahren gekennzeichnet durch die drei Pfeile 1, 2 und 3 in der Richtung der Parameterkoordinaten.

3. Zufallsbedingte Optimierungsmethoden:

Bei dieser Methode werden die Systemparameter statistisch unabhängig verändert und bei einer Ergebnisverbesserung des Wertes der Objektfunktion der Ort des Parametervektor im Parameterraum entsprechend nachgeführt. Durch dieses Verfahren ist es im Gegensatz zu den beiden ersten Methoden der Parameteroptimierung mög-

lich, jede beliebige Funktion (z. B. auch solche, deren partielle Ableitung nach den Systemparametern nicht existieren) zu optimieren, selbst wenn sie mehrere Extremwerte, Extremwerte am Rand, Unstetigkeiten und Lücken im Definitionsraum aufweist. Der Hauptnachteil des Verfahrens liegt darin, daß die Suchzeit recht groß werden kann.

Bei der Verwendung von Optimierungsmethoden liegt der Schwerpunkt auf den Gradientenverfahren, die besonders leistungsfähig sind. Vor allem eignen sich das diskrete Gradientenverfahren und das Gauß-Seidel-Suchschrittverfahren.

Oft empfiehlt es sich, verschiedene der drei erwähnten Optimierungsmethoden zu kombinieren. Man kann sich z. B. darauf beschränken, nur beim Erreichen eines relativen Extremwerts mit dem Gradientenverfahren zu arbeiten, und zwischen verschiedenen Extremwerten mit einer groben Zufallssuche zu arbeiten.

Ein Optimierungsproblem besteht aus zwei Hauptteilen, dem zu optimierenden System und dem Optimisator. Zur Simulation von Systemen, falls diese nicht direkt und in Echtzeit vorliegen, eignet sich der Analogrechner recht gut, da die Rechenelemente des Analogrechners sehr schnell arbeiten, so daß innerhalb kurzer Zeit viele Versuchsläufe möglich sind. Die Operationen des Optimisators sind vorwiegend digitaler Art. Der Optimisator wird durch Rechenalgorithmen realisiert, die mit diskreten Operationen arbeiten, welche am Digitalrechner oder am hybriden Zusatz eines Analogrechners programmiert werden müssen. Alle Optimisatoren sind demnach kombinierte Analog-Digital-Systeme.

Für eine nicht zu große Zahl von Systemparametern und bei einfachen Suchstrategien genügt für die Optimierungsaufgabe ein hybrider Analogrechner. Ab 4 oder 5 Parametern wird man damit nur noch selten auskommen. Sollen die Parameteränderungen automatisch durchgeführt werden, so wird man einen iterativen Analogrechner benützen. Man verwendet dann zwei Gruppen von Integrierern, von denen die eine mit langer Rechenzeit die Koeffizienten als Funktionen der Zeit erzeugen, während die anderen Integrierer mit kurzer Rechenzeit dazu dienen, das Gradientenverfahren (oder evtl. andere) durchzuführen. Bei großen Parameterzahlen kommt man ohne Digitalrechner nicht mehr aus. Man arbeitet dann zweckmäßigerweise z. B. mit repetierendem Rechnen, wobei die Pausenzeiten groß genug sein müssen für die Einstellung der neuen Parameterwerte.

7.2 Einige grundlegende mathematische Überlegungen

7.2.1 Die Berechnung von Extremwerten

Zur Ermittlung einer optimalen Lösung ist es notwendig, das Maximum oder Minimum bzw die Maxima und Minima einer das vorliegende System beschreibenden Kriteriumsfunktion zu bestimmen. Die möglichen Extremwerte müssen in mathematischen Ausdrücken erfaßt werden. Es ist also nötig, die Kriterien festzustellen, nach denen es möglich ist, Extremwerte zu finden, wenn die mathematische Beschreibung eines Systems vorliegt.

In der Differentialrechnung können die Extrempunkte einer einfachen Funktion $y = f(x)$, die von einer unabhängigen Variablen x abhängt, mit Hilfe der Ableitungen dieser Funktion bestimmt werden. Eine notwendige Bedingung für das Vorliegen eines relativen Extremwertes ist gegeben durch die Gleichung (7.2)

$$y' = \frac{dy}{dt}\Big|_{x_o} = 0 \qquad (7.2)$$

Dann liegt an der Stelle x_o unter Umständen ein relatives Maximum. Es gibt zwar keine relativen Extremwerte, die der Gleichung (7.2) nicht genügen, aber eine hinreichende Bedingung dafür, daß es sich tatsächlich um einen Extremwert handelt, liefert erst die Ungleichung (7.3). Und zwar erhält man ein

relatives Maximum, wenn gilt $y'' = \frac{d^2y}{dx^2}\Big|_{x_o} < 0$

relatives Minimum, wenn gilt $y'' = \frac{d^2y}{dx^2}\Big|_{x_o} > 0 \qquad (7.3)$

Gilt keines von beiden, so ist also $y''(x_o) = 0$, und es handelt sich um einen Wendepunkt mit waagerechter Tangente.

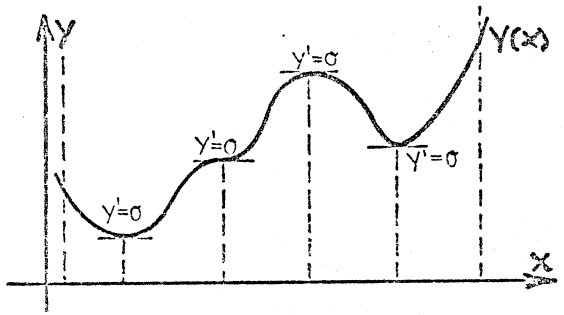


Bild 7.4 zeigt Extremwerte einer Funktion $y = f(x)$

Außerdem kann es passieren, daß das absolute Maximum oder Minimum der Funktion y=f(x) am Rand des untersuchten Definitionsbereiches liegt, d. h. an einer Stelle, für die Gleichung (7.2) nicht erfüllt ist. Deshalb erhält man aus den Gleichungen (7.2) und (7.3) nur die relativen Extremwerte der Funktion y(x) unter Ausschluß möglicher absoluter Extremwerte am Rand. Die Ränder des Definitionsbereichs müssen gesondert untersucht werden, z. B. anschließend an die Ermittlung des oder der relativen Extremwerte. Soll dies automatisch, also durch einen Optimisator untersucht werden, so werden wesentlich kompliziertere Methoden erforderlich. Bei den meisten Optimierungsaufgaben muß man davon ausgehen, daß das betrachtete System mehrere Extremwerte besitzt. Das bedeutet, daß ein am Rechner bestimmtes relatives Maximum oder Minimum von der Wahl des Anfangspunktes der Parametereinstellung abhängt. Wählt man einen anderen Anfangspunkt, so kann es also zur Ermittlung eines anderen relativen Extremwerts führen.

Normalerweise sind bei Optimierungsaufgaben Systeme zu optimieren, die von mehreren Parametern abhängen. Im allgemeinen Fall erhält man also eine Kriteriumsfunktion von der Art

$$Q = Q(x_1, \ldots, x_n) \tag{7.4}$$

Man erhält dann für das Vorhandensein von relativen Extremwerten die notwendige aber nicht hinreichende Bedingung, daß für alle partiellen Ableitung von Q nach den einzelnen Systemparametern gelten muß

$$\frac{\partial Q}{\partial x_1} = 0; \quad \frac{\partial Q}{\partial x_2} = 0; \quad \ldots \quad ; \quad \frac{\partial Q}{\partial x_n} = 0 \tag{7.5}$$

Eine ganz wesentliche Voraussetzung ist dabei wieder die Existenz aller partiellen Ableitungen von $Q(x_1, \ldots, x_n)$ nach den einzelnen Systemparametern, d. h. es können nur differenzierbare Funktionen bearbeitet werden.
Die Gleichungen (7.4) und (7.5) können etwas einfacher in Vektorschreibweise formuliert werden. Dann schreibt man

$$\vec{x} = (x_1, \ldots, x_n); \qquad Q = Q(\vec{x}) \tag{7.6}$$

kann

und man abgekürzt schreiben

$$\text{grad } Q = \nabla Q = \left(\frac{\partial Q}{\partial x_1}, \frac{\partial Q}{\partial x_2}, \ldots, \frac{\partial Q}{\partial x_n}\right) \tag{7.7}$$

Damit erhält man statt Gleichung (7.5)

$$\text{grad } Q(\vec{x}) = 0 \tag{7.8}$$

als notwendige Bedingung für relative Extremwerte von $Q(\vec{x})$.
Damit ist es nun möglich, die relativen bzw örtlichen Extremwerte einer Funktion $Q(\vec{x})$ in ihrer Abhängigkeit von den Systemparametern zu beschreiben und zu bestimmen.

Das allgemeine Optimierungsproblem hat nicht unbedingt eine eindeutige Lösung oder überhaupt eine Lösung. Meist werden mehrere örtliche Extremwerte existieren und nicht immer ist darin der absolute Extremwert enthalten. Dieser kann am Rand liegen. Außerdem versagt die ganze Gradientenmethode, wenn an irgendeiner Stelle des Parameter-Lösungsraumes nicht alle partiellen Ableitungen existieren. Sieht man von solchen Fällen ab, die sich allerdings oft schon durch den Gleichunstyp auszeichnen und daran erkennbar sind, so ist es recht einfach, mit Hilfe des Gradienten gemäß Gleichung (7.8) die Extremwerte zu bestimmen. Nach einem Vergleich der Extremwerte untereinander und einer Untersuchung der Ränder des Definitionsbereichs kann man den absoluten Extremwert endgültig gewinnen.

7.2.2 Die Beschreibung dynamischer Systeme

Die meisten Optimierungsprobleme gehören in das Gebiet der dynamischen Optimierung. Deshalb ist es vorteilhaft, das Verhalten - insbesondere das Zeitverhalten - solcher Systeme zu erörtern. Die mathematische Beschreibung dynamischer Systeme geschieht durch Differentialgleichungen. Man kann ein System n-ter Ordnung durch eine einzige Differentialgleichung n-ter Ordnung oder durch n Differentialgleichungen erster Ordnung beschreiben. Folgende Darstellungsweise ist allgemeingültig

$$\begin{aligned}
\dot{y}_1 &= f_1(y_1, \ldots, y_m; t; x_1, \ldots, x_n) \\
&\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\
\dot{y}_i &= f_i(y_1, \ldots, y_m; t; x_1, \ldots, x_n) \\
&\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\
\dot{y}_m &= f_m(y_1, \ldots, y_m; t; x_1, \ldots, x_n)
\end{aligned} \tag{7.9}$$

Dabei bedeutet

$$y_{i+1} = \frac{d^i y}{dt^i} \tag{7.10}$$

In Gleichung (7.9) sind die y_i die normalen unabhängigen Variablen des Systems (z. B. räumliche Koordinaten eines Körpers, in dem ein Prozeß stattfindet). Die Zeit t ist die Größe, nach der abgeleitet wird. Die Größen x_j sind die Systemparameter, die bei der Optimierungsaufgabe interessant werden.

Zur vollständigen Kennzeichnung eines dynamischen Systems benötigt man noch eine Reihe von Anfangsbedingungen

$$y_1(0) = y_{1o};\ y_2(0) = y_{2o};\ \ldots\ ;\ y_m(0) = y_{mo} \qquad (7.11)$$

Darin stellt jede Größe y_{io} den Wert der (i-1)-ten Ableitung zum Zeitpunkt t=0 dar. Da die Variablen y_i das Systemverhalten zu jedem Zeitpunkt vollständig festlegen, bezeichnet man sie häufig als die Zustandsvariablen des Systems. Ebenso kann man die n Parameter x_j als die Komponenten eines Parametervektors betrachten. Man definiert damit folgende Parameter

$$\vec{y} = (y_1, \ldots, y_m)\ , \qquad \vec{x} = (x_1, \ldots, x_n) \qquad (7.12)\ .$$

Nun ist es möglich, die ganze Gruppe von Differentialgleichungen (7.9) abgekürzt in Vektorschreibweise zu formulieren und es gilt

$$\dot{\vec{y}} = \vec{f}\,(\vec{y}\,;t;\vec{x}\,) \qquad (7.13)$$

Dazu gehören die Anfangsbedingungen

$$\vec{y}(0) = \vec{y}_o \qquad (7.14)$$

Durch die Anfangsbedingungen und die Zustandsvariablen ist ein dynamisches System in seinem Ablauf und seinen Verhaltensweisen eindeutig bestimmt. Allerdings kann der Ablauf von Systemparametern abhängen, die während des laufenden Prozesses konstant bleiben, aber von Lauf zu Lauf verändert werden können.

7.2.3 Die Kriteriumsfunktion

Die Optimierung statischer oder dynamischer Systeme bezieht sich immer auf eine bestimmte Kriteriumsfunktion, die vor der Auswertung von Rechenergebnissen festgelegt werden muß. Da diese Kriteriumsfunktion $Q(x_1,\ldots,x_n)$ in Abhängigkeit von den Systemparametern $x_1,\ldots,x_n$ praktisch beliebig gewählt werden kann, allerdings mit der Maßgabe, wirklich ein Kriterium für ein optimales Prozeßverhalten zu sein, muß Q mit großer Sorgfalt ausgesucht werden.

Die Kriteriumsfunktion ist eine gewöhnliche, d. h. differenzierbare Funktion der Systemparameter $x_1,\ldots,x_n$. Sie führt zu einem Extremwert des Systems, wenn Gleichung (7.8) gültig ist. Um diese Bedingung zu erfüllen, muß der Parametervektor bei jedem Rechenlauf so verändert werden, daß sich Q dem Extremwert nähert. Es muß also gelten, wenn $\vec{x}_i$ der Parametervektor beim i-ten Lauf ist,

$$Q(\vec{x}_{i+1}) = Q_{i+1} \gtrless Q_i = Q(\vec{x}_i) \qquad (7.15)$$

damit der Extremwert ein Maximum (Minimum) wird.

Betrachtet man einen Regelkreis, bei dem $A(y_1,\ldots,y_m;t;x_1,\ldots,x_n)$ die Ausgangsgröße und S der Sollwert ist, so kann man z. B. die folgenden Kriteriumsfunktionen wählen

$$Q(x_1,\ldots,x_n) = \int_0^T |(A - S)|\ dt \qquad (7.16)$$

oder

$$Q(x_1,\ldots,x_n) = \int_0^T (A - S)^2\ dt \qquad (7.17)$$

Gefordert ist für Q ein absolutes Minimum des Integrals über der Rechenzeit T.

Dagegen wäre folgende Kriteriumsfunktion ungeeignet

$$Q(x_1,\ldots,x_n) = \int_0^T (A - S)\ dt \qquad (7.18)$$

In Bild 7.5 ist zu sehen, daß sich bei einem Regelkreis z. B. eine Schwingung um den Sollwert ergibt, so daß sich bei der Integration der absoluten Abweichungen positive und negative Anteile zum großen Teil gegenseitig herausheben würden, d. h. man erhielte keine echte Auskunft über die Gesamtabweichungen vom Sollwert, sondern einen wesentlich kleineren mehr oder weniger zufällig entstandenen Wert. Deshalb wird von der Kosten- bzw Kriteriumsfunktion häufig verlangt, daß der Wert der zu integrierenden Funktion $z(x_1,\ldots,x_n)$ immer positiv sein muß.

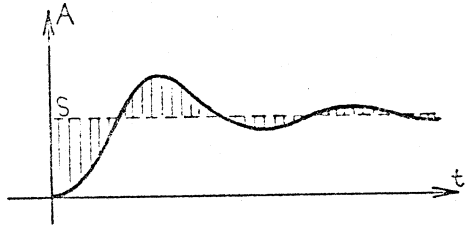


Bild 7.5 zeigt den möglichen Zusammenhang zwischen A und S

7.3 Parameteroptimierung nach dem Gradientenverfahren

7.3.1 Optimierung durch kontinuierlichen steilsten Abstieg(Anstieg)

Die Methode des kontinuierlichen steilsten Abstiegs bzw Anstiegs kann nur bei statischen Optimierungsaufgaben angewandt werden, da ja kontinuierlich in der Zeit optimiert wird. Bei dynamischen Systemen ist die Zeit aber selbst eine Systemkomponente, so daß der zu optimierende Prozeß und die zu beobachtende Auswirkung von Parameteränderungen ebenfalls zeitabhängig sind. Das Systemverhalten und die Kriteriumsfunktion seien durch eine Gruppe von linearen oder nichtlinearen algebraischen Gleichungen, abhängig von einer Reihe von Systemparametern gegeben. Wird dieses System z. B. an einem Analogrechner simuliert, so ist die Kriteriumsfunktion Q selbst eine Ausgabegröße der Schaltung. Der Parametervektor soll nun, beginnend bei einem zu bestimmenden Anfangspunkt, so verändert werden, daß er sich in Richtung des Lösungspunktes im Parameterraum bewegt. Das geschieht am schnellsten, wenn jede Komponente des sich ändernden Parametervektors $\vec{x}$ parallel zur entsprechenden Komponente des Gradienten grad Q verläuft. Man spricht dann von einer Einstellung der Parameteränderung in Richtung des steilsten Abstiegs (Anstiegs), wenn ein Minimum (Maximum) gesucht wird. Sind zwei Vektoren zueinander parallel, so wird das innere Produkt maximal, d. h. die Änderungsgeschwindigkeit des Parametervektors ist am schnellsten, wenn die Änderung tangential zum Gradientenvektor verläuft. Die Änderungsgeschwindigkeit der Kriteriumsfunktion ist gegeben durch

$$\frac{dQ}{dt} = \frac{\partial Q}{\partial x_1}\cdot\frac{dx}{dt}1 + \frac{\partial Q}{\partial x_2}\cdot\frac{dx}{dt}2 + \ldots + \frac{\partial Q}{\partial x_n}\frac{dx}{dt}n \qquad (7.19)$$

Voraussetzung für die Verwendung dieser Gleichung ist wieder die Existenz der partiellen Ableitungen von Q nach den Systemparametern. Das läßt sich formulieren in der Form

$$\frac{dx_i/dt}{dx_j/dt} = \frac{\partial Q/\partial x_i}{\partial Q/\partial x_j} \quad \text{oder in Vektorform} \quad \frac{d\vec{x}}{dt} = K(\nabla Q) \qquad (7.20)$$

Für K>0 : Methode des steilsten Anstiegs.
Für K<0 : Methode des steilsten Abstiegs.

Beim statischen System folgt die Ausgangsgröße unmittelbar der Eingangsgröße. Deshalb sind nur die drei normalen Rechenphasen notwendig und der Optimisator arbeitet in der Betriebsart Repetierendes Rechnen. Damit können alle relativen Extremwerte ermittelt werden.

7.3.2 Genäherte kontinuierliche Gradientenmethode bei dynamischen Optimierungsaufgaben

Die meisten Optimierungsaufgaben befassen sich mit der Optimierung dynamischer Objekte, d. h. mit Prozessen, die durch Differentialgleichungen beschrieben werden. Dabei ist es nicht möglich, wie bei statischen Prozessen, mit einer direkten Methode des kontinuierlichen steilsten An- bzw Abstiegs zu arbeiten, da in der Rechenschaltung Energiespeicher enthalten sind, so daß der Gradient der Kriteriumsfunktion nur im Verlaufe einer zeitlichen Integration ermittelt werden kann.

Bei der Näherungsmethode des kontinuierlichen steilsten Abstiegs bzw Anstiegs wird das System auf dem Analogrechner simuliert und die Kriteriumsfunktion wird als Systemausgabe ermittelt. Danach muß der Gradient der Kriteriumsfunktion grad Q berechnet werden. Jede Komponente des Gradienten muß der Änderungsrate bzw Änderungsgeschwindigkeit der entsprechenden Parameterkomponente proportional gemacht werden. Die Ausgabe des Gradienten besteht deshalb aus n verschiedenen Komponenten, wobei jede einer zeitlichen Änderung proportional läuft. Diese Komponenten werden integriert, um sofort die Einstellwerte der Parameter für die Systemsimulation zu erzeugen. Die Näherungsgradientenmethode kann so als nahezu kontinuierliches Optimierungsverfahren betrachtet werden, wenn die Rechenzeit des Systems klein ist gegen die Einstellzeit der Parameter. Dann fallen die Systemrechenzeiten so wenig ins Gewicht, daß der Vorgang der Gradientenbestimmung und der Parameternachführung nahezu zeitunabhängig erscheint.

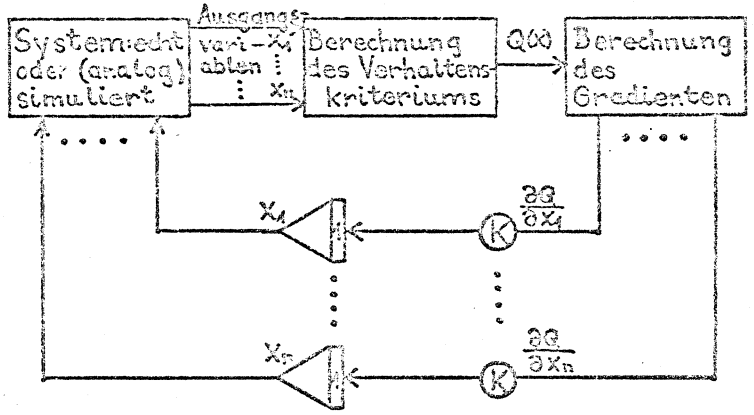


Bild 7.6 Instrumentierung der Methode des steilsten An- bzw Abstiegs

Bei der angenäherten Methode des steilsten An- bzw. Abstiegs wird die Länge jedes Schrittes durch den für K gewählten Wert bestimmt. Es ist durchaus möglich, eine Schrittgröße zu wählen, die so groß ist, daß sie über den Extremwert hinausführt und sogar zu einer Verschlechterung des Wertes der Kriteriumsfunktion führt. Wird also K nicht klein genug gewählt, so wird die Einstellschleife unstabil und es ergibt sich z. B. eine Pendelbewegung um den Extremwert. Andererseits zieht eine unnötig kleine Schrittgröße eine verschwenderisch große Menge von Iterationen nach sich. Demgemäß ist die Wahl von K, obwohl sie in weiten Grenzen unkritisch sein kann, eine wesentliche Frage bei der Optimierung des Gradientenverfahrens.

Der Algorithmus muß bei jedem Schritt des Verfahrens prüfen, ob sich der Wert von Q gemäß Gleichung (7.15) verbessert hat, ehe die vorgenommene Parameteränderung endgültig übernommen wird als Wert des neuen Anfangspunktes für den nächsten Verfahrensschritt. Wenn Q im Parameterraum metrisch ist, d. h. meßbar und immer positiv, wird bei geeigneter Wahl von K eine konvergierende Wertefolge von Q gegen den Wert des Extremums erzeugt.

7.3.3 Dynamische Optimierung durch diskrete Gradientenverfahren

7.3.3.1 Die Methode der diskreten Gradienten

Zur Vermeidung der schwierigen mathematischen Probleme, die der Versuch einer genäherten kontinuierlichen Parameteroptimierung bei dynamischen Systemen bereitet, ist es nötig, eine Kriteriumsfunktion Q zu wählen, die sich für iterative Optimierungsverfahren eignet. Das Verfahren beginnt mit der Wahl einer Kriteriumsfunktion, die durch Integration über eine bestimmte Zeitspanne aus einer geeignet zu wählenden Funktion entsteht. Sei z. B. $\vec{y}_s(t)$ der Sollwert eines Prozesses und $\vec{y}(\vec{x};t)$ der tatsächliche Wert, so kann man festlegen

$$Q(\vec{x}) = \int_{t_o}^{t_o+T} \left(\vec{y}_s(t) - \vec{y}(\vec{x};t)\right)^2 dt \qquad (7.21)$$

Dabei wird davon ausgegangen, daß der Parametervektor $\vec{x}$ während des Zeitintervalls $t_o \leqq t \leqq t_o+T$ konstant gehalten wird. Verwendet man diese Kriteriumsfunktion, so kann man das iterative Gradientenverfahren mit vier Schritten beschreiben bzw. instrumentieren.

1. Schritt:

Zu Beginn des Verfahrens muß ein Anfangsparametervektor $\vec{x}_o$ festgelegt und eingestellt werden.

2. Schritt:

Im Anfangspunkt $\vec{x}_o$ muß der Gradient grad Q_o der Kriteriumsfunktion ausgewertet werden, dabei gilt

$$\text{grad } Q_o = \left(\frac{\partial Q_o}{\partial x_1}, \frac{\partial Q_o}{\partial x_2}, \ldots, \frac{\partial Q_o}{\partial x_n}\right) \qquad (7.22)$$

Der Gradient von Q_o zeigt in die Richtung des steilsten An- bzw Abstiegs der Kriteriumsfunktion. Der Betrag von grad Q_o ist ein Maß für die Geschwindigkeit des An- bzw Abstiegs.

3. Schritt:

Nach der Berechnung des Gradienten von Q_o an der Stelle $\vec{x}_o$ wird mit Hilfe der Beziehung (7.23), die den vorher benützten kontinuierlichen Gleichungen für den steilsten An- bzw Abstieg entspricht, ein diskreter Parameter-Änderungsvektor berechnet.

$$\Delta\vec{x}_o = K\cdot\nabla Q(\vec{x}_o) \qquad (7.23)$$

mit steilstem Anstieg für $K > 0$ und
steilstem Abstieg für $K < 0$.

4. Schritt:

Dem Parametervektor ist ein neuer Wert zuzuweisen, so daß gilt

$$\vec{x}_o := \vec{x}_o + \Delta\vec{x} \qquad (7.24)$$

Dadurch erhalten alle einzelnen Systemparameter neue Werte zugewiesen gemäß Gleichung (7.25)

$$x_{io} := x_{io} + \Delta x_i = x_{io} + K\frac{\partial Q}{\partial x_i} \qquad (7.25)$$

Das entspricht einem Schritt in Richtung des Gradienten beim steilsten Anstieg bzw einem Schritt in entgegengesetzter Richtung beim steilsten Abstieg.

Für diese Folge von vier Rechenschritten läßt sich ein Algorithmus entwickeln, der durch iterative Parameteroptimierung längs einer Fallinie zum gesuchten Extremwert strebt. Ist auf diese Weise ein Extremwert angenähert, dann kann unter Umständen, von einem Näherungswert ausgehend, mit einer anderen Suchmethode (z. B. digital) der Wert noch genauer bestimmt werden. Das kann z. B. durch quadratische Interpolation der drei oder vier letzten Werte der Kriteriumsfunktion Q geschehen. Auf jeden Fall muß ein Algorithmus

ein Abbruchkriterium vorgesehen sein, das die Genauigkeit des Rechners und die Zahl der Rechenläufe berücksichtigt, die zum Zeitpunkt der Abfrage bereits vorgenommen wurden.

Bei der Instrumentierung des Verfahrens auf einem Rechner werden die Gradientenkomponenten durch die Differenzenquotienten angenähert

$$\frac{\partial Q}{\partial x_i} \approx \frac{Q(x_1,\ldots,x_i+\Delta x_i,\ldots,x_n)-Q(x_1,\ldots,x_i,\ldots,x_n)}{\Delta x_i} = \frac{Q_i-Q_o}{\Delta x_i} \qquad (7.26)$$

Dabei ist es z. B. möglich, alle $\Delta x_i = \Delta x$ = konstant zu wählen.

Die wesentlichen Operationen, die der Optimisator auszuführen hat, sind das Speichern der diskreten Werte Q_o und $(Q_i-Q_o)/\Delta x_i$. Dazu benutzt man die Analog-Speicher des Rechners, wie es im Bild (7.8) gezeigt wird. Zunächst wird die Kriteriumsfunktion für den Anfangspunkt $\vec{x}_o$ bestimmt, also Q_o, und gespeichert. Dann wird je eine Komponente von $\vec{x}_o$ verändert, dazu Q_i berechnet und $(Q_i-Q_o)/\Delta x_i$ gespeichert. Zur Speicherung aller n Gradientenkomponenten sind damit insgesamt (n+1) Analog-Speicher erforderlich. Mit Hilfe der gespeicherten Werte können die neuen Parameterwerte anhand von Gleichung (7.25) hergestellt werden.

Das Verfahren wird im Programmablaufplan (7.7) beschrieben. Die Bilder (7.8) und (7.9) zeigen die Instrumentierung des Verfahrens und das Zeitdiagramm der Steuerleitungen des Optimisators für ein Beispiel mit zwei Parametern.

Nach der Abarbeitung aller Quotienten $(Q_i-Q_o)/\Delta x_i$ werden in Bild (7.7) entsprechend der Instrumentierung eines hybriden Analogrechners alle neuen Parameterwerte für alle i parallel und gleichzeitig eingestellt. Bild (7.8) zeigt die Schaltung des analogen Teils des Optimisators. Zur Aufschaltung der Prüfschritte dienen Digital-Analog-Schalter, die von den Steuerleitungen $S_1, \ldots, S_n$ angesteuert werden. Q_o wird in den als Speicher geschalteten Integrierer O übernommen, wenn die Steuerleitung H_o erregt wird. Die Differenzen $\Delta Q = (Q_i-Q_o)$ werden in einem Summierer gebildet. Die mit dem Faktor K bewerteten Gradientenkomponenten werden in den Speichern 1,.. ..,n bei der Erregung der entsprechenden Steuerleitung H_i zu den alten Werten x_i addiert. Diese wiederum werden bei der Erregung von H_s parallel in die Speicher $1^*, \ldots, n^*$ übernommen und ins System zurückgeführt.

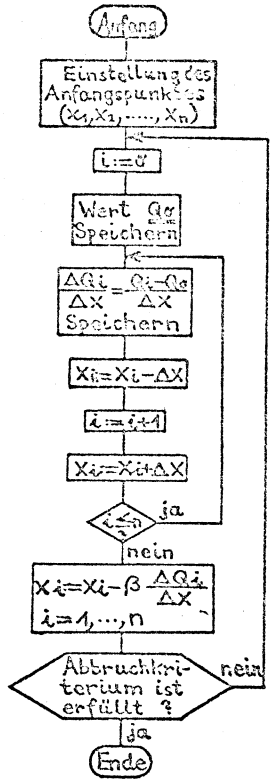


Bild 7.7 zeigt den Programmablaufplan für das diskrete Gradientenverfahren

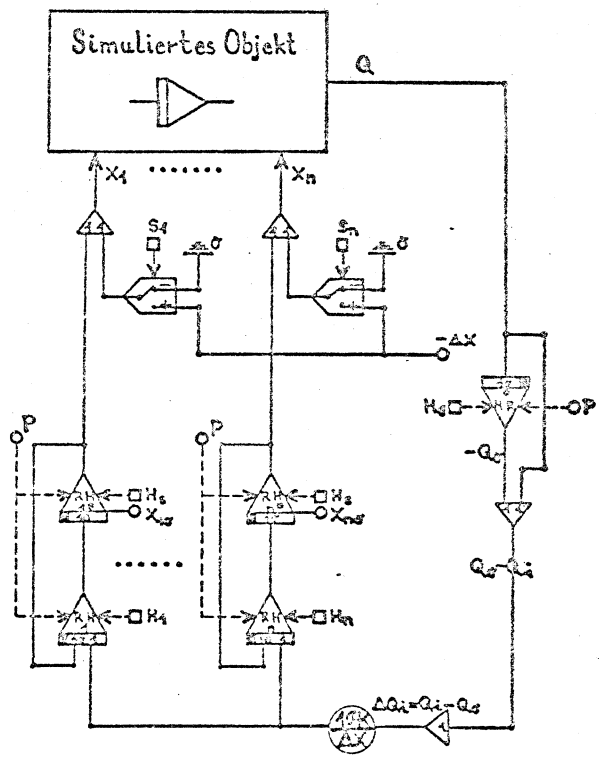


Bild 7.8 zeigt den analogen Teil des Optimisators beim diskreten Gradientenverfahren; es bedeutet □Anschlüsse auf dem Analogprogrammierfeld, ○ Anschlüsse auf dem Digitalprogrammierfeld

Die Steuerleitungen $H_o, \ldots, H_n$ und $S_1, \ldots, S_n$ sowie H_s stehen am Digitalprogrammierfeld zur Verfügung und werden durch eine aus digitalen Elementen aufgebaute Steuerschaltung gemäß dem Zeitdiagramm in Bild (7.9) erregt. Hierbei ist vorausgesetzt, daß es sich bei dem zu optimierenden Objekt um ein dynamisches System handelt. Q steht also erst nach dem Ablauf der Rechenzeit T zur Verfügung, und das System wird aus normalen Integrierern aufgebaut. Man arbeitet in der Betriebsart Iterierendes Rechnen, und man wählt zweckmäßigerweise $T_{R1} = T = m \cdot GT1$ und alle übrigen Zeiten gleich den Grundtakten GT1=GT2 (m ganze positive Zahl).

Die verschiedenen Steuersignale werden durch logische Verknüpfung der Registerausgänge $X_1, \ldots, X_n$ mit den Signalen T_{P1}, T_{R1}, T_{H1}, T_{P2}, T_{R2} erzeugt, die vom Steuergerät geliefert werden und den jeweiligen Stand des Objektes signalisieren

Die Methode des diskreten Gradientenverfahrens kar mit iterativen hybriden Analogrechnern bereits zufriedenstellend ausgeführt werden, wie die Beschreibung anhand der Bilder (7.7), (7.8) und (7.9) zeigt. Allerdings erfordert die Instrumentierung umfangreiche Logikelemente und Speicherelemente (verbraucht Integrierer). Daraus ersieht man, daß es noch günstiger ist, dieses Verfahren an Hybridrechnern zu programmieren, wo ein Digitalrechner vorhanden ist, auf dem sämtliche logischen Entscheidungen programmiert werden können.

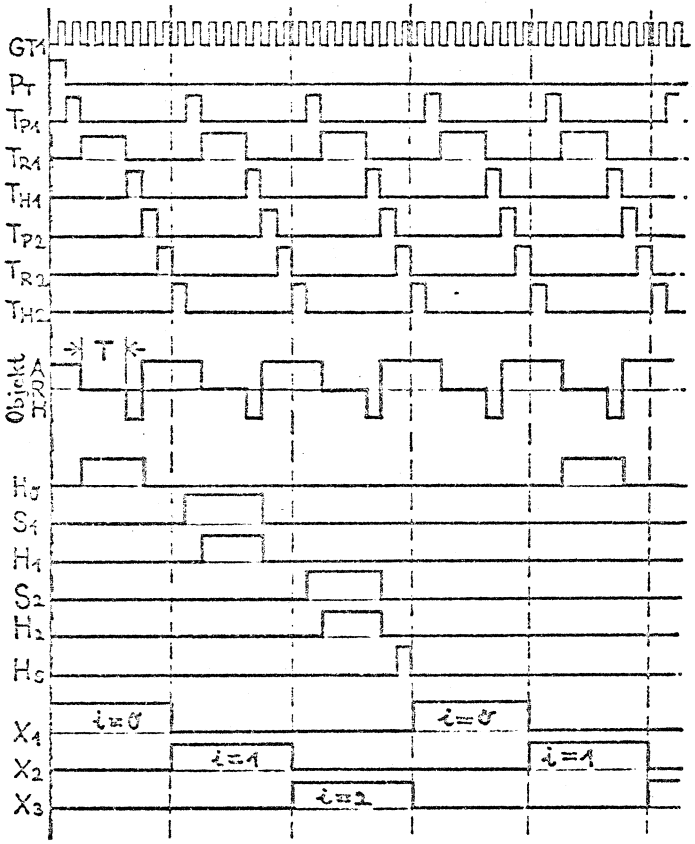


Bild 7.9 zeigt das Zeitdiagramm der Steuerleitungen des Optimisators bei der Methode diskreter Gradienten

7.3.3.2 Hybridinstrumentierung des diskreten Gradientenverfahrens

Bei der Instrumentierung auf einem Hybridrechner bietet sich für das diskrete Gradientenverfahren zur Ausnutzung der Fähigkeiten beider Rechner folgender Algorithmus an.

1. Schritt:
Das System, welches optimiert werden soll, wird auf dem Analogrechner simuliert und vom Digitalrechner aus werden die Werte der Systemparameter im Anfangspunkt im System eingestellt.

2. Schritt:
Auf dem Digitalrechner wird ein Verfahren für die Bestimmung der Parameter-Schrittgröße - abhängig von K - programmiert.

3. Schritt:
$Q(\vec{x}_o)$ wird auf dem Digitalrechner ausgewertet und gespeichert. Durch Digitalbefehl werden alle Parameter der Reihe nach geändert. Für je einen Parameter werden die $Q_i(x_1,\ldots,x_i+\Delta x_i,\ldots,x_n)$ nacheinander ausgewertet, d. h. der Ergebniswert vom Analogrechner übernommen, und abgespeichert. Sind alle n Komponenten von grad Q am Digitalrechner gemäß Gleichung (7.26) berechnet, so wird daraus der neue Gradient von Q gewonnen.

4. Schritt:
Sind alle (n+1) Rechenläufe, die zur Berechnung von grad Q benötigt werden, vorbei, so wird auf digitaler Seite der neue Parametervektor $\vec{x}_o$ gemäß Gleichung (7.25) berechnet und seine Komponenten werden am Analogrechner neu eingestellt.
Danach werden bis zur Erfüllung eines Abbruchkriteriums, das am besten digital zu programmieren ist, die Schritte 3 und 4 wiederholt.

7.3.3.3 Iterative Optimierung mit zyklischer Parametereinstellung

Die Notwendigkeit, n+1 Werte der Kriteriumsfunktion bzw ihres Gradienten zu speichern, kann man auf zwei Arten umgehen.
Man kann z. B. n+1 Simultansysteme auf dem Analogrechner simulieren, in denen gleichzeitig die verschiedenen Komponenten von grad Q berechnet werden können. Diese Methode benötigt allerdings wesentlich mehr analoge Rechenelemente und ist deshalb meist nicht anwendbar.
Um Rechenausrüstung zu sparen, kann man aber auch die Gradientenkomponenten nacheinander berechnen und das System jedesmal nach jeder Iteration wieder zu den Anfangswerten $\vec{x}_o$ zurückführen.
Bei dieser Alternativmethode werden also nicht alle n Parameter eingestellt, sondern nacheinander immer nur einer, für den dann sofort der neue Anfangspunkt $\vec{x}_o$ festgelegt wird. Man kann diese Methode kurz wie folgt beschreiben.

1. Schritt:
Anfangspunkt $\vec{x}_o$ festlegen und $Q_o = Q(\vec{x}_o)$ berechnen und speichern.

2. Schritt:
Die nächste gerade anstehende Komponente des Gradienten berechnen. Beim ersten Durchlaufen des 2-ten Schrittes wird die 1-te Komponente berechnet beim i-ten Durchlauf die i-te Komponente

$$\frac{\partial Q(x_1,\ldots,x_i+x_i,\ldots,x_n)}{\partial x_i}$$

bis i=n. Damit wird sofort der neue Parametervektor $\vec{x}_o := \vec{x}_o + \vec{\Delta x}_i$ eingestellt, wobei $\vec{\Delta x}_i$ ein Vektor in die Richtung der i-ten Komponente ist.

3. Schritt:
Der neue Parametervektor wird am Analogrechner eingestellt und es wird sofort der neue Wert von $Q_o = Q(\vec{x}_o)$ berechnet und gespeichert. Ist das Abbruchkriterium nicht erfüllt, so wird mit Schritt 2 fortgesetzt.

7.4 Parameteroptimierung durch sequentielle Parametereinstellung (Relaxationsverfahren)

7.4.1 Das Relaxationsverfahren (Gauß-Seidel-Verfahren)

Bei den bisher besprochenen Methoden war es erforderlich, den Gradienten der Kriteriumsfunktion vor oder gleichzeitig mit der Parametereinstellung zu berechnen. Eine derart komplizierte Einstellmethode ist nicht immer gerechtfertigt. Bedeutend einfachere Methoden können zu einer längeren Berechnungszeit führen, doch kann man dadurch Aufwand (z. B. bei der Analoginstrumentierung) oder Programmierzeit und Speicherplatzbedarf komplexer Unterprogramme am Digitalrechner einsparen.
Eine dieser einfacheren Methoden besteht in der Suche nach dem Extremwert durch Einstellen der verschiedenen Parameter über ihren zulässigen Gesamtbereich und in der Wahl des Wertes des Parametervektors, der zu einem Extremwert für die Kriteriumsfunktion Q führt. Bei dieser Methode braucht der Gradient von Q nicht berechnet zu werden und es wird auch weniger Analog-Speicher gebraucht.

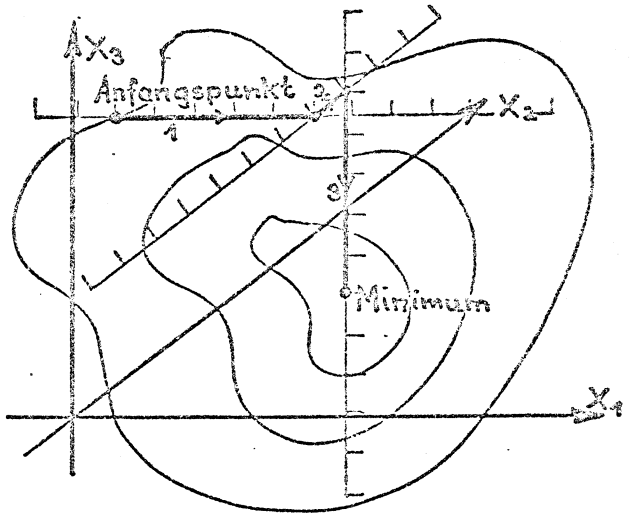


Bild 7.10 zeigt das Prinzip der Einparameter-Einstell-Methode

Auch im Falle des Relaxationsverfahrens oder Gau Seidel-Verfahrens wird mit der Wahl eines Anfangspunktes begonnen, d. h. am Analogrechner wird zunächst $\vec{x}_o$ eingestellt. Dann wird der erste Parameter x_1 im ganzen für ihn zulässigen Wertebereich variiert, bis eine Verbesserung der Kriteriumsfunktion $Q(x_1, x_{2o}, \ldots, x_{no})$ nicht mehr erreichbar ist. Dabei bleiben die anderen Systemparameter auf festen Werten $x_{2o}, x_{3o}, \ldots, x_{no}$, bis für x_1 der optimale Wert gefunden ist. Das Optimum von Q kann dann z.B.systematisch ermittelt werden, indem x_1 fortwährend um denselben Wert Δx_i erhöht oder erniedrigt wird.
Anschließend wird dasselbe Verfahren auf den zweiten Parameter x_2 angewendet usw, bis alle Parameter auf diese Weise durchgetestet worden sind. Allgemein wendet man dann für die Einstellung des i-ten Parameters ein Verfahren gemäß Gleichung (7.27) an.

$$x_i := x_i \pm K \cdot \Delta x_i \qquad (7.27).$$

Je nach Wahl des Anfangswertes von x_i wird das positive oder negative Vorzeichen in Gleichung (7.27) verwendet oder auch beide. Wenn der letzte Parameter auf diese Weise bearbeitet wurde, erhält man schließlich einen Parametervektor, der die Stelle des absoluten Extremwerts im ganzen untersuchten Parameterraum bezeichnet,und der zugehörige Wert von Q ist optimal für diesen Bereich.

Beim sequentiellen Verändern der einzelnen Parameter und Parameterwerte arbeitet der Analogrechner in der Betriebsart Iterierendes Rechnen. Im normalen Teilzyklus wird die Kriteriumsfunktion Q berechnet. Im komplementären Zyklus wird der neue Parameterwert eingestellt und eventuell,bei einer Verbesserung von Q, der neue bessere Wert der Kriteriumsfunktion und des Parameters gespeichert.

7.4.2 Ver ren des benachbarten Gitternetzes

Eine Alternativmöglichkeit des Suchverfahrens der sequentiellen Parametereinstellung besteht darin, alle Parameter x_i mit $+\Delta x_i$ bzw $-\Delta x_i$ zu erweitern. Alle damit erreichbaren Werte des Parametervektors werden nun daraufhin untersucht, ob sich der Wert von Q in Richtung auf den gesuchten Extremwert verändert. Der beste Wert von Q und der Wert des dazugehörigen Parametervektors werden gespeichert. Dann beginnt die Auswertung ausgehend von dem neuen Anfangspunkt auf's Neue. Dabei sind weniger Rechenläufe notwendig, allerdings ist es dafür möglich, daß man nur ein relatives Extremum findet aber nicht den absoluten Extremwert,weil nur ein Teil des Parameterraums abgetastet wird. Das Verfahren konvergiert unter Umständen langsam, wenn die Kriteriumsfunktion nur stückweise differenzierbar oder mehrfach zusammenhängend ist. Bei nichtlinearen Systemen kann beides geschehen. Um auch die weiteren Extremwerte zu finden, wenn es mehrere gibt, kann man z. B. nach jedem Optimierungslauf, der einen Extremwert ermittelt, mit neuen Anfangswerten für die Systemparameter die Optimierung wiederholen. Bild 7.11 zeigt schematisch die Methode des benachbarten Gitternetzes, die dünn ausgezogenen Linien, die am Ende einen Punkt tragen, stellen die möglichen Parametervariationen um $\pm\Delta x_i$ dar; die dick ausgezogenen Linien führen zu erfolgreichen Werten des Parametervektors. Sie führen zum Extremwert (im Bild ein Maximum).

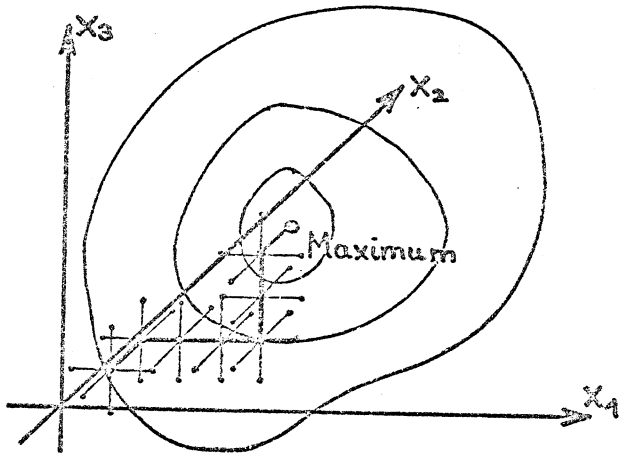


Bild 7.11 zeigt die Methode der benachbarten Gitternetze

7.4.3 Das Beispiel der allgemeinen Randwertaufgabe

Vorgegeben ist ein System mit unbekannten Anfangswerten z. B.

$$\dot{y}_1 = f_1(y_1,\ldots,y_n;t)$$
$$\ldots\ldots\ldots\ldots\ldots$$
$$\dot{y}_n = f_n(y_1,\ldots,y_n;t) \qquad (7.28)$$

mit den unbekannten Anfangsbedingungen

$$y_1(0), \ldots, y_n(0) \triangleq x_1,\ldots,x_n = \text{Systemparameter} \qquad (2.29)$$

Außerdem gebe es eine Reihe von Randwerten

$$U_1,\ldots,U_m \qquad (2.30)$$

die hier nicht näher ausgeführt werden müssen.

Zur Behandlung von Randwertproblemen mit einem Analogrechner verwendet man häufig das sogenannte Probierverfahren. Dabei werden zunächst willkürliche Anfangswerte für die Systemparameter eingestellt, das sind die unbekannten Anfangsbedingungen $x_1,\ldots,x_n$. Mit den gewählten Anfangswerten erhält man eine Lösung, die zum Zeitpunkt T, nach abgeschlossenem Rechenlauf, gemessen werden kann und normalerweise noch nicht den Randbedingungen des Systems genügt. Nun werden die Anfangsbedingungen systematisch verändert, bis die Lösungen immer besser die Randwerte erfüllen. Dazu wird immer ein Parameter so lange verstellt, bis er zu einer optimalen Kriteriumsfunktion Q führt, d. h. bis die Randwerte am besten erfüllt werden. Bei mehr als zwei unbekannten Anfangsbedingungen kann die Lösung nur noch am Hybridrechner gefunden werden. Für Differentialgleichungen höherer Ordnung ist eine Lösung nicht immer gewährleistet.

7.5 Zufallsbedingte Optimierungsverfahren

7.5.1 Der Grundalgorithmus

Bei der einen Zufallssuche wird die Kriteriumsfunktion Q an zufällig gewählten Punkten des Parameterraums berechnet, und durch zufällige Wahl von Veränderungen Δx_i der einzelnen Komponenten des Parametervektors wird eine Verbesserung von Q angestrebt. Es handelt sich sozusagen um ein zufälliges Kriechen in Richtung eines Extremwertes, wobei der Ausdruck Kriechen bereits andeutet, daß ein solcher Vorgang wesentlich länger dauern kann als bei anderen Verfahren. Man kann den Grundalgorithmus etwa wie folgt angeben (Beschreibung des Algorithmus in Bild 7.12).

1. Schritt:
Es wird ein Anfangsparametervektor $\vec{x}_o$ festgelegt. Dann wird $Q_o = Q(\vec{x}_o)$ berechnet.

2. Schritt.
Nun wird durch eine Folge rein zufälliger und voneinander unabhängiger Zufallswerte N_i für jede Komponente x_i des Parametervektors ein Zuwachs $\Delta x_i = C_i \cdot N_i$ berechnet, wobei C_i eine feste Konstante ist (z. B. alle C_i gleich). Die Folge der N_i ergibt im Mittelwert 0, d. h. sie schwanken nach Betrag und Vorzeichen in statistisch gleicher Verteilung. Damit erhält man insgesamt den neuen Parametervektor

$$\vec{x}_1 = \vec{x}_o + \Delta\vec{x} \qquad (7.31) \quad \text{mit } \Delta x_i = C_i \cdot N_i \qquad (7.32)$$

Die zufälligen Werte N_i kann man z. B. durch das Abtastgeräusch von Rauschgeneratoren am Analogrechner , durch Erzeugung von Pseudozufallsfolgen am Digitalrechner oder auf andere Weise erzeugen.

3. Schritt:
Zum neuen Parametervektor $\vec{x}_1$ wird der Wert der Kriteriumsfunktion durch einen Rechenlauf bestimmt. Ist $Q(\vec{x}_1) = Q_1$ gegenüber Q_o ein verbesserter Wert, so wird gesetzt

$$Q_o := Q_1 \quad \text{und} \quad \vec{x}_o := \vec{x}_1 \qquad (7.33)$$

Ist das Abbruchkriterium (z. B. Zahl der Versuche) noch nicht erfüllt, so wird mit Schritt 2 fortgesetzt.

Durch diese Methode sind die Systemgleichungen im Prinzip für jeden Punkt $\vec{x}_o$ im Parameterraum lösbar. Allerdings muß gewähr-

leistet sein, daß der Punkt, an dem Q berechnet werden soll, im erlaubten Parameterbereich liegt, d. h. strenggenommen muß im Algorithmus abgeprüft werden, ob alle Zuwächse $\Delta\vec{x}$ erlaubt sind. Deshalb ist es auch nicht sinnvoll,den Anfangspunkt für das Verfahren an den Rand des Parameterraums zu legen, der untersucht werden soll.

Der Grundalgorithmus kann unter Umständen erheblich verbessert werden, wenn man versucht,aus den Eigenschaften der Kriteriumsfunktion Q den größtmöglichen Nutzen zu ziehen. Es ist nämlich häufig der Fall, daß die einmal erfolgreiche Richtung des Änderungsvektors $\Delta\vec{x}$ auch bei nachfolgenden Schritte eine Verbesserung einbringt. D. h. es empfiehlt sich,eine gewisse Wechselbeziehung zwischen aufeinanderfolgenden Schritten herzustellen. Damit geht man von der reinen Zufallssuche also etwas ab.
Es ist z. B. möglich, folgendes zusätzliche Verfahren in den Grundalgorithmus einzubauen: Bringt ein Versuchswert $\vec{x}_1 = \vec{x}_o + \Delta\vec{x}$ eine Verbesserung von Q, so wird zunächst im Grundalgorithmus gesetzt $\vec{x}_o := \vec{x}_1$. Nun wird der erfolgreiche Änderungsvektor nochmals angesetzt und es gilt dann ein weiteresmal $\vec{x}_o := \vec{x}_o + \Delta\vec{x}$. Insgesamt wurde also $\Delta\vec{x}$ zweimal zum ursprünglichen Wert von $\vec{x}_o$ addiert.
Brachte aber der Versuch mit dem Änderungsvektor $\Delta\vec{x}$ eine Verschlechterung, so wird gesetzt $\vec{x}_o := \vec{x}_o - \Delta\vec{x}$ und es wird mit Schritt 3 weitergemacht.
Solche Techniken nennt man absolute Steuerung in positive oder negative Richtung.

Eine weitere Variationsmöglichkeit ist dadurch gegeben, daß man für die verschiedenen Komponenten x_i des Parametervektors verschiedene Faktoren C_i einführt, je nachdem, wie groß der erlaubte Bereich eines Parameters ist. Andererseits kann man für verschiedene Rechenläufe alle C_i um denselben Faktor oder Summanden verändern. Nähert sich das Verfahren z. B. bei einer globalen Suche örtlichen Extrempunkten, so kann die Schrittweite mit Hilfe der C_i verkleinert werden, damit das Verfahren nicht in zu großen Schritten über den Extrempunkt hinausschießt. Andererseits kann, von einem relativen oder örtlichen Extrempunkt ausgehend, die Schrittweite laufend vergrößert werden, um nicht zu viele Versuche zu benötigen, bis der ganze Parameterbereich wenigstens teilweise durchsucht ist.

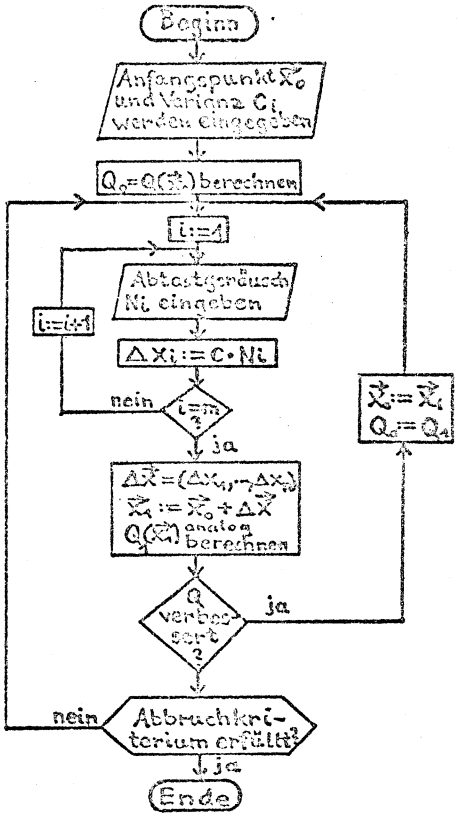


Bild 7.12 zeigt die Zufallssuche zur Ermittlung eines lokalen Extremwertes

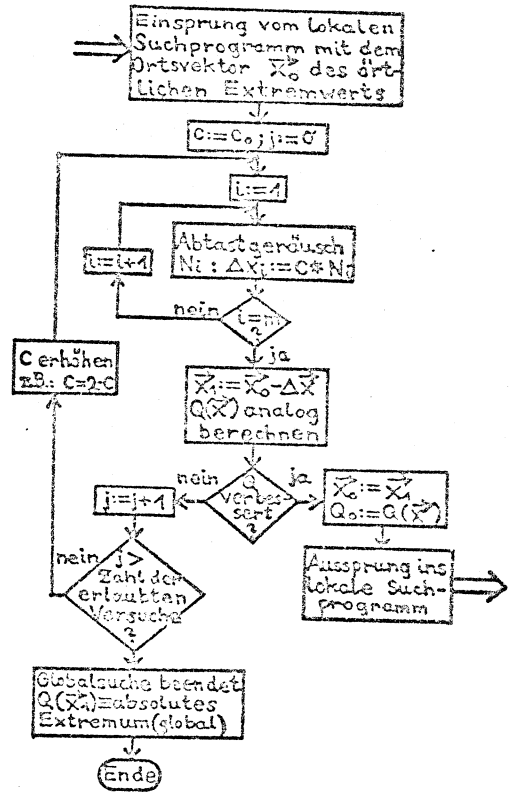


Bild 7.13 zeigt die Zufallssuche zur Ermittlung des globalen Extremwertes

7.5.2 Suche der lokalen und globalen Extremwerte

Der Grundalgorithmus zur Ermittlung des absoluten Extremwerts besteht aus den zwei Teilen der Bestimmung eines relativen bzw örtlichen Extremwerts und der Bestimmung des absoluten bzw globalen Extremwerts.

7.5.2.1 Die Suche nach einem lokalen Extremwert

1. Schritt:
Es wird ein Anfangsparametervektor $\vec{x}_o$ festgelegt und $Q_o = Q(\vec{x}_o)$ berechnet.

2. Schritt:

Nun wird ein Zufallsvektor $\Delta\vec{x} = (C_1 \cdot N_1, \ldots, C_n \cdot N_n)$ mit zufälligen, voneinander unabhängigen Werten $N_1, \ldots, N_n$ und zu vereinbarenden Faktoren $C_1, \ldots, C_n$ hergestellt und damit der Versuchsvektor $\vec{x}_1 = \vec{x}_o + \Delta\vec{x}$ gebildet.

3. Schritt:

Am Analogrechner wird der neue Wert $Q_1 = Q(\vec{x}_1)$ berechnet.

4. Schritt:

Q_1 und Q_o werden miteinander verglichen. Bringt Q_1 eine Verbesserung, so wird gesetzt $Q_o := Q_1$ und $\vec{x}_o := \vec{x}_1$.

5. Schritt:

Alle Fehlversuche, bei denen keine Verbesserung von Q eintrat, bzw alle Versuche überhaupt werden gezählt. Ist die zugelassene Zahl erreicht, so wird das Verfahren abgebrochen. Oder ein anderes Abbruchkriterium wird verwendet.

Die Schritte 1 - 5 werden sooft wiederholt, bis ein relatives Extremum von Q und damit ein mögliches Optimum des zu optimierenden Systems erreicht ist.

7.5.2.2 Die Suche nach dem globalen Extremwert

Sobald ein örtlicher Extremwert gefunden wurde, muß geprüft werden, ob es sich dabei um den globalen Extremwert handelt. Eine vernünftige Methode besteht darin, von neuem eine Zufallssuche durchzuführen. Diese bietet große Flexibilität, da die Suche so vorgenommen werden kann, daß sie die Orte vor allem untersucht, bei denen weitere Extremwerte besonders wahrscheinlich sind. In vielen Fällen ist der Raum nahe bei lokalen Extremwerten der wahrscheinlichste Ort für weitere Extremwerte. Man kann also z. B. Suchverfahren anwenden, die vor allem Punkte in der Nähe bereits bekannter Extremwerte untersuchen. Es werden z. B.,wie bei der lokalen Extremwertsuche, zufallsbedingte Parameterzuwächse zum Ortsvektor des bereits gefundenen lokalen Extremwerts hinzugezählt, wobei mit sehr kleinen Faktoren C_i begonnen wird und bei größerer Entfernung vom lokalen Extremwert die C_i immer größer gewählt werden, so daß nur noch recht grob der Parameterraum untersucht wird. Z. B. kann nach je einem Durchlauf des Verfahrens oder nach einer bestimmten Anzahl von Versuchen mit konstanten C_i jedes C_i verdoppelt werden, oder um einen konstanten Wert ΔC_i erhöht werden oder um einen konstanten Faktor (z. B. 1,02) erhöht werden .

Nach jeder möglichen Parameterverbesserung wird die Kriteriumsfunktion daraufhin untersucht,ob sie sich verbessert hat. Ist das der Fall, so wird der Wert des verbesserten Parametervektors als neues $\vec{x}_o$ gewählt und zum Suchverfahren für lokale Extremwerte gesprungen. Wird keine Verbesserung gefunden, so geht die globale Extremwertsuche weiter. Existiert im ganzen zulässigen Parameterbereich kein besserer Wert von Q, so wird das Verfahren beendet, und der zuletzt gefundene lokale Extremwert ist auch der globale Extremwert.

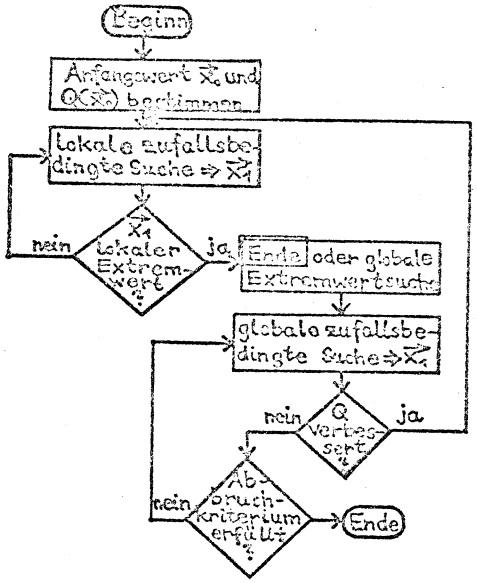


Bild 7.14 zeigt das kombinierte Verfahren der globalen Extremwertsuche

Die Konvergenz des lokalen Suchverfahrens kann gesichert werden, da sich dabei eine Folge von Werten der Kriteriumsfunktion Q ergibt, die monoton abnimmt und nach unten begrenzt ist, bzw monoton zunimmt und nach oben begrenzt ist. Die Schnelligkeit der Konvergenz ist allerdings eine andere Sache. Es ist nachweisbar, daß immer dann die Konvergenz gesichert ist für das allgemeine Verfahren, wenn sich auch bei der Gradientenmethode Konvergenz ergeben würde. Für den Fall nichtlinearer Systeme ist die Konvergenz nicht gesichert. Wird bei der globalen Suche der ganze Parameterraum abgesucht, so kann auf jeden Fall der globale Extremwert ermittelt werden. Die Frage ist nur, ob die Rechenzeit und Rechnerausstattung genügt.

8. Die Organisation der Zusammenarbeit zwischen Digital- und Analogrechner in einem hybriden Rechnersystem

Bereits in Kapitel 1 ist die unterschiedliche Arbeitsweise eines Digital- und eines Analogrechners gegenübergestellt worden. Sollen beide Rechnertypen zu einem hybriden System zusammengefügt werden, so müssen durch Hard- und Software die beiden Rechnertypen aneinander angepasst werden. Dabei muss das Gesamtsystem so organisiert werden, dass sich die Vorteile der Komponenten miteinander verbinden, nicht aber deren Nachteile.

8.1. Die Bausteine der Kopplung

Zur Zusammenarbeit eines Digital- und Analogrechners sind drei wesentliche Grundfunktionen erforderlich. Die Rechendaten müssen aus der analogen in die digitale Darstellung gewandelt werden. Die gesamte Steuerung des Analogrechners muss vom Digitalprogramm aus erfolgen. Schliesslich sind zur Synchronisation der Abläufe im Digital- und Analogrechner besondere Einrichtungen nötig. Bild 8.1 gibt eine Übersicht über die Einrichtungen eines Hybridsystems an einem Beispiel.

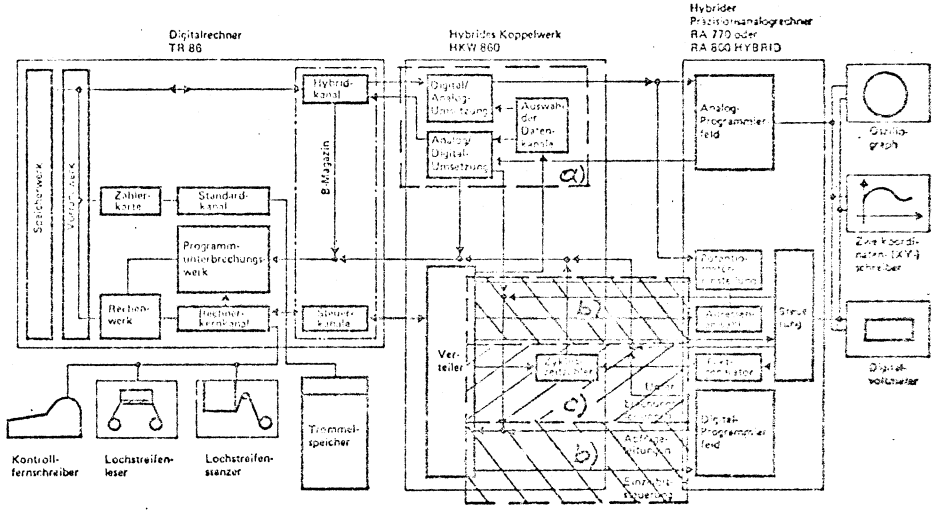


Bild 8.1 Blockschaltbild des Hybriden Rechnersystems HRS 860
Baugruppen zum Transfer der Rechendaten (a)), zur Steuerung des Analogrechners (b)) und zur Synchronisation (c)).

8.1.1 Wandlung der Rechendaten

Die Genauigkeit der Datenwandlung sollte sich an der typischen Genauigkeit der analogen Rechenelemente orientieren. Bei modernen Präzisionsanalogrechnern mit einer typischen Genauigkeit von 10^{-4} bedeutet dies einen 14 Bit Code einschliesslich Vorzeichen. Um eine Umcodierung im Digitalrechner (Zeitverlust!) zu vermeiden, wird man die Darstellung als linksbündige Festkomma-Dualzahl wählen.

Die Analog-Digitalumsetzung (ADU) ist im Prinzip ein Divisionsvorgang; es wird festgestellt, wie oft der entsprechende Teil der Referenzspannung im zu wandelnden Analogwert enthalten ist. Es wird häufig das in

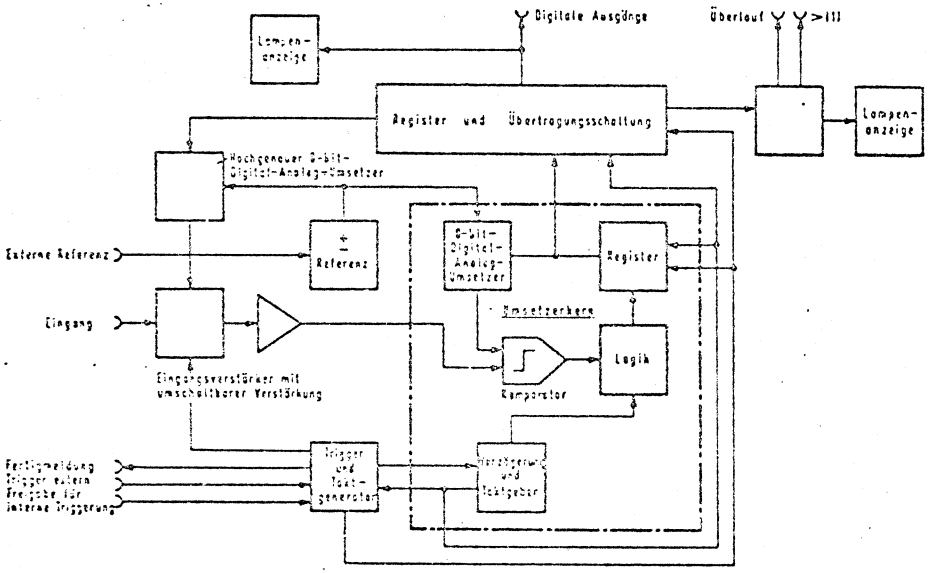


Bild 8.2 Zweibereichsverfahren bei der AD-Umsetzung

Bild 8.2 dargestellte Zweibereichsverfahren verwendet. Man wandelt dabei zunächst im Umsetzerkern mit einer Genauigkeit, die der halben Zahl von Dualstellen entspricht. Durch einen hochgenauen Digital-Analog-Umsetzer (DAU) wird dieser Wert in eine Analogspannung zurückgewandelt und von der Eingangsspannung subtrahiert. Die Differenzspannung wird so verstärkt, dass diese im gleichen Bereich liegt wie die Eingangswerte des Umsetzers. Mit dem selben Umsetzerkern können nun die restlichen Bits des Digitalwertes bestimmt werden. Man kommt auf diese weise mit einem Umsetzerkern von wesentlich geringerer Genauigkeit aus, benötigt aber wegen der doppelten Verwendung des Umsetzerkernes zusätzlich Zeit.

Im Umsetzerkern wird der Ausgang eines DAU systematisch durch Setzen verschiedener Bitkombinationen verändert. Ein Komparator stellt fest, ob die Differenz zwischen dem Eingangswert am Umsetzerkern u_e und dem Ausgang des DAU positiv oder negativ ist. Diese Entscheidung bestimmt die nächste einzustellende Bitkombination mit dem Ziel, dass $u_e - u_d$ gegen Null strebt (Verfahren der sukzessiven Approximation). Bild 8.3 zeigt ein Beispiel für den Ablauf des Approximationsvorganges für 4 Bit.

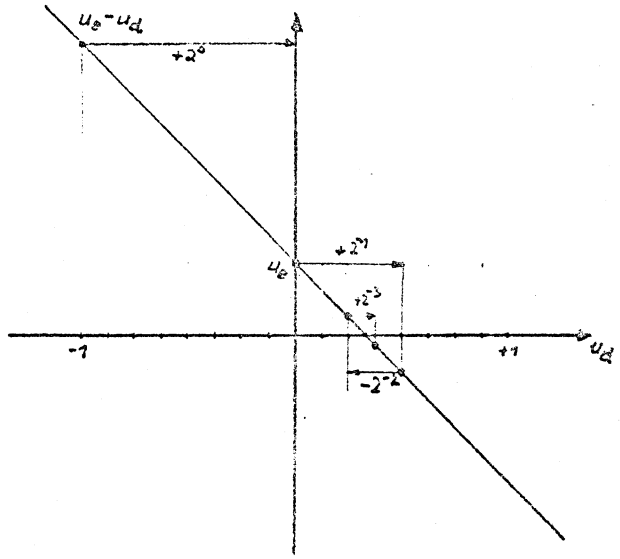


Bild 8.3 Zum Ablauf des Verfahrens der sukzessiven Approximation

Man kann nach diesem Prinzip den ganzen Wert wandeln. Dies stellt jedoch an den Komparator hohe Anforderungen. Liegt z.B. u_e sehr nahe bei 0.5, so ist bei der Bestimmung des nach dem Vorzeichen höchstwertigen Bits $u_e - u_d \approx 0$. Der Komparator muss seine Entscheidung mit der vollen Genauigkeit (z.B. 2^{-13} bei einem 14-Bit-Code) treffen, da sonst das Ergebnis im Verlauf der Wandlung nicht mehr korrigiert werden kann.

Das Zweibereichsverfahren (vergl. Bild 8.2.) hat hier günstige Eigenschaften. Ein Beispiel soll dies veranschaulichen. Der Eingangswert U_{in} wird im ersten Schritt in eine 8-stellige Dualzahl $U_{d,1}$ gewandelt. In einem hochgenauen DAU wird der Digitalwert, der im allgemeinen mit einem Fehler behaftet ist, in einen Analogwert zurückgewandelt und die Abweichung $U_{in} - U_{d,1}$ gebildet. Diese wird durch Umschalten des Eingangverstärkers um den Faktor $2^6 = 64$ verstärkt. Im zweiten Konvertierungsschritt wird dem Umsetzerkern der Wert

$$U_{e,z} = 64 \cdot (U_{in} - U_{d,1})$$

angeboten und in den Wert $U_{d,2}$ digitalisiert. Das Ergebnis ist dann

$$U_{in} = U_{d,1} + \frac{1}{64} \cdot U_{d,z} + \Delta U,$$

wobei ΔU den Konvertierungsfehler darstellt. Da im ersten Schritt die niedrigste ermittelte Stelle die Wertigkeit 2^{-7} hatte, überlappen sich die Teilergebnisse. Ein Fehler in der niedrigsten Stelle beim ersten Konvertierungsschritt wird also beim zweiten Schrict noch korrigiert. Man erreicht eine Genauigkeit von 2^{-13} mit einem Umsetzerkern der Genauigkeit 2^{-7}.

Die DAU's sind dazu vergleichsweise einfach aufgebaut (vgl. Bild 8.4). Sie bestehen aus einem Operationsverstärker mit einem Eingangsnetzwerk, dessen Widerstände in 2-er Potenzen abgestuft sind. Jeder Stelle der Dualzahl ist ein (elektronischer) Schalter des Bewertungswiderstandes zugeordnet, der dem Stellenwert entspricht.

Wird der Eingang des Netzwerkes nicht mit der Referenzspannung, sondern einem beliebigen Analogwert verbunden, so wird dieser mit dem Digitalwert multipliziert.
Da am Ausgang des DAU's der Wert so lange erhalten bleibt, bis in das zugeordnete Register ein neuer Digitalwert eingeschrieben wird, ergeben sich Treppenfunktionen am Ausgang. Mittels der in Bild 8.4. zusätzlich eingezeichneten Glättungseinheit ist es möglich, auch Polygonzüge zu erzeugen. Der zweckmässige Einsatz wird in einem späteren Kapitel erläutert.

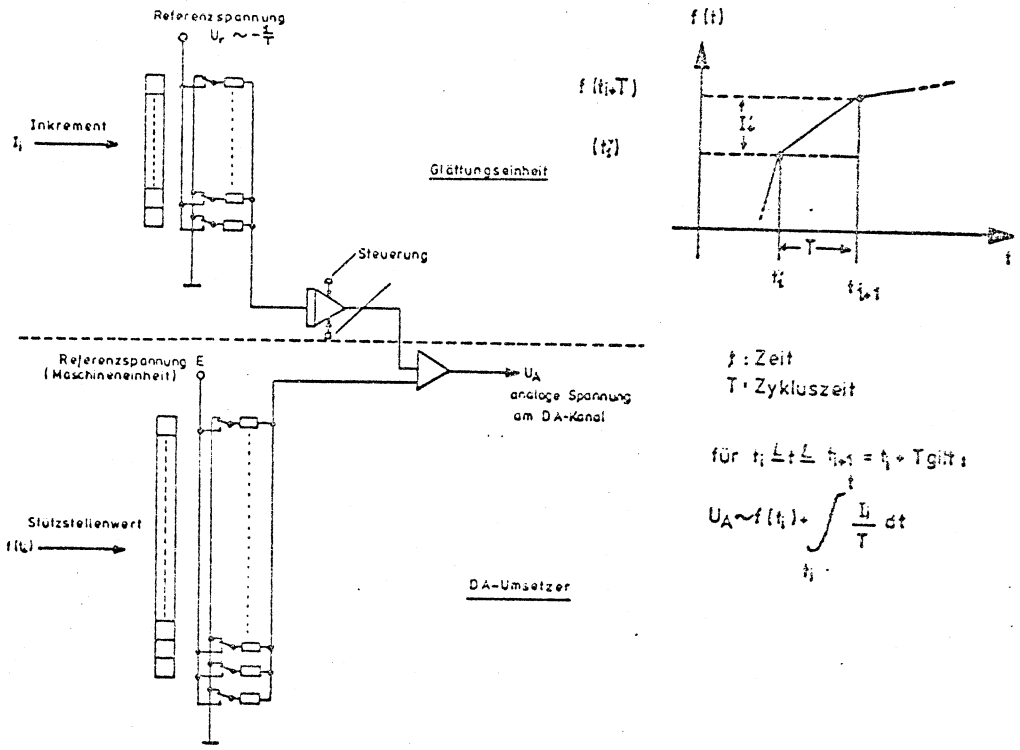


Bild 8.4. Prinzipschaltbild der DA-Umsetzung mit Glättung

Da im allgemeinen Werte zwischen verschiedenen Punkten des Programmierfeldes übertragen werden, müssen diese Werte auf die verschiedene Leitungsadressen verteilt werden.
Da ein genauer ADU teuer ist, so wird man diesen mit Zeitmultiplex mehrfach ausnutzen und den Verteiler auf die Leistungsadressen (Multiplexer) auf der Analogseite vorsehen (Bild 8.5). Dabei werden die Analogwerte der einzelnen Leitungen aber nicht gleichzeitig abgetastet(ΔT etwa 10μs). Dieser Zeitfehler kann durch Einfügen von Abtast und Halteverstärkern vermieden werden.

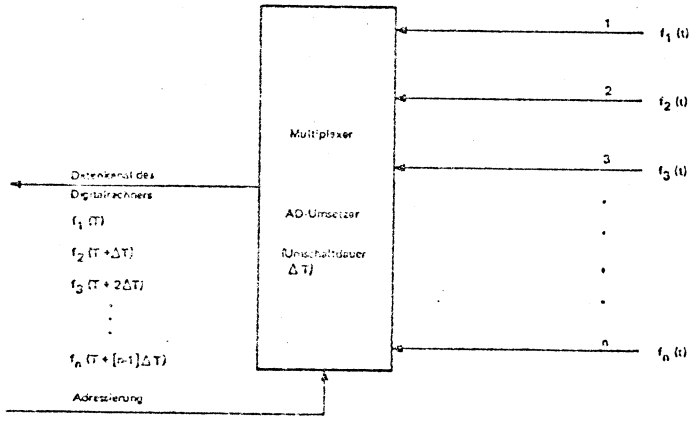


Bild 8.5. Anwahl der Leistungsadressen und Zeitfehler bei AD-Umsetzung

Diese halten nach Umschalten in die Betriebsart "Halten" die Werte an den Ausgängen konstant, bis alle Werte gewandelt sind (vgl. Bild 8.6).

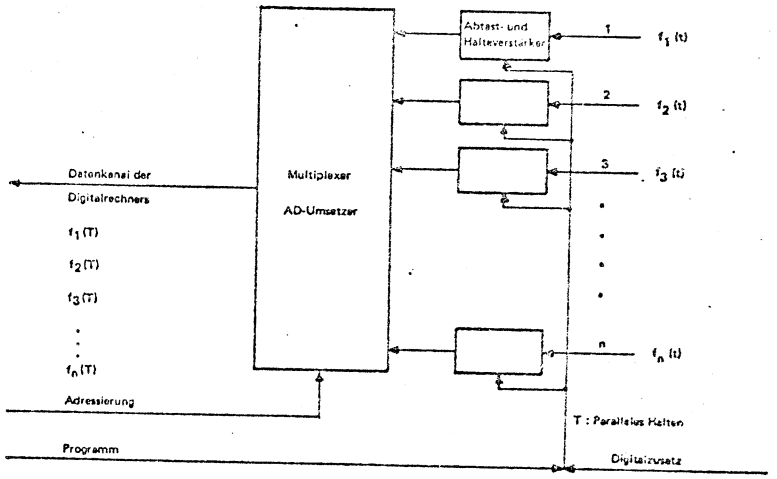


Bild 8.6. Verhinderung von Fehlern bei der AD-Umsetzung mit Hilfe von Abtast- und Halteverstärkern

In D/A-Richtung wird man für jede Leistung einen eigenen DAU vorsehen, da diese vergleichsweise preisgünstig sind. Die Verteilung der Daten auf die Leistungsadressen wird also auf der digitalen Seite erfolgen (vgl. Bild 8.7). Dabei entsteht ebenfalls eine Zeitverschiebung der Knickpunkte in Bild 8.4. in den einzelnen DAU's.
Durch doppelte Pufferung entsprechend Bild 8.8. kann auch diese Verschiebung vermieden werden.

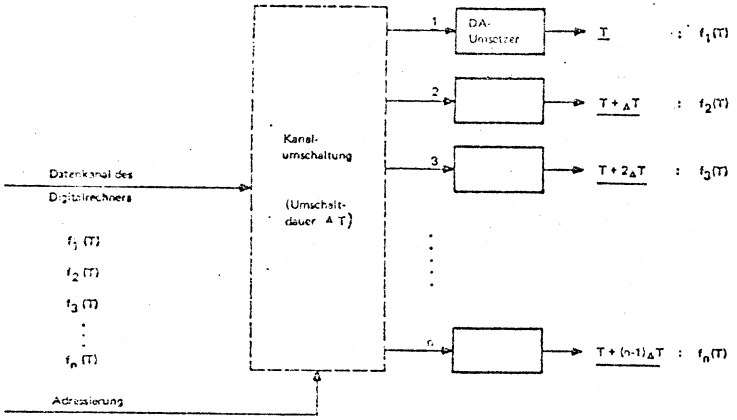


Bild 8.7. Zeitverschiebung bei DA-Umsetzung

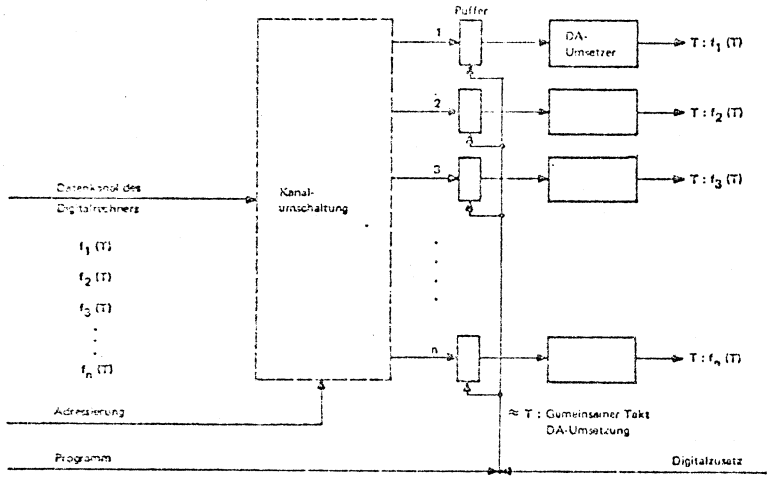


Bild 8.8. Verhinderung von Fehlern bei der DA-Umsetzung mit Hilfe der doppelten Pufferung

Damit nicht bei der Übertragung jedes einzelnen Wertes, die Leistungsadresse mit übertragen werden muss, sieht man die Möglichkeit eines blockweisen Datentransfers vor.
Dabei wird eine vorgegebene Anzahl von Leistungsadressen (Blocklänge) von einer gegebenen Anfangsadresse an bedient.
Ein Beispiel für den Ablauf der Datenübertragung ist in Bild 8.9. dargestellt.

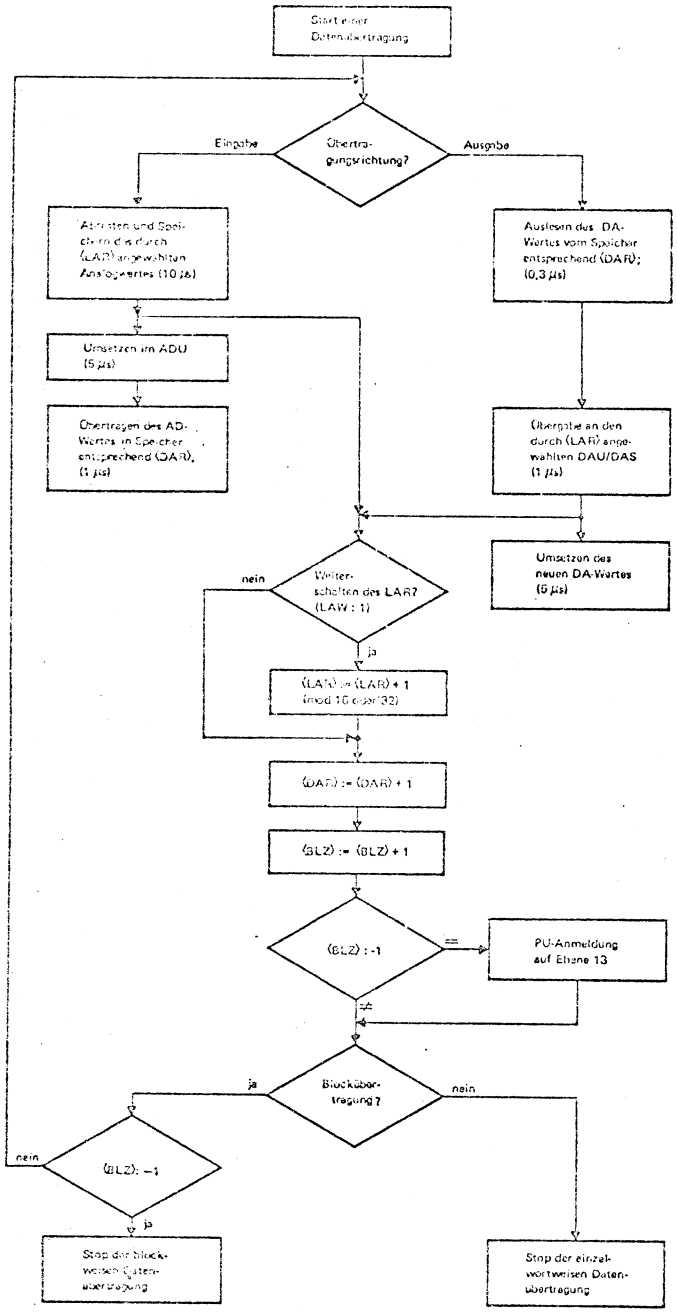


Bild 8.9. Ablauf der Datenübertragung über den Hybridkanal HRS 860

8.1.2 Die Steuerung des Analogrechners durch das Digitalprogramm

In einem hybriden Rechensystem erfolgt die Kontrolle des Ablaufs durch das Digitalprogramm. Dazu müssen die Bedienfunktionen (vgl. Kapitel 6), die normalerweise durch den Operateur am Analogrechner ausgeführt werden, durch das Digitalprogramm ausführbar sein. Entsprechende Steuer- und Anwahlbefehle müssen durch die Koppeleinrichtung decodiert und an das Bediengerät weitergegeben werden.

Benutzt man die Standard-Steuerprogramme, die im Bediengerät des Analogrechners realisiert sind, so muss das Digitalprogramm nach der Initialisierung Möglichkeiten haben,sich über den aktuellen Zustand des Steuerprogrammes zu informieren. Logische Variable, die den Zustand des Steuerprogrammes repräsentieren, müssen vom Digitalprogramm abgefragt werden können. In vielen Fällen -z.B. bei Optimierungsproblemen - wird der Wunsch bestehen, die Standard-Steuerprogramme zu modifizieren. Dies kann die individuelle Bestimmung der Zeiten für einzelne Programmphasen der Standard-Steuerprogramme sein, die normal durch die feste Einstellung der Zeitgeber am Analogrechner gegeben sind. Oder man möchte einzelne Integrieren oder Gruppen von Integrieren unabhängig vom Standard-Steuerungsprogramm betreiben. Insbesondere bei Optimierungsverfahren und bei der Aufzeichnung von Kurvenscharen wird dies häufig benötigt. Dazu müssen Werte von logischen Variablen auch vom Digitalrechner zum Analogrechner übertragen werden. Schliesslich können noch bei stückweise stetigen Funktionen in der Rechenschaltung Verzweigungen nötig sein. Als Beispiel dafür sei die Bewegung in inhomogenen Medien genannt wie das Eintauchen eines Körpers in Wasser. Beim Übergang des Körpers von Luft in Wasser ändert sich die Differentialgleichung des Bewegungsablaufes.

Bindeglied für diese Funktionen ist der Logikzusatz des Analogrechners. Dort befinden sich die Eingänge der Integrierer-Steuerschalter und die Zeitgeber . Zur freizügigen Realisierung vonVerzweigungen der Rechenschaltung ist der in Kapitel 3.5 beschriebene Komparator in zwei Teile zerlegt (vgl. Bild 8.10), nämlich den eigentlichen Komparator, dessen Ausgang eine logische Variable ist, und den Schalter, dessen Stellung durch eine logische Variable kontrolliert wird.

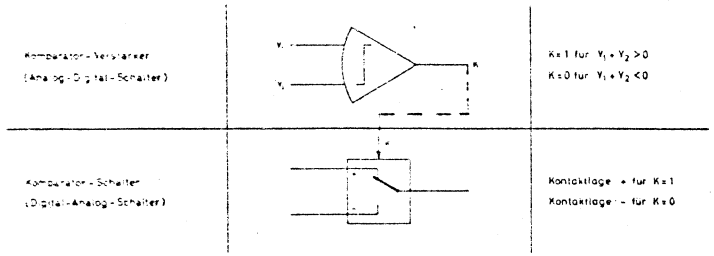
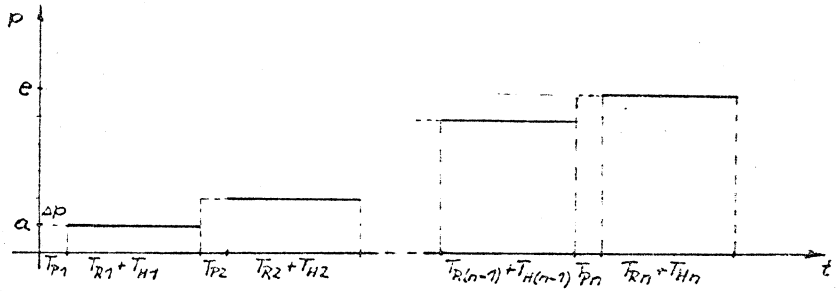


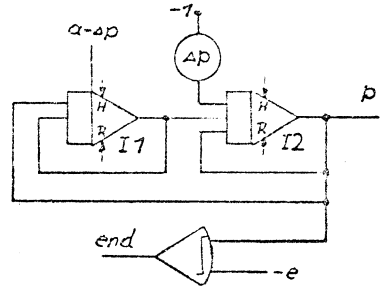
Bild 8.10 Aufteilung des Komparators in seine beiden Funktionen

Der Logikzusatz enthält ferner noch Elemente zum Aufbau kombinatorischer und sequentieller Netzwerke (Verknüpfungsglieder, Flipflops). Diese können am Programmierbrett des Logikzusatzes freizügig miteinander verbunden werden. Man hat damit zwei Möglichkeiten zur Realisierung eines individuellen Steuerungsablaufes bzw. beliebigen Verzweigungen der Rechenschaltung. Zum einen kann man den Ablauf durch ein Netzwerk entsprechend den Methoden zum Entwurf komb. und sequ. Netze realisieren. Dies erfordert u.U. viel Zeit für den Entwurf. Zum anderen kann man den Ablauf durch ein Digitalprogramm realisieren, muss dann aber die Werte der benötigten log. Variablen übertragen. Durch das Digitalprogramm und durch den Datentransfer wird der Digitalrechner belastet.

Ein vernünftiger Kompromiss wird darin bestehen, triviale Aufgaben am Logikzusatz direkt zu realisieren und den Digitalrechner dort einzusetzen, wo der Entwurf des Netzwerkes viel Arbeitsaufwand bedeutet. In Bild 8.11 (vgl. S.12) sind beide Möglichkeiten an einem Beispiel gegenübergestellt.



Verlauf des Parameters p über mehrere Rechenphasen zur Erzeugung einer einparametrigen Kurvenschar



Rechenphase	I 1			I 2		
	Zustand	H	R	Zustand	H	R
Pause	Hold	1	0	Run	0	0
Rechnen	Run	0	0	Hold	1	0
Halt		0	0		1	0
Standby	Init.Cond.	1	1	Init.Cond.	1	1

Bild 8.11 Erzeugung des Parameterwertes für einparametrige Kurvenscharen

Realisierung der Steuerung durch:

a) ein Steuerprogramm

```
10      if standby goto 50
        goto 10
50      hi1 =.true.
        ri1 =.true.
        hi2 =.true.
        ri2 =.true.

100     if pause goto 200
        goto 100
200     hi1 =.true.
        ri1 =.false.
        hi2 =.false.
        ri2 =.false.

        if end goto 1000
300     if rechnen goto 400
        goto 300
400     hi1 =.false.
        ri1 =.false.
        hi2 =.true.
        ri2 =.false.

500     if halt goto 100
        goto 500

1000    standby =.true.
        stop
```

b) parallele Logik

Rechenphase	Steuerleitung h	r	s
Standby	1	1	0
Pause	1	1	1
Rechnen	0	0	1
Halt	1	0	1

Verknüpfungen

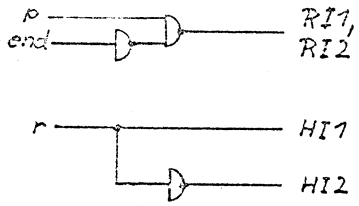


Bild 8.12 Steuerprogramm zur Steuerung der Integrierer in Bild 8.11

8.1.3 Die Synchronisation durch Programmunterbrechung

Bereits im Beispiel von Bild 8.11 bzw. 8.12 wurde eine Synchronisation des Digitalprogrammes mit den Rechenphasen des Analogrechners durch Abfragen von logischen Variablen erreicht. Dazu müssen die Abfragen an geeigneten Stellen des Programmes wiederholt werden, bis der Zustand, nach dem gefragt wurde, eingetreten ist. Die Reaktionszeit auf diesen neuen Zustand ist von dem zeitlichen Abstand der Abfragen abhängig. Werden sehr kurze Reaktionszeiten verlangt, so ist der Rechnerkern durch die Wiederholung der Abfragen voll ausgelastet.

Wesentlich effektivere Möglichkeiten zur Synchronisation bietet ein entsprechend ausgebautes Programmunterbrechungswerk. Da mit Unterbrechungssignalen in geringem zeitlichen Abstand gerechnet werden muss, und die einzelnen Unterbrechungsursachen hinsichtlich Schwankungen der Antwortzeit unterschiedlich empfindlich sind, ist es nicht möglich, das Unterbrechungswerk für die Dauer der Interruptroutinen global zu sperren (durch Setzen des Merklichts M im Bild 8.13). Die Folge davon wäre eine starke Schwankung der Antwortzeit durch Aktivitäten auf Interruptebenen niedrigerer Priorität. Man muss vielmehr erreichen, dass eine Interruptroutine nur durch Unterbrechungen auf Ebenen höherer Priorität unterbrechbar ist, und jeder Unterbrechungsebene eine entsprechende Priorität des Unterbrechungsprogrammes zuweisen. Dies kann durch Führen einer Prioritätsliste durch Software-Steuerung erfolgen. Die Unterbrechungswünsche werden hier in einer hardwareseitig festinstallierten Prioritätenfolge gespeichert. Sofern sich die Anlage im Zustand der Unterbrechbarkeit befindet, das Merklicht M also gelöscht ist (M = 0), führt der Unterbrechungswunsch höchster Priorität zu einem Interrupt des gerade laufenden Programms. Damit wird gleichzeitig der Unterbrechungswunsch gelöscht. Der Zustand der

Unterbrechbarkeit wird aufgehoben durch Setzen des Merklichts M (M = 1), der Befehlszählerstand des unterbrochenen Programms wird in einer der Priorität zugeordneten Speicherzelle abgelegt, und der in der nachfolgenden Speicherzelle stehende Befehl wird ausgeführt. Diese einem Interrupt zugeordneten Speicherzellen werden Interruptebene - kurz Ebene - genannt, und die zugeordnete Priorität wird als Hardwarepriorität bezeichnet. In der Interruptverwaltung wird jedoch eine softwareseitig vorgebbare Prioritätsliste mit dem Eingangsindex p und den nachfolgend definierten Elementen Ep geführt. Der Index p stellt eine Softwarepriorität - nachfolgend kurz Priorität genannt - dar. Ein nicht an die Interruptverwaltung angeschlossenes Objektprogramm hat in diesem Zusammenhang die niedrigste Priorität. Ein Element Ep kann folgende Zustände annehmen:

LEER: An dem Element ist kein Interruptprogramm angeschlossen (Startadresse des Programms fehlt)

WARTE: An dem Element ist ein Interruptprogramm angeschlossen. Dieses Programm wartet auf einen Startauftrag durch einen externen oder internen Interrupt. Ein externer Interrupt wird von einem Gerät, ein interner von einem Programm erzeugt (siehe unten)

START: Ein Interruptprogramm (oder Nebenprogramm,siehe unten) ist zum Start gemäß der Priorität p angemeldet

AKTIV: Das Interrupt- oder Nebenprogramm befindet sich im arbeitenden Zustand

UNTBR: Das Interrupt- oder Nebenprogramm wurde im Zustand AKTIV durch einen externen oder internen Interrupt unterbrochen, und die aktuellen Registerbestände sind gekellert

Die Interruptverwaltung wird hauptsächlich von den Geräte-Programmen des Systems benutzt. Für den normalen Anwendungsfall tritt sie nicht in Erscheinung. Zum Zwecke der Installation besonderer Anwendungsfälle im System stellt sie jedoch folgende Dienstleistungen zur Verfügung :

Eine Interruptebene der Hardwarepriorität p_1 kann an die Interruptverwaltung unter einer anderen Priorität p_2 angekoppelt werden. Gleichzeitig kann ein Interruptprogramm unter Ep_2 im Zustand WARTE angeschlossen werden.
Trifft dann ein Interrupt auf der Ebene p_1 ein, so wird wenn:

a) das laufende Programm höhere Priorität hat als es p_2 entspricht, Ep_2 in den Zustand START überführt und das laufende Programm fortgesetzt,

b) das laufende Programm niedrigere Priorität hat als es p_2 entspricht, das laufende Programm unterbrochen, d.h. vom Zustand AKTIV nach UNTBR und das Interruptprogramm (Ep_2) vom Zustand WARTE nach AKTIV überführt.

Ein aktives Programm hat sich mit einem Aufruf von der Interruptverwaltung abzumelden. Es wird dadurch von START nach WARTE überführt. Anschließend wird dasjenige Programm höchster Priorität nach AKTIV überführt, das sich im Zustand START oder UNTBR befand.

Während des Programmlaufs kann durch Aufruf der Interruptverwaltung ein interner Interrupt erzeugt und diesem eine Priorität zugewiesen werden. Dieser Aufruf wirkt so, als wäre ein externer Interrupt zum Zeitpunkt des Aufrufes eingetreten.
Durch Aufruf der Interruptverwaltung können ferner Elemente Ep vom Zustand WARTE in START überführt werden. Dies wird die Anmeldung von Nebenprogrammen genannt.

Die Interruptverwaltung durch Software hat den Nachteil, daß das Durchsuchen der Prioritätsliste viel Zeit beansprucht, wenn eine grössere Anzahl von Softwareprioritäten vorgesehen wird. Von Vorteil ist die Möglichkeit, die Prioritäten der angeschlossenen Programme zu ändern und das Erzeugen interner Interrupts.

Um die Belegung des Rechnerkernes durch die Interruptverwaltung zu vermeiden, wird man eine hardwaremässige Steuerung der Prioritäten der Unterbrechungsprogramme vorsehen. Dazu verfügt das modifizierte Programmunterbrechungswerk für das Hybride Rechnersystem HRS 860 (siehe Bild 8.13) über insgesamt 28 PU-Ebenen, denen die Indizes 1 - 28 zugeordnet sind, und deren Priorität mit steigendem Index abnimmt. Die PU-Wünsche laufen asynchron in das PUW ein und werden in ein Anmelderegister aufgenommen. Die Aufnahme wird asynchron und für die Ebenen 5 bis 28 entsprechend der Einstellung des vorgeschalteten Masken-Registers bedingt vorgenommen. Die Einstellung des Masken-Registers erfolgt durch den Rechnerkern mittels entsprechender Befehle. Aus dem Anmelde-Register wird die Information zu bestimmten Zeitpunkten in ein Auswert-Register übernommen.

Die Prioritätsentschlüsselung erfolgt durch eine Verknüpfungsschaltung, die das Auswert-Register ständig (d.h. ohne Zeitbedingung abfragt und deren 28 Ausgänge den vorrangigsten PU-Wunsch bzw. die niedrigste gesetzte Stelle des Auswert-Registers angeben ("1 aus 28"-Code). Die Weiterverarbeitung der Information speziell die Startbedingung für das PUW, ist abhängig von der Priorität der Programmunterbrechung.
Bei Programmunterbrechungen auf der Ebene 3 bis 28 wird das Merklicht M automatisch hardwaremäßig vom Programmunterbrechungswerk gesetzt, das damit das anlaufende Unterprogramm gegen erneute Unterbrechungen schützt. Der Programmierer kann diesen Schutz jederzeit durch ein "Lösche Merklicht" - Befehl aufheben. Spätestens bei Ende des Unterprogramms erfolgt eine Aufhebung der Sperre durch den "Lösche Merklicht"-Befehl in der ersten der drei Speicher-Festzellen (siehe unten).

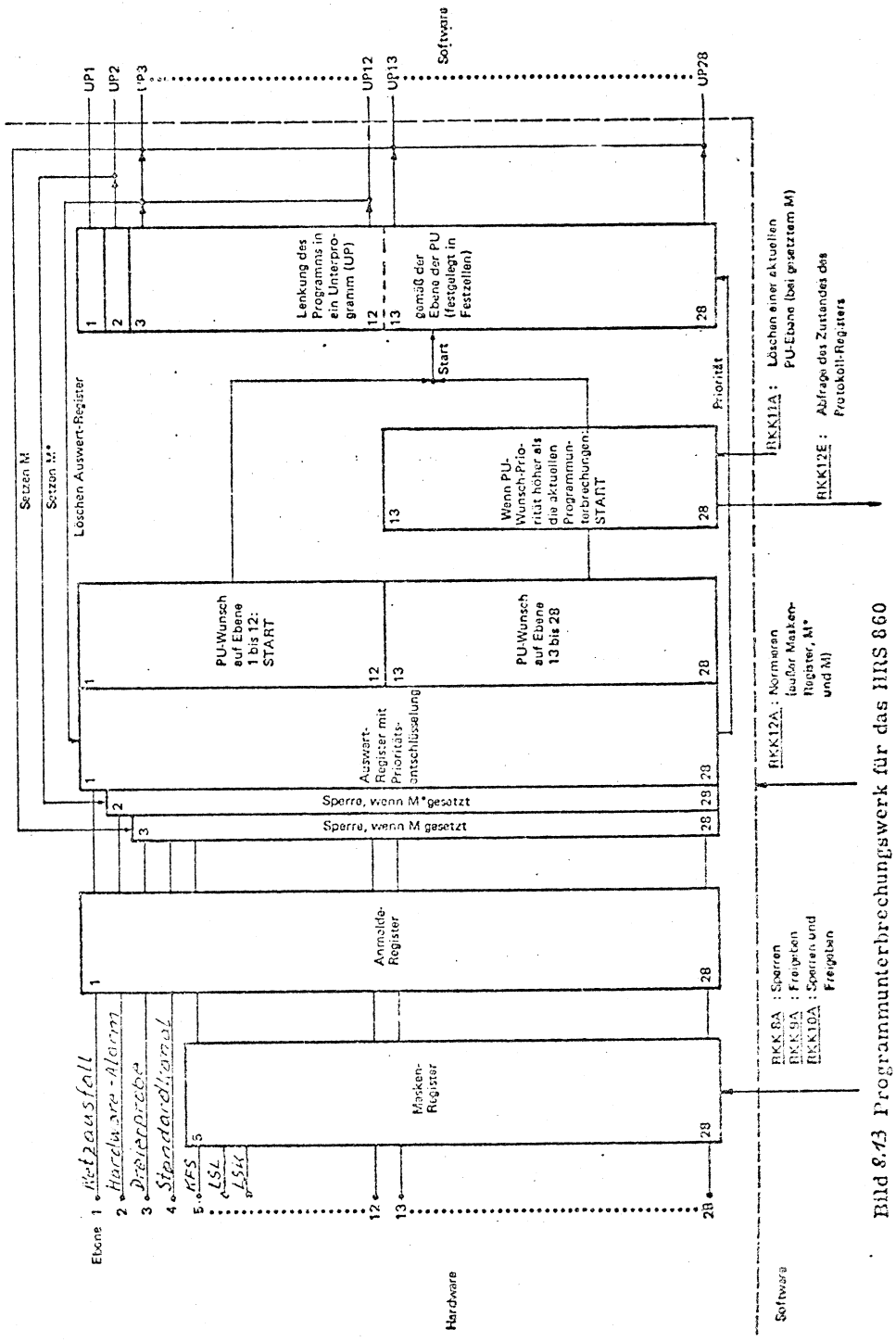


Bild 8.13 Programmunterbrechungswerk für das HRS 860

Unterbrechungsebenen 13 bis 28 (für den Betrieb im HRS 860):

Bei einer Unterbrechung auf einer der Ebenen 13 bis 28 wird das Ergebnis der Prioritätsentschlüsselung in ein Protokoll-Register übernommen, das die Aufgabe einer Buchführung über angefangene und noch nicht beendete (=aktuelle) PU's hat. Ein in das Auswert-Register gelangter PU-Wunsch wird also nur dann in das Protokoll-Register aufgenommen, wenn er eine höhere Priorität als alle bereits im Auswert-Register registrierten PU-Wünsche hat, da sich andernfalls die Ausgänge der Prioritätsentschlüsselung nicht ändern würden. Ein Setzen im Protokoll-Register ist hierbei das Startsignal für das PUW. Nach dem Startsignal sperrt das PUW das Auswert-Register gegen weitere PU-Wünsche ab, so daß eine erneute Unterbrechung bei aktiviertem PUW verhindert wird, und eine Simulierung des Unterprogramm-Sprungbefehls ungestört ablaufen kann. Hierbei wird die Sprungadresse durch die Prioritätsentschlüsselung erzeugt, d.h. der Sprung erfolgt nach Speicherzellen, die den einzelnen Unterprogramm-Ebenen fest zugeordnet sind. Nach der Bildung der Sprungadressen werden im Auswert-Register alle Stellen mit Ausnahme der durch das Protokoll-Register markierten Stellen gelöscht. Dadurch bleibt die Stellung der Prioritätsentschlüsselung erhalten, so daß - falls kein PU-Wunsch höherer Priorität in das Auswert-Register gelangt - keine erneute PU erfolgen kann.

Soll ein Programmunterbrechungswunsch höherer Priorität eine Unterbrechung des laufenden Unterprogramms hervorrufen, muß der Programmierer durch einen "Lösche Merklicht"-Befehl für die Aufhebung der automatischen Unterbrechungssperre durch das Merklicht M sorgen, falls dies nicht durch hybride Bibliotheksunterprogramme geschiet.

Unterbrechungsebenen 3 bis 12 (in der Regel für Standart-Peripherie):

Der Unterschied der Behandlung von Unterbrechungswünschen auf der Ebene 3 bis 12 zu denen auf den Ebenen 13 bis 28 besteht darin, daß kein Protokoll über die aktuellen Programmunterbrechungen geführt wird.

Der Start des PUW erfolgt direkt durch das Auswert-Register, das im Gegensatz zu oben nach der Erzeugung der Sprungadresse, die auch hier als Ergebnis der Prioritätsentschlüsselung des Auswert-Registers entsteht, völlig, d.h. in allen Stellen gelöscht wird. Damit springt die Prioritätsentschlüsselung in die Neutralstellung. Ein danach in das Auswert-Register gelangender PU-Wunsch führt ungeachtet der Priorität des gerade laufenden Programms eine Unterbrechung herbei, falls keine Sperre gesetzt ist.

Unterbrechungsebenen 1 und 2:

Die Unterbrechungsebenen 1 (Netzspannungsausfall, "Aus"- und "Stop"-Taste) und 2 (EA-Fehleralarm des Rechnerkernkanals und Programmschutzfehler) nehmen gegenüber den anderen Unterbrechungsebenen eine Sonderstellung ein, da sie nicht softwaremäßig mit dem Merklicht M sperrbar sind. Da sie im übrigen wie die Ebenen 3 bis 12 behandelt werden, führen PU-Wünsche auf den Ebenen 1 und 2 in jedem Falle sofort eine Unterbrechung herbei.

Den Unterbrechungsebenen 3 - 28 des PUW sind je 3 aufeinanderfolgende Festzellen im Kernspeicher zugeordnet. Der vom PUW erzeugte Unterprogramm-Sprung erfolgt in die zweite Festzelle, in der durch Abläufe im SU-Befehl die Rücksprungadresse in das unterbrochene Programm abgelegt wird. In der folgenden (dritten) Festzelle steht ein unbedingter Sprung in das eigentliche Unterprogramm. Der letzte Befehl des Unterprogramms muß ein Sprungbefehl in die erste Festzelle sein, in der ein "Lösch Merklicht"-Befehl steht. Nach Ausführung dieses Befehls wird die zweite Festzelle ausgelesen, und es erfolgt der Rücksprung in das unterbrochene Programm.

Bei den Ebenen 1 und 2 fehlt die Festzelle mit dem "Lösche Merklicht"-Befehl. Sie haben nur je 2 Speicherzellen. Unterprogramm-Sprung und Rücksprung erfolgen in die erste der beiden Zellen.

Per Befehl lassen sich am Programmunterbrechungswerk noch folgende Funktionen durchführen:

Löschen im Protokoll-Register:

Da auf den PU-Ebenen 13 bis 28 die jeweils aktuellen Ebenen (also angelaufene und noch nicht beendete Unterprogramme) im Protokoll-Register registriert werden, ist am Ende des Unterprogramms die entsprechende Stellen in diesem Register zu löschen. Dies erfolgt über den Rechnerkanal.

Dieser Befehl wird nur bei gesetztem Merklicht M richtig ausgeführt.

Normieren des Programmunterbrechungswerkes:

Das PUW kann unabhängig von der Rechnernormierung über den Rechnerkanal 12 normiert werden. Dabei wird das Auswertungsregister gelöscht. Dies wird benötigt, wenn das Unterbrechungsprogramm nicht an die Unterbrechungsstelle zurückkehrt, sondern zur Synchronisation den Hauptprogramm an anderer Stelle fortaetzen soll.

Abfrage des Protokoll-Registers:

Der Zustand des Protokoll-Registers kann über den Rechnerkernkanal eingelesen werden. Damit lassen sich z.B. für Prüfzwecke die aktuellen Programmunterbrechungen ermitteln.

Zur Zusammenarbeit von Digital- und Analogrechner:
wird die unabhängige Variable, die Zeit, mit Hilfe der vom Digitalrechner aus steuerbaren Einheiten

- Zykluszeit-Register (ZZR),
- Zykluszeit-Zähler (ZZZ) und
- einer Steuereinheit

diskretisiert.

Der prinzipielle Aufbau der Zeitsteuerung ist in Bild 8.14 gezeigt. Zweck der Einrichtung ist das Erzeugen von Programmunterbrechungen auf Ebene 16 an den Diskretisierungspunkten der Zeitachse. Digitale Funktionsberechnungen erfordern ständige Wiederholung der Berechnung für jeden Diskretisierungspunkt. Die Zeit für die Berechnung eines Wertes darf den Abstand der Diskretisierungspunkte (Zykluszeit) nicht überschreiten.

Der Zykluszeit-Zähler ist ein 7-Bit-Dualzähler, der mit einem vom Zykluszeit-Register übernommenen Anfangswert beginnend rückwärts zählt, bis er Null erreicht hat. In diesem Augenblick gibt er ein Signal für Zykluszeit-Ende (ZZE) ab, übernimmt sofort wieder den Startwert vom ZZR und beginnt erneut in Richtung Null zu zählen. Das ZZE-Signal meldet eine Programmunterbrechung auf Ebene 16 an, wenn keineSperre durch das Masken-Register des Programmunterbrechungswerkes gesetzt ist.
Die Zählerfrequenz des ZZZ wird zusammen mit seinem Anfangswert vom Inhalt des ZZR festgelegt, und zwar mit zwei Bitstellen für die Auswahl einer der Takt-Frequenzen

- 10 kHz
- 1 kHz
- 100 Hz
- frei wählbar, am Digital-Programmierfeld (DPF) aufschaltbarer "externer" Zählertakt.

Zählwert und Bitkombination für die Auswahl der Zählfrequenz lassen sich in das ZZR mit einem Ausgabe-Befehl über Rechnerkernkanal einschreiben.

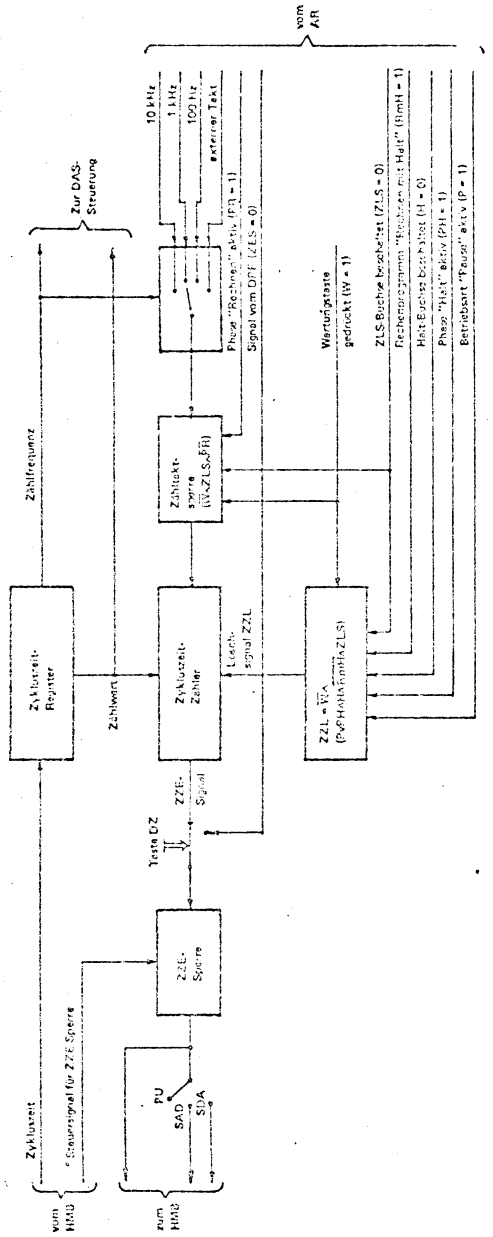


Bild 8.14 Blockschaltbild der Zeitsteuerung

8.2 Organisation des Rechenablaufes

In Kapitel 2 ist zwischen alternierender und simultaner Arbeitsweise von Digital- und Analogrechnern unterschieden worden. Bei der alternierenden Arbeitsweise befindet sich stets einer der beiden Rechner im wartezustand. Typische Anwendungsfälle sind die Optimierung und die Lösung von Randwertproblemen. Der Analogrechner übernimmt die Lösung der Zustandsgleichungen mit hoher Rechengeschwindigkeit, während der Digitalrechner die Berechnung der Einstellwerte für den nächsten Rechenzyklus übernimmt. Dies geschieht im allgemeinen in der Halt- bzw. Pausenphase des Analogrechners, während in der Phase "Rechnen" der Digitalrechner im Zustand "warten" ist.

Die Zeitsteuerung der Rechenphasen wird durch die zentrale Steuerung des Analogrechners vorgenommen. Dies ist für den Programmierer einfacher und es entfallen mögliche Zeitfehler durch die begrenzte Verarbeitungsgeschwindigkeit des Digitalrechners. Durch das Digitalprogramm erfolgt die Wahl der Betriebsart und der Start des Analogrechners. Es ist eine Synchronisation mit den Rechenphasen des Analogrechners (Phasensynchronisation) erforderlich, da zu jeder Rechenphase ein besonderer Zweig des Digitalprogrammes gehört. Bei Phasenwechsel muss der Zweig, der zur neuen Phase gehört, angesprungen werden. Ein Standardablauf für eine grosse Klasse von Problemen ist:

Rechenphase des Analogrechners	Dienst des Digitalrechners
Halt, Pause	Übernahme von Werten, die am Ende der vorhergehenden Rechenphase erreicht wurden; Berechnung und Übergabe der Parameterwerte für den neuen Rechenzyklus
Rechnen	warteschleife

Zur Phasensynchronisation dienen offene PU-Programme. Dabei wird das unterbrochene Programm nicht mehr fortgesetzt. Das offene PU-Programm ist vielmehr ein Teilstück des Hauptprogramms, das mit Hilfe der Unterbrechung auf einen extern ablaufenden Vorgang (in unserem Falle, den aktiven Analogrechner) synchronisiert wird. Es werden bei Benutzung der standardmässigen Programmsteuerung während der aktiven Phase des Analogrechners, die Unterbrechungen 'AR rechnet' und 'AR halt' dazu benutzt, das Hauptprogramm mit dem Phasenablauf des Analogrechners zu synchronisieren. Das heisst, dass durch Unterprogramme (ASTART, ANZEIGE, PHASW., PHALR) offene PU-Programme an die Ebenen für 'AR rechnet' bzw. 'AR halt' angeschlossen und sofort nach Eintreffen der Unterbrechung angesprungen werden. Nur so kann man den extremen Zeitbedingungen des Analogrechners gerecht werden.

Somit ist aber zwingend erforderlich, dass bei Eintritt in die aktive Phase und bei jedem Phasenwechsel jeder sonstige E/A-Verkehr ruht, um eine durch die PU's 'AR rechnet' oder 'AR halt' unterbrochene E/A-Routine nicht zu Ende geführt wird.

Bei der simultanen Arbeitsweise sind während der Phase "Rechnen" beide Rechner aktiv. Der Digitalrechner kann wegen der seriellen Arbeitsweise und der endlichen Arbeitsgeschwindigkeit nur diskrete Funktionen verarbeiten, während der Analogrechner kontinuierliche Funktionen über der Zeit als unabhängige Variable verarbeitet. Zur Anpassung muss die Zeitachse diskretisiert werden. Dies geschieht durch den Zykluszeitgeber. Dieser erzeugt zu aquidistanten Zeitpunkten eine Programmunterbrechung. Dadurch wird die Übernahme von Werten zu den Zeitpunkten t_i (i = 0, 1, ---, n) in den Digitalrechner veranlasst. Da die Berechnung von Funktionswerten

im Digitalrechner Zeit benötigt, stehen die berechneten Funktionswerte $z(t_i)$ erst zur Zeit $t_i + t_v$ für die Übertragung an den Analogrechner zur Verfügung. Neben der Diskretisierung entsteht also ein weiterer systembedingter Fehler durch die Zeitverzögerung.

Die Verzögerungszeit ist durch den Zeitbedarf des Digitalrechners bestimmt; dieser bestimmt auch den minimalen Abstand der Diskretisierungspunkte (Zykluszeit Δt). Da man im Interesse kleiner systembedingter Fehler mit der kleinstmöglichen Zykluszeit arbeiten wird, ergibt sich, dass zum Diskretisierungszeitpunkt t_{i+1} die Funktionswerte $z(t_i)$ an der Stelle t_{i+1} übergeben werden (vgl. Bild 8.15).

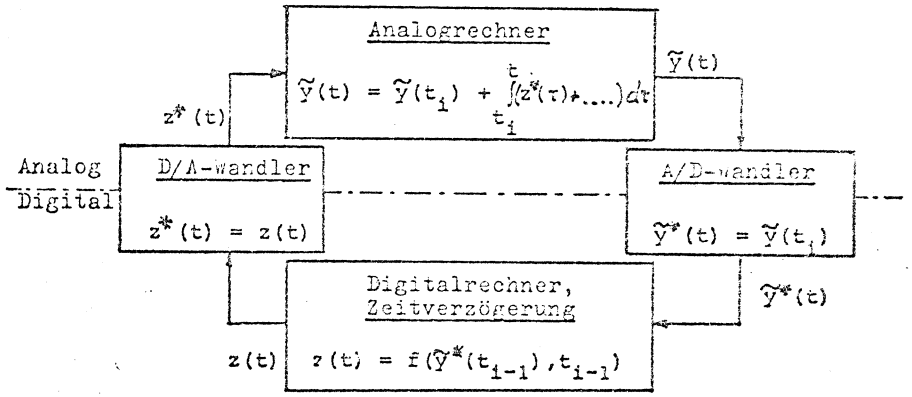


Bild 8.15 Arbeitsschema eines Hybridsystems für das Zeitintervall $t_i \leq t < t_{i+1}$.

Zur Korrektur des Zeitfehlers wäre eine Zeitverschiebung der Funktionen nötig. Dies ist ohne Verstoss gegen das Kausalitätsprinzip *nicht exakt durchführbar*. Eine angenäherte Korrektur ist jedoch möglich, wenn man an Stelle von $\tilde{y}(t_i)$ einen Näherungswert für $\tilde{y}(t_{i+1})$ zur Zeit t_i an den Digitalrechner übergibt. Entwickelt man $\tilde{y}$ an der Stelle t_i in eine Taylor-Reihe, so erhält man

$$\tilde{y}(t_{i+1}) = \tilde{y}(t_i) + \tilde{y}'(t_i) \cdot \Delta t + \ldots .$$

Damit ergibt sich als Näherungswert

$$\tilde{y}(t_{i+1}) \approx \tilde{y}(t_i) + \tilde{y}'(t_i) \cdot \Delta t.$$

In vielen Fällen wird man den Wert der Ableitung $\tilde{y}'(t_i)$ aus der Rechenschaltung entnehmen können (vgl. Bild 8.1b), so dass die Korrektur zweckmässig am Analogrechner vorgenommen wird; sie kann aber auch im Digitalrechner durchgeführt werden.

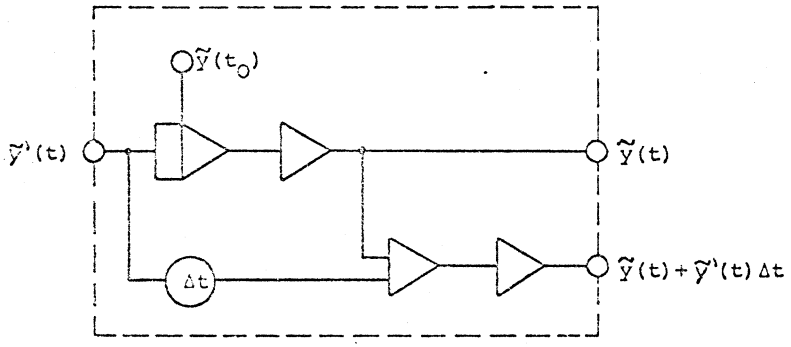


Bild 8.1b Schaltung am Analogrechner zur Korrektur der Verzögerungszeit im digitalen Teil der hybriden Schleife.

8.3 Abschätzung des systembedingten Fehlers

Die analogen Rechenelemente vermitteln eine Abbildung der Eingangsfunktionen in die Ausgangsfunktion im Raum der reellen stetig (in Sonderfällen stückweise stetigen) beschränkten Funktionen. Da numerische Prozesse Operationen mit Werten von Variablen bedeuten, muss die unabhängige Problemvariable (Zeitachse) diskretisiert werden. Die kontinuierlichen Funktionen müssen durch diskrete Funktionen ersetzt werden. Für ein endliches Zeitintervall kann man diese durch eine endliche Anzahl von Variablen repräsentieren. Für die Zusammenarbeit dieser verschiedenen Darstellungen ist also nicht nur die Wandlung einzelner Werte von der digitalen Darstellung in die analoge und umgekehrt nötig sondern die Abbildung diskreter Funktionen in kontinuierliche und umgekehrt. Digitale Abbildungen einerseits und analoge anderseits stellen demnach Operatoren in verschiedenen Räumen dar. Zur Zusammenarbeit benötigt man Operatoren, welche diese Räume ineinander abbilden.

Die Abbildung zeitkontinuierlicher Funktionen in diskrete Funktionen wird man in natürlicher Weise so vornehmen, dass der Funktionswert im Diskretisierungspunkt übernommen wird. Dazu werden die zu wandelnden analogen Funktionen in äquidistanten Abständen (Zykluszeit) abgetastet und der Analogwert digitalisiert. Die Zykluszeit muss so gewählt werden, dass nach deren Ablauf der Digitalrechner mit Sicherheit zur Verarbeitung der neuen Werte bereit ist. Sie hägt also von dem Umfang der in einem Zyklusintervall zu bearbeitenden numerischen Operationen ab.

Die Zyklusintervalle werden durch einen entsprechend genauen Taktgeber bestimmt. Die Operationen am Digitalrechner müssen damit synchronisiert werden. Hybride Rechnersysteme verfügen über die entsprechende Hard- und Software. Informationstheoretisch genügt es, die Zykluszeit entsprechend dem Abtasttheorem zu wählen; wegen der weiter unten hergeleiteten

Konsistenzbedingung muss man aber wesentlich kürzere Zykluszeiten wählen. Ist die Diskretisierung der kontinuierlichen Funktionen verhältnismässig unproblematisch, so gilt das nicht für die Konstruktion kontinuierlicher Funktionen aus diskreten. Der lokale Verlauf zwischen zwei Diskretisierungs - punkten muss aus dem globalen Verlauf sinnvoll konstruiert werden. Dies ist eine Interpolationsaufgabe. Für lineare Interpolation erhält man z.B. eine stetige Funktion, die allerdings in den Stützstellen nicht differenzierbar ist. Stellt man höhere Anforderungen an die Glattheit der Funktion, so kann man aufwendigere Interpolationsverfahren verwenden.

Interpolationsverfahren erfordern, dass für die Bestimmung des Verlaufes zwischen t_i und t_{i+1} zumindest der Wert für t_{i+1} (z.B. bei linearer Interpolation) bekannt sein muss. Berücksichtigt man, dass analoge und digitale Operationen im allgemeinen eine geschlossene Schleife bilden (vgl. Bild 8.17), so ist ersichtlich, dass dies nur erfüllbar ist, wenn man eine Zeitverzögerung der gewandelten Funktionen in Kauf nimmt. In Bild 3.1 ist auch bereits eingetragen, dass durch den Zeitbedarf des Digitalrechners ohnedies eine Zeitverzögerung von einer Zykluszeit entsteht.

Interpolation:

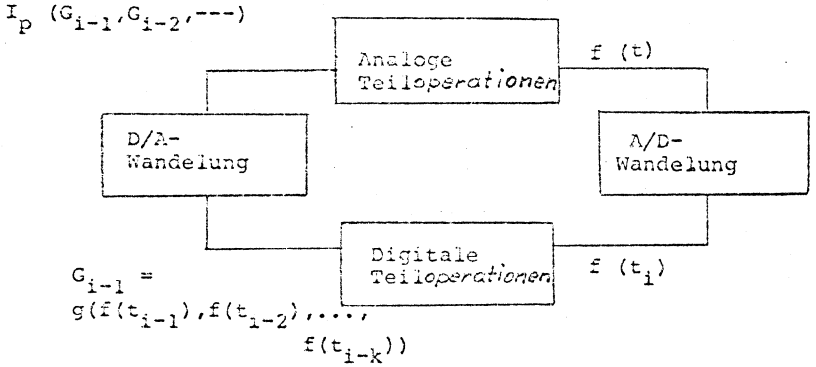
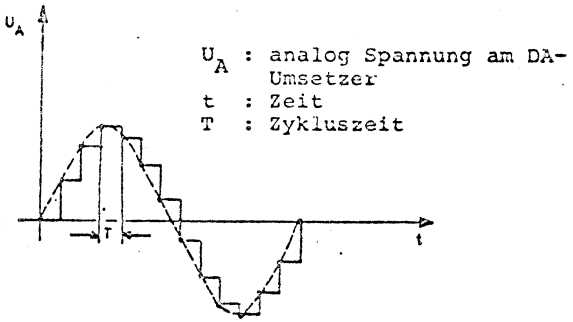
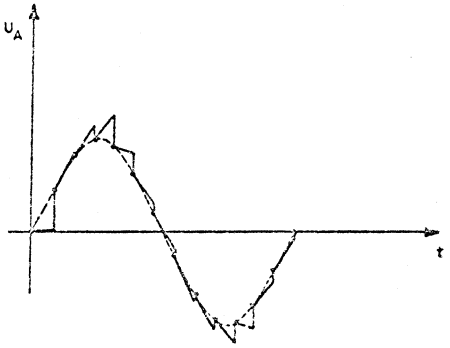


Bild 8.17 Geschlossene Schleife einer hybriden Simulation zur Zeit $t_i \leq t < t_{i+1}$.

An Stelle der Interpolation werden häufig daher Extrapolationsverfahren angewendet . Es sind zwei Verfahren gebräuchlich: die Verwendung von Treppen- oder Rampenfunktionen. Bei der Anwendung von Rampenfunktionen wird im digitalen Teil die Rampensteilheit zusätzlich als Näherungswert der Steigung der idealen Ausgangsfunktion berechnet. Bild 8.18 stellt beide Verfahren für eine Sinusfunktion als ideale Ausgangsfunktion gegenüber.



Extrapolation durch Treppenfunktion



Extrapolation durch Rampenfunktion

Bild 8.18 Extrapolation mittels Treppen- und Rampenfunktion

Bei beiden Verfahren erhält man Unstetigkeiten in den Stützstellen. Welches Verfahren besser ist, hängt vom Steigungs- und Krümmungsverhalten der idealen Funktion ab. Die Unstetigkeit wird in vielen Fällen weiter nicht stören z.B. wenn über diese Funktion integriert wird. Besonders stört jedoch diese Unstetigkeit, wenn dadurch Eigenfrequenzen von Teilsystemen angeregt werden. Zur Beseitigung der Unstetigkeiten verwendet man Tiefpässe, die jedoch wegen deren Einschwingverhalten Verzögerungszeiten bedeuten, so dass man sich den Verhältnissen bei der Verwendung von Interpolationsverfahren nähert.

Die Zeitverzögerung durch den Zeitbedarf des Digitalrechners und die Verfahren zur Funktionsrekonstruktion spielen eine besonders unangenehme Rolle. Aus der numerischen Analysis ist eine Konsistenzbedingung bekannt, die eine Aussage darstellt, unter welcher Bedingung die numerische Lösung gegen die exakte Lösung konvergiert, wenn die Schrittweite gegen Null strebt. Analog dazu wird man hier fordern, dass die hybride Lösung gegen die exakte Lösung konvergiert, wenn man nur die Zykluszeit beliebig klein wählt.
Eine derartige Konsistenzbedingung für die hybride Lösung von Differentialgleichungen lässt sich formulieren.
Gegeben sei das System von Differentialgleichungen

$$y'(t) = f(y(t),t); \quad y(t_o) = y_o . \tag{1}$$

Die Lösung von (1) wird dann dargestellt durch die Vektorfunktion

$$y(t) = y(t_o) + \int_{t_o}^{t} f(y(\tau),\tau)d\tau \tag{2}$$

Da die Näherungslösung $\tilde{y}$ (t) im allgemeinen Fall als kontinuierliche Funktion zwischen den Stützstellen t_i ermittelt wird, und die Lösung schrittweise fortgesetzt wird, gilt für die Näherungslösung die Gleichung

$$\tilde{y}(t) = \tilde{y}(t_i) + \int_{t_i}^{t} \tilde{f}(\tilde{y}(\tau),\tau)d\tau; \tag{3}$$

$$t_i \leq t < t_{i+1};\ i=0,1,\ldots,n$$

$$\tilde{y}(t_o)=y(t_o)$$

Damit lässt sich die Konsistenzbedingung formulieren. Es gilt:

Es sei f und $\tilde{f}$ stetig bezüglich y bzw. $\tilde{y}$ und t und es gelte die Lipschitz-Bedingung. Man betrachtet die Folge $\tilde{y}^{(\Delta t)}(t)$ der Näherungslösung gemäss (3) für $t_{i+1}-t_i=\Delta t \to 0$.
Dann ist notwendig und hinreichend für die Konvergenz der Folge $\tilde{y}^{(\Delta t)}(t)$ gegen die exakte Lösung y (t), dass

$$\tilde{f}(Y_i, t_i) = f(Y_i, t_i).$$

Dies besagt, dass die exakte Funktion f derart durch eine Näherungsfunktion $\tilde{f}$ ersetzt werden muss, dass die Funktionswerte in den Diskretisierungspunkten übereinstimmen.

Beweis:
Der Fehler e_{i+1} im Punkte t_{i+1} ergibt sich gemäss

$$\begin{aligned} e_{i+1} &= e_i + \int_{t_i}^{t_{i+1}} \tilde{f}(\tilde{y}(\tau),\tau)\,d\tau - \int_{t_i}^{t_{i+1}} f(y(\tau),\tau)d\tau \\ &= e_i + \int_{t_i}^{t_{i+1}} \tilde{f}(\tilde{y}(\tau),\tau)d\tau - \int_{t_i}^{t_{i+1}} \tilde{f}(y(\tau),\tau)\,d\tau \\ &\quad + \int_{t_i}^{t_{i+1}} \tilde{f}(y(\tau),\tau)d\tau - \int_{t_i}^{t_{i+1}} f(y(\tau),\tau)d\tau \end{aligned} \tag{4}$$

Dies kann man wie folgt abschätzen:

$$\begin{aligned} |e_{i+1}| \leq |e_i| &+ \left| \int_{t_i}^{t_{i+1}} (\tilde{f}(\tilde{y}(\tau),\tau) - \tilde{f}(y(\tau),\tau))d\tau \right| \\ &+ \left| \int_{t_i}^{t_{i+1}} \tilde{f}(y(\tau),\tau) - f(y(\tau),\tau))d\tau \right| \end{aligned} \tag{5}$$

Es sei

$$\zeta(\Delta t) = \max_{t\varepsilon(t_i,t_{i+1})} |\tilde{f}(y(t),t) - f(y(t),t)| \tag{6}$$

Wegen $\tilde{f}(y(t_i), t_i) = f(y(t_i), t_i)$ und der Stetigkeit von $\tilde{f}$ und f gilt

$$\zeta(\Delta t) \longrightarrow 0 \text{ für } \Delta t \longrightarrow 0$$

Damit ergibt sich mit der Lipschitz-Konstanten L

$$|e_{i+1}| \leq |e_i| + \Delta t(L|e_i| + \zeta(\Delta t))$$
$$= |e_i|(1+\Delta t \cdot L) + \Delta t \cdot \zeta(\Delta t)$$

Mit $e_0 = 0$ ist wegen $1+\delta < \exp(\delta)$ für $\delta > 0$

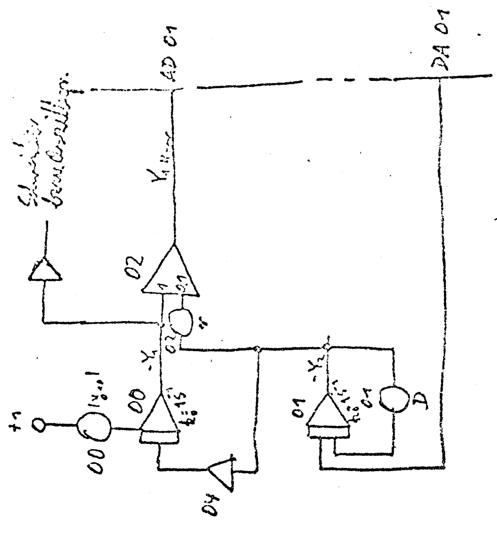
$$|e_i| \leq \frac{\exp(i \cdot \Delta t \cdot L)-1}{\Delta t \cdot L} \Delta t \cdot \zeta(\Delta t)$$

$$= \frac{\exp((t_i - t_0)L)-1}{L} \zeta(\Delta t) \qquad (7)$$

Wegen $\zeta(\Delta t) \to 0$ für $\Delta t \to 0$ ist die hinreichende Bedingung bewiesen. Die Notwendigkeit ergibt sich aus Existenz und Eindeutigkeit der Lösung. Der Beweis enthält auch die konstruktive Aussage, dass gemäss (7) der Fehler mit zunehmender Integration kummuliert.

Durch die Zeitverzögerung kann die Konsistenzbedingung von Sonderfällen abgesehen nicht erfüllt werden. Dieser Sachverhalt ist unbefriedigend, umreisst aber klar die Bedeutung dieses systembedingten Fehlers. Durch Benutzen der Taylor-Entwicklung kann man bei Übergabe der Funktionen an den Digitalrechner eine angenäherte Parallelverschiebung zur Kompensation der Verzögerungszeiten vornehmen. Dies ist jedoch nur für kurze Verzögerungen befriedigend, da im allgemeinen nur das lineare Glied der Taylor-Reihe zur Verfügung steht. Man ist daher gezwungen, den Fehler durch entsprechend kurze Zykluszeiten zu begrenzen. Dies erfordert eine effektive Programmierung des Digitalrechners.

```
      PLATFORM A
      INTEGER*2 AZELLE, BZELLE, WZEIT
      FRACTIONAL YIK, AUS, HILF
      COMMON YIK ,AUS, HILF
      EXTERNAL UPR
      A=1.0
      HILF=XFLFR(-2.0*A/3.142*0)
      WRITE (2,70) HILF
70    FORMAT (1H ,5HHILF=,F7.4)
      ASSIGN 100 TO BZELLE
      CALL*ZYDEF(UPR,WZEIT)
      CALL*ZYZEIT(15I)
      CALL*VEPRO
      GOTO 10
60    CALL*APAUSE
      WRITE (2,30) WZEIT
30    FORMAT (1H ,12HWARTEZEIT   ,I5)
      CALL*VEPRO
10    CALL*ARESET
      CALL*ADSINS(OI,YIK)
      AUS=HILF*SINFR(YIK)
      CALL*DASINS(OI,AUS)
      CALL*ASTART
50    CALL*PHABR(BZELLE)
20    CALL*ZYSYN
      GOTO 20
40    CALL*ADSINS(OI,YIK)
      AUS=HILF*SINFR(YIK)
      CALL*DASINS(OI,AUS)
      CALL*PHSYN(AZELLE)
      GOTO 50
100   CALL*TAH
      IF (ALARM('2000'H)) 200,200,60
200   IF (ALARM('1000'H)) 400,400,200
400   IF (ALARM('4000'H))40,40,300
300   CALL*APAUSE
      STOP
      END
      SUBROUTINE UPR
      FRACTIONAL YIK, AUS, HILF
      COMMON YIK,AUS, HILF
      SPECIAL 1
      CALL*DASINS(OI,AUS)
      AUS=HILF*SINFR(YIK)
      CALL*ADSINS(OI,YIK)
      RETURN
      END
```



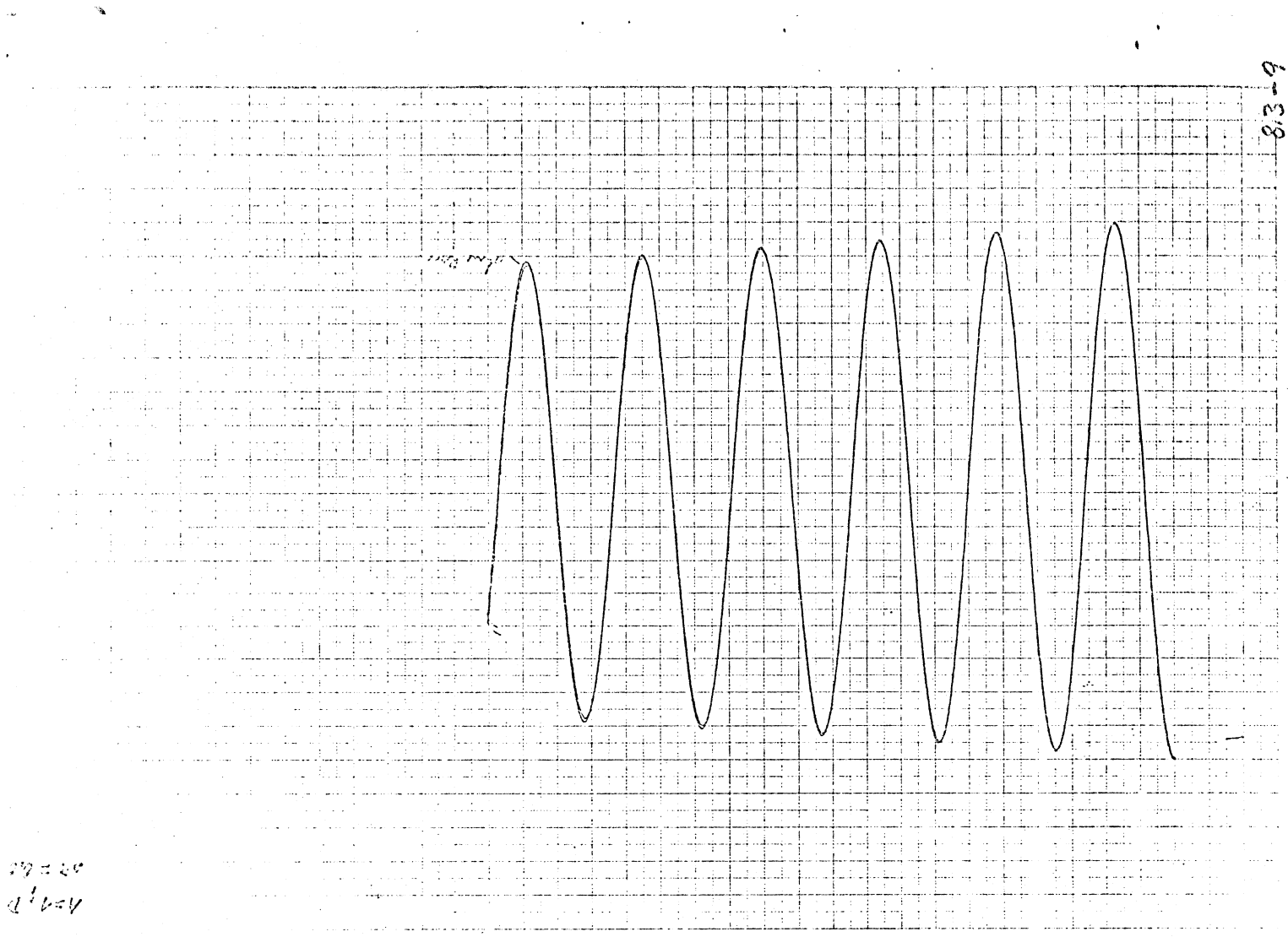
Dgl.: $\frac{dY_1}{d\tau} = Y_2$

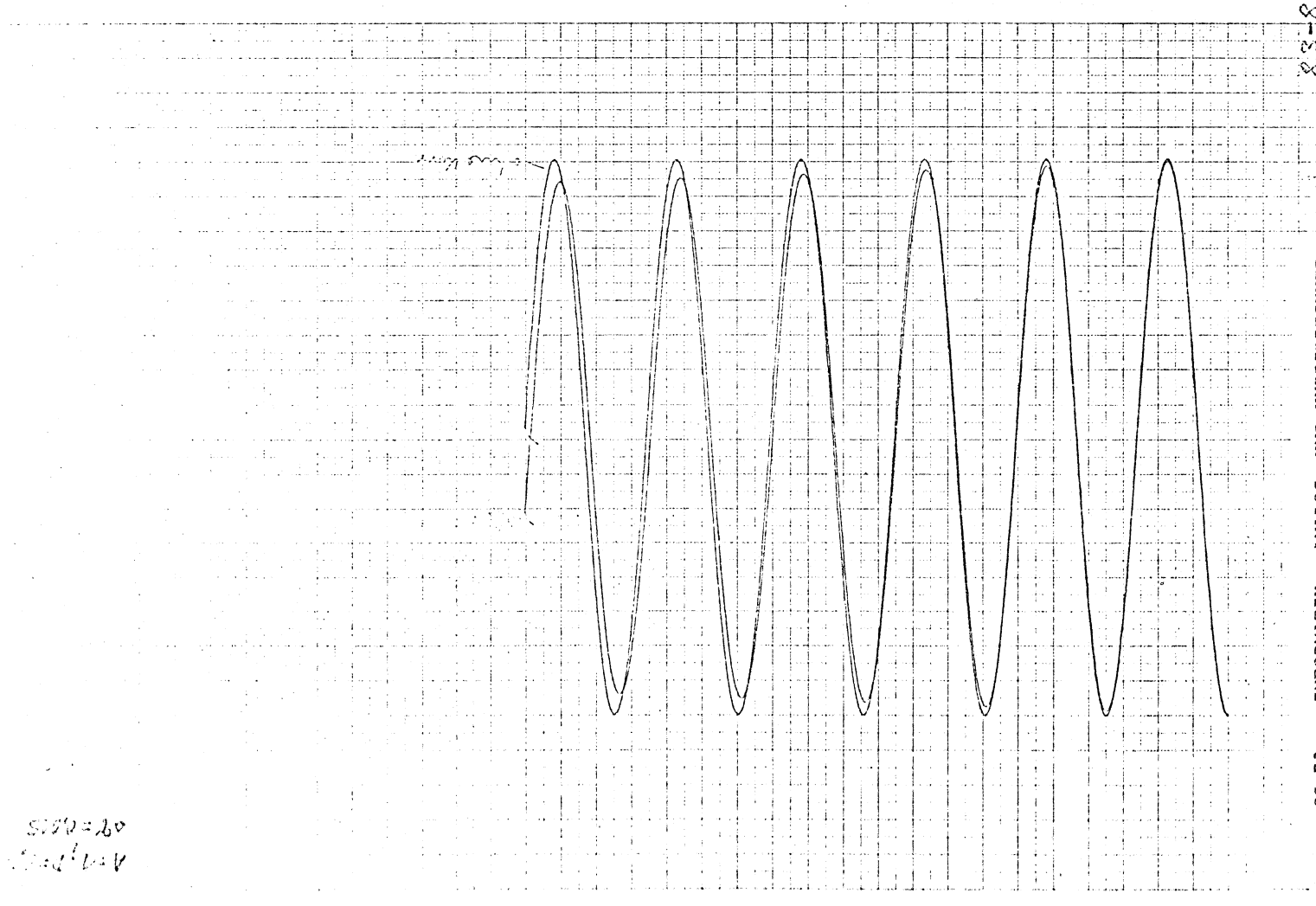
$$\frac{dY_2}{d\tau} = -\left(D \cdot Y_2 + \underbrace{\frac{A}{\pi/2} \cdot \sin\left(\frac{\pi}{2} \cdot Y_1\right)}_{\text{dig. berechnet}}\right)$$

Korrektur:

nach Taylor: $Y_{1,i+1} = Y_{1,i} + \frac{dY_1}{d\tau}\Big|_i \cdot \Delta\tau + \ldots$

$\Delta\tau = 0.005$ ⟶ Einstellwert für r: $r = 0.05$





§ 9 Die Grundsoftware eines Hybrid-Systems am Beispiel des HRS 860

9.1 Allgemeine Strukturprinzipien

Ein wesentliches Hindernis bei der Verbreitung und Anwendung hybrider Rechensysteme ist deren zeitaufwendige Programmierung, für die Spezialkenntnisse über das verwendete System in aller Regel erforderlich sind.

Problemorientierte Programmierung bedeutet anlagenunabhängige Beschreibung des zu lösenden Problems mit Ausdruckmitteln, die dem Problem und nicht der für die Lösung vorgesehenen Anlage angepaßt sind.
Die besonderen Möglichkeiten hybrider Rechensysteme, z.B. das interaktive Arbeiten, müssen allerdings weiterhin benutzt werden können. Da für hybride Rechnersysteme neben der Übersetzung in das digitale und analoge Programm noch zusätzlich Dienstleistungen (Skalierung, Herstellen und Überprüfen der Rechenschaltung) zu erbringen sind, ist ein ganzes Programmsystem notwendig.
Eine Problemorientierte Sprache für hybride Rechnersysteme sollte die besonderen Möglichkeiten dieser Systeme durch das Sprachkonzept unterstützen. Dazu gehört vor allem die gute Möglichkeit zum interaktiven Arbeiten während des Rechenablaufes.

Die Echtzeitdatenverarbeitung stellt hohe Anforderungen an die Verarbeitungsgeschwindigkeit und an die Reaktionsfähigkeit auf externe Signale.
Die in festem Zeitmaßstab bzw. in Echtzeit ablaufenden Vorgänge vermitteln einen besonders realistischen Eindruck vom Verhalten des zu untersuchenden Systems. Analog berechnete Größen müssen leicht dargestellt und laufend beobachtet werden können, ohne den E/A-Verkehr des Digitalrechners zu belasten; Parameter müssen am Analogrechner ebensoleicht verändert werden können. Dies sollte ergänzt werden durch die Möglichkeit, Parameter im Digitalrechner zu verändern und den Rechenablauf von der Bedienkonsole aus zu beenden oder mit korrigiertem bzw neuem Parametersatz zu wiederholen.

Damit die Vorteile des interaktiven Arbeitens am Analogrechner nicht verloren gehen, sollte es möglich sein, über die Bedienungskonsole die Adressen von analogen Variablen zu erfahren. Zum Testen der Programme müssen ausreichende Überwachungsmöglichkeiten des Programmlaufes vorgesehen werden. Durch den Übersetzungsgang gehen die Beziehungen zwischen den Problemgrößen und deren Lokalisierung zur Laufzeit des Programmes verloren. Da im Fehlerfalle diese Beziehungen jedoch herstellbar sein muß, ist es nötig, daß der Übersetzer den Benutzer durch Lokalisierungslisten über diese Zusammenhänge informiert, oder es muß durch einen Dump auf der Ebene der Quellensprache im Fehlerfalle alle relevante Information geliefert werden. Dies muß dann auch den Analogrechner mit einschließen, z.B. bei Übersteuerung eines Operationsverstärkers. Bei der Programmierung auf problemorientierter Ebene geht durch den Übersetzungsgang der Bezug zwischen den Größen des Quellenprogramms und deren Darstellung im erzeugten Maschinenprogramm verloren. Bei Fehlern muß der Benutzer ausreichend über die Fehlersituation informiert werden. Dazu werden durch das Programmierungssystem Fehlerroutinen angeschlossen, die entsprechende Fehlermeldungen ausgeben. Werden dabei Größen des Maschinenprogrammes ausgegeben, so sollten die Angaben, an welcher Stelle des Quellenprogrammes der Fehler auftrat und welche Werte die interessierenden Variablen besitzen, von einem Diagnoseprogramm (Dump) durch die Auswertung von Adreßbuch und Lokalisierungsliste mit direktem Bezug auf das Quellenprogramm geliefert werden. In hybriden Rechnersystemen kommt noch die Forderung nach zeitgerechter Ausführung dazu.

Bei der problemorientierten Programmierung von Hybridsystemen müssen auch Fehlersituationen des Analogrechners und des Interface mit erfaßt werden. Bei der Sicherstellung von Werten des Analogrechners muß beachtet werden, daß die Werte während der Abfrage nicht unzulässig verändert werden. Die Integrierer müssen in Halt gesteuert werden; ihre Werte sind sofort im Digitalrechner sicherzustellen, weil sie sich durch Drift ändern könnten. Die Abfrage der Potentiometerkoeffizienten muß am Schluß erfolgen, weil dadurch alle übrigen Werte zerstört werden.
Die Zuordnung von Adressen der Rechenelemente zu den Variablen des Quellenprogrammes ist nur über das vom Übersetzer erstellte Adreßbuch möglich. Zusammen mit dem Adreßbuch und der Lokalisierungsliste des Digitalprogrammes sollte dieses von Fehlerprogrammen zugänglich sein. Nötig ist dies insbesondere zur Neuskalierung bei Übersteuerung und um das Objektprogramm dialogfähig zu machen.

9.2 Das grundlegende Konzept des HRS 860

Die Software des Hybriden Rechnersystems HRS 860 basiert auf der Grundsoftware des Digitalrechners TR 86. Sie ist ein integrierter Bestandteil des Digitalsoftwarepaketes und ergänzt dessen Einheiten.
Um den extremen Zeitbedingungen der Hybridprogrammierung gerecht zu werden und dabei ein Höchstmaß an Bedienkomfort zu gewährleisten, wurde das TR 86-FORTRAN-System zeitoptimiert, wobei Hardwaremaßnahmen den schnellen Programmablauf unterstützen. Als Betriebssystem für HRS 860 wurde BESY 70 zugrundegelegt, ein Betriebssystem für Stapelverarbeitung.
Für den Betrieb des TR 86 FORTRAN-Compilers auf dem hybriden Rechnersystem HRS 860 steht eine Hybridversion zur Verfügung. Im Grundausbau werden etwa 6000 Speicherzellen benötigt. Eine Erweiterung der Anlage durch Peripheriegeräte erfordert weiteren residenten Speicherplatz.

Am Institut für Informatik sind als Peripheriegeräte der Kontrollfernschreiber, ein Lochstreifenleser und ein Lochstreifenstanzer, ein Plattenspeicher sowie ein Sichtgerät SIG 100 angeschlossen. Damit werden rund 11000 Speicherzellen des Kernspeichers belegt (von 32768).
Als Systemkonsole dienen Kontrollfernschreiber oder das Sichtgerät SIG 100.

Die Aufgaben des BESY 70 im Hybridsystem sind in erster Linie der Start des Hybridprogramms sowie die Abwicklung der Standard-Ein/Ausgabe-Vorgänge.

Der Ein/Ausgabe-Verkehr zwischen Digital- und Analogrechner (Steuerung, Datentransfer), der in der zeitkritischen Phase des Hybridprogramms abläuft, verwaltet sich selbst.
Außerhalb des zeitkritischen Teils des Hybridprogramms werden sämtliche Systemdienste des BESY 70 in Anspruch genommen, soweit sie benötigt werden und dynamisch aufgerufen werden können, Z.B. Ein/Ausgabe-Programm für Standardperipheriegeräte, für die Plattenverwaltung oder DUMP für die Fehlersuche im digitalen Teil eines Hybridprogramms.

Bei Hybrid-Ein/Ausgabe-Prozessen kann ein unbeabichtigtes Überschreiben von Speicherinhalten nur bei Dateneingabe auftreten. Diese Fehlermöglichkeit wird durch die entsprechenden hybriden Bibliotheksunterprogramme weitgehend ausgeschlossen. Für alle Nicht-Ein/Ausgabe-Befehle bleibt der Programmschutz erhalten.
Ein/Ausgabe-Befehle werden normalerweise über das Einsprungfenster des Betriebssystems (Zellen 122-256) in Form von Unterprogrammen im geschützten Speicherbereich ausgeführt.

Die nachfolgend aufgeführten Merkmale stellen für die Realzeitprogrammierung interessante Eigenschaften der normalen TR 86-Digitalsoftware dar bzw. sind Modifikationen, die für Echtzeitaufgaben eingeführt wurden.

1. Eine Einwort-Arithmetik mit den neuen Zahlentypen FRACTIONAL und FLOATSHORT sorgt im TR 86-FORTRAN-System für zeit- und speichereffektive Objektprogramme; damit die Ausführungszeiten der arithmetischen Ausdrücke auf FORTRAN-Ebene nur unwesentlich länger als auf Assembler-Ebene werden. Ebenso wurden Sedezimalkonstanten zur bequemen Ein/Ausgabe, zur Darstellung und Verarbeitung von Bitmustern eingeführt. Dazu wurden die nötigen arithmetischen und sonstigen Operationen mit Größen der Typen FRACTIONAL und FLOATSHORT geschaffen.
2. Zahlreiche hybride Bibliotheksprogramme ersparen dem Benutzer Hardware-Detailkenntnisse und erlauben volle Konzentration auf das Anwendungsproblem.
   Es bestent eine gemeinsame Hybridunterprogrammbibliothek für FORTRAN-IV- und TAS 86-Hybridprogrammierung mit zeitoptimierter Parameterübergabe bei den hybriden Bibliotheksunterprogrammen.
3. Es existiert ein Prioritätsprogrammunterbrechungssystem mit selektivem Sperren und Freigeben von Unterbrechungssignalen. Das Programmunterbrechungssystem umfaßt 28 hardwaremäßig realisierte Ebenen mit jeweils verschiedener Priorität und dazu Routinen für den dynamischen Anschluß von Unterprogrammen an Unterbrechungsebenen. Man verhindert Kollisionen bei Programmunterbrechungen durch gesteuertesMehrfachzuladen von Standardprogrammeinheiten.
4. Das Verkehrsprogramm VEPRO 860 dient für den Dialogverkehr während der Programmestphase und zur Überwachung des Programmlaufes
5. Das Prüfprogramm STATEST 860 dient zum statischen Testen der Analogschaltung. Das Anlagentestprogramm TST 860 zur schnellen Überprüfung von Koppelwerk und Analogrechner.
6. Zur digitalen Fehlerdiagnose an der TR86 dienen besondere Fehler-Routinen des Hybridsystems. Die Unterdrückung des Programmschutzalarms für Ein/Ausgabe-Befehle im Hybridbetrieb verkürzt die Programmausführungszeiten.

   Viele Komponenten der Hybridsoftware können auch für andere Realzeitanwendungen benützt werden.

9.3 Erweiterung der TR 86-Arithmetik

Wird der TR 86 für die Verarbeitung von Zahlenwerten eingesetzt, die mit einer Genauigkeit von 3 bis 4 signifikantenDezimalstellen geliefert oder benötigt werden(z.B. Verarbeitung analoger Meßwerte in einem Hybriden Rechensystem, Auswertung analoger Magnetband-Aufzeichnungen u.s.w.), so bedeutet die Verwendung des normalen Numeriksystem des TR 86 einen unangemessen, und in vielen Fällen auch untragbar hohen Aufwand an Rechenzeit undSpeicherbedarf.
Entsprechend den Anforderungen der Realzeitprogrammierung wurde das TR 86-FORTRAN-System in der Hybrid-Version durch ein Einwortarithmetiksystem erweitert. Es wurden neue Zahlentypen eingeführt und eine entsprechende Arithmetik geschaffen, die in das FORTRAN-System integriert ist. Die Einwortarithmetik ermöglicht kurze Programmlaufzeiten und speichersparende Codierung von Benutzerprogrammen. Der für die Erweiterung benötigte Speicherbedarf in der FORTRAN-Bibliotheksdatei beträgt ca. 3K TR 86-Worte, jedoch wird für ein Anwendungsprogramm jeweils nur ein Teil der Unterprogramme im Kernspeicher benötigt.

Um die Ausführungszeiten möglichst zu reduzieren, sind in der Einwortarithmetik nur Operationen mit typengleichen Operanden zulässig. Die Prüfung der Typengleichheit sowie der Zulässigkeit der Argumente arithmetischer Operationen erfolgt nicht zur Objektzeit, sondern bereits während der Übersetzung des Programmes. Die Gesamtheit dieser Maßnahmen ermöglicht Ausführungszeiten, die denen reiner TAS 86-Programmierung nahe kommen.

Die neu eingeführten Zahlentypen von der Länge eines TR 86-Ganzwortes (=24 Bitstellen) sind FRACTIONAL und FLOATSHORT.

FRACTIONAL- Zahlen sind echte Brüche, also Einwort-Festpunktzahlen, mit einer Genauigkeit von 6 Dezimalen.

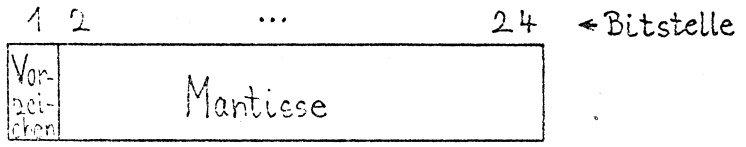


Bild 9.1 zeigt schematisch eine FRACTIONAL-Zahl

Konstanten vom Typ FRACTIONAL werden geschrieben als Dezimalziffernfolge, die genau einen Punkt enthalten muß und die mit dem Buchstaben V abschließt. Die Konstante muß zwischen -1 und 1 liegen, d.h. vor dem Punkt darf höchstens eine Null stehen.
Beispiel:

0.25V

FLOATSHORT-Zahlen sind Einwort-Gleitpunktzahlen mit einer Genauigkeit von 4-5 Dezimalen.

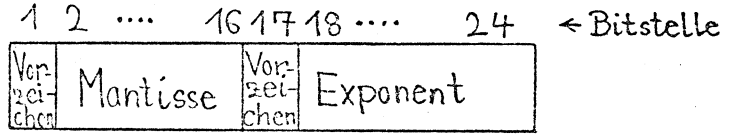


Bild 9.2 zeigt schematisch eine FLOATSHORT-Zahl

Konstanten vom Typ FLOATSHORT beginnen mit einer Mantisse aus Dezimalziffern, welche wahlweise eine ganze Zahl oder ein Bruch sein kann. Danach steht der Buchstabe W, der den Exponententeil einleitet, welcher mindestens eine und höchstens 3 Ziffern enthalten darf,wahlweise mit einem vorausgestellten Vorzeichen.
Beispiele:

0.25W0
2.50W-11

Zusätzlich wurden noch sogenannte Sedezimalkonstanten eingeführt, das sind Folgen von höchstens 6 Sedezimalziffern, die in Apostrophe gesetzt sind und vom Buchstaben H abgeschlossen werden. Sedezimalziffern sind
0,1,2,3,4,5,6,7,8,9,A(=10),B(=11),C(=12),D(=13),E(=14)
und F(=15).
Beispiele:

'12345'H
'456'H
'123ABC'H

Eine Sedezimal-Konstante ist grundsätzlich vom Typ I*2.
Eine I*2-Konstante ist eine Folge von Dezimalziffern, die von dem Buchstaben I abgeschlossen wird; es handelt sich um ganze Einwort-Festpunktzahlen. Diese gehören zur normalen TR86-Ausstattung.
Beispiel:

1024I

Man kann Variablen und Felder vom Typ FRACTIONAL und vom Typ FLOATSHORT deklarieren und sie mit den entsprechenden Konstanten belegen (Typ-Anweisungen FRACTIONAL und FLOATSHORT). Ebenso gibt es Formelfunktionen und FUNCTION's, deren Ergebnisse und Argumente vom Typ FRACTIONAL oder FLOATSHORT sei müssen (FRACTIONAL FUNCTION und FLOATSHORT FUNKTION).

Ein System von Unterprogrammen erweitert das normale Numerik-System des TR 86.
Die erweiterte Einwort-Arithmetik besteht aus einer Sammlung von Unterprogrammen für die Ausführung von arithmetischen Grundoperationen mit typengleichen Operanden, sowie für eine Anzahl von Funktionen, die zum Teil den normalen Standardfunktionen von FORTRAN IV entsprechen, teilweise jedoch den besonderen Bedürfnissen einer zeitlich optimalen Auswertung von physikalischen Meßgrößen angepaßt sind. Diese Unterprogramme können sowohl auf FORTRAN- als auch auf Assembler-Ebene benutzt werden. Die einzelnen Unterprogramme sind im Rahmen der genannten FORTRAN-Erweiterung in die FORTRAN-Bibliothek integriert und werden von dort bei Bedarf an das übersetzte Programm montiert.

Größen der Typen FRACTIONAL und FLOATSHORT können durch folgende Operatoren verknüpft werden (die Operanden müssen vom selben Typ sein):

Wertzuweisung
Addition, Subtraktion, Multiplikation, Division
Wurzel, Quadrat, Reziprokwert
Exponentialfunktion, Logarithmus
Winkelfunktionen
Absolutbetrag, Skalierung
Vergleichsoperationen
Logische Operatoren
Schift
Konvertierungsfunktionen für die Umformung aller Einwort- und Zweiwort-Zahlentypen

9.4 Hybride Unterprogramme der TR 86-FORTRAN-Bibliothek

9.4.1 Allgemeines

Die hybriden Unterprogramme bilden eine Erweiterung der TR 86-FORTRAN-Bibliothek um zusätzliche Standard-Bibliothekselemente, die geschlossen an das übersetzte Programm montiert werden, in dem sie benötigt werden.

Der Aufruf der hybriden Unterprogramme erfolgt als SUBROUTINE mit der Anweisung der Form

CALL*NAME (V1,V2,...,VN)

NAME ist hierbei der Name der Routine, V1 bis VN sind die Argumente. Die Zahl der Argumente ist von Fall zu Fall verschieden. Der Stern innerhalb des Aufrufs steuert den Zugriff zur Erweiterung der TR 86-FORTRAN - Bibliothek und eine optimale Parameterübergabe. Um die Ausführungszeiten der hybriden Unterprogramme zu reduzieren, erfolgt - im Gegensatz zur normalen SUBROUTINE - die Übergabe des ersten Parameters soweit möglich im Akkumulator, die Zulässigkeit des Typs der Argumente wird bereits während der Übersetzung geprüft. Darüber hinaus ermöglicht der Aufruf von CALL* die Benutzung von Parameterlisten variabler Länge.

Innerhalb einer Code-Prozedur können die hybriden Bibliotheks-Unterprogramme mit

```
SU NAME
OP(E) V1
  :
OP(E) K VN
```

aufgerufen werden. NAME ist der Name des Unterprogrammes, V1 bis VN sind Parameteradressen (evtl. mit Ersetzung). Der Operationsteil OP ist bedeutungslos. K markiert den letzten Parameter (Bit 8 = 1).

Die Realzeitprogrammierung bedingt, daß innerhalb einer laufenden Programmeinheit die gleichen Standardfunktionen benutzt werden können, wie in den Unterbrechungsprogrammen. Diese Programmtechnik setzt Standardunterprogramme mit wiedereintrittsfester Struktur voraus oder erfordert das Mehrfachzuladen von Standardfunktionen. Das Mehrfachzuladen ist das zeiteffektivere Verfahren, das jedoch einen etwas höheren Kernspeicheraufwand bedingt. Die zeitaufwendigere wiedereintrittsfeste Programmierung benötigt den Speicherplatz für mehrfaches Ablegen von Programmeinheiten nicht. Dafür ist jedoch der Aufbau eines dynamischen Kellers notwendig zum Retten der Parameter, zur Sicherung des Rücksprungs sowie zum Speichern von Zwischenergebnissen. Zudem besitzen die wiedereintrittsfesten Unterprogramme einen umfangreicheren Organisationskopf. Der daraus resultierende Speicherbedarf verringert den Vorteil in Bezug auf Speicheraufwand bei wiedereintrittsfester Programmierung. Das Mehrfachladen von Standardprogrammeinheiten bietet bei etwas größerem Speicheraufwand Geschwindigkeitsvorteile, die für die Bearbeitung von Realzeitproblemen genutzt werden können.

Das Mehrfachzuladen von Standardroutinen zur Vermeidung von Kollisionen bei benutzerprogrammierter Programmunterbrechung wird durch die Anweisung

SPECIAL I bzw.

*32 I oder *SP I in Assemblerprogrammen

gesteuert. Die Spezialnummer I kann die Werte 0 bis 255 annehmen; ist keine angegeben, so wird "0" eingesetzt. Einander unterbrechende, abgeschlossene FORTRAN-Programmeinheiten müssen verschiedenen Spezialnummern zugeordnet werden.

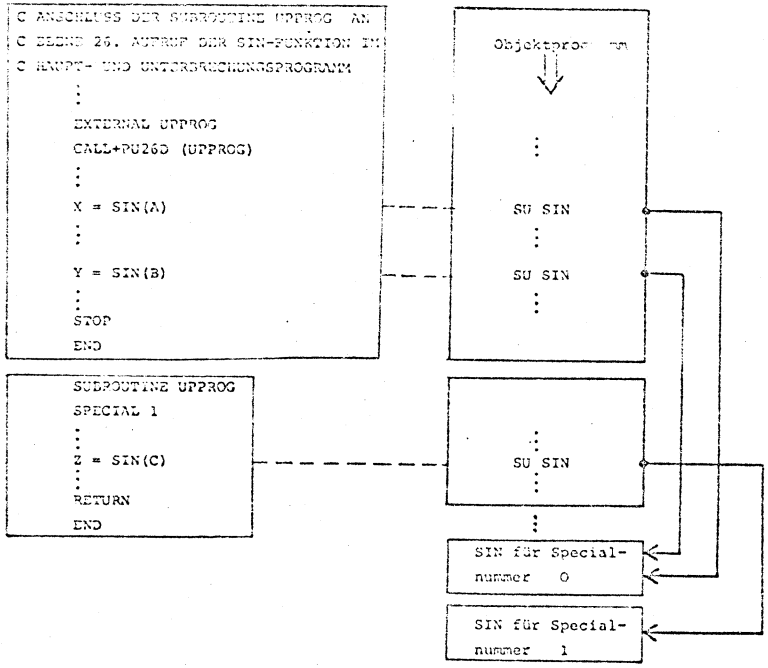


Bild 9.3 zeigt das Mehrfachzuladen von Standardfunktionen

9.4.2. Einstellen der Rechenprogrammarten und Betriebsarten

Am Analogrechner lassen sich alle vorhandenen Rechenprogrammarten einstellen. Es handelt sich um

- Dauerrechnen (mit dem Unterprogramm ADAUER)
- Rechnen mit Halt (ARMHLT)
- Repetierendes Rechnen (AREPET)
- Automatisches Iterierendes Rechnen (AITAUT)
- Iterierendes Rechnen mit Handstart (AITHND)

Für jedes dieser Unterprogramme beträgt der Zeitbedarf rund 20 Millisekunden.

Weitere Unterprogramme erlauben die Einstellung der Betriebsarten des Analogrechners, z.B.

- Pause (APAUSE)
- Halt (AHALT)

- Potentiometereinstellung (APOSET)
- Statisches Programmprüfen (ASTARP)
- Dynamisches Prüfen (ADYNPR)

Beim Einstellen der Betriebsart Rechnen (ASTART) wird die eingestellte Rechenprogrammart gestartet und die Programmunterbrechungen "Analogrechner beginnt zu rechnen" sowie "Analogrechner hält" werden versorgt.

Beim Weiterbefehl (AWEITER) wird ein mit ASTART veranlaßter Rechenablauf nach dem Erreichen der Haltphase fortgestartet. Das Unterprogramm AWEITER darf nur bei den Rechenprogrammarten "Rechnen mit Halt" und "Iterierendes Rechnen mit Handstart" aufgerufen werden.

Durch das hybride Unterprogramm SPRSET kann mit Hilfe eines spezifischen Bitmusters, das an den Analogrechner über die Steuerprogrammregister SPR1 und SPR2 ausgegeben wird, sowohl eine beliebige Rechenprogrammart als auch (soweit zulässig) eine beliebige Betriebsart eingestellt werden.

9.4.3 Synchronisation der Rechenabläufe und Behandlung des Zykluszeitzählers

Der Zykluszeitzähler dient zur Synchronisation gewisser Rechenabläufe, z.B. bei der Programmierung der Programmunterbrechungsebenen.

Es gibt hybride Unterprogramme zur Voreinstellungeiner Zykluszeit T_z, die zwischen 1 und 1275 Millisekunden betragen darf (ZYZEIT), zur Voreinstellung des Zählwerts und der Zählfrequenz des Zykluszeit-Zählers durch zwei getrennte Parameter (ZYTAKT), sowie zum Anschluß eines Unterprogramms an die Programmunterbrechungsebene 16 zum Aufruf beim Zykluszeit-Ende (ZYDEF).

Beim ZYDEF wird zusätzlich die minimale Wartezeit bis zum nächsten Zykluszeit-Ende gemessen und auf eine Speicherzelle eingetragen.

Weitere Unterprogramme synchronisieren den Rechenablauf zwischen Digital- und Analogrechner. Es ist möglich, in Form einer offenen Programmunterbrechung auf den Programmunterbrechungsebenen 17 oder 18 eine Fortsetzungsadresse des Digitalprogramms anzuschließen, die nach der nächsten Programmunterbrechung "Analogrechner beginnt zu rechnen" bzw "Analogrechner hält" angesprungen wird (PHADR).

Ebenso kann die auf das Unterprogramm PHSYN jeweils folgende Adresse des Hauptprogramms - also die Adresse der auf PHSYN folgenden Anweisung - als Fortsetzungsadresse des Digitalprogramms an die Programmunterbrechungsebenen 17 oder 18 angeschlossen werden. Bei PHSYN wird außerdem das Hauptprogramm unterbrochen und die Wartezeit gemessen, bis eine der Programmunterbrechungen "Analogrechner beginnt zu rechnen" bzw " "Analogrechner hält" eintrifft.

Nach dem Aufruf von ZYSYN wird bis zum nächsten Zykluszeit-Ende gewartet, die minimale Wartezeit innerhalb des Zyklus gemessen (wichtig bei mehrfachem Aufruf von ZYSYN) und dann die nach ZYSYN folgende Anweisung des Hauptprogramms angesprungen (Anschluß an PU-Ebene 16).

Weitere Unterprogramme dienen zum Erzeugen von Wartezeiten als ganzzahlige Vielfache von 10 Mikrosekunden (WARTE) und zur Prüfung, ob die Haltephase des Analogrechners ausreichend dimensioniert ist (TAH). Ist die Haltphase zu klein, so kann mit FDPHE das Programm mit einer definierten, vom Benützer selbst geschriebenen Fehlerbehandlung fortgesetzt werden.

9.4.4 Datenübertragung

Die Übertragung von Daten in Digital-Analog- bzw Analog-Digital-Richtung wird durch eine Reihe von hybriden Unterprogrammen geregelt, die sofort ausgeführt werden, d.h. ohne Bezug zu bestimmten Zeitereignissen. Soll der Datentransport in bestimmten gleichbleibenden Zeitabständen erfolgen, so muß man bestimmte Zeitereignisse (z.B. das Zykluszeitende) damit verknüpfen. Am günstigsten verwendet man dazu die Befehle ZYZEIT bzw ZYTAKT zur Voreinstellung der Zykluszeit und hängt dann mit PUDEF bzw PU16D ein Unterprogramm an die Programmunterbrechungsebene 16, das alle Anweisungen enthält, die für den Datentransfer nötig sind. Mit ZYSYN kann innerhalb einer Programmschleife der Datentransport dann synchronisiert werden.

Die Datenübertragung einzelner Werte dauert in beiden Richtungen (DASING, ADSING) jeweils knapp 50 Mikrosekunden. Außerdem ist es möglich, über benachbarte Datenkanäle gleichzeitig Daten zu übergeben (DABLOC, ADBLOC). Wird der blockweise Datentransfer durch eine gesonderte Voreinstellung außerhalb der kritischen Rechenzeit vorbereitet (DAiDEF, ADiDEF), so dauert der eigentliche Datentransfer jeweils nur knapp 30 Mikrosekunden in beide Richtungen für alle Kanäle gemeinsam (DAiST,ADiST). Bei allen Datentransfers wird das Hauptprogramm bereits vor dem Ende der Datenübertragung wieder für neue Anweisungen freigegeben, d.h. softwaremäßig werden die Übertragungsvorgänge zwar gestartet, aber danach geht die Ausführung an Hardware-Einrichtungen über.

Da die Digital-Analog-Umsetzer in den vier Betriebsarten UMSETZEN, INTERPOLIEREN FEIN, INTERPOLIEREN GROB und MULTIPLIZIEREN arbeiten können, kann die Betriebsart für einen Block von benachbarten Kanälen voreingestellt werden (DAUVOR).

Weitere Unterprogramme bereiten die Berechnung extrapolierter Werte zusätzlich zur Einstellung der Betriebsart der D-A-Umsetzer vor (EXiDEF,INiDEF) bzw bewirken die Übergabe extrapolierter Datenwerte(EXiST,INiST).

Mit dem Unterprogramm TEXU kann abgefragt werden, ob an einigen D-A-Kanälen zu große Extrapolationszuwächse auftreten.

Es ist außerdem möglich, einen einzelnen Analogwert über einen zu bezeichnenden Analog-Digital-Kanal einzulesen und sofort als Funktionswert zu verwenden (ADWERT).

Das Ende eines laufenden Datentransfers über den Hybridkanal kann mit ADASYN abgewartet werden. Dann ist gesichert, daß Speicherzellen nicht gleichzeitig durch den noch laufenden Datentransfer und durch andere Unterprogramme angefordert werden.
Mit TADU wird geprüft, ob ein eingelesener Analogwert bei übersteuertem Analog-Digital-Umsetzer gemessen wurde.
Beispiel:

```
      FRACTIONAL DAFLD(10)
      INTEGER*2 WZEIT,AZELLE
      DATA DAFLD / 10*0.V/
      EXTERNAL ZYUP
C ANSCHLIESSEN DER SUBROUTINE ZYUP AN PUE 16 FÜR ZZE
      CALL*ZYDEF(ZYUP,WZEIT)
C EINSTELLEN DER ZYKLUSZEIT TZ = 20MS
      CALL*ZYZEIT(20I)
C EINSTELLEN DER RECHENART REPET. RECHNEN
      CALL*AREPET
C VORBEREITEN EINER BLOCKWEISEN DATENUEBERTRAGUNG IN DA-RICHTUNG AB
C LEITUNGSADR. LA = 3 MIT BLOCKAENGE 4; LA = 0 SEI DEM FELDELEMENT
C DAFLD(1) ZUGEORDNET
      CALL*DA1DEF(3I,DAFLD(4).4I)
      ASSIGN 20 TO AZELLE
      CALL*ASTART
C BEGINN DEF BEFEHLSFOLGE FUER DEN ERSTEN ZYKLUS
      :
```



```
      CALL*PHADR(AZELLE)
      :
10    CALL*ZYSYN
C BEGINN DER BERECHNUNG DER NEUEN WERTE FUER DAS AUSGABEFELD
      :
      GOTO 10
C BEGINN DER BEFEHLSFOLGE FUER DIE HALTPHASE
20    ...
      .
      .
      STOP
      END
      SUBROUTINE ZYUP
C BEGINN DES UNTERPROGRAMMS, DAS MIT JEDER ZZE-PU DURCHLAUFEN WIRD
      CALL*DA1ST
      RETURN
      END
```

9.4.5 Anwahl und Einstellung analoger Rechenelemente

Einige Unterprogramme gestatten die Anwahl von Rechenverstärkern oder Potentiometern und die Analog-Digital-Ausgabe der anliegenden Meßwerte.

Dabei kann ein beliebiges Rechenelement so angewählt werden, daß der Meßwert am Analog-Digital-Kanal mit der höchsten Nummer (Nummer 15 oder 31 je nach Ausbau des Rechners) übertragen wird (ANWAHL).

Ein einzelner Meßwert kann auch direkt als Funktionswert verwendet werden (MWERT).
Ebenso können die Meßdaten eines Blockes von Rechenelementen in ein Meßfeld übertragen werden(MWLES). Jeder Meßwert wird in rund 60 Millisekunden ermittelt und an den Digitalrechner übertragen.
In umgekehrter Richtung können Daten an den Analogrechner übertragen werden zur Einstellung von Potentiometerwerten. Mit POTSET (nicht mit APOSET zu verwechseln) kann ein Block von

Servopotentiometern eingestellt werden, deren Anwahladressen in einem Adreßfeld anzugeben sind.
Einstellzeit zwischen 0.1 und 3 Sekunden.

9.4.6 Verarbeitung binärer Information am Digitalprogrammierfeld

Mit Hilfe eines Abfragewords kann der Zustand von 24 Abfrageleitungen abgefragt werden. Man erhält ein 24-stelliges Bitmuster, das mit einer Maske (ein Bitmuster) so durch eine logische UND-Verknüpfung verbunden wird, daß jeweils zusammengehörige Bitstellen ein Ergebnisbit an derselben Stelle im Bitmuster ergeben. Das Ergebnis wird als Funktionswert bereitgestellt (ALAND).

Bitstelle des Abfragewortes	Bedeutung der gesetzten Bitstelle
1	Hybridkanal aktiv
2	AR übersteuert
3	AR in Phase "Rechnen"
4	AR in Phase "Halt"
5	DAU/DAS-Betriebsarteinstellung aktiv
6	Steuerbefehl mit RKK-Adresse 13 nicht beendet
7	Potentiometer-Einstellung unmöglich
8	Anwahl des Rechenelements möglich
9	HKW nicht betriebsbereit (Taste "Wartung") oder am AR die Tasten "Fremd2 und "Extern" nicht akriviert
10 - 24	Am DPF frei verfügbare Abfrageleitungen mit der Buchsenbezeichnung AL(10 bis 24) mit 1-Potential beschaltet.

Bild 9.3 zeigt die Bedeutung der Bits des Abfragewortes

Soll Information vom Digitalrechner an Buchsenfelder des Digitalzusatzes ausgegeben werden, so muß zunächst ein Bitmuster (durch Sedezimalzahlen rechtsbündig dargestellt) vorgegeben werden.

Die Bitstellen 13 bis 24 des Parameterwortes können an die Buchsen PA1 bis PA12 einschließlich dem Taktimpuls zur Übernahme der Information in Flip-Flops (Buchse PAT) überschrieben werden (RÖSET).

Ebenso können die Bits 19 bis 24 eines aktuellen Parameters an das sechsstellige Register PA1 bzw PA2 übertragen werden (R1SET bzw R2SET).
Schließlich kann man noch die Bitstellen 13 bis 24 eines aktuellen Parameters an das zwölfstellige Register PPA des Digitalprogrammierfeldes übergeben (R3SET).

Beispiel:

```
      :
C EINLESEN DES ZUSTANDES DER ABFRAGELEITUNG DES HKW,
C AUSBLENDEN DER
C BITSTELLEN 1 BIS 18 UND 22 BIS 24 UND SPRUNG NACH ANWEIS. 17
C FALLS ALLE BITSTELLEN 19 BIS21 DEN WERT NULL BESITZEN
      IF(ALAND('38'H)) 16,17,16
16    ...
      :
17    ...
      :
```

9.4.7 Programmierung des X-Y-Schreibers

Von den Ein-/ Ausgabe-Geräten des Analogrechners (Oszillograph, X-Y-Koordinatenschreiber,Digitalvoltmeter, Kienzle-Drucker, A-D-Kanäle) ist der X-Y-Schreiber durch Unterprogramme steuerbar. Dies geschieht in der Art eines Plotters, d. h. der X-Y-Schreiber kann auch unabhängig vom Analogrechner benutzt werden, um Kurvenzüge auszugeben.
Es gibt z. B. Unterprogramme für die Normierung des Zeichensystems und die Justierung des X-Y-Schreibers (PLOTJU), für die Festlegung der gewählten Zeichenebene auf dem Zeichenpapier (PLOTEB) sowie für die Festlegung der gewählten Abbildungsbereiche in X-Richtung und in Y-Richtung (PLOTSK) und das Achsenkreuz (PLOTXY).

Es ist möglich, Kurven mit äquidistanten bzw. beliebig verteilten Stützstellen bei wählbarer Interpolationsart und wählbarer Zeichengeschwindigkeit (PLOTK1 bzw PLOTK2) zeichnen zu lassen.
Weiterhin kann eine beliebige Folge von Symbolen gezeichnet werden mit wählbarem Anfangspunkt sowie wählbarer Symbolgröße und Schreibrichtung.

Weitere hybride Unterprogramme dienen zur Versorgung der Programmunterbrechungsebenen (siehe Abschnitt 9.5) und zur freien Programmierung von Fehlerdiagnoseprogrammen (siehe Abschnitt 9.7).

9.5 Das Programmunterbrechungswerk des HRS 860

Das modifizierte Programmunterbrechungswerk für das Hybride Rechnersystem HRS 860 verfügt über insgesammt 28 PU-Ebenen, denen die Indizes 1 - 28 zugeordnet sind, und deren Priorität mit steigendem Index abnimmt. Die PU-Wünsche werden in ein Anmelderegister aufgenommen. Die Aufnahme wird für die Ebenen 5 bis 28 entsprechend der Einstellung des vorgeschalteten Masken-Registers bedingt vorgenommen.
Aus dem Anmelde-Register wird die Information zu bestimmten Zeitpunkten in ein Auswert -Register übernommen.

Die Prioritätsentschlüsselung erfolgt durch eine Verknüpfungsschaltung, die das Auswert-Register ständig (d.h. ohne Zeitbedingung) abfragt
Die Weiterverarbeitung der Information, speziell die Startbedingung für das PUW, ist abhängig von der Priorität der Programmunterbrechung.
Der STart des PUW erfolgt direkt durch das Auswert-Register.
Nach dem Startsignal sperrt das PUW das Auswert-Register gegen weitere PU-Wünsche ab, so daß eine erneute Unterbrechung bei aktiviertem PUW verhindert wird.

Das Programmunterbrechungswerk welches bei der Verwendung des FORTRAN-Hybrid-Compilers im HRS 860 benutzt wird, verhindert, daß auf einer der Ebenen 13 - 28 ein Eingriff erfolgt, solange noch das PU-Programm einer Ebene derselben oder einer höheren Priorität aktiv ist.

Laufende Unterbrechungsprogramme dieser Ebene können durch Unterbrechungssignale höherer Priorität unterbrochen werden. Durch selektives Sperren und Freigeben von Unterbrechungsebenen ist eine dynamische Prioritätszuweisung während des Hybridprogrammlaufs möglich.
Ein in das Auswert-Register gelangter PU-Wunsch wird also nur dann in das Protokoll-Register aufgenommen, wenn er eine höhere Priorität als alle bereits im Auswert-Register registrierten PU-Wünsche hat.
Eine Anmeldung wird aber im PUW gespeichert. Da aber jeweils nur eine Anmeldung pro Ebene gespeichert werden kann, ist es notwendig, alle PU-Routinen möglichst kurz zu gestalten, damit keine Anmeldung der gleichen oder einer niedrigen Ebene verloren geht.

Unterbrechungsebenen 13 bis 28 (für den Betrieb im HRS 860):
Soll ein Programmunterbrechungswunsch höherer Priorität eine Unterbrechung des laufenden Unterprogramms hervorrufen, muß der Programmierer durch einen "Lösche Merklicht"-Befehl für die Aufhebung der automatischen Unterbrechungssperre durch das Merklicht M sorgen, falls dies nicht durch hybride Bibliotheksunterprogramme geschieht.
Unterbrechungsebenen 3 bis 12 (in der Regel für Standard-Peripherie): Ein in das Auswert-Register gelangter PU-Wunsch führt ungeachtet der Priorität des gerade laufenden Programms eine Unterbrechung herbei, falls keine Sperre gesetzt ist.
Unterbrechungsebenen 1 und 2:

Die Unterbrechungsebenen 1 (Netzspannungsausfall, "Aus"- und "Stop"-Taste) und 2 (EA-Fehleralarm des Rechnerkernkanals und Programmschutzfehler) nehmen gegenüber den anderen Unterbrechungsebenen eine Sonderstellung ein, da sie nicht softwaremäßig (mit dem Merklicht M) sperrbar sind. Da sie im übrigen wie die Ebenen 3 bis 12 behandelt werden, führen PU-Wünsche auf den Ebenen 1 u. 2 in jedem Falle sofort eine Unterbrechung herbei.

Nummer der Unterbrechungsebene		Zugehöriges Unterbrechungsereignis
1		Netzspannungsausfall, Betätigen von "Stop"-oder "Aus"-Taste
2		EA-Fehleralarm des RKK, Programmschutzfehler
3		Dreierprobenalarm
4		Unterbrechung durch Eingriffswerk
5		Unterbrechung durch Kontrollschreiber
6		Unterbrechung durch Lochstreifenleser
7		Unterbrechung durch lochstreifenstanzer
8 - 12		frei für weitere RKK-Peripherie
13	HYK	Abschluß eines blockweisen Datentransfers über den Hybridkanal
14	ARÜ	AR übersteuert
15	ADÜ	ADU übersteuert
16	ZZE	Zykluszeit-Ende
17	ARH	Beginn der Phase "Halt" des Analogrechners
18	ARR	Beginn der Phase "Rechnen" des Analogrechners
19 - 28		An den Buchsen UB(5 bis 14) des DPF beliebig aufschaltbare Unterbrechungssignale

Bild 9.4 zeigt die 28 Programmunterbrechungsebenen

In einem Hybridprogramm werden, bei Benutzung des Standardmäßigen Programmsteuerung während der aktiven Phase des Analogrechners, die Unterbrechungen 'AR rechnet' und 'AR hält' dazu benutzt, das Hauptprogramm mit dem Phasenablauf des Analogrechners zu synchronisieren. Beim Aufruf der Unterprogramme ASTART,AWEITER, PHSYN, PHADR werden offene PU-Programme (s.u.) an die Ebenen für 'AR rechnet' bzw 'AR hält' angeschlossen und sofort nach Eintreffen der Unterbrechung angesprungen.

Bei Eintritt in die aktive Phase und bei jedem Phasenwechsel muß jeder sonstige E/A-Verkehr ruhen, da eine durch die PU's 'AR rechnet' oder 'AR hält' unterbrochene E/A-Routine nicht zu Ende geführt wird. Ein E/A-Verkehr zwischen zwei Phasenwechselnist ebenfalls nicht erlaubt.
Ist ein E/A-Verkehr nicht zu umgehen, z.B. wenn Werte auf die Magnettrommel übernommen werden soll, so kann dies am besten dadurch geschehen, daß während des E/A-Verkehrs der Analogrechner über die Steuerbuchse H (Halt) am DPF in der Halt- oder Pausenphase festgehalten und nach dem E/A- Ende der normale Rechenablauf des Analogrechners wieder freigegeben wird.

Nicht erlaubt sind in PU-Programmen der Aufruf von Betriebssystemsprogrammen und alle darauf aufbauende Unterprogramme, vor allem nicht die READ- oder WRITE-Anweisung.

Bei offenen PU-Programmen wird das unterbrochene Programm nicht mehr fortgesetzt.
Nach durchlaufen eines geschlossenen PU-Programms wird das unterbrochene Programm an der unterbrochenen Stelle mit den alten Registerinhalten und Merklichterzuständen fortgesetzt.

Der Anschluß von Unterprogrammen an Unterbrechungsebenen erfolgt über PU-Kommandos in VEPRO 860 oder über Bibliotheksunterprogramme.

Eine Reihe von Unterprogrammen versorgt die einzelnen PU-Ebenen und das PU-Werk. Damit ist es möglich, einen Teil oder die Gesamtheit der PU-Ebenen 3 - 28 durch Setzen des Merklichts M zu sperren (PUSPE, PUMSK1) bzw durch Löschen des Merklichts M freizugeben. (PUAKT, PUMSK0). Zeitbedarf: 6 Mikrosekunden für die Befehlsausführung. Weitere Unterprogramme der Hybridbibliothek erlauben das Kurzschließen einer oder mehrerer der PU-Ebenen 13 - 28 (PUNOP), das Setzen der Sperrmaske für die Pu- Ebenen 5 - 28 (PUMSKS). Das Einlesen oder Retten des Protokollregisters (PUAND , PUNORM), das Normieren des PU-Werkes und anderes.

Der Anschluß von Unterprogrammen (FORTRAN-Subroutinen oder Code-Prozeduren) an eine der PU-Ebenen $13 \leqq ki \leqq 28$ geschieht durch die Befehle PUkiD bzw PUDEF, die beide je etwa 35 Mikrosekunden Zeitbedarf haben.

Beispiel:

```
      FRACTIONAL V
      COMMON V
      EXTERNAL PUPROG
      V = ...
C ANSCHLUSS VON PUPROG AN PUE 21
      CALL*PU21D(PUPROG)
      :
      SUBROUTINE PUPROG
      COMMON V
      SPECIAL 21
C AUSGABE VON V UEBER DA-KANAL 3
      CALL*DASING(31,V)
      RETURN
      END
```

9.6 Das Verkehrsprogramm des HRS 860 (VEPRO 860)

VEPRO 860 ermöglicht das Austesten, Überwachen und Verändern hybrider Arbeitsprogramme in Form eines Dialogs zwischen Benutzer und Hybridsystem.
Durch Hinzufügen besonderer, vom Benutzer zu definierenden Kommandos können während des Programmlaufs Programme dynamisch verändert werden.
VEPRO 860 läßt sich auch außerhalb der Hybridprogrammierung verwenden, z.B. zum Überwachen spezieller Hardwareeinrichtungen (Experimente).

VEPRO 860 erlaubt in bequemer Weise den Anschluß von Unterbrechungs- und Leerroutinen an Programmunterbrechungsebenen, den Einbau dynamischer Stops beim Auftreten von Unterbrechungsereignissen sowie das Löschen der Einsprungzellen für die Unterbrechungsebenen 13 - 28 (vgl. PU-Kommandos). Als Dialogmedium dient die Systemkonsole des Digitalrechners(Kontrollfernschreiber oder Sichtgerät SIG 100).

VEPRO 860 erlaubt die Verwendung der folgenden von hybriden Bibliotheksunterprogrammen (siehe Abschnitt 9.4) geregelten Kommandos.

Programmwahl am Analogrechner

Einstellen der Betriebsarten

Einstellen der Zykluszeit

Potentiometer einstellen

Anwahl von Rechenelementen und Einlesen des Ausgangswertes

Disjunktives Setzen und Löschen der Parallelausgabe-Register

Einlesen des Zustands der Abfrageleitungen

Datentransfer

Anschließen von Unterbrechungs- oder Leerroutinen für die Programmunterbrechungsebenen 13 bis 28

Rückkehr in das Betriebssystem

Fortsetzen des Hybridprogramms

Direkter Sprung an eine Adresse im FORTRAN-Hauptprogramm

Soll die Zahl der in VEPRO 860 verfügbaren Kommandos vergrößert werden, so kann der Benutzer im Arbeitsprogramm eine beliebige Anzahl weitere Kommandos definieren.

Das Verkehrsprogramm kann auf verschiedene Weise aktiviert und benutzt werden. Man unterscheidet drei Arten von Aufrufen:

1. Aufruf:

VEPRO 860 wird durch das Kommando

:START, 6000;

an BESY 70 gestartet und steht dem Benutzer unabhängig von einem Arbeitsprogramm als ein autonomes Dienstprogramm des Betriebssystems zur Verfügung. Durch Eingabe der entsprechenden Kommandos können dann die Betriebszustände des Analogrechners und des Koppelwerkes festgestellt bzw. verändert werden. Durch das Kommando

END

erfolgt der Rücksprung in das Betriebssystem, das sich anschließend zur Entgegennahme weiterer Kommandos über die Systemkonsole meldet.

2. Aufruf:

Aus einem hybriden Arbeitsprogramm heraus erfolgt der Eintritt in VEPRO 860 in Form eines Unterprogrammsprungs durch den Befehl

SUE 198 U bzw. SU VEPRO in TAS 86 oder durch

CALL * VEPRO in FORTRAN

Durch geeignete Kommandos können die gewünschten Untersuchungen bzw. Änderungen im Arbeitsprogramm vorgenommen werden. Es darf eine beliebige Anzahl von Kommandos nacheinander eingegeben werden.

Der Rücksprung in das Hauptprogramm erfolgt nicht automatisch sondern durch das Rückkehrkommando

HPOST;

(HYbrid-Fortstart) an die Stelle, an der das Hauptprogramm verlassen wurde.

3. Aufruf:

Dieser Aufruf ermöglicht die Simultanarbeit von Arbeitsprogramm und VEPRO 860. Der Benutzer kann während des Laufes des Realzeitprogramms dem Verkehrsprogramm ein Kommando erteilen. Sobald das Arbeitsprogramm einen Befehl

SUE 199 bzw. SU KABRUF für Assembler oder

CALL * KABRUF

für FORTRAN vorfindet - zweckmäßigerweise sollte dieser Sprungbefehl außerhalb des zeitkritischen TEils des Realzeitprogramms liegen - wird ein Sprung in eine Abfrageroutine von VEPRO 860 erzeugt. Hier wird zunächst festgestellt, ob ein Kommando eingegeben wurde, wenn nicht, erfolgt ein automatischer Rücksprung in das Hauptprogramm. Liegt jedoch ein vollständiges Kommando vor, so wird es mit der Liste der in VEPRO 860 zulässigen und evtl.zusätzlich definierter Kommandos verglichen und entsprechend ausgeführt. Anschliessend erfolgt automatisch der Rücksprung in das Hauptprogramm.

9.7 Fehlerdiagnose im HRS 860-System

Im TR 86-FORTRAN-System treten drei Typen von Fehlermeldungen auf:

Fehlermeldung der Verwaltungsprogramme

Fehlermeldung der Compilerläufe

Fehlermeldung des Objektprogramms

Das Auftreten eines Fehlers wird durch eine Fehlermeldung angezeigt.

Der FORTRAN-Compiler meldet bei der Übersetzung von FORTRAN-Programmen alle statisch erkennbaren Programmierfehler. Ist ein Programm frei von solchen statischen Fehlern, so bleibt doch eine Reihe von Fehlerquellen erhalten, die sich erst beim Lauf des übersetzten Programms auswirken und auch dann erst ermittelt werden können. Dem Erkennen von derartigen Fehlern dienen die dynamischen Tests auf Objektebene.

Das TR 86-FORTRAN-System erlaubt es, FORTRAN-Objektprogramme in zwei verschiedenen Versionen, der Laufversion und der Testversion, zu erstellen und abzuarbeiten.

In beiden Versionen werden während der Objektphase Kontrollen durchgeführt und gegebenenfalls Fehler gemeldet. Zu den Kontrollen und Fehlermeldungen der Laufversion kommen jedoch in der Testversion noch eine ganze Reihe weiterer wichtiger Kontrollen und Fehlermeldungen hinzu, die insbesondere das Austesten von FORTRAN-Programmen erleichtern sollen.

Die Testversion unterscheidet sich von der Laufversion in 8 Standardfunktionen. Diese 8 Standardfunktionen sind in zwei Ausführungen vorhanden. Weitere Unterschiede zwischen der Lauf- und der Testversion eines FORTRAN-Objekts gibt es nicht. Der Zeit- und Platzbedarf der Testversion ist gegenüber dem der Laufversion größer.

Es ist besonders wichtig, zu wissen, in welcher Programmiereinheit und in welcher Anweisung der Fehler auftrat und welchen Wert die Variablen und Fehler dieser Programmiereinheit zum Zeitpunkt des Fehlerereignisses hatten. Auskunft hierüber erteilt der FORTRAN-Dump.

Der FORTRAN-Dump erleichtert es, Fehler zu lokalisieren, die erst beim Ablauf des FORTRAN-Programms auftreten. Nach dem Auftreten eines Fehlers und dem Abbruch des Objektlaufs wird eine Angabe darüber ausgedruckt, bei welcher Anweisung und in welcher Programmeinheit der Fehler auftrat, und welche Unterprogramme bis dahin aufgerufen wurden. Die Fehlersuche wird dadurch erleichtert, daß alle Variablen und Felder der betreffenden Programmeinheit und alle COMMON-Größen in ihren aktuellen Werten ausgedruckt werden.

Nach der Lokalisierung der Fehlerstelle werden die Variablen, Felder und , falls verlangt, die COMMON-Größen der fehlerauslösenden Programmeinheit, und nur diese, mit ihren aktuellen Werten abgedruckt.
Für die Steuerung der Ausgaben der im FORTRAN-Programm benutzten Größen sind folgende Möglichkeiten vorgesehen:

Im Fehlerfall werden
a) nur Variablen,
b) nur Variablen, einschließlich der im COMMON liegenden Variablen,
c) Variablen und Felder,
d) Variablen und Felder, einschließlich der im COMMON liegenden Variablen und Felder

der fehlerauslösenden Programmeinheit in ihren aktuellen Werten ausgedruckt.

Bei Ausführung des FORTRAN-Dumps werden nicht mehr benötigte Teile des FORTRAN-Objektprogramms im Kernspeicher überlagert.

Zur Erleichterung der Fehlerdiagnose und um definierte Fehler abzufangen, so daß der Benutzer die Art der Fortsetzung oder des Abbruchs des Programmes selbst programmieren kann, dienen einige Unterprogramme der hybriden Programmbibliothek.
Bei allen Programmen dieser Gruppe werden die standardmäßigen Behandlungen der jeweiligen Fehler durch einen Sprung in ein frei programmiertes Fehlerfortsetzungsprogramm ersetzt, bzw. es werden die standardmäßigen Fehlerbehandlungen wieder regeneriert, sofern sie zu einem früheren Zeitpunkt ersetzt

wurden. Zeitbedarf: jeweils rund 10 Mikrosekunden je Unterprogramm. Die Fehlermeldung vom Analogrechner erfolgt über die Rechnerkanäle 13 und 14 in Form spezifischer Bitmuster eines Abfragewortes.

Folgende Aufgaben werden erledigt:

Fehlerbehandlung bei Analogrechner-Übersteuerung (FDARU)

Fehlerbehandlung bei Analog-Digital-Umsetzer-Übersteuerung (FDADU)

Fehlerbehandlung bei Zyklus-Zeit-Ende-Zeitfehler (FDZZE)

Fehlerbehandlung bei Phasenende-Zeitfehler (FDPHE)

Fehlerbehandlung bei Potentiometer-Einstellfehler(FDPOT)

Fehlerbehandlung bei arithmetischen Fehlern (FDARI)

Wird ein solches Unterprogramm mehrfach aufgerufen, so können im selben Programm verschiedene Fehlerbehandlungen für denselben Fehler bewirkt werden, je nachdem an welcher Stelle des Hauptprogramms der Fehler auftritt.

Beispiel:

```
C BEISPIEL FUER FDARU
      INTEGER*2 AD.WZ
      ASSIGN 10 TO AD
C ANSCHLUSS EINER FEHLERMELD. FUER AR UEBERST.
      CALL*FDARU(AD)
      CALL*ARMHLT
1     CALL*ASTART
      :
      CALL*PHSYN(WZ)
C ANSCHLUSS DER STANDARDMAESSIGEN FEHLERBEHANDLUNG
C FUER AR-UEBERSTEUERUNG
      CALL*FDARU(0I)
      :
10    CALL*PUNOP(17I)
      CALL*PUAKT
      WRITE(1,11)
11    FORMAT ('AR ÜBERSTEUERT')
      GOTO 1
      :
```

10. Lösung von partiellen Differenzialgleichungen mit Hybridsystemen

Partielle Differentialgleichungen (part. Dglen) treten bei der Beschreibung orts- und zeitunabhängiger Vorgänge auf. Sie unterscheiden sich von den bisher behandelten Fällen dadurch, dass Ortskoordinaten als weitere unabhängige Variable auftreten. Mit deren Hilfe kann man eine umfassendere Klasse kontinuierlicher Systeme beschreiben.
Die Beschreibung realer Systeme durch gewöhnliche Dgen stellt häufig bereits eine Idealisierung dar (z.B. das Modell des "starren Körpers"). Einige Beispiele für Vorgänge, die durch part. Dglen beschrieben werden können, sind:

- Wärmeleitung
- Diffusionsvorgänge
- Wellenmechanismen
- räumlich verteilte Parameter

10.1. Grundsätzliche Betrachtungen

Bei einer Reihe von Aufgaben führt ein Produktansatz zu einer Trennung der Variablen. Man erhält damit Eigenwertaufgaben, die mit den entsprechenden Methoden zu lösen sind. Dieser analytische Ansatz stellt jedoch keine allgemeine verwendbare Lösungsmethode dar.
Eine Möglichkeit der Bearbeitung ergibt sich, wenn in der partiellen Differentialgleichung die Ableitung nach einer oder auch mehreren Variablen durch Differenzenausdrücke ersetzt werden. DAbei sind stets Konvergenz- und Stabilitätsfragen zu berücksichtigen. Ein Ersatz der Ableitungen durch finite Ausdrücke kann in verschiedenster Weise erfolgen. So können beispielsweise in einer Differentialgleichung mit t und x als unabhängigen Variablen entweder die Ableitungen nach x oder nach t oder auch die Ableitung nach beiden Variablen durch Differenzenausdrücke ersetzt werden. Man erhält als Ergebnis dieser Zerlegung ein System gewöhnlicher Differentialgleichungen bzw. bei Ersatz aller

Ableitungen ein algebraisches Gleichungssystem. Diese Systeme sind linear oder nicht linear, je nachdem, ob die zugrunde liegende partiellen Differentialgleichungen linear oder nicht linear sind. Dadurch, daß für eine Ableitung Differenzenausdrücke unterschiedlicher Gestalt und Genauigkeit verwendet werden können, ergeben sich für ein Problem verschiedene Möglichkeiten von Ersatzsystemen. Ebenso sind evtl. vorkommende Ableitungen in den Anfangs- und Randwerten durch Differenzenausdrücke zu ersetzen.

Bei der Näherung eines Differentialquotienten mit Hilfe eines endlichen Differenzausdrucks bestehen zwei Möglichkeiten: die Zentral-, Vorwärts- und die Rückwärtsdifferenz-Näherung. Können die Gleichungen explizit gelöst werden, so erhält man ein Verfahren, das rechnerisch verhältnismäßig einfach ist, das jedoch nicht immer stabil ist.

Ein implizites Verfahren, vermeidet zwar Unstabilitäten, erfordert jedoch die Lösung einer großen Anzahl von Simultangleichungen.

Die Tabelle in Bild 10.1 gibt einen Überblick über gängige Approximationen von Differentialquotienten durch Differenzenausdrücke.

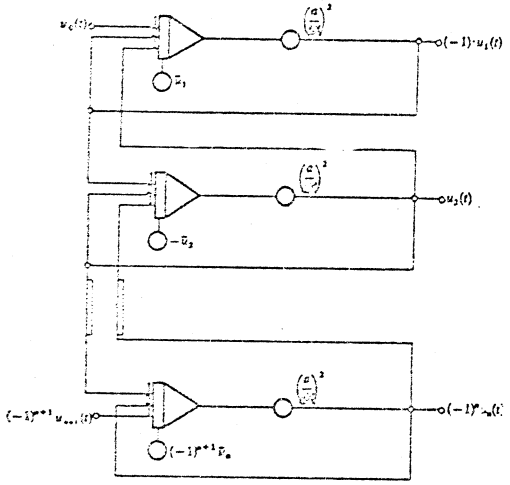


Bild 10.3. Prinzipschaltbild für die diskretisierte Wärmeleitungsgleichung

Ein weiteres Beispiel ist die Wellengleichung

$$\frac{\partial^2 u}{\partial t^2} = a^2 \frac{\partial^2 u}{\partial x^2} \qquad (10.5)$$

mit $u(x,0) = \varphi_1(x)$; $\left[\frac{\partial u}{\partial t}\right]_{(x,0)} = \varphi_2(x)$ (10.6)

und

$$u(0,t) = \psi_1(t); \; u(1,t) = \psi_2(t). \qquad (10.7)$$

Durch Diskretisierung erhält man das Dgl.-System

$$\frac{d^2 u_i}{dt^2} = \frac{a^2}{\Delta x^2}(u_{i+1} - 2u_i + u_{i-1}), \quad i = 1,2,\ldots,n \qquad (10.8)$$

mit den Anfangsbedingungen

$$u_i(0) = \varphi_1(x_i), \; \left[\frac{du_i}{dt}\right]_{t=0} = \varphi_2(x_i). \qquad (10.9)$$

Die dazugehörige Rechenschaltung zeigt Bild 10.4.

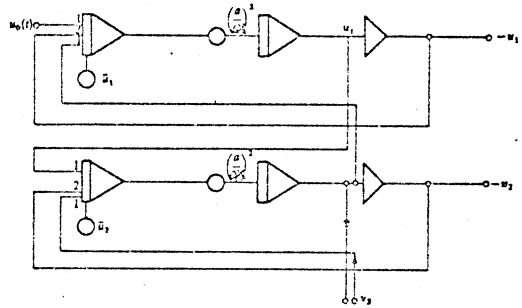


Bild 10.4 Prinzipschaltung für die diskretisierte Wellengleichung

Das Verfahren hat den Nachteil, dass die maximale Anzahl der Diskretisierungspunkte durch die Anzahl der verfügbaren Integrierer begrenzt ist. Von Vorteil ist, dass das Verfahren günstiges Stabilitätsverhalten aufweist.

10.3 Diskretisierung der Zeit (Serienmethode)

Es ist auch umgekehrt möglich, die Zeit t zu quantisieren und x als kontinuierliche Variable beizubehalten. Man erhält mit $t_k = t_o + k\Delta t (k = 0,1,2,\ldots)$ das Ersatzdifferenzialgleichungssystem der Wärmeleitungs-Gleichung (10.1)

$$\frac{d^2 u_k}{dx^2} = \frac{1}{a^2 \Delta t}(u_k - u_{k-1}) \qquad (k=1,2,\ldots), \qquad (10.9)$$

wobei die u_k Funktionen von x sind. Mit den Anfangs- und Randbedingungen von Gleichung (10.2) und (10.3) erhält man

$$u_o(x) = u(x,0) = \varphi(x) \qquad (10.10)$$

und

$$u_k(0) = \psi_1(t_k); \quad u_k(1) = \psi_2(t_k) \qquad (10.11)$$

Das somit formulierte Ersatzproblem - im folgenden Verfahren II genannt - unterscheidet sich grundlegend von dem oben behandelten - im fogenden Verfahren I genannt - . Während dort ein echtes simultanes System von Differentialgleichungen entstand, können hier die Gleichungen für k=1,2,.. nacheinander gelöst werden, wobei lediglich dafür zu sorgen ist, daß bei der Lösung der k-ten Gleichung die Größe u_{k-1} als Lösung der zuvor behandelten Gleichung zur Verfügung steht, was z.B. mit Hilfe eines Funktionsspeichers erfolgen kann.

Verfahren I liefert ein Anfangswertproblem eines Systems erster Ordnung, das in einer Rechenschaltung bestehend im wesentlichen aus n-1 Integratoren und zwei Funktionserzeugern (für u_o und u_n) parallel für alle u_i gelöst werden kann (Parallelmethode). Dagegen liefert Verfahren II ein Randwertproblem zweiter Ordnung, das nacheinander für k=1,2,... gelöst wird mit einer Rechenschaltung, die u.a. zwei Integratoren und einen Funktionsspeicher enthält (Serienmethode) Bild 10.5 kennzeichnet den wesentlichen Unterschied des Lösungsaufbaus bei Verfahren I und Verfahren II, d.h. bei einer Parallel- und einer Serienmethode. Während beim vorliegenden Beispiel Verfahren I stabil ist, erweist sich Verfahren II als instabil, wie man sich leicht überlegt, wenn die homogenen Differentialgleichungen der Systeme (10.4) und (10.9) und ihre Lösungskomponenten betrachtet. Wenn sich auch die Instabilität auf Grund der vorgegebenen Randwerte nicht voll auswirken kann, so ist doch zumindest mit Genauigkeitsverlusten zu rechnen.

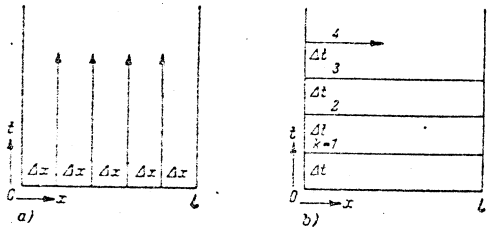


Bild 10.5 Integrationsrichtung der Ersatzsysteme
a) Parallelmethode, b) Serienmethode

Für die Serienmethode ist ein Funktionsspeicher nötig, der sich bei Hybridsystemen durch den Digitalrechner realisieren lässt. Bild 10.6 zeigt die Lösung mit Hilfe eines hybriden Rechensystems.

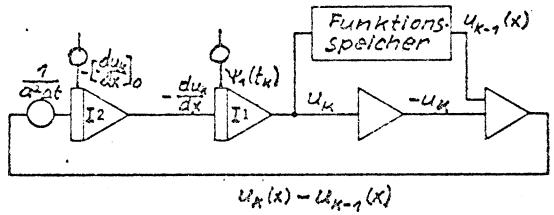


Bild 10.6. Serienmethode zur Lösung der eindimensionalen Wärmeleitungsgleichung

Die Anfangsbedingung für I2 muss in aufeinanderfolgenden Lösungsschritten so lange variiert werden, bis die Randbedingung $u_k(1) = \psi_2(t_k)$ erfüllt ist. Die Unterschiede der beiden Verfahren leiten sich davon ab, dass bei den meisten Problemen Anfangswerte für die Zeitkoordinate und Randwerte für die Raumkoordinaten gegeben sind.

Für konventionelle Analogrechner sind Parallelmethoden geeignet. Zu beachten ist, daß mit wachsendem Umfang der Probleme und mit Verkleinerung der Schrittweiten der Aufwand an Rechenelementen anwächst. Für iterative und Programmgesteuerte Analogrechner können dagegen Serienmethoden in Betracht gezogen werden. Eine Verkleinerung der Schrittweite hat keine Vergrößerung des Schaltungsaufwandes, sondern eine Vergrößerung des Rechenzeitaufwandes zur Folge, weil entsprechend mehr Schritte durchzurechnen sind. Jedoch sind stets bei der Aufstellung der Ersatzsysteme Konvergenz- und Atabilitätsprobleme zu beachten, die auch auf die Schrittweiten der quantisierten Variablen Einfluß haben. So erweist sich im vorliegenden Beispiel, behandelt mit Verfahren II, daß eine Verkleinerung der Schrittweite Δt die Instabilität vergrößert.

10.4 Diskretisierung von Ort und Zeit

Diese Methode wird für Digitalrechner immer angewendet. Man erhält ein System von gewöhnlichen Gleichungen, die gelöst werden müssen. Für Hybridsysteme ist dies ein Alternativverfahren, beispielhaft demonstriert durch die von Karplus beschriebene Methode, das als digitalrechnerorientiert bezeichnet werden kann. Hier wird die zu lösende Aufgabe zunächst für die herkömmliche Digitalrechnerlösung programmiert. Dann wird das Flußdiagramm sorgfältig analysiert, um zu bestimmen, ob das Ersetzen irgendeiner Schleife im Digitalprogramm durch Analogtechniken eine beachtliche Verringerung der Gesamtrechnerzeit zur Folge haben und andere bedeutende Vorteile bieten würde. Die Analog-Hardware wird daher als Unterprogramm betrachtet, die bei der Digitalrechnerlösung verwendet werden. Da Analoggeräte parallel arbeiten, wird die für das Unterprogramm erforderliche Lösungszeit von der Kompliziertheit der Berechnungen unabhängig und wird gänzlich von der Abtast- und Umwandlungsgeschwindigkeit der Anpassung bestimmt. Das unten beschriebene System ist nur ein Beispiel für diese neue Methode der Hybridrechnungen.

Zweck der zu beschreibenden Rechnertechnik und des Systems ist, bei technischen Aufgaben Hilfe zu leisten, die folgenden Gleichungen genorchen:

$$\frac{\partial}{\partial x}\left(\sigma(\phi)\frac{\partial \phi}{\partial x}\right) = k(\phi)\frac{\partial \phi}{\partial t} + f(\phi)$$

$$\frac{\partial}{\partial x}\left(\sigma(\phi)\frac{\partial \phi}{\partial x}\right) = k(\phi)\frac{\partial^2 \phi}{\partial t^2} + f(\phi) \qquad (8.20)$$

$$\frac{\partial^2}{\partial x^2}\left(\sigma(\phi)\frac{\partial^2 \phi}{\partial x^2}\right) = k(\phi)\frac{\partial^2 \phi}{\partial t^2} + f(\phi) \qquad (8.21)$$

$$\frac{\partial}{\partial x}\left(\sigma(\phi)\frac{\partial \phi}{\partial x}\right) = f(\phi) \qquad (8.22)$$

$$(8.23)$$

sowie ähnlichen Gleichungen in zwei und drei Raumdimensionen, wobei geänderte Formen dieser Gleichungen Glieder enthalten wie

$$k(\phi)\frac{\partial \phi}{\partial x}, \quad \frac{\partial f(\phi)}{\partial x}, \quad \frac{\partial^2 f(\phi)}{\partial x^2} \quad \text{und} \quad \frac{\partial^2 f(\phi)}{\partial t^2}$$

Der Einfachheit halber wird die Methode nur für die Raumdimension x beschrieben; die Ausweitung auf zwei und drei Dimensionen ist jedoch einfach. Zur Vereinfachung der Indexschreibweise wird das in Bild 8.11. dargestellte Schema angewandt. Ein typischer Punkt im Zeit/Raum-Gebiet wird mit 0 und benachbarte Punkte in der x- und t-Richtung werden mit 1 bis 10 bezeichnet. Im allgemeinen sind bei jedem Schritt einer Berechnung die Potentiale auf den Linien t_o und $t_o - \Delta t$ bekannt, während die Potentiale der Linie $t_o + \Delta t$ zu bestimmen sind.
Um die Verwendung eines einfachen und wirtschaftlichen Analogsystems für die Bearbeitung aller wichtiger nichtlinearer Feldaufgaben zu ermöglichen, werden zunächst

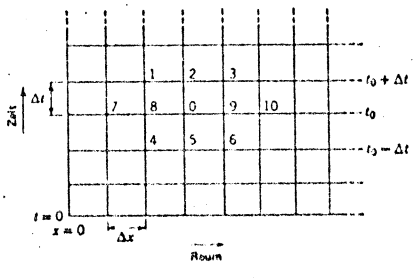


Bild 8.11 Raster endlicher Differenzen.

Gleichungen der allgemeinen Form von 8.20, 8.21 und 8.23. neu geordnet und dann tranformiert, damit sie wie folgt aussehen.

$$\frac{\partial^2 \phi}{\partial x^2} + F\left(\phi, \frac{\partial \phi}{\partial x}, \frac{\partial \phi}{\partial t}, \frac{\partial^2 \phi}{\partial t^2}, \ldots\right) = 0 \qquad (8.24)$$

Ebenso werden Gleichungen wie 8.22 , die den biharmonischen Operator enthalten, in folgende Form umgeschrieben:

$$\frac{\partial^4 \phi}{\partial x^4} + F\left(\phi, \frac{\partial \phi}{\partial x}, \frac{\partial \phi}{\partial t}, \frac{\partial^2 \phi}{\partial t^2}, \ldots\right) = 0 \qquad (8.25.)$$

Dann wird die an jedem Netzpunkt anwendbare Gleichung endlicher Differenzen neu geordnet, um folgende Form anzunehmen:

$$\phi_1 - 2\phi_2 + \phi_3 + \phi_i = 0 \qquad (8.26.)$$

Alle nichtlineare Glieder in den Gleichungen werden im Glied ϕ_i zusammengefaßt, so daß ϕ_i eine stark nichtlineare Funktion der Potentiale $\phi_0, \phi_1, \ldots, \phi_{10}$ ist. Die Technik für die Anordnung der nichtlinearen PDG in dieser Form wird in der angegebenen Literatur beschrieben. Für jeden endlichen Differenz-Rasterpunkt im Raumgebiet wird eine Analogrechnerschaltung gebaut, die die Gl. 8.26 erfüllt. Diese Schaltungen, die als Knotenbausteine bezeichnet werden, arbeiten alle simultan und sind wie in Bild 8.12. dargestellt miteinander verbunden. Die Eingabe jedes Bausteins entspricht der „Anfangsbedingung" ϕ_i, während die Ausgabe ϕ_2 das Potential zum Zeitpunkt $t_o + \Delta t$ darstellt, also die Aufgabenlösung für diesen Zeitschritt.
Das Netz der analogartigen Knotenbausteine wird in einer geschlossenen Schleife an einen Digitalrechner angeschlossen, um eine Hybridanlage zu bilden. Der Digitalrechner wird zum Speichern der Lösung, der Potentiale zum Zeitpunkt $t_o + \Delta t$ an den einzelnen Knotenpunkten benutzt, damit diese Daten in den

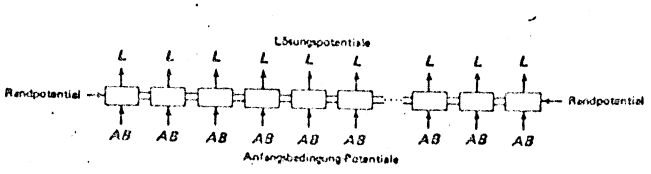


Bild 8.12 Anordnung von Knotenbausteinen. (AB = Anfangsbedingung, L = Lösung)

$$\frac{\partial}{\partial x}\left(\sigma(\phi)\frac{\partial \phi}{\partial x}\right) = k(\phi)\frac{\partial \phi}{\partial t} + f(\phi)$$

$$\frac{\partial}{\partial x}\left(\sigma(\phi)\frac{\partial \phi}{\partial x}\right) = k(\phi)\frac{\partial^2 \phi}{\partial t^2} + f(\phi) \qquad (8.20)$$

$$\frac{\partial^2}{\partial x^2}\left(\sigma(\phi)\frac{\partial^2 \phi}{\partial x^2}\right) = k(\phi)\frac{\partial^2 \phi}{\partial t^2} + f(\phi) \qquad (8.21)$$

$$\frac{\partial}{\partial x}\left(\sigma(\phi)\frac{\partial \phi}{\partial x}\right) = f(\phi) \qquad (8.22)$$

$$(8.23)$$

sowie ähnlichen Gleichungen in zwei und drei Raumdimensionen, wobei geänderte Formen dieser Gleichungen Glieder enthalten wie

$$k(\phi)\frac{\partial \phi}{\partial x}, \quad \frac{\partial f(\phi)}{\partial x}, \quad \frac{\partial^2 f(\phi)}{\partial x^2} \quad \text{und} \quad \frac{\partial^2 f(\phi)}{\partial t^2}$$

Der Einfachheit halber wird die Methode nur für die Raumdimension x beschrieben; die Ausweitung auf zwei und drei Dimensionen ist jedoch einfach. Zur Vereinfachung der Indexschreibweise wird das in Bild 8.11. dargestellte Schema angewandt. Ein typischer Punkt im Zeit/Raum-Gebiet wird mit 0 und benachbarte Punkte in der x- und t-Richtung werden mit 1 bis 10 bezeichnet. Im allgemeinen sind bei jedem Schritt einer Berechnung die Potentiale auf den Linien t_o und $t_o - \Delta t$ bekannt, während die Potentiale der Linie $t_o + \Delta t$ zu bestimmen sind. Um die Verwendung eines einfachen und wirtschaftlichen Analogsystems für die Bearbeitung aller wichtiger nichtlinearer Feldaufgaben zu ermöglichen, werden zunächst

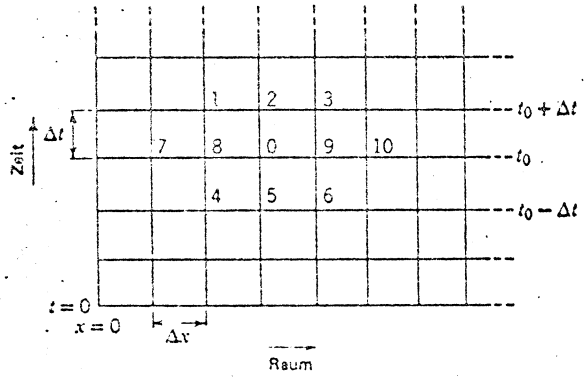


Bild 8.11 Raster endlicher Differenzen.

Gleichungen der allgemeinen Form von 8.20, 8.21. und 8.23. neu geordnet und dann tranformiert, damit sie wie folgt aussehen:

$$\frac{\partial^2 \phi}{\partial x^2} + F\left(\phi, \frac{\partial \phi}{\partial x}, \frac{\partial \phi}{\partial t}, \frac{\partial^2 \phi}{\partial t^2}, \ldots\right) = 0 \qquad (8.24)$$

Ebenso werden Gleichungen wie 8.22 , die den biharmonischen Operator enthalten, in folgende Form umgeschrieben:

$$\frac{\partial^4 \phi}{\partial x^4} + F\left(\phi, \frac{\partial \phi}{\partial x}, \frac{\partial \phi}{\partial t}, \frac{\partial^2 \phi}{\partial t^2}, \ldots\right) = 0 \qquad (8.25.)$$

Dann wird die an jedem Netzpunkt anwendbare Gleichung endlicher Differenzen neu geordnet, um folgende Form anzunehmen:

$$\phi_1 - 2\phi_2 + \phi_3 + \phi_i = 0 \qquad (8.26.)$$

Alle nichtlineare Glieder in den Gleichungen werden im Glied ϕ_i zusammengefaßt, so daß ϕ_i eine stark nichtlineare Funktion der Potentiale $\phi_0, \phi_1, \ldots, \phi_{10}$ ist. Die Technik für die Anordnung der nichtlinearen PDG in dieser Form wird in der angegebenen Literatur beschrieben. Für jeden endlichen Differenz-Rasterpunkt im Raumgebiet wird eine Analogrechnerschaltung gebaut, die die Gl. 8.26 erfüllt. Diese Schaltungen, die als Knotenbausteine bezeichnet werden, arbeiten alle simultan und sind wie in Bild 8.12. dargestellt miteinander verbunden. Die Eingabe jedes Bausteins entspricht der „Anfangsbedingung" ϕ_i, während die Ausgabe ϕ_2 das Potential zum Zeitpunkt $t_o + \Delta t$ darstellt, also die Aufgabenlösung für diesen Zeitschritt.
Das Netz der analogartigen Knotenbausteine wird in einer geschlossenen Schleife an einen Digitalrechner angeschlossen, um eine Hybridanlage zu bilden. Der Digitalrechner wird zum Speichern der Lösung, der Potentiale zum Zeitpunkt $t_o + \Delta t$ an den einzelnen Knotenpunkten benutzt, damit diese Daten in den

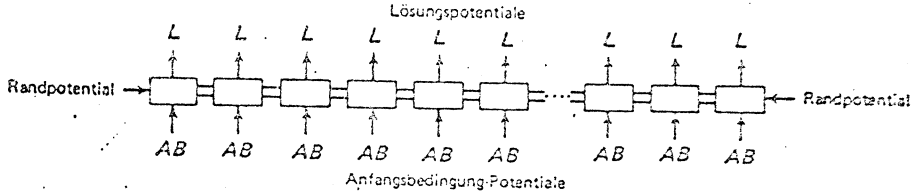


Bild 8.12 Anordnung von Knotenbausteinen. (AB = Anfangsbedingung, L = Lösung)

folgenden Zeitschritten als Anfangsbedingungen verwendet werden können. (Der Digitalrechner führt auch eine Reihe anderer wichtiger Funktionen durch.) Zur Umwandlung der Analogspannungen, die gleichzeitig an den Ausgangsklemmen aller Knotenbausteine erscheinen, in serielle Digitalform wird ein Multiplexer (Abtastgerät) zum stichprobenartigen Abtasten des Potentials ϕ_2 der einzelnen Knotenbausteine verwendet. Diese Gleichspannungen werden mit Hilfe eines Analog/Digital-Umsetzers in einen Digitalcode umgewandelt und im Speicher des Digitalrechners gespeichert. Anschließend werden diese Daten oder ihre geänderten Formen aus dem Digitalrechner ausgegeben, in Analogform zurückverwandelt und mit Hilfe eines zweiten Abtastgeräts (Verteilers) an die Eingangsklemmen der Knotenbausteine angelegt. Die Art der PDG, Änderungen in t, Nichtlinearitäten und Ungleicheiten in der räumlichen Anordnung der Rasterpunkte längs der Raumkoordinaten spiegeln sich nur in ϕ_i. Daher können die Knotenbausteine völlig gleich und von sehr einfacher wirtschaftlicher Bauart sein.

Der Vorteil des Hybridverfahrens liegt in der Tatsache, daß die Elemente der Analogschaltung parallel arbeiten und ungeachtet der Größe der Matrix in weniger als zwei Mikrosekunden zur richtigen Lösung, d. h. zur Umkehrung der Matrix gelangen. Daher wird die Gesamtberechnungszeit für jeden iterativen Unterzyklus durch die Zeitdauer bestimmt, die für die Umwandlung der seriellen Digitalinformation in parallele Analogform und umgekehrt erforderlich ist.

Zur Lösung der Gruppe von Differentialgleichungen für jeden Zeitschritt wird eine Anordnung von Analog-Knotenbausteinen verwendet. In solchen Analogsystemen erscheinen die Größen der abhängigen Variablen (die Potentiale $\phi(x, t)$ als Gleichspannungen, deren Größe den dargestellten Variablen proportional ist. Dementsprechend werden die Anfangsbedingungen für jeden Zeitschritt in Form von Gleichspannungen an das Netzwerk angelegt, und die Lösungspannungen werden aus dem Netzwerk ebenfalls in Form von Spannungen ausgegeben.

So nehmen die Knotenbausteine für die Lösung von 8.26. die einfache Form einer Analogaddierschaltung an, die einen hochverstärkenden Gleichstrom-Betriebsverstärker nebst einer Anzahl von Meßwiderständen enthält. Man kann die Gleichung 8.26. auch mit Hilfe eines rein passiven Analogsystems behandeln. In diesem Fall wird das Analognetz zu einem ein-, zwei-, oder dreidimensionalen, rechteckigen Gitter aus Meßwiderständen, und das Potential ϕ_i (modifiziert durch die Addition des Potentials ϕ_2 wird an die einzelnen Netzknotenpunkte jeweils über einen anderen Widerstand angelegt.

Es ist möglich, den Digitalrechner für die Erzeugung nichtlinearer Funktionen und das Analogsystem nur für die Matrixumkehrung zu verwenden. Als Alternative kann man einen Analogfunktionsgenerator in die Hybridschleife einfügen. Dieses Gerät arbeitet dann zeitlich parallel, so daß es die gewünschten nichtlinearen Funktionen für alle Punkte sequentiell erzeugt.

Das in Bild 8.13 dargestellte Rechnersystem ist eine geschlossene Schleife von analogen und digitalen Bauteilen mit digitalen Techniken für die Funktionserzeugung. Unter den dargestellten Geräten bedürfen lediglich die Aufgaben des

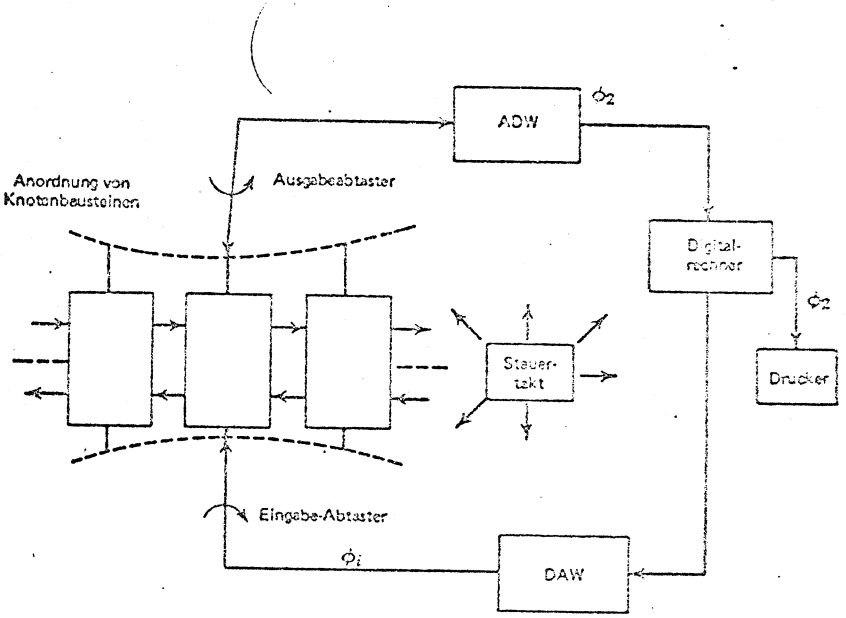


Bild 8.13 Hybridanlage für die DRDZ-Methode.

Digitalrechners einer weiteren Erläuterung. Diese können wie folgt zusammengefaßt werden:

1. Der Digitalrechner dient als Speicher für die Potentiale ϕ_2 (die Lösungen für die einzelnen Berechnungszyklen) und stellt diese bei den folgenden Berechnungszyklen als Anfangsbedingungen (ϕ_0, ϕ_4, ϕ_5 . . . ϕ_{10}) zur Verfügung.
2. Beim Fehlen eines Analog-Funktionsgenerators dient er zur Erzeugung des Potentials ϕ_i.
3. Da ϕ_i im allgemeinen eine Funktion des unbekannten Potentials ϕ_2 ist, sind für jedes Zeitinkrement mehrere iterative Rechnerabläufe durch die Hybridschleife erforderlich. Für jeden Netzpunkt werden Anfangswerte für ϕ_2 angenommen und die Berechnungen fortgesetzt, bis die Potentiale ϕ_2 in zwei aufeinanderfolgenden Iterativzyklen keine bemerkenswerte Änderung mehr aufweisen. Der Digitalrechner dient der Prüfung aufeinanderfolgender ϕ_2 auf solche Konvergenz. Sobald ein vorgegebenes Konvergenzkriterium erfüllt ist, ermöglicht der Digitalrechner das Weitergehen zum folgenden Zeitinkrement.

Der Hybridrechnerbetrieb umfaßt für die Lösung von Gleichungen wie 8.20 bis 8.23 folgende allgemeine Schritte:

Schritt 1. Das zu untersuchende ein-, zwei- oder dreidimensionale Feld wird durch Gitterpunkte endlicher Differenz dargestellt; für jeden Punkt wird ein Knotenbaustein vorgesehen. Diese Bausteine werden, wie oben beschrieben, miteinander verbunden, und an die den Feldgrenzen benachbarten Bausteine werden Randpotentiale angelegt.

Schritt 2. Die nichtlinearen Funktionen werden in das System eingeführt, indem man sie entweder im Digitalrechnerspeicher unterbringt oder den Analogfunktionsgenerator entsprechend einstellt.

Schritt 3. Die spezifizierten Ausgangsbedingungen und das Zeitinkrement Δt werden in den Digitalrechner eingespeist.

Schritt 4. Nun wird das Rechensystem auf automatischen Betrieb eingestellt. Für jedes Zeitinkrement wird eine Reihe von Unterzyklen durchgeführt, bis die Konvergenz erreicht ist. Dann wird die Lösung für diesen Aufgabenschritt, ϕ_2, für die einzelnen Netzpunkte ausgedruckt, und der Rechner geht automatisch zum nächsten Zeitschritt weiter.

Schritt 5. Nach Beendigung einer Anzahl von Zeitschritten hält der Rechner an.

Der Vorteil der Hybridtechnik gegenüber einer reinen Digitalrechnung liegt darin, daß die Berechnung der nichtlinearen Glieder und die Matrixinversion viel schneller erfolgen können. Bei der Verwendung einer reinen Digitalmethode wie in Bild 8.6 erfordert die Bearbeitung einer zweidimensionalen nichtlinearen Aufgabe mindestens sechs Multiplikationen, vier Divisionen und sechs Subtraktionen je Netzpunkt und Iterativunterzyklus. Demgegenüber erfordert die Hybridmethode weniger als 15 Mikrosekunden je Knotenpunkt und Unterzyklus schon mit handelsüblichen Anpassungsschaltungen, wenn der Analoganlage genügend Zeit zum Einspielen gewährt wird. Außerdem führt die Verwendung des Analognetzes schneller zur Konvergenz, so daß für die einzelnen Schritte im Zeitgebiet bedeutend weniger Unterzyklen erforderlich sind. Wird ein Analogfunktionsgenerator verwendet, so kann man zusätzlich Rechenzeit sparen. Darüber hinaus ermöglicht das Vorhandensein von Analogfunktionsgeneratoren die bequeme Einstellung und Änderung der nichtlinearen Funktionen, eine Eigenschaft, die für Konstrukteure von besonderem Interesse ist.

8.11 Monte-Carlo-Methode

Wird die in Abschnitt 8.7 beschriebene Monte-Carlo-Methode an einem Digitalrechner durchgeführt, so ist die Hauptschwierigkeit in den zahlreichen Zufallsgängen zu suchen, die an jedem Lösungspunkt durchgeführt werden müssen. Typischerweise sind 500 bis 1000 Zufallsgänge erforderlich, um eine Genauigkeit besser als 1 % zu erzielen. Die Anwendung der Hybridtechniken ermöglicht die Durchführung der Zufallsgänge im Analogbereich unter Verwendung eines Rauschgenerators, während der Digitalrechner für die Steuerung sowie für die Addition und Mittelung der Potentiale an den Randschnittpunkten dient. Little hat diese Methode bei einer verhältnismäßig langsamen Analoganlage mit elliptischen und parabolischen PDG beschrieben. Handler hat die Hybrid-Monte-Carlo-Methode auf weitere PDG ausgedehnt und den ultraschnellen Hybridrechner ASTRAC . erwendet.

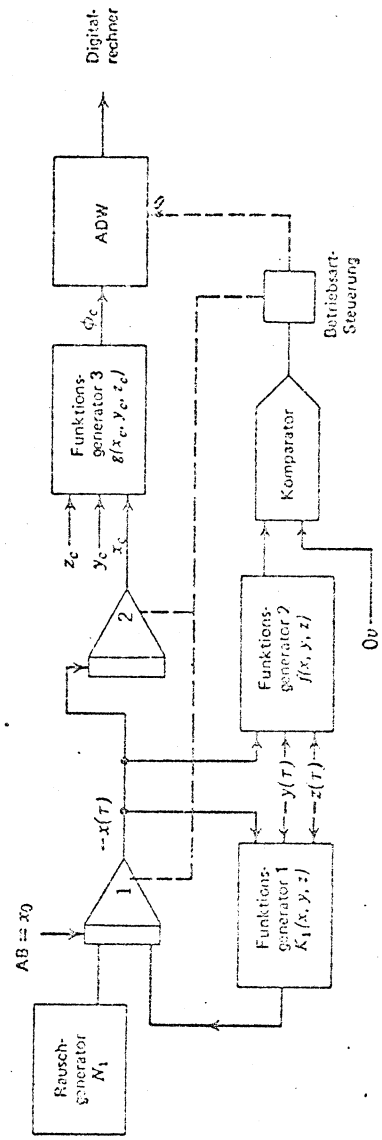


Bild 8.14 Hybridanlage für die Monte-Carlo-Methode.

§ 11. Simulation kontinuierlicher Systeme in Echtzeit

11.1. Aufgabenstellung

Simulation ist die rechnerische Nachbildung des Verhaltens eines dynamischen (physikalischen) Systems, wobei der Unterschied zur normalen Lösung einer Rechenaufgabe darin besteht, daß bei der Simulationsaufgabe das System in allen Einzelheiten und für beliebige Anfangsbedingungen oder Störfunktionen durch das Rechenprogramm nachzubilden ist. Dies bedeutet, daß alle Zustandsgrößen des Systems simultan berechnet und in der Form ausgegeben werden müssen, wie man sie am System selbst auch beobachten kann. Die Systemparameter sowie die eventuellen Anfangswerte und Störfunktionen müssen frei wählbar sein.

Sollen insbesondere alle Änderungen der Zustandsgrößen des Simulationsprogrammes im gleichen Zeitmaßstab erfolgen, wie beim zu simulierenden System selbst, so ist dies eine E c h t z e i t - S i m u l a t i o n.

Werden an eine Echtzeitsimulation extreme Zeitbedingungen gestellt, d.h. müssen die Antworten des Systems extrem schnell vorliegen, dann ist i.a. die Programmierung des Simulationsprogrammes immer mit Schwierigkeiten verknüpft. Die geforderten kurzen Antwortzeiten bewirken häufig eine beträchtliche Einschränkung der zur Verfügung stehenden Mittel. So werden insbesondere bei der digitalen Simulation Anforderungen an das Programm und den Rechner gestellt, die bei extremen Zeitbedingungen - wenn überhaupt - nur mit Mühe zu erfüllen sind, oder aber die Echtzeitbedingungen können nicht gewährleistet werden. Bei einer fehlerfreien Echtzeit-Simulation darf aber von außen nicht zu unterscheiden sein, ob die unter bestimmten aufgeprägten Bedingungen beobachteten Zustandsgrößen vom Simulationsprogramm geliefert werden oder vom System selbst. Vielfach bringen die bekannten Vorteile des Analogrechners hinsichtlich der Rechengeschwindigkeit eine beachtliche Verbesserung der erlaubten Zeitbedingungen. Bezüglich der Eigenfrequenzen des zu simulierenden Systems gelten die Ausführungen von § 2, d.h. Systeme mit Eigenfrequenzen über 1Hz müssen analog, Systeme mit Eigenfrequenzen unter 1Hz müssen digital simuliert werden. Die Gesichtspunkte für eine geeignete Aufgabenteilung bei einer hybriden Simulation, die in § 2 angeführt sind, behalten ihre volle Bedeutung, denn nur bei Berücksichtigung der spezifischen Rechnereigenschaften ist der optimale Einsatz eines Hybrid-Systems gewährleistet. Bei der Simulation unter Einbeziehung von Echtteilen treten die Probleme der Echtzeitsimulation besonders konzentriert auf, denn hier sind die Auswirkungen einer fehlerhaften Simulation i.a. unmittelbar und drastisch zu bemerken.

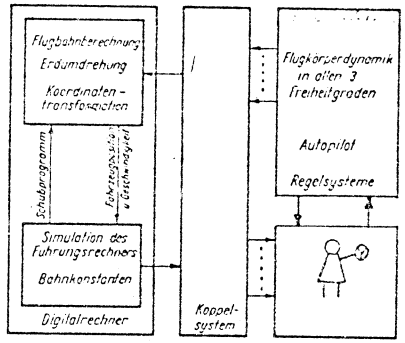


Bild 11.1 Simulation eines gelenkten Flugkörpers auf einem hybriden Rechnersystem unter Einbeziehung von Echtteilen.

Stellt der gelenkte Flugkörper in Bild 11.1 einen Satelliten dar, so muß für alle die Fälle, in denen die Piloten direkt das Flugverhalten beeinflussen, der Mensch als Echtteil mit in die Simulation einbezogen werden. Um ein wirklichkeitsgetreues Verhalten von interessierenden Systemgrößen zu erhalten, muß dafür gesorgt werden, daß die Reaktion auf das Verhalten des Echtteiles, also des Menschen in diesem Fall, in genau der Art und Weise in der gleichen zeitlichen Abhängigkeit wie beim realen System auftritt.

11.2 Beschreibung kontinuierlicher Systeme

Das Verhalten von kontinuierlichen Systemen wird i. a. durch Differentialgleichungen oder Systeme von Differentialgleichungen beschrieben.
In der Regelungstechnik, wo vielfach lineare Systeme behandelt werden, hat sich eine aus der Laplace-Transformation hervorgegangene Operatorenschreibweise durchgesetzt, die besonders günstig ist bei Darstellung von reinen Übertragungssystemen.
Bei nichtlinearen Systemen, bei denen das Überlagerungsprinzip nicht anwendbar ist, da die einzelnen Größen durch nichtlineare Beziehungen miteinander verknüpft sind, muß aber auf die Beschreibung mit Differentialgleichungen zurückgegriffen werden. Dabei ist es nicht sinnvoll, Systeme von Differentialgleichungen durch Eliminieren auf eine einzige Differentialgleichung entsprechend hoher Ordnung zurückzuführen.

Die allgemein ansetzbare Darstellung in Form einer Zustandsgleichung erhält man, wenn eine Differentialgleichung oder ein Differentialgleichungssystem höherer Ordnung in ein System erster Ordnung aufgelöst wird. Die Zustandsgleichung hat dann in Vektorenschreibweise folgende Gestalt:

$$\frac{d\,\underline{x}(t)}{dt} = \underline{f}\left[\underline{x}(t), \underline{u}(t), t\right] \qquad (11.1)$$

mit dem Zustandsvektor $\underline{x}(t) = \begin{pmatrix} x_1(t) \\ \vdots \\ x_n(t) \end{pmatrix}$ (11.2)

dem Eingangsvektor $\underline{u}(t) = \begin{pmatrix} u_1(t) \\ \vdots \\ u_m(t) \end{pmatrix}$ (11.3)

dem Ausgangsvektor $\underline{y}(t) = \begin{pmatrix} y_1(t) \\ \vdots \\ y_\ell(t) \end{pmatrix}$ (11.4)

Ein gegebenes System läßt sich damit in zwei Teilsysteme aufspalten: ein speicherfreier Teil und ein Systemteil, in dem vorherige Zustände

gespeichert werden.
Für den Ausgangsvektor gilt folgende Beziehung:

$$\underline{y}(t) = \underline{g}\left[\underline{x}(t), \underline{u}(t), t\right] \qquad (11.5)$$

Hat man es mit einem linearen System zu tun, so vereinfachen sich die Gleichungen (11.2.1) und (11.2.5).

$$\frac{d\,\underline{x}(t)}{dt} = \underline{A}(t) \cdot \underline{x}(t) + \underline{B}(t) \cdot \underline{u}(t) \qquad (11.6)$$

$$\underline{y}(t) = \underline{C}(t) \cdot \underline{x}(t) + \underline{D}(t) \cdot \underline{u}(t) \qquad (11.7)$$

Bei einem linearen, zeit-invarianten System sind die Matritzen $\underline{A}$, $\underline{B}$, $\underline{C}$ und $\underline{D}$ keine Funktionen der Zeit, sondern konstant.

Auch verteilte Parametersysteme, die durch partielle Differentialgleichungen beschrieben werden, können durch eine Zustandsgleichung der Form (11.1) angenähert werden, indem man nur eine Variable als kontinuierliche Variable bestehen läßt und nach den anderen diskretisiert.
Lineare Systeme mit variablen Koeffizienten lassen sich durch ein System linearer Zustandsgleichungen mit konstanten Koeffizienten beschreiben, sofern für die Koeffizientenfunktionen ein analytischer Zusammenhang gegeben ist.

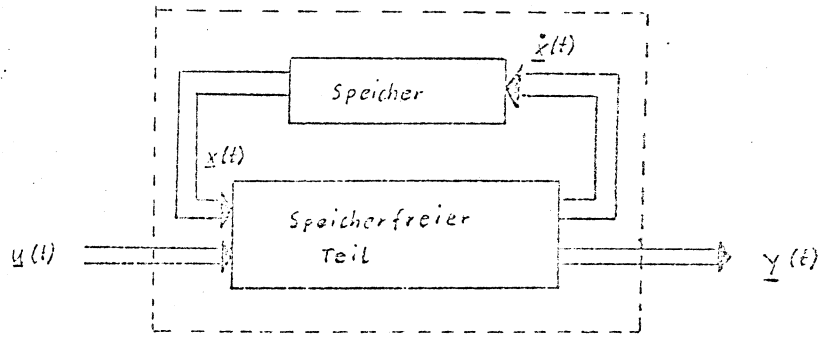


Bild 11.2: Schematische Darstellung eines kontinuierlichen Systems

11.3 Probleme der numerischen Integration

Wie bei der Lösung der Zustandsgleichung (11.1) bzw. der rechnerischen Nachbildung des Ausgangsvektor (Gleichung 11.5.) auf dem Analogrechner zu verfahren ist, wurde in § 4 gezeigt. Da aber schon bei etwas umfangreicheren Simulationsaufgaben das Leistungsvermögen des Analogrechners hinsichtlich Genauigkeit und Nachbildung von Teilsystemen allein nicht mehr ausreicht, wird i.a. ein Hybrid-System benötigt. Die hervorstechendste Eigenschaft des Analogrechners ist seine Fähigkeit, exakte Integrationen durchzuführen. Dem Bemühen, möglichst alle auftretenden Integrationen analog zu realisieren, sind aber in Form der zur Verfügung stehenden Datenkanäle Grenzen gesetzt. Zudem sollte immer beachtet werden, daß jene Zeitersparnis, die man durch analoge Durchführung einiger Operationen gewinnt, größer sein muß als der Zeitverlust, den die zusätzlichen Datentransfers kosten.
Ist man daher auf Grund von diesen Überlegungen gezwungen, verschiedene Integrationen auf dem Digitalrechner durchzuführen, so ist der Zustandsvektor des digitalen Untersystems mit Hilfe eines digitalen Algorithmus zu lösen, der den jeweils nächsten Zustand des Systems zur Zeit $t + \Delta t$ bestimmt. Symbolisch wird dies durch folgende Gleichung dargestellt.

$$\underline{x}(t+\Delta t) = \underline{f}^*[\underline{x}(t),\underline{u}(t),t] \qquad (11.6)$$

Die zeitgerechte Lösung der Gleichung (11.6) ist das zentrale Problem bei der digitalen bzw. hybriden Simulation unter Beachtung von Echtzeitbedingungen. Eine Übersicht über das Verhalten der einzelnen Integrationsmethoden ist nötig, um die Eignung für die Lösung spezieller Simulationsaufgaben abschätzen zu können.
Insbesondere interessieren:

Genauigkeit der gelieferten Lösung
Rechenzeitbedarf
Verhalten gegenüber Unstetigkeiten im Schrittzeitpunkt
Verhalten gegenüber Unstetigkeiten innerhalb eines Intervalles

Vor allem die benötigte Rechenzeit kann bei der Auswahl eines bestimmten Verfahrens bei Echtzeit-Simulation oder hybrider Simulation entscheidend sein.

Nichtiterative, vorwärtsschreitende Integrationsmethoden haben die allgemeine Form:

$$y_{i+1} = a_i y_i + a_{i-1} y_{i-1} + \dots + a_{i-p} y_{i-p} + h[b_{i+1} y'_{i+1} + b_i y'_i + \dots + b_{i-p} y'_{i-p}] \qquad (11.7)$$

Zur Berechnung von $y(t)$ an der Stelle $t = a + h(i+1)$ werden also die vorhergehenden Integralwerte an den Stellen $t = a + ih$, $a + (i-1)h$, $\dots, a + (i-p)h$ ebenso wie der Wert des Integranden an der Stelle $t = a + (i+1)h$ selbst sowie an den Stellen $t = a + ih, a + (i-1)h, \dots, a + (i-p)h$ mit herangezogen. Ist insbesondere $b_{i+1}=0$, so spricht man von einem offenen Verfahren. Daneben unterscheidet man noch Ein- und Mehrschrittverfahren.
Bei den Einschrittverfahren werden nur die Werte der unmittelbar vorhergehenden Stelle $t = a + ih$ verwendet.
Allgemein lassen sich Einschrittverfahren wie folgt darstellen:

$$y_{i+1} = y_i + \phi(y_i, t_i, h)h \qquad (11.8)$$

Zu dieser Klasse gehören die Eulerformel (Rechteckintegration), das Verfahren nach Heun (Trapezregel), die Taylorentwicklung, sowie alle Runge-Kutta-Methoden. Da bei diesen Methoden nur auf Werte der vorhergehenden Stelle zurückgegriffen wird, bezeichnet man sie als selbststartend, da beim ersten Schritt die zuerstbenötigten Werte in Form von Anfangsbedingungen vorliegen.

Die gebräuchlichen Mehrschrittverfahren lassen sich in folgender allgemeiner Form darstellen:

$$y_{i+1} = \varphi(y_i, y_{i-1}, y_{i-2}) + \phi(y_{i+1}, y_i, y_{i-1}, y_{i-2}, t_{i+1}, t_i, t_{i-1}, t_{i-2}; h)h \qquad (11.9)$$

Bei ihnen werden also auch Werte benötigt, die an Stellen auftraten, die um mehr als einen Schritt zurückliegen. Zu diesen Mehrschrittverfahren zählen fast alle Prediktor-Korrektor-Verfahren, die Adams-Verfahren und das Verfahren nach Milne. Da die Methoden dieser Gruppe zu Beginn der Berechnung mehr als nur die zur Verfügung stehenden benachbarten Werte benötigen, ist hier eine Anlaufrechnung mit einem Einschrittverfahren durchzuführen. In diesem Verhalten liegen die unterschiedlichen Reaktionen von Ein- und Mehrschrittverfahren auf Unstetigkeiten begründet. (Siehe Tabelle).

Eine eingehendere Kenntnis der verschiedenen Methoden erfordert deren Einsatz bei hybrider Simulation oder bei Echtzeitsimulation. Hier ist die Schrittweite h durch die Abtastfrequenz fest vorgegeben. Bei diesen Anwendungen sind häufig einfache, d. h. nicht rechenzeitintensive Verfahren, wie das Eulerverfahren vorteilhaft. Da die Schrittweite und damit das Abtastintervall entsprechend klein gewählt werden kann, ist das Verhalten gegenüber Unstetigkeiten besser und die Genauigkeit insgesamt vielfach größer. Das Runge-Kutta-Verfahren ist nur dann sinnvoll, wenn die rechten Seiten besonders einfach sind. Bei einer Integration am Analog-Rechner muß bei der digitalen Berechnung der rechten Seiten darauf geachtet werden, ob bei Differentialgleichungssystemen alle Komponenten des Lösungsvektors an den Digital-Rechner übergeben werden können. Am Analog-Rechner verursachen Iterationsschritte im Intervall $[t_i, t_{i+1}]$ umfangreiche Integriererssteuerungen zur Speicherung der entsprechenden Werte. Bei Echtzeitsimulation entsteht ein weiteres Problem dadurch, daß die Ergebnisfunktionen des Digital-Rechners in Form einer Treppenkurve vorliegen. Übergibt man eine solche Treppenkurve über die Koppelelektronik direkt auf die Echteile, so ist dies wegen deren Einschwingverhalten äußerst störend. Vielfach wird hier der Analog-Rechner als einstellbarer Tiefpaß zur Glättung dazwischengeschaltet. Es ist aber unter Einbeziehung der Zeitbilanz zu überlegen, ob spezielle Verfahren, die neben dem Funktionswert noch die Ableitung übergeben, zusammen mit geeigneten Approximationsmethoden nicht bessere Ergebnisse liefern.

Die Fehler der Integrationsverfahren lassen sich in zwei Gruppen einteilen:

Abbruchfehler (Diskretisierungsfehler, Truncation Error)
Rundungsfehler (Round-off-error)

Der Abbruchfehler entsteht durch den Ersatz der "kontinuierlichen" durch eine "diskrete" Integration. Der Abbruchfehler drückt sich in der Ordnung des Verfahrens aus, er wird angegeben durch das Glied, in dem die Übereinstimmung mit der Taylorreihe abbricht. Der Rundungsfehler hat seine Ursache in der digitalen Zifferndarstellung, denn beim Digital-Rechner wird jede Größe durch endlich viele Ziffernstellen repräsentiert. Er ist abhängig von der Wortlänge der Maschine. Insbesondere entsteht ein akkumulierter Rundungsfehler durch die Verknüpfung der mit lokalem Rundungsfehler behafteten Größen. Der akkumulierte Rundungsfehler hängt von vier Größen ab:

- vom lokalen Rundungsfehler (Wortlänge)
- vom Problem (Art und Anzahl der Rechenoperationen)
- vom Integrationsverfahren (Art u. Anzahl d. Rechenoperat.)
- von der Schrittweite (Anzahl der Rechenoperationen)

Die Stabilität der Lösungsfunktion ist gegeben, wenn die Lage der Wurzeln des charakteristischen Polynom die Bedingung erfüllt, daß es sich um periodische oder abklingende Vorgänge handelt. Der Begriff "numerische Instabilität" des Integrationsverfahrens beinhaltet etwas anderes. Numerische Instabilität - bedingt durch den akkumulierten Rundungsfehler - tritt dann ein, wenn sich die Ergebnisfunktion des Verfahrens immer mehr von der exakten Lösungsfunktion entfernt.
Die Wahl des Integrationsverfahrens und der Schrittweite h sollte auch unter diesem Gesichtspunkt erfolgen.

Tabelle für verschiedene Integrationsverfahren

	Einschritt-V.		Mehrschritt-Verf.	
	Euler	Runge-K. (4O)	Adams	P. C.
Genauigkeit	h^2	h^5	$h^3 \ldots h^5$	$h^3 \ldots h^5$
selbststartend	ja	ja	nein	nein
Funktionsauswertungen (Rechenzeit!)	1	4	1 (ältere Funkte müssen mitgespeichert werden)	2
Unstetigkeit im Schrittzeitpunkt	wird richtig verarbeitet		Neustart erforderlich	
Unstetigkeit während eines Intervalles	Ergebnis abhängig von der Lage des Sprunges zwischen zwei Stützstellen. Je mehr Stützstellen, desto besseres Ergebnis		Neustart erforderlich	

11.4. Schlussbemerkungen

Simulationssysteme zur Simulation dynamischer Systeme in Echtzeit werden benötigt, um Echtteile in das Simulationssystem einbeziehen zu können. Als Echtteil in diesem Sinne ist auch der Mensch anzusehen, wenn dieser unmittelbar am Simulationsablauf beteiligt ist. Bedingt durch die Operationszeit des Digitalrechners und die Notwendigkeit der Funktionstransformation diskreter Funktionen in kontinuierliche entstehen Zeitfehler, die nur für entsprechend kurze Zykluszeiten befriedigend korrigiert werden können. Dies führt dazu, dass die Zykluszeit wesentlich kürzer gewählt werden muss, als sich aus dem Abtasttheorem ergibt. Bei umfangreichen Systemen mit schnell ablaufenden Vorgängen stösst man auch heute noch an die Leistungsfähigkeit digitaler Rechenanlagen. Auch ist es fraglich, ob die extrem hohen Kosten für die Verwendung superschneller Digitalrechner ökonomisch zu rechtfertigen sind.

Die Verwendung hybrider Rechensysteme bietet eine Reihe von Vorteilen. Zunächst enthalten diese standardmässig alle Hardware- und Software-Einrichtungen zur Synchronisation und zum Datenaustausch. Durch den programmierbaren Analogrechner kann die Funktionstransformation den Bedürfnissen des Einzelfalles angepasst werden. Die Zeitbillanz des Digitalrechners kann durch zwei Massnahmen entscheidend verbessert werden. Teilsysteme mit rasch ablaufenden Vorgängen können als analoge Modelle realisiert werden. Dadurch kann man ohne Verlust an Genauigkeit mit grösseren Zykluszeiten arbeiten. Durch die parallele Arbeitsweise des Analogrechners kann auch der Zeitbedarf des Digitalrechners pro Zyklus verringert werden, so dass man Zykluszeiten erreicht, die für die geforderte Genauigkeit erforderlich sind. Schliesslich kann ein analoges Teilmodell ohne Ändern der Programmierung durch ein Echtteil ersetzt werden. Digitale Teilmodelle wird man dann in erster Linie dort verwenden, wo analoge Teilmodelle nicht befriedigend dargestellt werden können. Die Effektivität anderer Möglichkeiten - z.B. linearisierte analoge Teilmodelle zu verwenden und die Koeffizienten der Linearisierung von Zyklusintervall zu Zyklusintervall digital neu zu berechnen - ist noch nicht genügend untersucht worden.

Nachteil[illegible] für die Verwendung ist vor allem die Programmierung der analogen Systemkomponente. Wünschenswert wäre eine Programmierung auf problemorientierte Ebene durch Verwendung einer CSSL-Sprache. Allerdings sind bekannte Programmiersprachen zur Simulation kontinuierlicher Systeme bisher nur in digitalen Simulationssystemen ohne Berücksichtigung der Echtzeit-Forderungen implementiert worden.

Bei Verwendung einer problemorientierten Programmiersprache für die Simulation, mit der sowohl die Struktur des Modells als auch die Steuerung und Protokollierung des Simulationsablaufs formuliert werden muß sind vier verschiedene Sprachebenen gemäß Bild 11.3 sinnvoll.
Damit ein solches Programmiersystem aber dem erforderlichen Komfort zu bieten imstande ist - verbunden damit ist eine entsprechend geringe Effizienz - sind verschiedene Gesichtspunkte zu beachten:

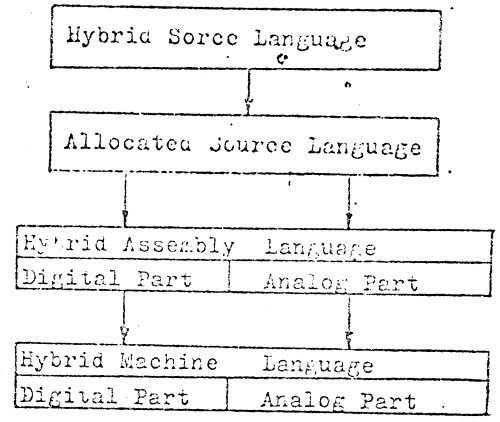


Bild 11.3: Sprachebenen des Programmiersystems

Eine Übersetzung aus der hybriden Quellsprache in die unterteilte Quellsprache, bei der eine Systemaufteilung auf die beiden Rechner zu erfolgen hat, sollte automatisch unter Wahrung von Benutzerwünschen durchgeführt werden. Für den analogen Systemteil sind dem Benutzer beim Skalieren Testhilfen zu gewähren. Außerdem ist ein leistungsfähiges Patchingsystem unbedingt notwendig, das die Verschaltung der analogen Rechenschaltung selbständig übernimmt.
In diesen Problemkreisen wird z.Zt. an verschiedenen Orten gearbeitet, wobei für manche Teilprobleme auch schon brauchbare Lösungen existieren.

Literaturverzeichnis zur Vorlesung "Analog- und Hybridrechner"

Bekey-Karplus: Hybridsysteme (zu §§ 2, 7, 8, 10)
Kohlhammer-V.

Giloi: Kursmaterialien zur Automatisierung (zu §§ 1, 11)
TU Berlin, Nr. 27 a

Giloi-Lauber: Analogrechnen (zu §§ 3, 7)
Springer V.

Heinhold-Kulisch: Analogrechnen (zu §§ 3, 7)
BI-Verlag

Kley: Elektronische Hybridrechner (zu § 1)
Franckh'scher-V.

Schwarz: Analogprogrammierung (zu §§ 3, 5, 6)
VEB Fachbuch Leipzig

Sydow: Programmierungstechnik f. elektrische Analogrechner (zu §§ 5, 6, 11)
VEB Technik Berlin

TfK-Fachbuch: Rechenanleitung für Analogrechner (zu § 4)

TfK-Forschungsberichte, Telefunken-Zeitung Jahrgang 39 (1966) Heft 1 (zu §§ 2, 8, 11)