theoreme eine grosse Rolle, welche Aufschluss geben über die Anzahl der Knoten der Oberschwingungen. Der Verfasser überträgt diese Theorie auf die Differenzenrechnung, womit Oszillations-Theoreme auch in der numerischen Mathematik ihre Anwendung finden können.

Nr. 5

Tabellen zur Erzeugung von Funktionen einer und zweier Variablen mit linearen Potentiometern. Von *Moheb Aziz Abdel-Messih* (1954). 33 Seiten. Broschiert Fr. 5.– (DM 5.–).
Katalog für die Erzeugung von rationalen Funktionen durch lineare Potentiometer. Es werden alle Schaltungen mit einem oder zwei Potentiometern zusammengefasst. Die Tabellen bilden ein Hilfsmittel für die Darstellung empirischer Funktionen durch Potentiometer-Schaltungen.

Nr. 6

Die mathematischen Grundlagen für die Organisation der elektronischen Rechenmaschine der Eidgenössischen Technischen Hochschule. Von *John Robert Stock* (1956). 76 Seiten. Broschiert Fr. 8.– (DM 8.–).
In den Jahren 1952–56 wurde im Institut für angewandte Mathematik der ETH, in Zusammenarbeit mit verschiedenen Firmen, eine elektronische Rechenmaschine konstruiert, deren Grundkonzeption sich in verschiedener Hinsicht von anderen Rechenautomaten unterscheidet. Die vorliegende Abhandlung beschreibt die für den Bau der ERMETH massgebenden Prinzipien sowie den allgemeinen Aufbau und schliesslich auch die Details der Abläufe der Rechenoperationen.

Nr. 7

Der Quotienten-Differenzen-Algorithmus. Von *Heinz Rutishauser* (1957). 74 Seiten. Broschiert Fr. 9.– (DM 9.–).
Zusammenfassung von drei in der Zeitschrift für angewandte Mathematik und Physik (ZAMP) erschienenen Arbeiten des Verfassers, nebst einigen Erweiterungen und Ergänzungen.

Nr. 8

Refined Iterative Methods for Computation of the Solution and the Eigenvalues of Self-Adjoint Boundary Value Problems. By *M. Engeli, Th. Ginsburg, H. Rutishauser* and *E. Stiefel* (1959). 107 pages. Fr. 17.– (DM 17.–).
Darstellung einiger bewährter Verfahren und numerischer Experimente über die iterative Lösung von Randwertaufgaben bei elliptischen partiellen Differentialgleichungen unter besonderer Berücksichtigung der Randwertprobleme der Elastizitätstheorie.

Die Reihe wird fortgesetzt

Mitteilungen aus dem Institut für angewandte Mathematik
AN DER EIDGENÖSSISCHEN TECHNISCHEN HOCHSCHULE IN ZÜRICH
HERAUSGEGEBEN VON PROF. DR. E. STIEFEL

Nr. 3

Automatische Rechenplanfertigung bei programmgesteuerten Rechenmaschinen

von

Heinz Rutishauser

BIRKHÄUSER VERLAG · BASEL/STUTTGART
1961

Nachdruck 1961
Printed in Switzerland

Mitteilungen aus dem Institut für angewandte Mathematik
AN DER EIDGENÖSSISCHEN TECHNISCHEN HOCHSCHULE IN ZÜRICH
HERAUSGEGEBEN VON PROF. DR. E. STIEFEL

Nr. 3

Automatische Rechenplanfertigung bei programmgesteuerten Rechenmaschinen

von

Heinz Rutishauser

BIRKHÄUSER VERLAG · BASEL/STUTTGART
1961

Nachdruck 1961
Printed in Switzerland

INHALTSUEBERSICHT

EINLEITUNG

Für die numerische Behandlung eines mathematischen Problems mit programmgesteuerten digitalen Rechengeräten muss der Rechenplan, d.h. die Gesamtheit der Befehle für die Durchführung der einzelnen Rechenschritte, entweder der Maschine auf einem Lochstreifen zugeführt werden oder dann in ihrem innern Speicherwerk vorhanden sein.

Es ist bekannt, dass die Rechenplanfertigung, d.h. die Aufstellung eines solchen Rechenplanes für ein bestimmtes Problem, oft eine erhebliche Arbeit ist, und in extremen Fällen kann sogar ein beträchtlicher Teil des gesamten Aufwandes auf die Vorbereitungsarbeit entfallen. Es sind daher Geräte zur Vereinfachung der Rechenplanfertigung konstruiert worden, wie etwa die "Coding machine" für Mark III (Vgl. [1] [1]). Ferner macht K.Zuse in seinem "Allgemeinen Plankalkül" [7] wesentlich weiter reichende Ansätze zur Automatisierung der Rechenplanfertigung, die jedoch ebenfalls die Konstruktion spezieller Geräte erfordern.

Demgegenüber gewann der Verfasser dieser Arbeit schon vor längerer Zeit die Ueberzeugung, dass es möglich sein müsse, die programmgesteuerte Rechenmaschine selbst dank ihrer Vielseitigkeit als Planfertigungsgerät zu verwenden. Dies würde also bedeuten, dass man mit diesen Rechenmaschinen nicht nur numerische Probleme löst, sondern auch Rechenpläne "berechnet".

Obgleich nun die beschriebenen Methoden im Prinzip auch mit schon bestehenden Maschinen durchgerechnet werden könnten, ist es für die Formulierung der Verfahren doch vorteilhaft, den folgenden Betrachtungen eine bestimmte, allerdings erst projektierte Maschine zugrunde zu legen, deren mathematische Eigenschaften, soweit sie im folgenden benötigt werden, in §1 beschrieben sind.

1) Eckige Klammern verweisen auf das Literaturverzeichnis.

Die in dieser Arbeit verwendeten Begriffe und Bezeichnungen sind im einzelnen nicht mehr erklärt, es wird vielmehr auf die bisher erschienene Literatur verwiesen, insbesondere [4] und [5]

Schliesslich sei noch erwähnt, dass der Verfasser an der GaMM-Tagung 1951 in Freiburg i.Br. über gewisse Teile dieser Arbeit vorgetragen hat (Vgl. auch [9]).

§ 1. KURZE BESCHREIBUNG EINER PROGRAMMGESTEUERTEN RECHENMASCHINE 2).

1.1. Zahlsystem, Zahlenspeicher.

Es ist vorgesehen, Zahlen im Dezimalsystem und zwar in halblogarithmischer Form darzustellen :

$$x = a.10^b$$

Der Exponent b kann zwischen -31 und +31 variieren. Der Faktor a, der der Nebenbedingung $|a| < 10$ unterliegt, wird mit 9 Dezimalen nach dem Komma angegeben; die einzelnen Dezimalen sind nach Aiken [3]) dual verschlüsselt. Damit benötigt eine Zahl insgesamt 48 Dualstellen zu ihrer Darstellung, nämlich eine für das Vorzeichen, 40 für den Faktor a, 6 für den Exponenten b und eine weitere für ein spezielles Zeichen (Vgl. §1.4).

Es sei aber schon hier betont, dass |a| auch kleiner als 1 sein kann, denn bei diesem Projekt erfolgt ähnlich wie bei BARK [8] keine Normalisierung [4]) nach der Subtraktion, sondern erst

2) Diese Beschreibung ist einem Projekt entnommen, das gegenwärtig am Institut für angewandte Mathematik der ETH unter der Leitung von Prof.Dr. E. Stiefel ausgearbeitet wird.

3) Die Ziffer z wird durch die Dualzahl 3+z+3.sgn(z-4,5) dargestellt. (Vgl. auch [1] , [4]).

4) Unter der Normalisierung verstehen wir die bei den meisten anderen Maschinen dieser Art übliche Kommaverschiebung bei a unter entsprechender Reduktion von b, so dass $1 \leq |a| < 10$ wird.

unmittelbar vor der Multiplikation oder Division. Dieser Umstand ermöglicht u.a. die Bestimmung der einer Zahl x nächstliegenden ganzen Zahl durch Addition von $0{,}000\ 000\ 000 \times 10^9$.

Zur Aufnahme von Zwischenresultaten ist ein Zahlenspeicher mit 1000 Speicherzellen (von 0 bis 999 numeriert) vorgesehen. Im Zusammenhang mit diesem Speicher werden später einige Abkürzungen verwendet, nämlich :

(n) für die Zahl in der Speicherzelle n.

⟨x⟩ für die Adresse der Zahl x, d.h. für die Nummer der Zelle, in welcher die Zahl x gespeichert ist.

x→n für: "Speichere die Zahl x in der Zelle n".

1.2. Befehlsgebung.

Die geplante Maschine ist eine Einadressmaschine, jeder Befehl ist durch eine 6-stellige Zahl dargestellt, deren Ziffern die folgende Bedeutung haben sollen :

2 3 1 6 8 7

- 6 8 7 : Adresse der Zahl auf die die Operation ausgeübt werden soll.
- 1 : Indexziffer (Vgl. §1.6)
- 2 3 : Operationsziffern; diese bestimmen die auszuführende Operation, (Vgl. §1.3).

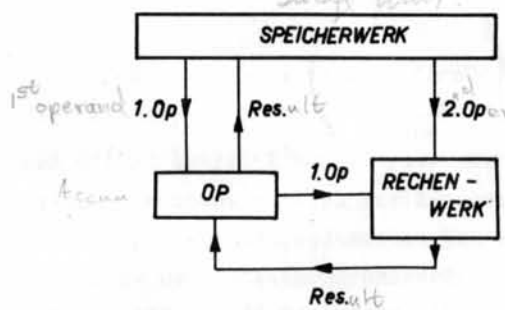
Die Befehle werden in der Maschine gespeichert, hierzu dient ein besonderer Befehlsspeicher mit 1000 Zellen, der 2000 Einadressbefehle aufnehmen kann (2 Befehle pro Zelle). Die Befehle werden in der Reihenfolge ausgeführt, in der sie gespeichert sind, nur durch die sog. Sprungbefehle (Vgl. §1.5) wird die normale Reihenfolge unterbrochen.

1.3. Die Organisation des Rechenwerks.

In Ruhestellung oder zwischen 2 Rechenoperationen ist immer eine Zahl im sog. Operandenregister Op, welche später je nachdem als Augend, Minuend, Multiplikand oder Dividend verwendet werden kann [5].

Zur Ausführung einer arithmetischen Operation wird dann durch einen entsprechenden Befehl der zweite Operand vom Speicherwerk ins Rechenwerk übertragen und sogleich die befohlene Operation ausgeführt. Das Resultat derselben geht wieder nach Op, von wo aus es jederzeit gespeichert werden kann; dabei wird aber Op nicht gelöscht. Zum Eingeben eines neuen ersten Operanden in Op unter vorgängiger Löschung desselben dient ein besonderer Befehl A n (Vgl. die Aufstellung auf der folgenden Seite). Durch diese Organisation wird insbesondere vermieden, dass bei der Multiplikation einer der beiden Faktoren durch einen besonderen Befehl bereitgestellt werden muss.

Fig. 1 : Schematische Darstellung des Rechenwerks.



5) Diese Art der Organisation findet sich im wesentlichen schon im Rechengerät Z4 von K.Zuse, scheint aber sonst nicht gebräuchlich zu sein.

Liste der Rechen- und Speicherbefehle.

Bezeichnung der Operation	Operationsziffern	Adresse	Wirkung der Operation	Dauer in ms
Leerbefehl	00	irrelevant	wirkungslos	40
A	01	n	(n) ⟶ Op	40
+	02	n	(Op)+(n) ⟶ Op	40
-	03	n	(Op)-(n) ⟶ Op	40
×	06	n	(Op)×(n) ⟶ Op	160
:	07	n	(Op):(n) ⟶ Op	560
Maj	08	n	Max [(Op),(n)] ⟶ Op	40
Min	09	n	Min [(Op),(n)] ⟶ Op	40
Sgn	10	irrel.	Sgn(Op) ⟶ Op	40
|x|	11	irrel.	|(Op)| ⟶ Op	40
-1	12	irrel.	-(Op) ⟶ Op	40
N	19	irrel.	(Op) wird normalisiert	40
S	20	n	(Op) ⟶ n	40

1.4. Ein- und Ausgang; die Zahl Q.

Das Eingeben der Anfangswerte eines Problems in die Maschine erfolgt durch einen Lochstreifen, den man zuvor mit einem Handlocher hergestellt hat. Auf diesem Lochstreifen sind die einzugebenden Zahlen nach mathematischen Grundsätzen zu Gruppen (sog. Blöcken) zusammengefasst. Die einzelnen Blöcke können von beliebiger und insbesondere untereinander von verschiedener Länge sein. Das Ende eines solchen Blocks, und damit automatisch der Anfang des nächsten, ist durch ein besonderes Zeichen Q, welches ebenfalls als Zahl aufgefasst wird und mit in die Maschine eingeht, gekennzeichnet. Ein solches Q-Zeichen steht auch am Anfang des Lochstreifens; ferner gibt man das Ende desselben zweckmässig durch zwei oder mehrere aufeinanderfolgende Q-Zeichen an. Damit sieht ein Zahlenstreifen etwa wie folgt aus :

Q.....Q.....Q... ...Q.....QQ

1.Block 2.Block letzter Block

Diese Zahl Q kann gespeichert werden und man kann auch mit ihr rechnen: Das Resultat irgend einer Rechenoperation ist genau dann Q, wenn mindestens einer der Operanden Q ist. Von den übrigen Zahlen unterscheidet sich Q durch die 48.te Dualstelle, welche bei allen normalen Zahlen 0, bei Q dagegen 1 ist.

Wird nun ein Zahlenstreifen in eine der Ablesestationen der Rechenmaschine eingelegt, so läuft er automatisch bis zum nächsten Q-Zeichen und zwar so, dass die darauffolgende Zahl zum Ablesen bereit ist; das erste Q-Zeichen geht also nicht in die Maschine ein. Das eigentliche Abtasten der Zahl und Uebertragen derselben ins Rechenwerk erfolgt auf einen entsprechenden Befehl hin, den wir aber im folgenden nicht brauchen.

Auch den Rechenplan gibt man vor Beginn der Rechnung auf einem Lochstreifen (sog. Befehlsstreifen) in die Maschine ein; hierbei tastet die Maschine immer zwei Befehle miteinander ab und füllt damit eine Befehlsspeicherzelle.

Es ist nun, wie bereits erwähnt, das Ziel dieser Arbeit, Rechenpläne durch die Maschine berechnen zu lassen; hierbei ist es zweckmässig, die errechneten Befehle sogleich zu lochen. Hierzu dient ein spezieller Befehl:

LB : Loche von der in Op stehenden Zahl $a.10^b$, ohne Rücksicht auf den Exponenten, die letzten 6 Dezimalen von a.

Als weitere Befehle für den Ein- und Ausgang von Zahlen sind natürlich Befehle zum Lochen aller 48 Dualstellen einer Zahl, zur Entgegennahme von Zahlen von einer Tastatur, sowie zum Drucken von Resultaten mit einem der Rechenmaschine angeschlossenen Druckwerk vorgesehen.

1.5. Die bedingten Befehle.

Da ein Einadressbefehl nur 6 Dezimalen beansprucht, kann man in jeder Zelle 2 Befehle unterbringen. Als Adresse eines Befehls bezeichnen wir einfach seine Zellennummer im Befehlsspeicher und unterscheiden dann die beiden Befehle in derselben Zelle n als Befehl n,0 und n,5 (linker und rechter Befehl).

Die Befehle gelangen wie schon erwähnt normalerweise in der Reihenfolge zur Ausführung, in der sie gespeichert sind, also etwa 10^5,0 - 105,5 - 106,0 - 106,5 - etc. Gewisse Befehle - die sogenannten Sprungbefehle - bewirken eine Unterbrechung der normalen Reihenfolge :

Bezeichnung der Operation	Operationsziffern	Adresse	Wirkung des Befehls
C	35	n	Unbedingter Sprung : Führe als nächsten Befehl den Befehl n,0 aus, ohne jedoch Op zu löschen.
Cc	36	n	Bedingter Sprung : Handle wie unter C, sofern (Op) positiv ist, andernfalls gehe normal weiter zum nächsten Befehl.
CQ	37	n	Handle wie unter C, falls (Op) = Q ist, andernfalls gehe normal weiter zum nächsten Befehl.
Co	38	n	Handle wie unter C, falls (Op) = 0 ist, andernfalls gehe normal weiter zum nächsten Befehl.

Der nach einem Sprung als erster auszuführende Befehl n,0 heisst der Zielbefehl des Sprungs; anschliessend sollen die Befehle bis zum nächsten Sprungbefehl wieder in normaler Reihenfolge ausgeführt werden.

1.6. Das Rechnen mit Befehlen.

Die Maschine besitzt ausser den je 1000 Speicherzellen für Zahlen und Befehle noch 9 Register zur Aufnahme von dreistelligen ganzen Zahlen ohne Vorzeichen, nämlich die sogenannten Indexregister IR_k (k=1,...,9). Diese Indexregister haben folgende Funktion: Wenn in einem Befehl die Indexziffer nicht 0, sondern k ist, so bewirkt dies, dass die Adresse dieses Befehls modulo 1000 um die im Indexregister k stehende Zahl vergrössert wird, bevor er zur Ausführung kommt [6]). Beispiel : Steht die Zahl 993 in IR_4, so bewirkt der Befehl 01 4 586 nicht Ablesen aus der Zelle 586, sondern aus der Zelle 579 ($\equiv 586+993 \bmod 1000$).

Zum Speichern in einem Indexregister dient ein besonderer Befehl:

$$SI_k \quad (= 21\ 0\ 00k) \quad (k=1,2,..,9) \ ,$$

durch welchen von der in Op stehenden Zahl $a.10^b$ die letzten 3 Dezimalen von a in das Indexregister k übertragen werden. Durch eine vorangehende Addition von 10^9 erreicht man, dass gerade die Hunderter, Zehner und Einer von (Op) in IR_k gespeichert werden, ausserdem erscheinen dann negative Zahlen als sog. 10-Komplemente.

---

6) Eine solche Anordnung wurde bereits von T.Kilburn [2] vorgeschlagen (sog. "B-Tube"). Dagegen wird bei anderen Projekten, insbesondere Mark III, der Inhalt eines Indexregisters nicht zur Adresse eines Befehls addiert, sondern lediglich an deren Stelle eingesetzt. Die additive Wirkung der Indexregister bringt aber wesentliche Vorteile.

Die Indexregister ermöglichen also das Aufrufen einer Zahl aus dem Speicherwerk, deren Adresse zuvor von der Maschine selbst errechnet wurde. Man kann die Indexziffer aber auch in einem Sprungbefehl einsetzen, z.B. 35 4 586. Der Zielbefehl dieses Sprungs ist dann nicht 586,0, sondern 586,0 + (IR_4) .

Ferner steht noch der folgende Befehl im Zusammenhang mit dem Indexregistern :

BZ k (=22 0 00k) (k=1,2,..,9).

Durch diesen Befehl wird der momentane Stand des sog. Befehlszählers (sequence counter) - das ist die um 1 vergrösserte Adresse dieses Befehls - im Indexregister k gespeichert, ohne dass Op verändert wird. Man braucht diesen Befehl, wenn man an einer nicht a priori bekannten Stelle eines Rechenplans auf einen Unterplan springen und nachher die Rechnung an der gleichen Stelle wieder fortsetzen will.

1.7. Konstante im Rechenplan.

Es ist oft bequemer, ganze Zahlen (insbesondere Adressen) mit Hilfe des folgenden Befehls in die Rechnung einzuführen, anstatt sie vor Beginn der Rechnung zu speichern :

Z n : Lösche das Register Op und setze die dreistellige ganze Zahl n hinein.

Ist in diesem Befehl die Indexziffer von 0 verschieden, etwa gleich k, so wird sinngemäss die Zahl n+(IR_k) nach Op übertragen. Dies kann insbesondere dazu dienen, eine Zahl aus einem Indexregister abzulesen.

§2. DIE BERECHNUNG EINES RECHENPLANS.

Es handelt sich hier darum, aus einer gegebenen Formel (Klammerausdruck), wie etwa

$$[A_1 : (A_2 + A_3)] - (A_1 \times A_2 \times A_3) \geqslant B \qquad 7) \qquad (2.1)$$

den zugehörigen Rechenplan zu berechnen.

2.1. Darstellung eines Klammerausdrucks.

Es ist klar, dass man der Rechenmaschine einige Angaben über die Natur des Klammerausdruckes machen muss, damit sie den gesuchten Rechenplan berechnen kann. Zu diesem Zweck kleidet man den Klammerausdruck in numerische Gestalt :

Wir betrachten zur Erläuterung das oben angeführte Beispiel (2.1); dieser Ausdruck enthält als "Elemente" Klammerzeichen, Operationszeichen, Operanden und ein Resultat; das letztere soll in Zukunft ebenfalls unter die Operanden gerechnet werden. Wir bezeichnen nun diese Elemente ohne Rücksicht auf ihre spezielle Natur der Reihe nach mit $E_1, E_2, \ldots, E_N$, und stellen der ganzen Formel noch ein leeres Element E_o voran. Im Beispiel (2.1) ist also speziell:

$$E_1 = [\ ,\ E_2 = A_1\ ,\ E_3 = :\ ,\ \ldots\ldots,\ E_{19} = B\ .$$

Es werden im folgenden allerdings einige Voraussetzungen über den Klammerausdruck gemacht: Einmal muss dieser mathematisch sinnvoll sein, indem z.B. nicht 2 Operationszeichen unmittelbar

---

7) Das Zeichen $\geqslant$ (sog. "Ergibt"-Zeichen) soll nach einem Vorschlag von K.Zuse andeuten, dass nicht eine Bedingungsgleichung vorliegt, sondern das aus den links stehenden Grössen errechnete Resultat mit B bezeichnet werden soll. In $x \geqslant p$ bedeutet das Ergibtzeichen sinngemäss, dass p den Wert x annehmen soll.

aufeinander folgen dürfen. Ferner sollen Ausdrücke wie ab+c immer als (a × b) + c geschrieben werden [8]). Es ist jedoch statthaft, in einer Klammer mehr als 2 Operanden zu verknüpfen, wenn die verknüpfenden Operationen entweder lauter Multiplikationen oder dann ausschliesslich Additionen und Subtraktionen sind. Die Maschine führt natürlich solche mehrfachen Verknüpfungen in einzelnen Schritten aus, z.B. (a+b-c+d) als ([(a+b)-c] +d) .

Ein Klammerausdruck K wird nun so arithmetisiert, dass jedem seiner Elemente E_k zwei Zahlen a_k und b_k zugeordnet werden. Zunächst sind die a_k wie folgt definiert :

$a_o = 0$

$a_k = a_{k-1} + 1$, falls E_k eine öffnende Klammer oder ein Operand ist.

$= a_{k-1} - 1$, falls E_k eine schliessende Klammer oder ein Operationszeichen ist. (2.2)

a_{N+1} ist das Q-Zeichen und gibt den Schluss des Klammerausdruckes an.

Ferner sind die Zahlen b_k wie folgt festgelegt :

Für k=0 oder für eine öffnende Klammer ist b_k = 010000 , das ist der als Zahl geschriebene Befehl für "Ablesen aus der Zelle 0."

Für das Ergibt-Zeichen oder für eine schliessende Klammer ist b_k = 200000 , das ist der Befehl S 0.

Ist E_k ein Operationszeichen, so ist b_k der Befehl für die Ausübung dieser Operation auf die in Zelle 0 stehende Zahl, also:

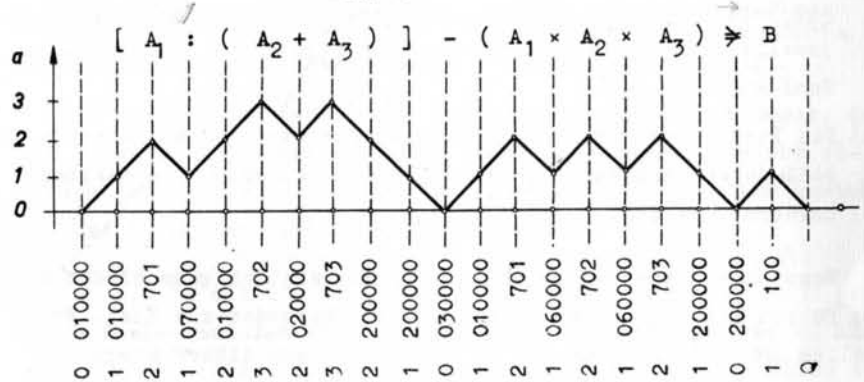
b_k= 020000 für Addition, = 030000 für Subtraktion, = 060000 für Multiplikation und = 070000 für Division.

Ist schliesslich E_k ein Operand, so ist b_k seine Adresse.

8) Diese Voraussetzung kann eliminiert werden, wie C.Böhm zeigt (Diss. ETH, noch unveröffentlicht).

Für den als Beispiel angeführten Klammerausdruck (2.1) zeigt die folgende Aufstellung die Zahlfolgen a_k und b_k, sowie eine graphische Darstellung der a_k :

Fig. 2



Dabei ist angenommen, dass die Zahlen A_i, bzw. B in den Zellen 700+i, bzw. 100 gespeichert seien. Aus der graphischen Darstellung erkennt man sofort, dass den "Bergspitzen" die Operanden des Klammerausdruckes, den "Tälern" dagegen die Operationszeichen entsprechen, während die Klammern in den "Hängen" liegen.

Es zeigt sich nun aber, dass es genügt, die Zahlenfolge b_k allein in die Maschine einzugeben, indem diese daraus die a_k gemäss

$a_o = 0$

$a_k = a_{k-1} + \mathrm{sgn}(15000 - b_k)$ (k=1,...,N) (2.3)

selbst berechnen kann. In der Tat ist ja $b_k < 15000$ für einen Operanden (Adressen sind immer kleiner als 1000), sowie für eine

öffnende Klammer, während für schliessende Klammern und Operationszeichen gilt : $b_k > 15000$.

Mit Hilfe der beiden Zahlenfolgen a_k und b_k, die einen Klammerausdruck vollständig charakterisieren, kann nun die Maschine folgende Aufgabe selbsttätig lösen :

a) Numerische Auswertung einer Formel, wobei die als Operanden einzusetzenden Zahlen durch ihre Adressen bestimmt sind. Zu diesem Zweck wird der Ausdruck von innen her unter Ausführung der jeweils befohlenen Operationen schrittweise zu einer einzigen Zahl abgebaut.

b) Die Formel soll nicht numerisch ausgewertet werden, sondern es soll der Rechenplan für die numerische Auswertung derselben berechnet werden.

Wenn auch diese beiden Aufgaben grundsätzlich verschieden sind, so führen doch in beiden fast dieselben Methoden zum Ziel. Wir wollen uns aber der Aufgabe b) zuwenden, denn dieser kommt für die geplante Maschine eine grössere Bedeutung zu. Die zur Auswertung eines Klammerausdrucks oder zur Berechnung des zugehörigen Rechenplans notwendige Rechnung ist nämlich sehr umfangreich, jedenfalls sehr viel länger als die Durchrechnung derselben Aufgabe mit Hilfe des fertigen Rechenplans. Es ist deshalb unzweckmässig, denselben Klammerausdruck mit Hilfe der Zahlfolgen a_k und b_k beispielsweise etwa 100-mal auszuwerten, anstatt zuerst den Rechenplan zu berechnen, der dann 100-mal benützt wird. Nur bei sehr schnellen elektronischen Rechenmaschinen dürfte der Rechenaufwand für die Aufgabe a) tragbar sein.

2.2. Das Prinzip der Rechenplanberechnung [9]).

Zu einem gegebenen Klammerausdruck K müssen zuerst die b_k-Werte in der richtigen Reihenfolge manuell in einen Zahlenstreifen gelocht werden, wozu das Planfertigungsgerät dienen kann. Es hat zu diesem Zweck eine Tastatur, die ausser den Operationszeichen und Adressen auch Klammern und Ergibtzeichen aufweist. Sind mehrere solche Ausdrücke vorhanden, so trennt man sie durch Q-Zeichen. In der Terminologie von §1.4 erscheint also jeder Klammerausdruck auf diesem Zahlenstreifen als ein Block.

Die eigentliche Berechnung des Rechenplans beginnt damit, dass die b_k-Werte nacheinander in die Maschine eingehen, welche sogleich nach (2.3) die zugehörigen a_k-Werte berechnet und zusammen mit den b_k in der nachstehenden Reihenfolge speichert :

$$a_o b_o a_1 b_1, \ldots, a_N b_N \, Q \quad .$$

Diese Zahlen heissen die Begleitwerte des Klammerausdrucks. Alsdann wird die Höhe H des Klammerausdrucks, d.h. das Maximum aller a_k nach folgender Rekursionsformel berechnet :

$$\left.\begin{aligned} h_o &= 0 \\ h_k &= \mathrm{Max}[\, h_{k-1}, a_k] \qquad (\text{für } k=1,2,\ldots,N) \\ \text{Dann ist } H &= h_N \, . \end{aligned}\right\} \qquad (2.4)$$

---

9) Im weiteren Verlauf der Arbeit wird oft auf einzelne Felder der am Schluss der Arbeit befindlichen Strukturdiagramme 1-3 hingewiesen, und zwar findet man

Feld 101 - 122 im Strukturdiagramm 1
Feld 201 - 214 " 2
Feld 301 - 315 " 3

Hierauf wird der durch die Werte a_k, b_k beschriebene Klammerausdruck in endlich vielen, noch näher zu beschreibenden Reduktionsschritten abgebaut, während gleichzeitig die gesuchte Befehlsreihe aufgebaut und sofort gelocht wird. Der entstehende Lochstreifen ist direkt als Befehlsstreifen verwendbar.

Wir bezeichnen den ursprünglich gegebenen Klammerausdruck mit K_1, seinen Zustand nach p-1 Reduktionsschritten mit K_p. Entsprechend bezeichnet man die Elemente von K_p mit E^p_k, seine Begleitwerte mit a^p_k, b^p_k $(k=0,1,\ldots,N_p)$ und seine Höhe mit H_p.

2.3. Verlauf eines Reduktionsschrittes.

Wir gehen aus vom Klammerausdruck K_p, der aus dem ursprünglich gegebenen nach p-1 Reduktionsschritten entstanden ist. Da die Reduktion bei den innersten Klammern ansetzen muss, besteht nun der p.te Reduktionsschritt darin, diese an Hand der a^p_k-Werte herauszusuchen und dann zu verarbeiten. Man erkennt nämlich die am stärksten in Klammern gekleideten Operanden daran, dass für sie a^p_k den Maximalwert H_p erreicht. Da hierbei in der Regel für mehrere Operanden $a^p_k = H_p$ ist, sucht man unter diesen zuerst denjenigen mit dem kleinsten k heraus :

Es sei i_{p-1} ein im vorangehenden Reduktionsschritt bestimmter Indexwert mit der Eigenschaft, dass $a^p_k < H_p$ für $k < i_{p-1}$. Dann bestimmt man, mit $k=i_{p-1}$ beginnend, die kleinste Zahl k mit der Eigenschaft $a^p_k = H_p$ (Vgl. Feld 107 im Strukturdiagramm 1); diesen Index nennen wir i_p. Es ist also

$$a^p_k \begin{cases} < H_p & \text{für } k < i_p \\ = H_p & \text{für } k = i_p \end{cases} \qquad (2.5)$$

Es kann der Fall eintreten, dass es einen solchen Indexwert i_p garnicht gibt, d.h. es ist für alle k zwischen i_{p-1} und N_p

$a^p_k < H_p$. In diesem Falle verkleinert man H_p um 1 und sucht mit k=0 beginnend, weiter nach i_p (Feld 105, 106).

$E^p_{i_p}$ ist dann notwendigerweise ein Operand, wie ein Blick auf die Formeln (2.2) und Fig. 2 unmittelbar zeigt. Ferner ist in der Regel auch $E^p_{i_{p+2}}$ ein Operand und $E^p_{i_{p+1}}$ das dazwischenliegende Operationszeichen. Allgemein stehen m Operanden [10]) $E^p_{i_{p+2\mu}}$ $(\mu=0,1,..,m-1)$ und m-1 Operationszeichen $E^p_{i_{p+2\mu-1}}$ $(\mu=1,2,\ldots,m-1)$ in derselben Klammer, und es ist nun die Befehlsfolge für diese Operationen inkl. die abschliessende Speicherung des mit R_p bezeichneten Zwischenresultates aufzustellen. Diese Befehlsfolge lautet nun für eine Einadressmaschine der in §1. beschriebenen Art :

	Operation	Operand
1. Befehl	A	$E^p_{i_p}$
2. Befehl	$E^p_{i_{p+1}}$	$E_{i_{p+2}}$
⋮	⋮	⋮
m. Befehl	$E^p_{i_{p+2m-3}}$	$E^p_{i_{p+2m-2}}$
m+1. Befehl	S	R_p

Gemäss den Festsetzungen von §2.1 sind für den k.ten dieser Befehle $(k=2,\ldots,m)$ $b^p_{i_{p+2k-2}}$ die Adressen des Operanden und $b^p_{i_{p+2k-3}}$ die von 4 Nullen gefolgten Operationsziffern für die Operation $E^p_{i_{p+2k-3}}$. Man erhält deshalb die numerische Form dieses Befehls einfach durch Addition der beiden Zahlen $b^p_{i_{p+2k-3}}$ und $b^p_{i_{p+2k-2}}$. Diese Regel gilt auch noch für k=1, weil das Element $E^p_{i_{p-1}}$ entweder eine öffnende Klammer oder dann das Leerelement E_o sein muss, somit die Zahl $b^p_{i_{p-1}}$ gerade den Befehl für das Ablesen aus der Zelle 0 darstellt.

---

10) m kann von der Maschine als die kleinste positive Zahl mit der Eigenschaft $a^p_{i_{p+2m}} < H_p$ leicht bestimmt werden (Vgl. Feld 110).

Für die Adresse des Zwischenresultates R_p, das im weitern Verlauf der Reduktion wieder als Operand auftreten wird, kann man eine laufende Nummer (wir wählen 1000-p) einsetzen, vorausgesetzt dass die betreffenden Zellen während der Rechnung wirklich frei sind. Somit erhält man die gesuchte Befehlsreihe für die Verknüpfung der m Operanden in derselben Klammer wie folgt :

1. Befehl : $b^p_{i_{p-1}} + b^p_{i_p}$
2. Befehl : $b^p_{i_{p+1}} + b^p_{i_{p+2}}$

.
.
.

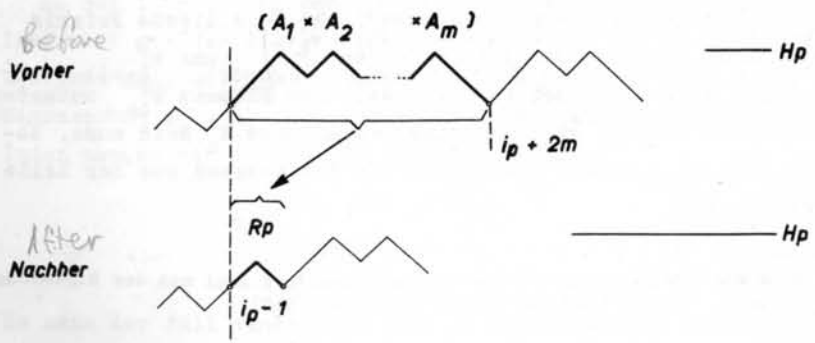
m. Befehl : $b^p_{i_{p+2m-3}} + b^p_{i_{p+2m-2}}$

m+1. Befehl : 201000 - p

Man braucht diese Zahl nur noch in den Befehlsstreifen zu lochen (Feld 111) wozu der Befehl LB dient (Vgl. §1.4).

Des weiteren besteht der p-te Reduktionsschritt darin, dass diese m Operanden samt den beiden Klammern und den m-1 Operationszeichen, also insgesamt 2m+1 Elemente E^p_k (nämlich $k=i_p-1$ bis

Fig. 3



$k=i_p+2m-1$), durch das Resultat R_p ersetzt werden (Vgl. Fig. 3). Ausserdem werden auf diese Weise 2m Plätze frei, die nachfolgenden Elemente sind daher um 2m Plätze nachzurücken (Feld 116-119). Damit ist der neue Klammerausdruck K_{p+1} wie folgt definiert :

$$E^{p+1}_k = \begin{cases} E^p_k & \text{für } k < i_p - 1 \\ R_p & \text{für } k = i_p - 1 \\ E^p_{k+2m} & \text{für } k = i_p, i_p+1, \ldots, N_p \end{cases}$$ [11].

$$N_{p+1} = N_p - 2m.$$

$$H_{p+1} = H_p$$

Entsprechende Aenderungen erfahren natürlich die Begleitwerte a_k und b_k. Nur für den neuen Operanden R_p, also für $k=i_p-1$, muss man a^{p+1}_k und b^{p+1}_k definieren: Da R_p ein Operand ist, während $E^p_{i_{p-1}}$ eine öffnende Klammer war, bleibt $a_{i_{p-1}}$ nach (2.2) unverändert, wie auch Fig. 3 zeigt. Dagegen muss der neue b-Wert an dieser Stelle gleich der Adresse von R_p, also 1000 - p sein. Schliesslich setzt man noch $a^{p+1}_{N_{p+1}} = Q$, d.h. auch das Q-Zeichen am Schluss muss um 2m Plätze nachgerückt werden.

Fig. 4 veranschaulicht den schrittweisen Abbau des als Beispiel angeführten Klammerausdrucks (2.1) und rechts aussen den gleichzeitigen Aufbau der Befehlsreihe.

---

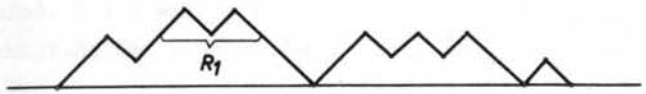
11) Die Maschine bestimmt den Index N nicht durch Abzählen, sondern immer als die kleinste positive Zahl mit der Eigenschaft, dass $a_{N_{p+1}} = Q$ ist. Hierzu dient der Befehl CQ.

Fig. 4

Abbau des Klammerausdrucks

Aufbau der Befehlsreihe

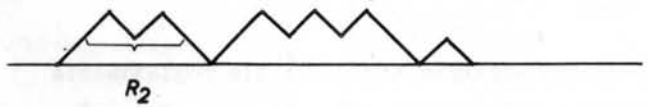
K_1: $[\,A_1 : (A_2 + A_3)\,] - (A_1 \times A_2 \times A_3) \Rightarrow B$



1. Red. : $H_1=3$, $i_1=5$, $m=2$; $R_1 = A_2+A_3$

A	702
+	703
S	999

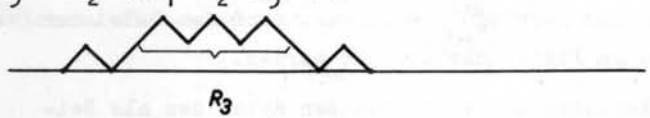
K_2: $[\,A_1 : R_1\,] - (A_1 \times A_2 \times A_3) \Rightarrow B$



2. Red. : $H_2=2$, $i_2=2$, $m=2$; $R_2 = A_1 : R_1$

A	701
:	999
S	998

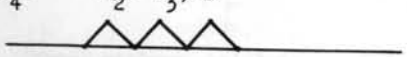
K_3: $R_2 - (A_1 \times A_2 \times A_3) \Rightarrow B$



3. Red. : $H_3=2$, $i_3=4$, $m=3$; $R_3 = A_1 \times A_2 \times A_3$

A	701
×	702
×	703
S	997

K_4: $R_2 - R_3 \Rightarrow B$



4. Red. : $H_4=1$, $i_4=1$, $m=3$; $B = R_2 - R_3$ (Schluss)

A	998
-	997
S	100

Fin

2.4. Operationen mit nur einem Operanden.

Die Berechnung des Rechenplans mit Hilfe der Begleitwerte beruht wesentlich darauf, dass jede Rechenoperation immer die beiden anliegenden Operanden verknüpft. Wenn aber Operationen mit nur einem Operanden, wie etwa $|x|$ oder $\sqrt{x}$, aber auch sin x oder e^x auftreten, müssen besondere Vorkehren getroffen werden. Wir machen folgende Fallunterscheidung :

a) Operationen mit einem Operanden, die von der Maschine durch einen einzigen Befehl ausgeführt werden (z.B. $|x|$) :

Von der Maschine aus gesehen, folgen diese Operationszeichen dem Operanden nach, und eine typische Befehlsfolge lautet:

Für $|a+b-c| \Rightarrow d$

A	a
+	b
-	c
|x|	
S	d

b) Operationen, die nicht in die Maschine eingebaut sind und deshalb mit Hilfe eines Unterplans ausgeführt werden müssen :

Die Befehlsreihe lautet für ein einfaches Beispiel, nämlich sin $(a+bx) \Rightarrow y$ wie folgt :

A	x	} Berechnung des Arguments
×	b	
+	a	
BZ	4	Speichert 1+Befehlszählerstand in IR_4
C	sin	Sprung nach Sinus-Unterplan.
------		} Leerbefehle

S	y	Speichern des Resultats.

Bei dieser Anordnung muss im Sinus-Unterplan berücksichtigt werden, dass das Argument vom Hauptplan her noch in Op steht (BZ verändert Op nicht). Ebenso muss der im Unterplan berechnete

Funktionswert vor dem Zurückgehen auf den Hauptplan nach Op gebracht werden, weil die weitere Rechnung dies verlangt. Zu diesem Zweck ist auch vorausgesetzt, dass Op von den Sprungbefehlen nicht verändert wird.

Da nach dem Unterplan der Befehl ⟨BZ⟩ +2 ausgeführt werden soll und auf Grund des BZ-Befehls die Zahl ⟨BZ⟩ +1 in IR_4 steht, wird das richtige Zurückgehen auf den Hauptplan durch den Befehl C 4 001 am Ende des Unterplans bewirkt (Die beiden Leerbefehle hängen mit dem Umstand zusammen, dass nur ein linker Befehl in einer Befehlsspeicherzelle ein Zielbefehl sein kann).

In beiden Fällen a) und b) haben wir unmittelbar vor dem Speicherbefehl (welcher einer schliessenden Klammer oder einem Gleichheitszeichen entspricht) noch ein Operationszeichen, auf welches kein Operand mehr folgt. Dies stört natürlich die Aufstellung der Zahlfolgen a_k und b_k. Zur Abhilfe fügt man unmittelbar nach dieser Operation einen blinden Operanden mit der Adresse 0 ein, wie wenn die Operation auf diesen Operanden auszuüben wäre. Ausserdem muss man im Fall b), wenn also die Operation durch einen Unterplan ausgeführt wird, noch den BZ-Befehl und die beiden Leerbefehle einfügen. Die Maschine erkennt den Fall b) daran, dass der b_k-Wert für ein solches Operationszeichen ≥ 350000 ist (Vgl. Feld 121). Insgesamt ergeben sich folgende Regeln für das Aufstellen der Begleitwerte bei Operationen mit nur einem Operanden :

Der Operand, bzw. der Ausdruck auf den die Operation anzuwenden ist, wird mit dem Operationszeichen in eine Klammer gekleidet; das Operationszeichen und der blinde Operand kommen an den Schluss.

Für das Operationszeichen wird als Begleitwert b_k eingesetzt: Im Falle a) die Operationsziffern mit 4 nachfolgenden Nullen, im Falle b) der Befehl für das Ausrufen des betreffenden Unterplans.

Unter Beachtung dieser Regeln verläuft die Bestimmung der a_k und die Berechnung des Rechenplans unverändert nach §2.2 und 2.3.

2.5. Anwendung auf Differentialgleichungen.

Der Rechenplan für die numerische Integration gewöhnlicher Differentialgleichungen zerfällt in zwei Teile, nämlich einen permanenten Plan, der durch die angewendete Integrationsmethode bestimmt wird, und einen individuellen Plan, der nur von der speziellen Differentialgleichung abhängt. Den letzteren pflegt man von Fall zu Fall herzustellen und in den permanenten Plan einzufügen [12]).

Nun kann man aber eine Differentialgleichung auch als Klammerausdruck betrachten und den zugehörigen Rechenplan durch die Maschine ausrechnen und in den permanenten Plan einfügen lassen. Wie in §2.4 gezeigt wurde, bereitet dabei auch das Auftreten elementarer transzendenter Funktionen in der Differentialgleichung keine Schwierigkeiten. Wenn damit bei einfach gebauten Differentialgleichungen auch kein grosser Zeitgewinn verbunden ist, so eliminiert man so doch durch das Ausschalten der manuellen Rechenplanfertigung eine Fehlerquelle. Im Hinblick auf den beträchtlichen Zeitverlust, den Fehler in einem Rechenplan verursachen, ist die automatische Rechenplanfertigung von ganz besonderer Bedeutung.

---

12) Wie man solche Unterpläne in einen Rechenplan einfügt, ist bei [3] , (Part II, Vol. III), sowie bei [6] ausführlich beschrieben.

§ 3. ANWENDUNG AUF ZYKLISCHE PROBLEME.

3.1. Das Strecken von zyklischen Rechenplänen.

Die Methoden von §2 beziehen sich nur auf Probleme mit linearer Struktur, die in der angewandten Mathematik nur eine untergeordnete Rolle spielen. Die Methoden sollen deshalb modifiziert werden, damit sie auch auf zyklische Probleme angewendet werden können.

Dabei treten insofern zwei verschiedene Problemstellungen auf, als man für eine zyklische Rechenaufgabe entweder einen zyklischen oder durch explicites Hinschreiben aller Operationen einen entsprechend längeren linearen Rechenplan herstellen kann (Diesen Uebergang zum linearen Plan nennen wir "Strecken eines zyklischen Rechenplans"). Der lineare (gestreckte) Rechenplan ist zwar viel länger als der zyklische und hinsichtlich der Grenzen der laufenden Indices gebunden, löst dafür aber die Rechenaufgabe meist wesentlich schneller, weil beim zyklischen Plan zu den eigentlichen Rechenoperationen noch solche für die Steuerung des Rechenablaufs hinzukommen. Allerdings fallen die Nachteile des zyklischen Plans umso weniger ins Gewicht, je grösser der Rechenaufwand pro Wert des laufenden Index ist.

Bei mehrfach zyklischen Problemen wird man mit Vorteil wenigstens die innersten Zyklen strecken, währenddem man die zyklische Natur der übergeordneten Zyklen nicht ändert. Die so erreichte Beschleunigung würde bei der in §1 beschriebenen Maschine die Rechenzeit für die Matrizenmultiplikation auf 40% des ursprünglichen Betrages reduzieren. Zusammenfassend sei also festgestellt :

Wenn es dank der Flexibilität der Maschine möglich ist, derart komprimierte Rechenpläne herzustellen wie bei dem in [4] , §4.7

angeführten Beispiel der Matrizenmultiplikation, so geht dies stets auf Kosten der Rechengeschwindigkeit, und zwar scheint eine Verdopplung der Rechenzeit nicht ungewöhnlich zu sein. Es empfiehlt sich deshalb - insbesondere bei einer nicht sehr schnellen Maschine -, von der Flexibilität nicht bis zum Aeussersten Gebrauch zu machen, sondern lieber längere Rechenpläne in Kauf zu nehmen und die Struktur zu vereinfachen. Um aber die gestreckten Rechenpläne, die sehr lang werden können, nicht manuell lochen zu müssen, kann man sie, wie im folgenden gezeigt werden soll, durch die Maschine ausrechnen lassen.

3.2. Darstellung der laufenden Indices in einem Klammerausdruck.

Wenn in einem Klammerausdruck bei einzelnen Operanden laufende Indices auftreten, so bedeutet dies, dass die numerische Auswertung wiederholt durchzuführen ist (nämlich einmal für jede Wertekombination der Indices), wobei die Adressen der betreffenden Operanden mit jedem Durchlauf in bestimmter Weise zu ändern sind.

Wollen wir nun einen solchen Rechenplan automatisch ausrechnen lassen, so müssen wir der Maschine mitteilen, in welcher Weise die Adresse eines Operanden $u_{\iota_1 \iota_2 \ldots \iota_n}$ von den Indices $\iota_1, \iota_2, \ldots, \iota_n$ abhängt. Der folgende Vorschlag beschränkt sich allerdings auf solche Fälle, wo diese Abhängigkeit linear ist : Sei

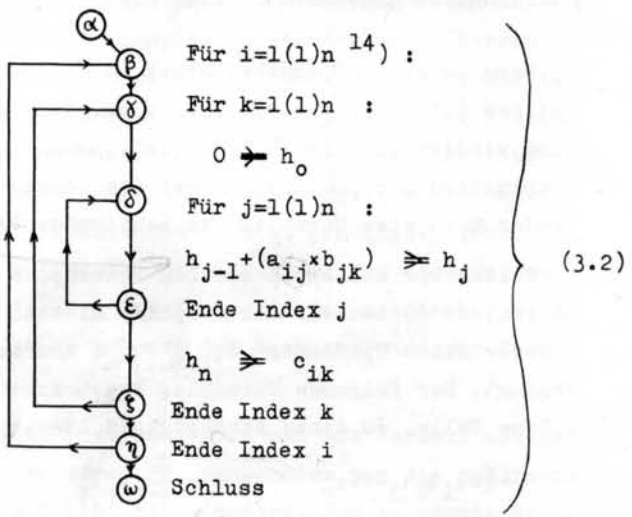
$$\langle u_{\iota_1 \iota_2 \iota_3 \cdots \iota_n} \rangle = b + \sum_{\varrho=1}^{n} \iota_\varrho I_\varrho \qquad (3.1)$$

Dann ordnet man dem Operanden u nicht nur 2 Begleitwerte a,b zu, sondern noch die sog. Indexbegleitwerte $10^{-3}I_1$, $10^{-6}I_2, \ldots, 10^{-3n}I_n$, welche die Abhängigkeit der Adresse von den Indices bestimmen. Wenn beispielsweise zu einem Element E die 4 Begleitwerte 3(=a), 52(=b), 2.10^{-3} und -50.10^{-9} gehören, so bedeutet dies, dass die Adresse von E von zwei Indices ι_1 und ι_3 abhängt :

$\langle E \rangle = 52 + 2\iota_1 - 50\iota_3$.

Die zu einem Element E gehörigen Begleitwerte werden nacheinander in der Reihenfolge a,b, $10^{-3}I_1$, $10^{-6}I_2$, ... etc. gespeichert, wobei man aber zur Entlastung des Speicherwerks einen verschwindenden Indexbegleitwert überhaupt weglässt [13]). Auf den letzten Indexbegleitwert folgt dann der a-Wert des nächstfolgenden Elementes des Klammerausdrucks.

Schliesslich muss noch festgelegt werden, welche Werte die auftretenden Indices durchlaufen. In einem Problem, das sich aus mehreren Formeln zusammensetzt, sieht dies etwa wie folgt aus (Es handelt sich hierbei um die Matrizenmultiplikation) :



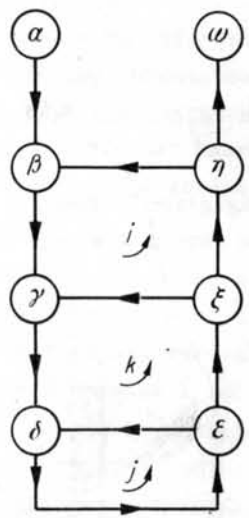
(3.2)

13) Dies kann man, da die eindeutige Zuordnung der Indexbegleitwerte zu den verschiedenen Indices bereits durch die Faktoren 10^{-3} gewährleistet wird.

14) Im Text von §3, und §4, sowie im Strukturdiagramm Fig.7 steht zur Vereinfachung der Schreibweise gelegentlich i,j,k anstelle der Indices ι_1, ι_2, ι_3. Man vermeide aber Verwechslungen mit dem in Text von §2, sowie in den Strukturdiagrammen 1,2,3, Fig. 6 und Fig. 8 zur Numerierung der Elemente eines Klammerausdrucks verwendeten Index i.

Offenbar muss man nun ausser den eigentlichen Klammerausdrücken auch die Vorschriften "Für i=..." und "Ende Index i", irgendwie numerisch verschlüsseln. Zuvor wollen wir uns aber an Hand der Struktur des Problems [15]) die Bedeutung dieser Vorschriften klar machen: Mit der Vorschrift "Für i= 1(1)n" beginnt ein neuer Index i zu laufen, demnach entspricht diese Vorschrift dem Punkt β in Fig.5, ebenso die Vorschrift "Für k=1(1)n" dem Punkt γ. Formeln, die nur einmal durchzurechnen sind - in unserm Beispiel gibt es keine solchen - wären der Strecke $\alpha\beta$ oder $\eta\omega$ zuzuordnen. Aber die Formel $0 \geqslant h_o$, die zwar keinen laufenden Index enthält, aber doch für jedes i und k auszuwerten ist, gehört zum Segment $\gamma\delta$. Auf $\delta\varepsilon$ haben wir dann 3 laufende Indices und beim Punkt ε wird geprüft, ob der Index j seinen Endwert erreicht hat. Wenn nicht, so springt man unter gleichzeitiger Erhöhung von j um 1 zum Punkt δ zurück, wenn ja, so erlöscht der Index j und die Rechnung geht weiter zur Formel $h_n \geqslant c_{ik}$, die dem Segment $\varepsilon\zeta$ angehört. Wir sehen also, dass der Punkt ε der Vorschrift "Ende Index j" entspricht. Analoges gilt für die Punkte ζ und η .

Fig. 5



Für das folgende definieren wir noch einen Stufenindex s. Jedem Punkt der Rechnung (d.h. jedem Punkt des Strukturdiagramms) ordnen wir eine Zahl s zu, die die Anzahl der laufenden Indices in diesem Punkt angibt; dieselbe Zahl s ordnen

15) Vgl. Fig. 5; ein ausführliches Strukturdiagramm folgt in Fig. 7.

wir auch dem zuletzt eingeführten Index zu und nennen sie seine Stufe. Damit hat in unserm Beispiel i die Stufe 1, j die Stufe 3, k die Stufe 2.

Die Matrizenmultiplikation wird durch die Formeln (3.2) vollständig beschrieben, indem diese das Strukturdiagramm eindeutig bestimmen, wie dies links neben den Formeln (3.2) angedeutet ist. Es genügt also, die Beschreibung (3.2) des Problems unter Beachtung der nachstehenden Regeln numerisch zu verschlüsseln:

1) Nach jedem Klammerausdruck (die Verschlüsselung derselben ist bereits besprochen) folgt ein Q-Zeichen.
2) Eine Vorschrift "Für ι_ς =u(v)w" wird durch 5 Zahlen verschlüsselt, auf welche ein Q-Zeichen folgt, um sie von nachfolgenden Klammerausdrücken oder weiteren Vorschriften zu trennen. "Für" wird durch eine sehr grosse Zahl, z.B. 10^{12} dargestellt, dann der Index ι_ς durch die Nummer der Zelle, die für die Speicherung seines zahlenmässigen Wertes vorgesehen ist; wir reservieren hierfür die Zelle $45+5_\varsigma$. Dann folgen die Zahlen u,v,w.
3) Die Vorschrift "Ende Index ι_ς" wird unabhängig von ς durch 2 aufeinanderfolgende Q-Zeichen dargestellt, ebenso das Ende des ganzen Problems.

Zur Erläuterung werden rechts nebenstehend die Begleitwerte [16]) für den Ausschnitt

"Für j=1(1)10" (α)

α { . . Q 10^{12} 55 1 1 10 Q

(Fortsetzung S.29)

16) Die Begleitwerte a sind weggelassen (Vgl. Fussnote 18 auf der folgenden Seite).

$h_{j-1}+(a_{ij}\times b_{jk}) \geqslant h_j$ (β)

"Ende Index j" (γ)

aus der Matrizenmultiplikation gezeigt, wobei folgende Speicherzellen reserviert sind :

99 für h_j (unabhängig von j, denn man braucht die alten h-Werte nicht mehr).

89+10i+j (Zellen 100-199) für a_{ij}

189+10j+k (Zellen 200-299) für b_{jk}

β { 99 + (89 10.10^{-3} 10^{-6} × 189 10.10^{-6} 10^{-9}) $\geqslant$ 99

γ { Q Q

. . .

3.3. Berechnung des Rechenplans für ein zyklisches Problem [17]).

Die Reihe der Begleitwerte, die nach §3.2 ein Problem charakterisieren, werde zunächst in einen Zahlenstreifen gelocht [18]) und dann in unveränderter Reihenfolge mitsamt den Q-Zeichen ins Speicherwerk der Maschine eingegeben. Aus diesen Begleitwerten soll nun die Maschine den gestreckten Rechenplan berechnen.

Diese Berechnung geht so vor sich, dass die Maschine ähnlich wie in §2 die Reihe der Begleitwerte absucht und die Befehlsreihe aufbaut, wobei aber die Abhängigkeit der Adressen von den Indices berücksichtigt werden muss : Wenn auf die Begleitwerte a_k, b_k eines Elementes E_k eine nicht-ganze Zahl folgt, so ist diese nicht a_{k+1}, sondern ein Indexbegleitwert für das Element E_k, der gemäss (3.1) mit $10^{3\varsigma}$ und mit dem Indexwert zu multiplizieren und zu b_k zu addieren ist. Dies erfordert natürlich einige Aenderungen im Strukturdiagramm 1 zwischen Feld 110 und Feld 121 (Vgl. Fig.6).

17) Vgl. Fussnote 9, Seite 15.

18) Dabei kann man die Begleitwerte a auch weglassen, da die Maschine diese nach (2.3) aus den b berechnen kann.

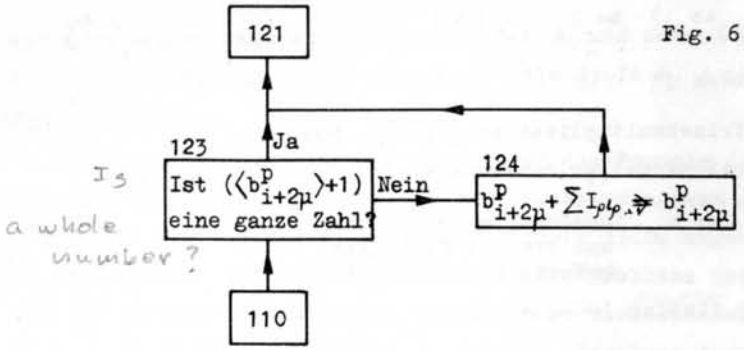


Fig. 6

Sobald man aber in der Reihe der Begleitwerte auf eine Zahl 10^{12} stösst, so zeigt diese eine neue Vorschrift "Für.." an; ein neuer Index ι_ϱ beginnt zu laufen und man findet in den folgenden 4 Zellen die Spezifikation des Index, sowie Anfangswert , Schrittweite und Endwert desselben. Nun wird diese Vorschrift insofern ausgeführt als der in §3.2 eingeführte Stufenindex s um 1 vergrössert wird und der Anfangswert des neuen Index ι_ϱ sogleich in die hierfür vorgesehene Zelle $45+5_\varrho$ übertragen wird, ebenso die Schrittweite in $46+5_\varrho$ und der Endwert in $47+5_\varrho$ (Feld 204 im Strukturdiagramm 2). Ausserdem muss aber die Adresse des neuen Index ι_ϱ irgendwo gespeichert werden, und zwar auf eine solche Weise, dass die Adressen der Indices niederer Stufe noch erinnert werden und sofort verfügbar sind, wenn s wieder verkleinert wird. Dies kann am besten dadurch geschehen, dass die Adresse jedes neu in Kraft tretenden Index sogleich in Zelle s gespeichert wird. Es ist dann also :

(s) = Adresse des laufenden Index $(=45+5_\varrho)$

((s))= Wert des laufenden Index $(=\iota_\varrho)$.

Anschliessend kann die Verarbeitung der auf die Vorschrift folgenden Klammerausdrücke in Angriff genommen werden (Feld 206), bis ein doppeltes Q-Zeichen das Ende der Wirksamkeit der Vorschrift

anzeigt (Feld 207). Alsdann ist der laufende Index um den vorgeschriebenen Betrag zu vergrössern (Feld 212) und es sind alle Klammerausdrücke von dem Punkt an, wo der Index eingeführt wurde, mit dem neuen Indexwert nochmals zu verarbeiten. Dies geht so weiter, bis der Index seinen Endwert erreicht hat (Feld 209); er erlischt dann beim nächsten doppelten Q-Zeichen und es ist s um 1 zu verkleinern (Feld 210). Die Berechnung des Rechenplans ist zu Ende, wenn s wieder den Wert o erreicht hat und dann nochmals ein doppeltes Q-Zeichen folgt.

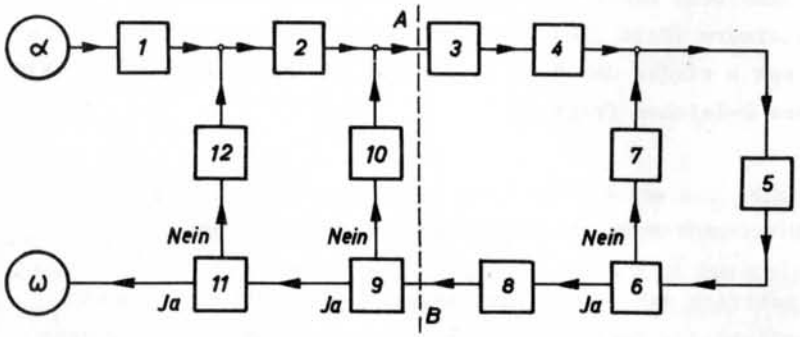
3.4. Teilweises Strecken von zyklischen Rechenplänen.

Bei mehrfach zyklischen Problemen ist es meist unrationell, einen vollständig gestreckten Rechenplan herzustellen, und zwar aus folgendem Grund : Das Strecken der innersten Zyklen (die zu den Indices höchster Stufe gehören) kann eine beträchtliche Verkürzung der Rechenzeit zur Folge haben, währenddem dann das Strekken auch der übrigen Zyklen kaum noch nennenswerte Vorteile bringt, sondern nur den Rechenplan enorm verlängert. Da aber das soeben beschriebene Verfahren nur die vollständige Streckung aller Zyklen erlaubt, muss es noch für die teilweise Streckung ergänzt werden. Wir wollen dies am Beispiel der Matrizenmultiplikation $\sum_1^n a_{ij}b_{jk} \geqq c_{ik}$ (Vgl. Strukturdiagramm Fig.7) erläutern : Von den 3 Zyklen ist der j-Zyklus der innerste und soll deshalb gestreckt werden ; dieser Vorgang entspricht der expliziten Ausschreibung der obigen Summe : $a_{i1}b_{1k} + a_{i2}b_{2k} + \ldots + a_{in}b_{nk} \geqq c_{ik}$. Der entstehende Rechenplan ist dann nur noch zweifach zyklisch, nämlich über i und k.

Fig. 7 zeigt das Strukturdiagramm für die Matrizenmultiplikation. Es ist zweckmässig, dasselbe längs der gestrichelten Linie zu zerschneiden, denn der eliminierende Index j kommt nur in

der rechten Hälfte vor, das Strecken des Rechenplans spielt sich also ausschliesslich in der rechten Hälfte ab. Für die linke Hälfte stellt man den Rechenplan separat her, es muss nur in den Punkten A und B der Anschluss an den berechneten Plan hergestellt werden.

Fig. 7



Legende :

1 :	$1 \hookrightarrow i$	7 :	$j+1 \hookrightarrow j$
2 :	$1 \hookrightarrow k$	8 :	$z_n \Rightarrow c_{ik}$
3 :	$1 \hookrightarrow j$	9 :	Ist k=n ?
4 :	$0 \Rightarrow z_o$	10 :	$k+1 \hookrightarrow k$
5 :	$z_{j-1}+(a_{ij} \times b_{jk}) \Rightarrow z_j$	11 :	Ist i=n ?
6 :	Ist j=n ?	12 :	$i+1 \hookrightarrow i$

Für die Berechnung der Befehlsreihe muss man die Index-Abhängigkeit der Adressen nach j einerseits und nach i,k anderseits zerlegen : Sei E ein Operand, dessen Adresse wie folgt dargestellt werden kann : $\langle E \rangle = b + j.J + f(i,k)$. Dann speichert man f(i,k) in einem der Indexregister, z.B. IR_5 und ruft den Operanden E mit einem Befehl auf, der 5 als Indexziffer und b+j.J als Adresse enthält. Da nun aber f(i,k) im rechten Teil des Strukturdiagramms konstant ist, kann man für diesen den Rechenplan nach den bisherigen Methoden berechnen (mit einem einzigen laufenden Index j), indem man die Verarbeitung der Grösse f(i,k) in die linke Hälfte

verlegt.

In unserm Beispiel treten nur 3 Elemente mit variablen Adressen auf, nämlich a_{ij}, b_{jk} und c_{ik} . Es werde gespeichert :

a_{ij} in Zelle $\alpha + j + [ni]$
b_{jk} in Zelle $\beta + nj + [k]$
c_{ik} in Zelle $\gamma + [ni + k]$.

Die eckig eingeklammerten Bestandteile entsprechen den f(i,k), wir weisen ihnen die Indexregister Nr. 1, bzw. 2, bzw. 3 zu, müssen dann aber bei der Aufstellung des Rechenplans für die linke Hälfte beachten, dass mit jeder Aenderung der Indices i,k (Feld 1, 2, 10 und 12 in Fig. 7) auch eine Aenderung der in den Indexregistern 1,2,3 gespeicherten Werte verbunden ist.

Das dem rechten Teil des Strukturdiagramms entsprechende Problem für das wir den Rechenplan auszurechnen haben, kann nun wie folgt formuliert werden :

$$\left.\begin{array}{l} 0 \Rightarrow z_o \\ \text{Für } j=1(1)n \\ z_{j-1}+(a_{ij} \times b_{jk}) \Rightarrow z_j \\ \text{Ende Index } j \\ z_n \Rightarrow c_{ik} \cdot \\ \text{Schluss.} \end{array}\right\} \quad (3.4)$$

Die Aufstellung der Begleitwerte und die Berechnung des Rechenplans erfolgt nach §3.2 und 3.3, nur ist darauf zu achten, dass die Grössen a_{ij}, b_{jk} und c_{ik} mit Hilfe der Indexregister aufgerufen werden und deshalb ihren Adressen die Indexziffern zuzufügen sind. Die Begleitwerte b und die Indexbegleitwerte lauten damit für diese 3 Grössen :

$$\left.\begin{array}{l} 1000 + \alpha \\ 10^{-6} \end{array}\right\} \text{für } a_{ij}, \quad \left.\begin{array}{l} 2000 + \beta \\ n.10^{-6} \end{array}\right\} \text{für } b_{jk}, \quad 3000 + \gamma \quad \text{für } c_{ik}$$

3.5. Eine Anwendung spezieller Natur.

Es sei der gestreckte Rechenplan für die Multiplikation einer Matrix $\{a_{ik}\}$ mit einem Vektor zu berechnen, wobei gewisse Elemente der Matrix als 0 vorausgesetzt werden können. Dieser insbesondere beim Arbeiten mit Systemen von linearen Differentialgleichungen recht häufig eintretende Umstand erlaubt bei vollständiger Streckung eine Verkürzung des Rechenplans, die sehr erheblich ins Gewicht fallen kann. Das Problem stellt sich wie folgt :

```
Für i=1(1)n
 0  ⩥ y_i
Für k=1(1)n
y_i+(a_ik× x_k) ⩥ y_i                    (3.5)
Ende Index k
Ende Index i
Schluss
```

Die Berechnung des Rechenplanes erfolgt im Wesentlichen nach den bisherigen Verfahren. Da aber die Formel $y_i+(a_{ik}\times x_k) \geqq y_i$ nur für solche Wertepaare i,k auszuwerten ist, für die $a_{ik} \neq 0$ ist, ist für $a_{ik} = 0$ die Berechnung der zugehörigen Befehle zu unterdrücken, was durch eine kleine Aenderung im Strukturdiagramm 2 erreicht werden kann.

3.6. Variable Indexgrenzen.

Die in §3.3 angegebenen Methoden zur Rechenplanberechnung leiden etwas unter der Einschränkung, dass für die Indexgrenzen feste Zahlen eingesetzt werden müssen, während doch in der Praxis gerade sehr oft der Fall eintritt, dass ein Index zwischen Grenzen läuft, die von anderen Indices abhängen. Ein klassisches Beispiel in dieser Richtung ist die Auflösung eines

linearen Gleichungssystems $\Sigma a_{ik}x_k=b_i$ nach dem Eliminationsverfahren von Gauss-Banachiewicz (Vgl. [3,6]):

```
Für k=1(1)n
 Für i=1(1)k-1
  a_ik ⩥ h_o
 Für j=1(1)i-1
h_j-1+(t_ij×t_jk) ⩥ h_j
 Ende Index j
-(h_i-1:t_ii) ⩥ t_ik
 Ende Index i
 Für i=k(1)n
  a_ik ⩥ h_o
 Für j=1(1)i-1
h_j-1+(t_ij×t_jk) ⩥ h_j                  (3.6)
 Ende Index j
 h_i-1 ⩥ t_ik
 Ende Index i
Ende Index k
Für i=n(-1)1
 b_i ⩥ h_o
 Für j=i+1(1)n
h_j-1+(t_ij× x_j) ⩥ h_j
 Ende Index j
 h_n ⩥ x_i
 Ende Index i
 Schluss .
```

Es ist aber leicht möglich, die Theorie auch noch auf variable Indexgrenzen auszudehnen. Sei eine der Indexgrenzen in "Für ι_ϱ = u(v)w" linear von Indices niedrigerer Stufe abhängig, z.B. $w = w_0 + \Sigma w_\varrho \iota_\varrho$. Dann stellt man die w_ϱ genau so wie die I_ϱ in §3.2 dar, nämlich durch $w_\varrho . 10^{-3\varrho}$ und speichert diese Grössen anschliessend an w_0.

In diesem Fall ist bei der Rechenplanberechnung jedesmal zu prüfen, ob die Indexgrenzen variabel sind, was sich durch Auftreten nicht-ganzer Zahlen unter den auf "Für" folgenden Werten äussert. Dies bedingt einige Aenderungen im Strukturdiagramm 2 (ähnlich der in Fig. 6 angegebenen Aenderungen im Strukturdiagramm 1 für die Indexabhängigkeit der Adressen).

Ferner muss man bei jeder in Kraft tretenden Vorschrift "Für ι_ϱ=u(v)w" prüfen, ob $\frac{w-u}{v} + 1$ (das ist die Anzahl der Werte, die ι_ϱ durchlaufen soll) positiv ist, andernfalls bleiben die folgenden Klammerausdrücke bis zum nächsten auf der gleichen Stufe stehenden doppelten Q-Zeichen unwirksam. Dieser Fall tritt insbesondere in (3.6) mehrmals auf; u.a. tritt für k=1 die Vorschrift "Für i = 1(1)k-1" [zweite Zeile in (3.6)], die ja dann "Für i=1(1)0" lautet, gar nicht in Kraft, so dass die nachfolgenden Zeilen bis zur Vorschrift "Ende Index i" (achte Zeile) übersprungen werden.

§ 4. BERECHNUNG EINES ZYKLISCHEN RECHENPLANS [19].

Für eine schnelle elektronische Rechenmaschine dürfte der in §3.1 geltend gemachte Grund für das Strecken von Rechenplänen nur noch in geringem Masse bestehen. Aus diesem Grunde seien hier unter Verzicht auf eingehende Behandlung einige Zeilen der Berech-

19) Vgl. Fussnote 9, Seite 15

nung eines ungestreckten Rechenplans für ein zyklisches Problem gewidmet. Im Zusammenhang mit §3.4 sei erwähnt, dass man die Rechenplanberechnung durch Kombination der in §3 und §4 beschriebenen Methoden auch so steuern kann, dass nach freier Wahl einzelne Zyklen des Problems gestreckt werden können, während die übrigen auch auf dem berechneten Plan ihre zyklische Natur beibehalten.

Die Berechnung eines zyklischen Rechenplans ist durch den Umstand erschwert, dass die Adressen der in den Formeln auftretenden Operanden in ganz verschiedener Weise von den Indices abhängen können, wie gerade das in §3.4 behandelte Beispiel der Matrizenmultiplikation zeigt. Man kann nun aber einfach so vorgehen, dass man jede indexabhängige Adresse vor der betreffenden Operation berechnet, in einem Indexregister speichert und dann den Operanden mit Hilfe der Indexziffer anruft.

Für die tatsächliche Durchführung der Berechnung eines zyklischen Rechenplans ist zunächst zu bemerken, dass die Verschlüsselung eines Problems genau gleich erfolgt wie in §3. Ferner geht die Rechenplanberechnung im Prinzip auch hier so vor sich, dass die Maschine die Begleitwerte absucht und dabei die Befehlsreihe aufbaut, wie es in §2.3 beschrieben ist. Aber es bestehen gegenüber §3 wesentliche Unterschiede in Bezug auf die laufenden Indices : Die Befehlsreihe ist für jeden Klammerausdruck nur einmal aufzustellen, dafür müssen dann die Befehle bei der nachfolgenden numerischen Auswertung für jede Wertekombination der Indices einmal ausgeführt werden. Hierzu sind natürlich die nötigen Vorkehren in den berechneten Rechenplan einzubauen. Dies bedingt:

a) <u>Für die Operanden</u> : Vor jeder Klammer müssen die darin enthaltenen Operanden auf variable Adressen untersucht werden und für jeden solchen Operanden muss die Befehlsfolge für die Berechnung der Adresse und Speicherung derselben in einem Indexregister aufge-

stellt werden. Erst dann berechnet man die Befehlsfolge für die eigentliche numerische Auswertung der Klammer.

Beispielsweise stellt man für den Operanden e_{ij} mit $\langle e_{ij} \rangle$ = $100 + 30i + 2j$ aus den Indexbegleitwerten 100, 30.10^{-3}, 2.10^{-6} die folgende Befehlsreihe her :

```
Z  0 100
S  0 099
Z  0 030
×  0 050   (× i)
+  0 099
S  0 099
Z  0 002
×  0 055   (× j)
+  0 099
SI 0 001
```

welche die Speicherung von $\langle e_{ij} \rangle$ in IR_1 veranlasst. Dafür ist dann in den Befehl, welcher den Operanden e_{ij} aus dem Speicherwerk aufrufen soll, die Adresse 0, aber die Indexziffer 1 einzusetzen. Fig. 8 zeigt die hierfür notwendigen Aenderungen im Strukturdiagramm 1, die zwischen Feld 107 und 109 anzubringen sind.

b) <u>Für die Indices</u> : Kommt man beim Absuchen der Begleitwerte zu einer neuen Vorschrift "Für =..", so bedeutet dies den Beginn eines neuen Unterzyklus. Es ist also wie in §3 der Stufenindex s um 1 zu erhöhen (Feld 3o3), dann die Adresse des neuen Index (nämlich $45+5\rho$) in die Zelle s zu übertragen und ferner die aus den 6 ersten Befehlen in (3.7) bestehende Befehlsfolge zu berechnen, welche zur Speicherung von Anfangswert, Schrittweite und Endwert des neuen Index in den hierfür bestimmten Zellen $45+5\rho$, $46+5\rho$, $47+5\rho$ dient:

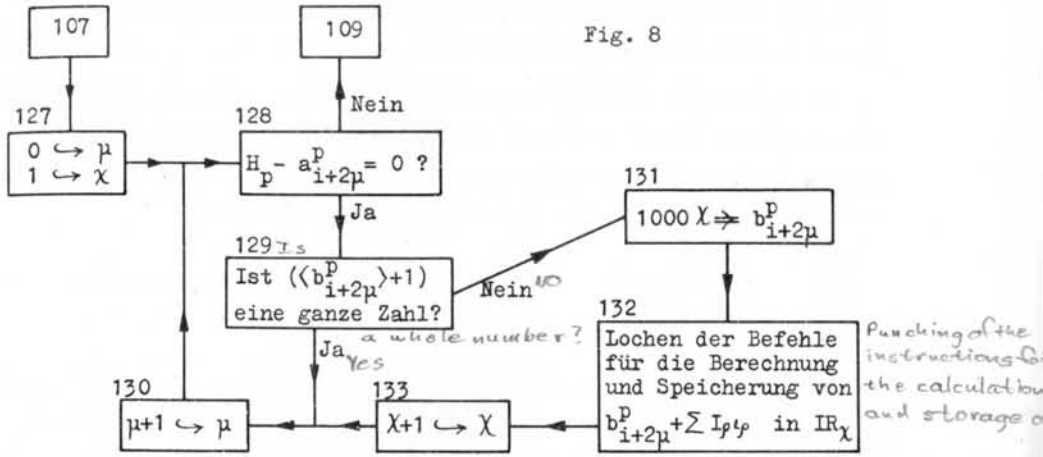


Fig. 8

```
Z   (r+2)
S    (s)
Z   (r+3)
S   (s)+1
Z   (r+4)
S   (s)+2
BZ    0 009
A     0 000
SI    0 008
Z     9 002
S     8 080
Leerbefehl
```

(3.7)

Die weiteren 6 Befehle in (3.7) haben den Zweck, die Adresse des auf diese 12 Befehle folgenden Befehls in der Zelle 80+s zu speichern. Dieser Befehl ist nämlich der Zielbefehl des Sprungs, der vom Ende des Zyklus wieder an seinen Anfang zurückführt (Entsprechend einer der Verbindungslinien $\eta \to \beta$, $\zeta \to \gamma$, $\varepsilon \to \delta$ in Fig.5). Seine Adresse muss gespeichert werden, damit man am Ende des Zyklus den erwähnten Sprungbefehl richtig formulieren kann. Der Leerbefehl hat zur Folge, dass dieser Sprungbefehl die gewünschte Wir-

kung auch dann hat, wenn der Zielbefehl in der rechten Hälfte einer Speicherzelle gespeichert ist.

Anschliessend erfolgt die Verarbeitung der nachfolgenden Klammerausdrücke (Feld 306) bis zum nächsten doppelten Q-Zeichen, wo wieder einige spezielle Befehle in den Rechenplan einzubauen sind, welche für den laufenden Index ι_ϱ wie folgt lauten :

```
Z  0  080
+  0  000
SI 0  009
A  9  000
SI 0  009
A  0  045+5ϱ
-  0  047+5ϱ
Co 9  000
A  0  045+5ϱ
+  0  046+5ϱ
S  0  045+5ϱ
```

(3.8)

Diese Befehle prüfen, ob der laufende Index seinen Endwert erreicht hat und rufen unter gleichzeitiger Erhöhung desselben um die Schrittweite an den Anfang des betreffenden Zyklus zurück, wenn dies noch nicht der Fall ist (Feld 309). Die ersten 5 dieser Befehle bereiten den Sprungbefehl Co 9 000 vor, indem sie die Adresse des Zielbefehls aus der Zelle s+80 holen und in IR_9 speichern. Die Maschine kann die in (3.8) auftretenden Adressen $45+5_\varrho$ $46+5_\varrho$, $47+5_\varrho$ leicht selbst in den berechneten Rechenplan einsetzen, da ja nach Voraussetzung $45+5_\varrho$ in der Zelle s gespeichert ist.

Weiter wird beim doppelten Q-Zeichen der Stufenindex s um 1 verkleinert (Feld 310); alsdann kann das Absuchen der Begleitwerte fortgesetzt werden. Wenn der Index s auf 0 gesunken ist und dann nochmals ein doppeltes Q-Zeichen folgt, so ist das Problem zu Ende; man setzt den Schlussbefehl in den Rechenplan ein (Feld 308 und 315).

Das Auftreten variabler Indexgrenzen bringt keine neuen Schwierigkeiten ausser den bereits in §3.6 genannten; man muss in die Befehlsreihe für die Speicherung der Indexgrenzen (3.7) die Berechnung derselben aus den Begleitwerten einbauen.

ANHANG : STRUKTURDIAGRAMME 1 - 3.

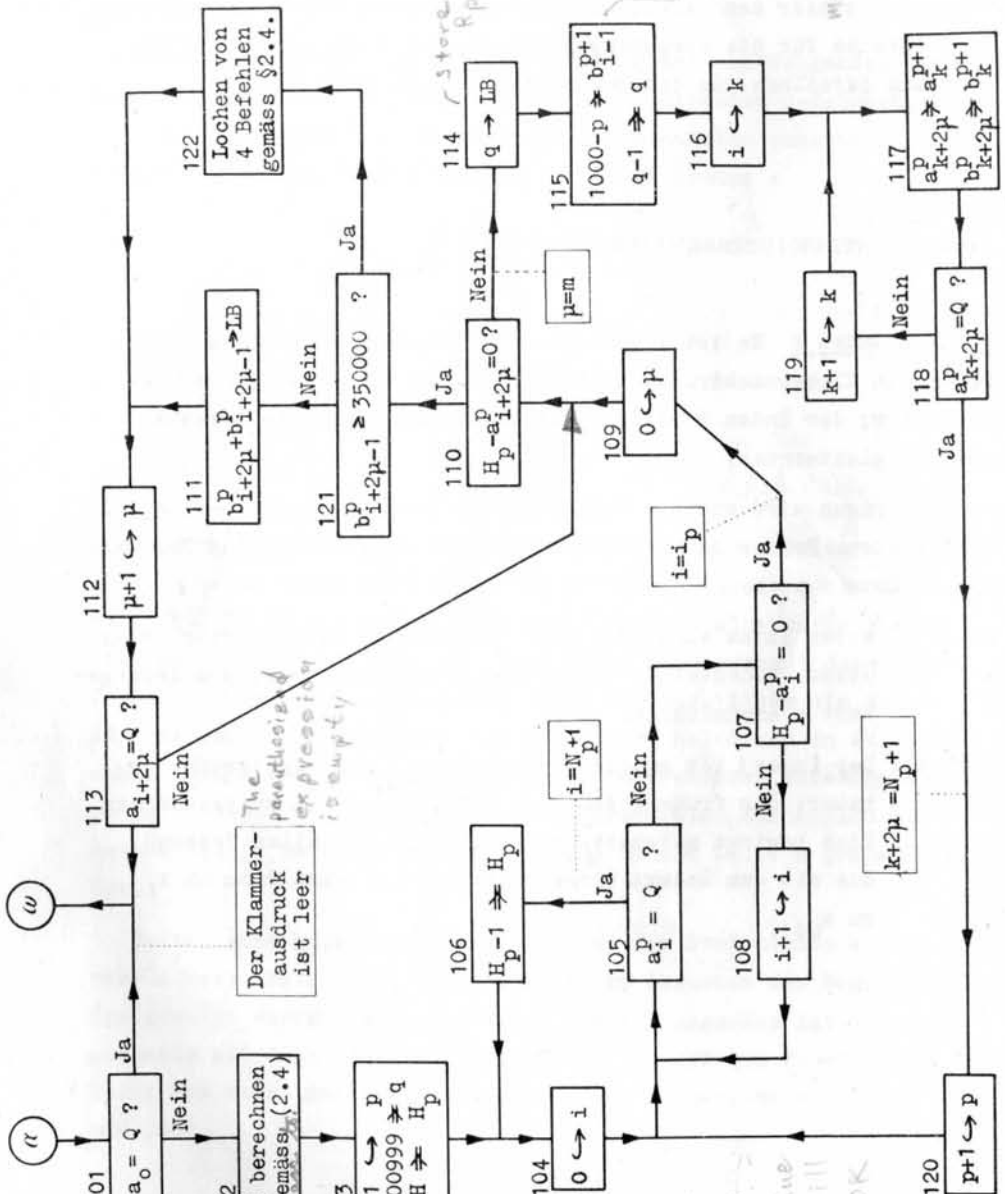
Erläuterungen : Es ist angenommen, die Begleitwerte eines auszuwertenden Klammerausdrucks seien bereits in den Zellen 500ff gespeichert; der Index r bedeutet im folgenden immer die Adresse eines Begleitwertes.

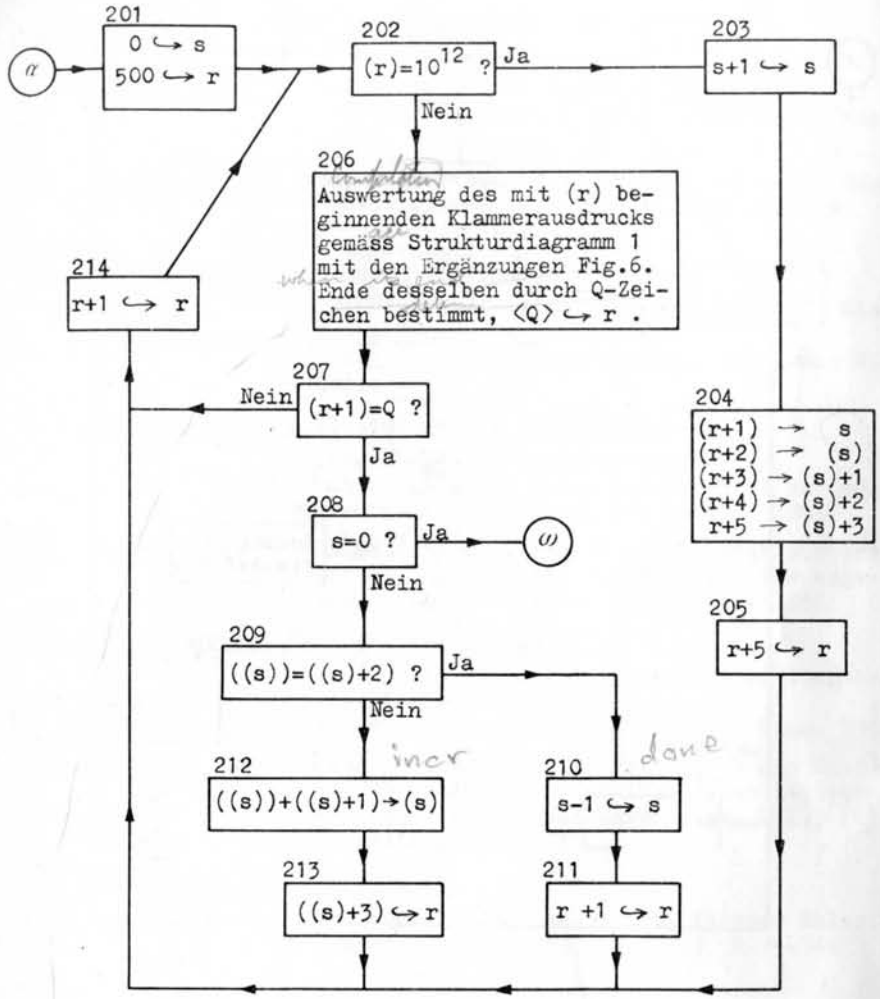
Im übrigen wird auf die Erklärung der Zeichen (),⟨ ⟩, → in §1.1 hingewiesen. Ferner sei an dieser Stelle die Bedeutung des Zeichens $\hookrightarrow$ erläutert :

$a \hookrightarrow \iota$, wobei links eine Zahl oder ein bereits eingeführter Index steht, bedeutet : Es ist ein neuer Index ι mit dem Anfangswert a einzuführen.

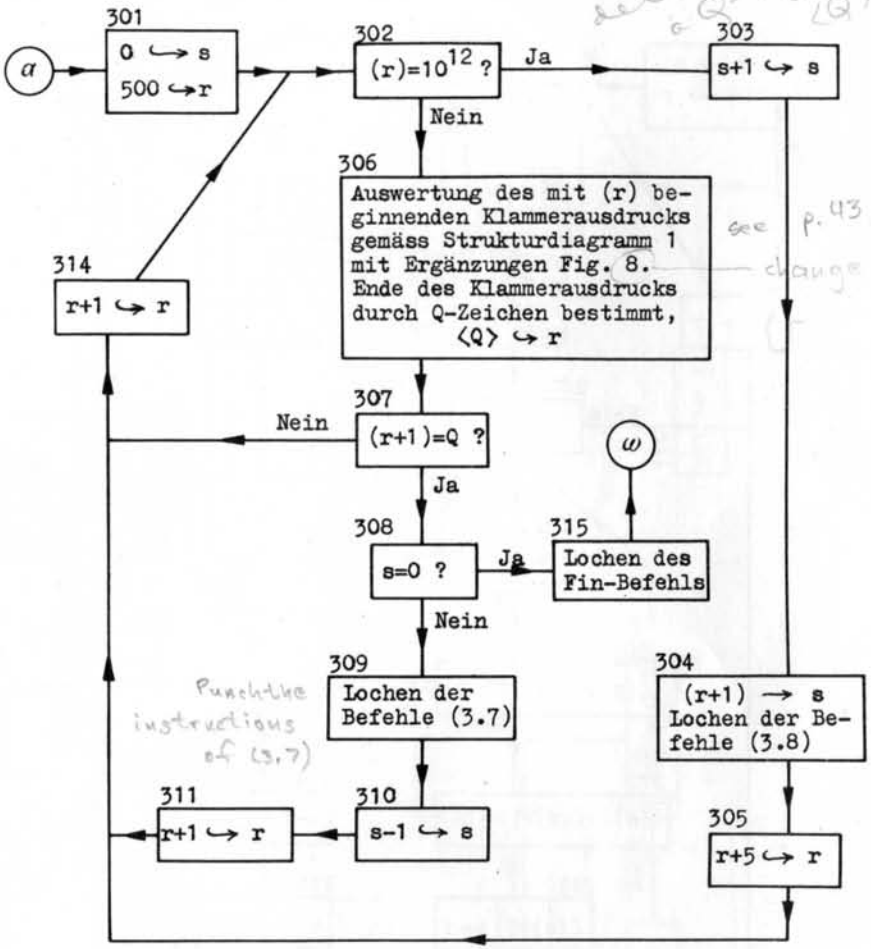
$\iota+1 \hookrightarrow \iota$: Der Index ι ist an dieser Stelle um 1 zu vergrössern, genauer: der frühere Wert $\iota+1$ ist fortan mit ι zu bezeichnen. Dies bewirkt automatisch die Umbenennung aller Grössen, die mit dem Index ι versehen sind, z.B. wird dadurch x_ι zu $x_{\iota-1}$

Strukturdiagramm 1





Strukturdiagramm 2



Strukturdiagramm 3

LITERATURVERZEICHNIS

[1] Aiken, H.H. Description of a Magnetic Drum Calculator (Harvard University Press, Cambridge, Mass. 1952), 318 S.

[2] Kilburn, T. The University of Manchester Universal High Speed Digital Computing Machine. Nature, Vol. 164 (1949) S. 184ff.

[3] Neumann, J.v. and Goldstine, H.H.
Planning and Coding of Problems for an Electronic Computing Instrument.
Institute for Advanced Study, Princeton N.J. 1947.

[4] Rutishauser H., Speiser A. und Stiefel E.
Programmgesteuerte Rechenmaschinen.
Mitteilungen Nr.2 aus dem Institut für angewandte Mathematik der ETH. Zürich 1951.

[5] Speiser, A. Entwurf eines elektronischen Rechengerätes.
Mitteilung Nr.1 aus dem Institut für angewandte Mathematik der ETH. Zürich 1950

[6] Wilkes, M.W., Wheeler, D.G., Gill, St.
The Preparation of Programs for an Electronic Digital Computer.
Addison Wesley Press, Cambridge, Mass. 1951

[7] Zuse, K. Ueber den allgemeinen Plankalkül als Mittel zur Formulierung schematisch kombinativer Aufgaben. Archiv der Mathematik, Vol.1 (1948/49), S.441-449.

[8] Kjellberg, G., Neovius, G.
The BARK, a Swedish General Purpose Relay Computer. MTAC, Vol 5. (1951), S.29-34.

[9] Rutishauser H. Automatische Rechenplanfertigung bei programmgesteuerten Rechenmaschinen. (Kurze Mitteilung). ZAMP, Vol.3 (1952), S.312-313.

Birkhäuser Verlag · Basel und Stuttgart

Mitteilungen aus dem Institut für angewandte Mathematik an der Eidgenössischen Technischen Hochschule Zürich

Herausgegeben von Prof. Dr. E. Stiefel

Nr. 1

Entwurf eines elektronischen Rechengerätes unter besonderer Berücksichtigung der Erfordernis eines minimalen Materialaufwandes bei gegebener mathematischer Leistungsfähigkeit. Von *Ambros P. Speiser* (2. Auflage 1954). 67 Seiten. Broschiert Fr. 7.– (DM 7.–).
Vorstudien für die Entwicklung eines Rechenautomaten mittlerer Grösse, angepasst an die Bedürfnisse eines mathematischen Instituts. Grundlage für die elektronische Rechenmaschine ERMETH der Eidgenössischen Technischen Hochschule Zürich. Der Entwurf sieht 1000 Elektronenröhren und 300 elektromagnetische Relais vor; als Speicher dient eine magnetische Trommel.

Nr. 2

Programmgesteuerte digitale Rechengeräte (elektronische Rechenmaschinen). Von *Heinz Rutishauser, Ambros Speiser* und *Eduard Stiefel* (1951). 102 Seiten. Broschiert Fr. 9.– (DM 9.–).
Inhalt: Grundlagen und wissenschaftliche Bedeutung – Organisation und Arbeitsweise – Arithmetische Prinzipien – Vorbereitung von Rechenplänen – Physikalische Grundlagen.

Nr. 3

Automatische Rechenplanfertigung bei programmgesteuerten Rechenmaschinen. Von *Heinz Rutishauser* (Nachdruck 1961.) 45 Seiten. Broschiert Fr. 8.– (DM 8.–).
Vorstudien für die Automation des Programmierens für elektronische Rechenautomaten.

Nr. 4

An Oscillation Theorem for Algebraic Eigenvalue Problems and its Applications. Von *Frank W. Sinden.* (1954.) 57 Seiten. Broschiert Fr. 7.– (DM 7.–).
In der Theorie der Eigenwertprobleme bei gewöhnlichen Differentialgleichungen (z. B. kritische Drehzahlen) spielen die sogenannten Oszillations-