Programmierung elektrotechnischer Probleme beim Einsatz von Digitalrechnern

Autor(en): **Dommel, H.**

Objekttyp: **Article**

Zeitschrift: **Bulletin des Schweizerischen Elektrotechnischen Vereins : gemeinsames Publikationsorgan des Schweizerischen Elektrotechnischen Vereins (SEV) und des Verbandes Schweizerischer Elektrizitätswerke (VSE)**

Band (Jahr): **54 (1963)**

Heft 25

PDF erstellt am: **07.07.2024**

Persistenter Link: https://doi.org/10.5169/seals-916543

BULLETIN

DES SCHWEIZERISCHEN ELEKTROTECHNISCHEN VEREINS

Gemeinsames Publikationsorgan des Schweizerischen Elektrotechnischen Vereins (SEV)
und des Verbandes Schweizerischer Elektrizitätswerke (VSE)

Programmierung elektrotechnischer Probleme beim Einsatz von Digitalrechnern

Von *H. Dommel*, München

681.14 – 523.8.518.5

Der Digitalrechner wird zunehmend zur Lösung elektrotechnischer Probleme eingesetzt. Sicher wird er eines Tages für den Ingenieur ein selbstverständliches Hilfsmittel bei grösseren Aufgaben sein, wenn Rechenschieber und Tischrechenmaschine nicht mehr ausreichen. Unbehagen gegenüber Digitalrechnern möchte der Verfasser als unbegründet entkräften und aufzeigen, wie einfach Programmieren ist.

Nach einem Überblick über Aufbau und Wirkungsweise des Digitalrechners und sein Befehlssystem wird anhand konkreter Beispiele das Vorgehen beim Programmieren gezeigt. Die Anfertigung eines Maschinenprogrammes aus einem Flussdiagramm wird kurz skizziert; die Programmierung in der Formelsprache ALGOL wird wegen ihrer wachsenden Bedeutung eingehender beschrieben. Am Problem der Kurzschlussberechnung soll abschliessend die praktische Anwendung gezeigt werden.

Bei Verwendung eines Digitalrechners liegt die schöpferische Arbeit in der präzisen Festlegung des Lösungswegs. Die anschliessende Formulierung eines Programmes ist dann nur noch eine einfache Aufgabe.

Le calculateur numérique est de plus en plus souvent utilisé pour résoudre des problèmes d'électrotechnique. Il deviendra certainement l'outil courant de l'ingénieur pour les calculs d'une certaine ampleur, pour lesquels la règle à calcul et la simple machine à calculer ne suffisent plus. L'auteur montre que les calculateurs numériques n'ont rien de très mystérieux et que la programmation n'est pas bien compliquée.

Après une brève description de la construction et du fonctionnement du calculateur numérique et de son système d'ordination, il donne quelques exemples concrets de programmation, puis décrit schématiquement la manière d'établir un programme de calculateur en partant d'un graphe de fluence. Il s'étend ensuite plus longuement sur le langage de programmation (langage algorithmique, ALGOL), dont l'importance ne cesse de s'accroître. Pour terminer, l'auteur indique, à titre d'application pratique, le problème du calcul du cas d'un court-circuit.

Lors de l'utilisation d'un calculateur numérique, le travail créateur consiste à déterminer d'une façon précise la voie de la solution. L'établissement d'un programme n'est ensuite qu'une tâche relativement simple.

1. Einleitung

Programmgesteuerte elektronische Rechenanlagen — auch Digitalrechner genannt — brachten einen umwälzenden Fortschritt für das Zahlenrechnen im Bereich der Wissenschaft, Technik und Wirtschaft. Überall, wo Daten in irgendeiner Form «verarbeitet», z. B. umgerechnet oder sortiert werden müssen, können Digitalrechner unvergleichlich schneller als der Mensch Routineaufgaben erledigen. In zunehmendem Masse werden Digitalrechner deshalb auch zur Lösung von elektrotechnischen Aufgaben verschiedenster Art herangezogen.

Zahlreiche Probleme der Elektrotechnik lassen sich mit genügender Genauigkeit analytisch lösen. Hier liegt ein sehr weites Anwendungsgebiet für den Digitalrechner. Während man früher einfache und rasch auswertbare «Faustformeln» gerne benutzt und dafür Ungenauigkeiten in Kauf genommen hat, besteht heute die Tendenz, bei Digitalrechnern weitgehend exakte Rechenverfahren zu verwenden. Einerseits fällt der Mehraufwand bei den hohen Rechengeschwindigkeiten meist kaum ins Gewicht und anderseits können dadurch Genauigkeitsabschätzungen entfallen, die bei Faustformeln immer etwas schwierig sind, vor allem wenn auf neuen Gebieten wenig Rechenerfahrungen vorliegen. Manche Probleme lassen sich überhaupt erst mit Hilfe eines Digitalrechners analytisch lösen. Hiezu zählen u. a. die Berechnungen des Betriebsverhaltens elektrischer Netze [1; 2][1]) eine Aufgabe, die vorher nur mittels Modellmessungen zu lösen war.

[1]) Siehe Literatur am Schluss des Aufsatzes.

Immer mehr wird der Digitalrechner auch für solche Aufgaben eingesetzt, wo sich durch Verändern von Parametern eine Vielzahl von möglichen Lösungen ergibt, aus denen dann nach bestimmten Gesichtspunkten nur die günstigste ausgesucht wird. Typische Beispiele dieses Aufgabentyps sind die Entwurfsberechnungen für elektrische Maschinen [3] und für Transformatoren [4]. Interessant ist in diesem Zusammenhang die Verwendung von Digitalrechnern für Netzplanungen [5], wo viele mögliche Varianten «durchgespielt» werden, um u. a. mit Methoden der Wahrscheinlichkeit Planungsunterlagen zu gewinnen.

In Zukunft wird man den Digitalrechner vermutlich immer mehr auch zur Steuerung technisch komplizierter Vorgänge einsetzen. So wird zur Zeit die Frage diskutiert, ob die Lastverteilung in einem Netz von einem Digitalrechner automatisch nach wirtschaftlichen Gesichtspunkten gesteuert werden soll [6; 7]. Auch die zentrale Überwachung und Steuerung grosser Dampfkraftwerke kann eine «Prozess-Rechenanlage» übernehmen [8]. Bei allen Steuerungsaufgaben müssen Messwerte direkt der Rechenanlage zugeführt werden, die dann auf Grund von daraus berechneten Resultaten ihrerseits wieder die Auslösung von Regel- und Warnimpulsen veranlasst.

Auf allen Gebieten nimmt der Digitalrechner dem Ingenieur lästige Routinearbeit ab und macht ihn für schöpferische Aufgaben frei. Es ist deshalb lohnenswert, die Möglichkeiten dieses neuen, leistungsfähigen Hilfsmittels richtig zu erkennen. «Da der Digitalrechner nun eine Hilfskraft ist, die überhaupt keine Einsicht in das hat, was sie tut, muss man ihr eine Arbeitsvor-

schrift geben, die bis ins kleinste Detail hinein den Ablauf der durchzuführenden Arbeiten eindeutig festlegt» [10]. Wie eine solche Arbeitsvorschrift in ein «Programm» gekleidet wird, versucht dieser Aufsatz im folgenden an Hand einfacher Beispiele zu schildern, ohne dabei auf spezielle Digitalrechner einzugehen.

2. Die Lösung eines technisch-wissenschaftlichen Problems

Bei der Behandlung eines technisch-wissenschaftlichen Problems gelangt man schematisch über 4 Stufen zur Lösung:

1. Idealisierung des Problems durch Schaffung einer Modellvorstellung;
2. Beschreibung des Modells durch einen mathematischen Ansatz;
3. Auswahl eines konstruktiven Lösungsverfahrens;
4. Wertmässige Lösung, und zwar entweder

a) ziffernmässig (von Hand, Tischrechenmaschine, Digitalrechner);
b) analog (Rechenschieber, Netzmodell) oder
c) graphisch.

Handelt es sich beispielsweise um die Berechnung der Kurzschlussströme in einem Netz, so könnte man in der 1. Stufe das Netz durch eine für Kurzschlussberechnungen zulässige Ersatzschaltung darstellen, in der 2. Stufe hiefür die Knotenpunktgleichungen anschreiben, in der 3. Stufe als Lösungsverfahren die Eliminationsmethode wählen und in der 4. Stufe mit einer Tischrechenmaschine die Werte ziffernmässig ermitteln. Es ist klar, dass eine Modellvorstellung die physikalische Wirklichkeit nur für das spezielle Problem richtig wiedergibt. So ist bei einer Freileitung das «Modell» eines Ohmschen Widerstandes zwar brauchbar für die Berechnung der stationären Ströme und Spannungen bei Gleichstrom, aber unbrauchbar für die Berechnung elektromagnetischer Ausgleichsvorgänge oder gar für die Berechnung des Durchhangs.

Während die 1. Stufe eine rein ingenieurmässige Aufgabe ist, wird in der 3. Stufe oft die Hilfe eines Mathematikers oder eines mathematisch versierten Ingenieurs notwendig sein. Als Beispiel für die Wahl eines Lösungsverfahrens sei die Auflösung von n linearen Gleichungen mit n Unbekannten betrachtet, eine Aufgabe, die sich bei fast allen Netzberechnungen stellt. Tabelle I zeigt abgerundet die von *R. Sauer* [9] angegebenen Rechenzeiten für einen Digitalrechner, einmal bei Verwendung des Eliminationsverfahrens nach *Gauss* [11], zum anderen bei Verwendung der bekannten Cramerschen Determinanten-Regel. Die mit steigender Zahl n der Unbekannten hoffnungslos wachsenden Rechenzeiten bei der Cramer-Regel zeigen drastisch, welche Sorgfalt bei der Wahl eines Lösungsverfahrens notwendig ist. Schon einfache Probleme erfordern oft sehr gründliche mathematische Kenntnisse.

Ungefähre Rechenzeiten bei 200 Operationen/s zur Auflösung von n linearen Gleichungen mit n Unbekannten [9]

Tabelle I

Anzahl n der Unbekannten	Elimination nach Gauss	Cramersche Determinanten-Regel
5	1,5 s	1 min
10	10 s	85 Tage
20	1 min	10^{11} Jahre

Es sei ausdrücklich darauf hingewiesen, dass der Digitalrechner ausschliesslich auf der letzten und 4. Stufe seine Dienste als Hilfskraft anbieten kann.

3. Der Digitalrechner

Aufbau und Wirkungsweise eines Digitalrechners seien, ohne auf Einzelheiten einzugehen, an Hand von Fig. 1 erläutert. Der Digitalrechner besteht im wesentlichen aus:

1. Rechenwerk, das der Tischrechenmaschine eines Rechners vergleichbar ist und wie diese verschiedene Register zur Ausführung der 4 Grundrechnungsarten besitzt;
2. Speicher, vergleichbar den Papierblättern eines Rechners, auf denen er Rechenablauf und Zwischenergebnisse notiert;
3. Steuerwerk, das die steuernde Tätigkeit des Rechners übernimmt und
4. u. 5. Ein-und Ausgabegeräten, die eine Verbindung des Digitalrechners mit der Aussenwelt herstellen.

Der Speicher kann entweder Zahlen (Ausgangszahlen, Zwischenergebnisse) oder auch Befehle aufnehmen; der Inhalt einer Speicherzelle wird «Wort» genannt. Ein Wort kann also ein Befehl oder eine Zahl sein. Beträgt die «Wortlänge» des Digitalrechners z. B. 10 Dezimalstellen, so wird die Rechnung grundsätzlich 10-stellig ausgeführt; die Befehle wären dann als 10-stellige Zahlen verschlüsselt[2]). Beim Rechnen holt sich das Steuerwerk fortlaufend Befehl für Befehl (Wort für Wort) aus den Zellen des Speichers, entschlüsselt sie und führt die entsprechenden Operationen im Rechenwerk aus. Die Leistungsfähigkeit eines Digitalrechners wird wesentlich durch Art und Grösse des Speichers bestimmt. Je weniger Zeit zum Aufsuchen eines Wortes im Speicher notwendig ist («Zugriffszeit»), desto schneller läuft die Rechnung ab. Die Zugriffszeit bei rotierenden Trommelspeichern mit magnetischer Oberfläche beträgt 5...20 ms, bei Ferritkern-Speichern 5...15 μs und bei den neuerdings verwendeten magnetischen Dünnschicht-Speichern bis herab zu 100 ns. Die Kapazität von Trommelspeichern liegt bei 4000...30000 Worten, von Magnetkernspeichern bei 2000...60000 Worten.

Eingabegeräte sind notwendig, um die Befehlsliste und die Zahlen, mit denen gerechnet werden soll, dem Digitalrechner mitzuteilen; Ausgabegeräte braucht man zum Ausdruck der Ergebnisse. Als Eingabegeräte werden hauptsächlich Magnetbandeinheiten, Lochstreifen- und Lochkartenleser sowie Fernschreiber verwendet, als Ausgabegeräte Magnetbandeinheiten, Lochstreifen- und Lochkartenstanzer, Schnelldrucker, Analogsichtgeräte und Fernschreiber.

Einen groben Überblick über die mittleren Rechengeschwindigkeiten[3]) einiger kommerzieller Digitalrechner gibt Tabelle II.

32'472-482

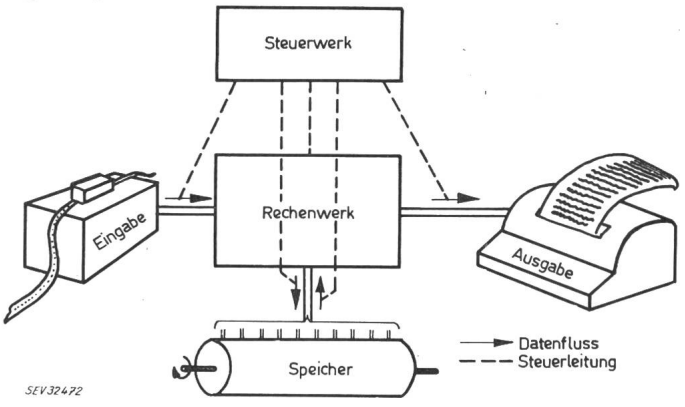
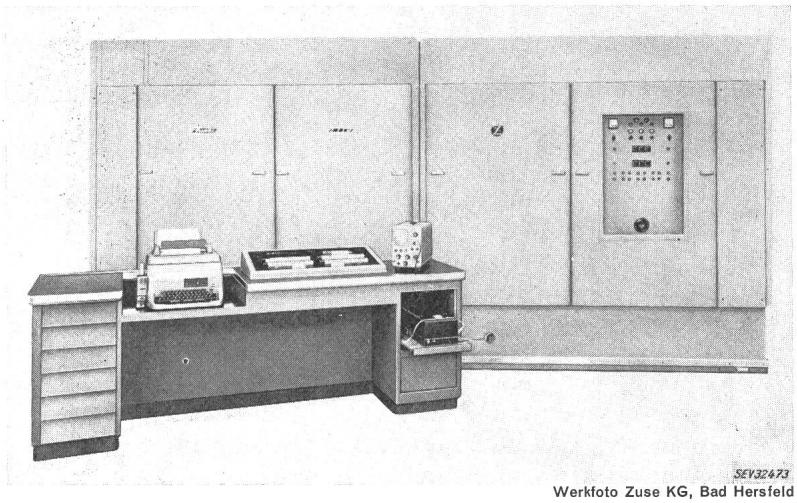


Fig. 1
Funktionsschema eines Digitalrechners

[2]) Von Digitalrechnern mit variabler Wortlänge sei hier abgesehen.
[3]) Die mittlere Rechengeschwindigkeit gilt für ein gedachtes Programm, das 25 % Additionen, 25 % Multiplikationen und 50 % organisatorische Befehle (z. B. Lesen aus Speicher) enthält.



SEV32473

Werkfoto Zuse KG, Bad Hersfeld

Fig. 2
Digitalrechner Z 22

Fig. 2 zeigt einen Digitalrechner, Fig. 3 die Ausgabe der Ergebniskurven auf einem Analogsichtgerät.

Mittlere Operationsgeschwindigkeiten von Digitalrechnern [12]

Tabelle II

Rechenmaschine	Mittlere Operationsgeschwindigkeiten Operation/s
Tischrechenmaschine	0,2
Z 22 (Zuse KG)	20
IBM 650	200
S-2002 (Siemens)	2 000
IBM 704	10 000
IBM 7090	50 000
TR4 (Telefunken)	70 000
Larc (Remington Rand)	100 000
Stretch (IBM)	1 000 000

Die wesentlichen Vorteile eines Digitalrechners sind: hohe Rechengeschwindigkeit, die jedoch nur sinnvoll ist bei automatischem Rechenablauf. Ausserdem hat der Digitalrechner hohe Stellengenauigkeit und wesentlich geringere Fehlerhäufigkeit als der Mensch. Er ist ausserdem relativ billig und sofort einsatzfähig.

4. Das Befehlssystem eines Digitalrechners

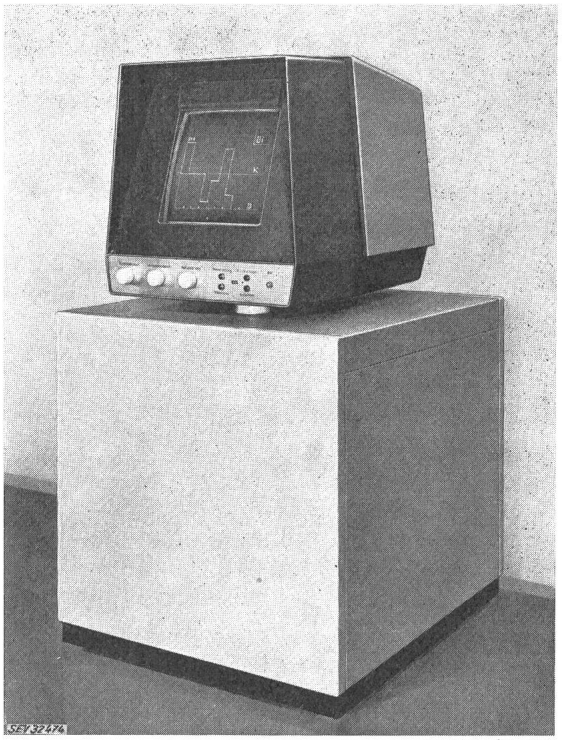
Die volle Wirksamkeit des Digitalrechners liegt in seiner Fähigkeit, mit Befehlen rechnen zu können, die er sich Wort für Wort aus dem Speicher holt und ausführt. Dadurch wird ein automatischer Rechenablauf von der Dateneingabe über die Rechnung bis zur Ergebnisausgabe möglich. Die Gesamtheit aller Befehle, die ein Digitalrechner ausführen kann, nennt man sein Befehlssystem; dieses Repertoire bildet seine «Intelligenz.»

Obwohl die Befehlssysteme der verschiedenen Digitalrechner mehr oder weniger voneinander abweichen, sind doch gewisse Ähnlichkeiten vorhanden. Tabelle III zeigt typische Befehle, wie sie fast jeder Digitalrechner ausführen kann[4]). Zur Ausführung der Befehle dienen die verschiedenen Register des Rechenwerks. Für die 4 Grundrechnungsarten werden gewöhnlich 2 Register benötigt, Multiplikandenregister und Akkumulatorregister. Sehr vorteilhaft beim Programmieren sind Indexregister, die ganze Zahlen aufnehmen können und zum Zählen und Adressenändern verwendet werden. Die Befehlsdarstellung in Tabelle III wird «memotechnischer Code» oder «Externcode» genannt. Bei der Eingabe der Befehlsliste in den Speicher werden die im memotechnischen Code geschriebenen Befehle (Externcode) automatisch in Zahlen verschlüsselt, mit denen dann der Digitalrechner intern arbeitet (Interncode). Ein mit Lochstreifen oder Lochkarten eingegebener Befehl «ADD 123» könnte bei einer Wortlänge von 10 Dezimalen intern etwa als 0001110123 verschlüsselt sein. Der memotechnische Code dient nur der leichteren Verständlichkeit beim Niederschreiben des Programms.

Die Befehle von Tabelle III haben einen Operationsteil, der die auszuführende Operation festlegt, und einen Adressteil, der meist die Zellennummer angibt, in der eine an der Operation beteiligte Zahl zu finden ist. Man kann die Befehle von Tabelle III einteilen in:

1. Arithmetische Befehle (1...4) für die Durchführung der Grundrechnungsarten;
2. Organisatorische Befehle, und zwar
a) Transportbefehle (5 und 6) für den Verkehr zwischen Speicher und Rechenwerk,
b) Regiebefehle (7...9) für den Verkehr mit der Aussenwelt,
c) Indexbefehle (10 und 11) und
d) Sprungbefehle (12...14)

Bei den Befehlen 7...10 ist der Adressteil bedeutungslos, beim Befehl 11 ist der Adressteil der Operand selbst. Bei Ausführung der Befehle 6 und 9...14 bleibt der Inhalt des Akkumulatorregisters unverändert. Zu Beginn der Rechnung wird dem Digitalrechner durch Einstellen des Befehlszählregisters die Nummer derjenigen Speicherzelle mitgeteilt, aus der er den 1. Befehl zu holen hat. Danach sorgt dieses Befehlszählregister



SEV32474

Werkfoto Siemens

Fig. 3
Analogsichtgerät

[4]) Auf die Wiedergabe des Befehlssystems eines speziellen Digitalrechners wird aus Gründen der Anschaulichkeit bewusst verzichtet. Bei Verwendung von Formelsprachen, z. B. ALGOL, braucht der Programmierer ohnehin das Befehlssystem eines speziellen Digitalrechners nicht mehr zu kennen.

Tabelle III

Nr.	Befehl		Wirkung	Ergebnis im
	Operationsteil	Adressteil		
1	ADD	*a*	ADDiere Inhalt der Zelle *a* zum Inhalt des Akkumulatorregisters	Akkumulatorregister
2	SUB	*a*	SUBtrahiere Inhalt der Zelle *a* vom Inhalt des Akkumulatorregisters	
3	MULT	*a*	MULTipliziere Inhalt der Zelle *a* mit Inhalt des Akkumulatorregisters	
4	DIV	*a*	DIVidiere durch Inhalt der Zelle *a* den Inhalt des Akkumulatorregisters	
5	BRING	*a*	BRINGe Inhalt der Zelle *a* ins Akkumulatorregister	Akkumulatorregister und Zelle *a*
6	SPEICHER	*a*	SPEICHERe Inhalt des Akkumulatorregisters nach Zelle *a*	
7	LESE	—	LESE 1 Wort über Eingabegerät ins Akkumulatorregister	Akkumulatorregister
8	DRUCK	—	DRUCKe Inhalt des Akkumulatorregisters am Ausgabegerät	
9	STOP	—	führe keinen weiteren Befehl mehr aus	
10	ISETZ	—	besetze Indexregister mit dem (auf eine ganze Zahl gerundeten) Inhalt des Akkumulatorregisters	Indexregister
11	ISUB	*a*	vermindere Inhalt des Indexregisters um den Wert *a*	
12	SPRUNG	*a*	setze Rechnung mit Befehl aus Zelle *a* fort	
13	ISPRUNG	*a*	setze Rechnung dann mit Befehl aus Zelle *a* fort, wenn Inhalt des Indexregisters negativ ist (ansonsten mit nächstem Befehl)	
14	ASPRUNG	*a*	setze Rechnung dann mit Befehl aus Zelle *a* fort, wenn Inhalt des Akkumulatorregisters negativ ist (ansonsten mit nächstem Befehl)	

dafür, dass die Rechnung mit den im Speicher unmittelbar folgenden Befehlen solange fortgesetzt wird, bis ein STOP-Befehl erscheint. Durch die Sprungbefehle 12...14 kann von dieser normalen, sukzessiven Folge in der Befehlsausführung abgewichen werden. Die «bedingten» Sprungbefehle 13 und 14 bewirken eine Abweichung von der normalen Folge nur dann, wenn eine bestimmte Bedingung erfüllt ist. Hiedurch kann der Digitalrechner logische Entscheidungen selbst treffen. Das Befehlsrepertoire kommerzieller Digitalrechner reicht von etwa 20 Befehlen bis über 1000 Befehle bei sog. Mikrobefehlssystemen.

Viele Digitalrechner lassen bei den Befehlen 1...6 und 12...14 Änderungen des Adressteils mit Hilfe des Indexregisters zu. Eine «Adressenmodifikation», mit der die Adresse um den Inhalt des Indexregisters erhöht wird, möge für die Befehle von Tabelle III durch Anfügen von «I» an den Adressteil gekennzeichnet sein. Der Befehl «ADD 123I» würde dann folgendermassen interpretiert, wenn bei Ausführung des Befehls gerade die Zahl 12 im Indexregister steht: «Addiere Inhalt der Zelle 135 (=123+12) zum Inhalt des Akkumulatorregisters».

Die hier beschriebenen Befehle sind die einer Einadress-Maschine. Es gibt auch Zweiadress-Maschinen, bei der ein Additionsbefehl etwa lauten würde «ADD 123/129» (ADDiere zum Inhalt der Zelle 123 den Inhalt der Zelle 129 mit Ergebnis im Akkumulatorregister).

Für die arithmetischen Befehle ist die interne Darstellung der Zahlen wichtig[5]). Man unterscheidet Festkomma-Darstellung und Gleitkomma-Darstellung. Die meisten Digitalrechner können wahlweise in Festkomma- oder in Gleitkomma-Arithmetik rechnen. Fig. 4 zeigt eine Darstellung von Festkommazahlen, Gleitkommazahlen und Befehlen bei einer Wortlänge von 10 Dezimalen. Die Vorzeichen sind dabei in Wirklichkeit als Zahl verschlüsselt (z. B. 2 für — und 1 für +). Bei Festkommazahlen ist das Komma unmittelbar nach dem Vorzeichen zu denken. Es dürfen deshalb bei Festkomma-Arithmetik niemals Zahlen auftreten, die dem Betrage nach grösser oder gleich 1 sind, was sich durch geeignete Masstabsfaktoren für die Rechnung erreichen lässt. Wesentlich flexibler ist die Gleitkomma-Darstellung. Hiebei wird ein Teil des Wortes als Exponent benützt in einer halblogarithmischen Darstellung $m \cdot 10^b$. Die Ergebnisse im Rechenwerk sind meist normalisierte Gleitkommazahlen, d. h. die unmittelbar nach dem Vorzeichen und dem gedachten Komma stehende Ziffer ist von Null verschieden. Beispiele für normalisierte Gleitkommazahlen in Darstellung von Fig. 4 sind

$$+\ 328561 + 03 = +\ 328{,}561$$
$$-\ 100210 + 00 = -\ \ 0{,}10021$$
$$+\ 217563 - 02 = +\ \ 0{,}00217563.$$

Eine nicht normalisierte Gleitkommazahl wäre + 032856 + 04 = + 328,56.

5. Die Programmierung

Die schöpferische, mathematische Aufgabe bei der Lösung eines Problems liegt in der Auswahl oder der Neuentwicklung geeigneter Lösungsverfahren. Der Rechenablauf muss hiebei

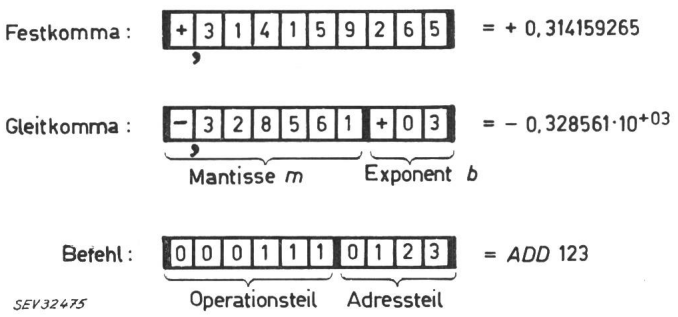


Fig. 4
Beispiel für interne Zahlen- und Befehlsdarstellung

[5]) Es sei angenommen, dass der Digitalrechner intern im Dezimalsystem rechnet. In Wirklichkeit arbeiten viele Digitalrechner intern im Dualsystem.

mit Sorgfalt in allen Einzelheiten festgelegt sein, bevor ein «Programm» geschrieben werden kann. Fehlerabschätzungen und Schätzungen des Rechenaufwands sind unerlässlich, wenn bei umfangreichen Problemen der Einfluss der Rundungsfehler und die erforderlichen Rechenzeiten grössere Bedeutung gewinnen.

Hat man sich für ein bestimmtes Lösungsverfahren entschieden, so muss die numerische Formulierung aufgegliedert werden in eine Folge arithmetischer und logischer Operationen. Eine solche Folge nennt man «Rechenplan» oder «Algorithmus».

Als *Beispiel A* [10] sei die Auflösung von 2 Gleichungen mit 2 Unbekannten betrachtet,

$$a_{11}\,x_1 + a_{12}\,x_2 = y_1$$
$$a_{21}\,x_1 + a_{22}\,x_2 = y_2$$

In der üblichen Sprache und mit den herkömmlichen arithmetischen Formeln liesse sich der Rechenplan für die Berechnung von x_1 und x_2 etwa folgendermassen ausdrücken:

«Berechne aus den Zahlen a_{11}, a_{12}, a_{21}, a_{22}, y_1 und y_2 den Wert $d = a_{11}\,a_{22} - a_{12}\,a_{21}$; wenn $d \neq 0$ ist, berechne

$$x_1 = (y_1\,a_{22} - y_2\,a_{12})/d\,;$$
$$x_2 = (y_2\,a_{11} - y_1\,a_{21})/d\,;$$

ansonsten zeige an, dass die Determinante Null ist.»

Würde der Fall $d = 0$ nicht gesondert behandelt, so ergäbe sich theoretisch $x_1 = x_2 = \infty$. Jeder Digitalrechner kann aber nur Zahlen bis zu einer bestimmten Grösse (z. B. 10^{50}) darstellen; bei Überschreitung dieses Betrages bleibt der Digitalrechner mit Fehleranzeige stehen. Der automatische Rechenablauf wäre dadurch gestört.

Formeln dürfen in einem Rechenplan nicht als Gleichungen sondern als Wertzuweisungen verstanden werden. So bedeutet beispielsweise die obige Formel $d = a_{11}\,a_{22} - a_{12}\,a_{21}$: «Setze die Zahlenwerte für a_{11}, a_{12}, a_{21}, a_{22} in die Formel ein und weise den aus der numerischen Auswertung resultierenden Zahlenwert der Variablen d zu». Um zu betonen, dass es sich bei Formeln in Rechenplänen nicht um Gleichungen handelt, schreibt man Wertzuweisungen in der Form:

$$a_{11} \times a_{22} - a_{12} \times a_{21} \Longrightarrow d$$

oder

$$d := a_{11} \times a_{22} - a_{12} \times a_{21}$$

Bei Verwendung von Digitalrechnern entspricht einer Wertzuweisung das Abspeichern eines Zahlenwertes in diejenige Zelle, die für die betreffende Variable reserviert ist. Der Unterschied zwischen Gleichung und Wertzuweisung wird deutlich an dem Beispiel $x := x + 1$. Als Gleichung wäre diese Formel sinnwidrig; als Wertzuweisung bedeutet sie, dass zum Zahlenwert für x die Zahl 1 zu addieren ist und das Ergebnis wiederum der Variablen x zugeordnet wird.

5.1 Das Flussdiagramm

Einen anschaulichen Überblick über die Folge der arithmetischen Operationen und logischen Entscheidungen vermittelt die graphische Darstellung des Rechenplans. Ein solches graphisches Schema wird als «Flussdiagramm» bezeichnet. Die arithmetischen Operationen sowie Ein- und Ausgabeoperationen schreibt man hiebei z. B. in rechteckige Kästchen, logische Entscheidungen in Kreise. Pfeile geben die Richtung des Rechenablaufs an.

In Fig. 5 ist das Flussdiagramm für Beispiel *A* dargestellt. Es zeigt erstens, dass an bestimmten Stellen des Rechenplans

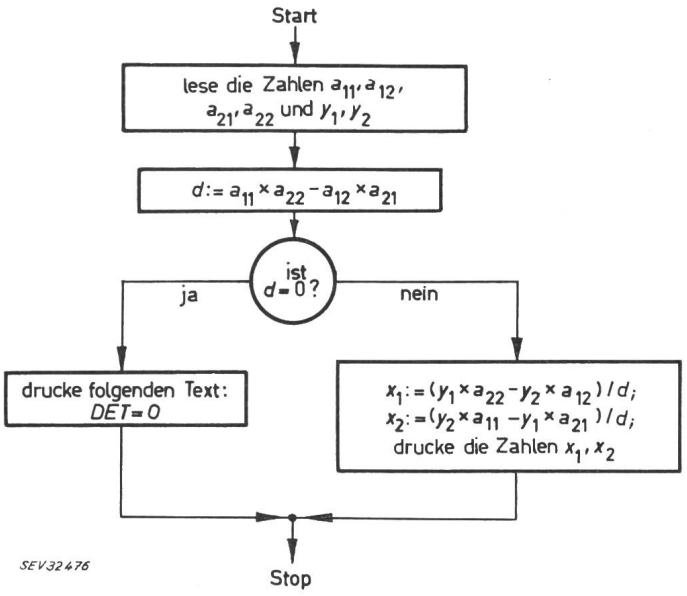


Fig. 5
Flussdiagramm für Beispiel *A*

logische Entscheidungen zu treffen sind, mit denen die Richtung des weiteren Rechenablaufs festgelegt wird. Zweitens lässt es erkennen, dass Rechenpläne so allgemein gehalten werden können, dass man damit die Lösung für beliebige Parameter bekommt. Rechenpläne (und daraus entwickelte Programme) sollen grundsätzlich so beschaffen sein, dass sie allgemeingültig sind. Ein guter Rechenplan zur Auflösung linearer Gleichungen würde deshalb nicht nur für 2 Gleichungen mit 2 Unbekannten geeignet sein, sondern im allgemeinen für n Gleichungen mit n Unbekannten; der Wert n würde dann lediglich durch die Speicherkapazität des Digitalrechners begrenzt.

Im Flussdiagramm von Fig. 5 wird jeder Teil (jedes Kästchen) gerade einmal durchlaufen. Solche «Geradeaus»-Programme sind nur für kleine Probleme möglich; bei grösseren Problemen würde die dazu notwendige Folge von Operationen sehr bald zu lang. Für umfangreiche Probleme wird das Programm wesentlich dadurch verkürzt, dass man weitgehend induktive und iterative Prozesse verwendet; der Rechenplan enthält dann Operationsfolgen, die schleifenartig mehrmals durchlaufen werden. Solche «Schleifen» sind im folgenden an Hand konkreter Beispiele erklärt.

Als *Beispiel B* sei die Berechnung der Produktsumme

$$y = \sum_{i=1}^{n} a_i\,b_i$$

mit Hilfe einer «Induktionsschleife» betrachtet. Der zweckmässige Rechenplan liesse sich in Worten wie folgt ausdrücken:

«Setze $y := 0$;
rechne für $i = 1, 2, 3, \ldots, n$ jeweils $y := a_i\,b_i + y$»

Zunächst wird der Variablen y der Wert 0 zugeordnet, dann bei $i = 1$ der Wert $a_1\,b_1$, bei $i = 2$ der Wert $a_1\,b_1 + a_2\,b_2$ usw., bis schliesslich bei $i = n$ die Variable y den Wert

$$\sum_{i=1}^{n} a_i\,b_i$$

angenommen hat. Fig. 6 zeigt das zugehörige Flussdiagramm. Dabei sollen die Zahlen a_i und b_i erst während des Rechenprozesses eingelesen werden. Die Induktionsschleife wird n-mal ($n \geq 0$) durchlaufen und erst bei $i = n$ verlassen. Mit wach-

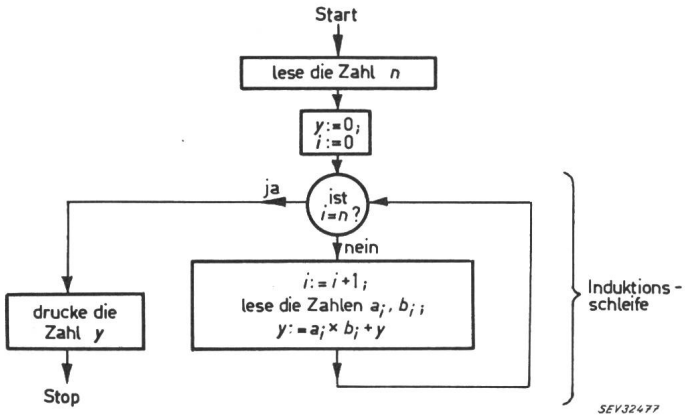


Fig. 6
Flussdiagramm für Beispiel *B*

sendem Paramter n nimmt auch die Folge der auszuführenden Operationen zu. Dadurch, dass gleichartige Rechnungen in einer Schleife «dynamisch» öfters durchlaufen werden, bleibt die «statische» Länge des Rechenplans jedoch gleich gross für beliebige n.

Die Verwendung einer «Iterationsschleife» sei am *Beispiel C* gezeigt: Am Ende einer Gleichspannungsleitung mit dem Widerstand R werde nach Fig. 7 die Leistung P_2 entnommen mit konstanter Spannung U_1 am Anfang der Leitung; gesucht seien der Strom I und die Spannung U_2 am Ende der Leitung. Es gelten die beiden Gleichungen:

$$U_2 = U_1 - IR \tag{1}$$

und

$$I = \frac{P_2}{U_2}$$

Setzt man Gl. (2) in Gl. (1) ein, so ergibt sich eine quadratische Gleichung für U_2. Dieses direkte Lösungsverfahren soll hier jedoch nicht verwendet werden sondern ein Iterationsprozess, der schrittweise zur Lösung führt und sich folgendermassen formulieren lässt:

«1. Schätze die Spannung U_2 auf $U_2 = U_1$;
2. Berechne damit I aus Gl. (2) und danach U_2 aus Gl. (1);
3. Wenn berechneter und geschätzter Wert um mehr als die gewünschte Genauigkeit ε abweichen, dann nehme den berechneten Wert U_2 als neuen Schätzwert und wiederhole 2. und 3.; ansonsten drucke I und U_2.»

Fig. 8 zeigt das zugehörige Flussdiagramm. Wie oft die Iterationsschleife bis zum Erreichen der gewünschten Genauigkeit durchlaufen wird, ist vorher nicht zu sagen. Die Anzahl

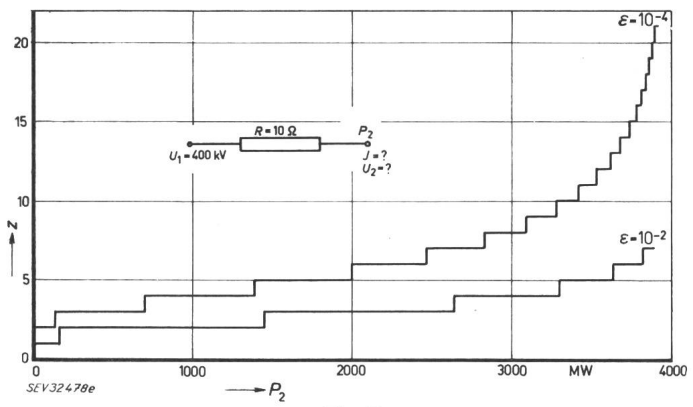


Fig. 7
Anzahl der Iterationsschritte z als Funktion von P_2
P_2 Leistung am Ende; ε relative Genauigkeit; U_1 Spannung am Anfang; U_2 Spannung am Ende; J Strom; R Ohmscher Widerstand; z Anzahl der Iterationsschritte

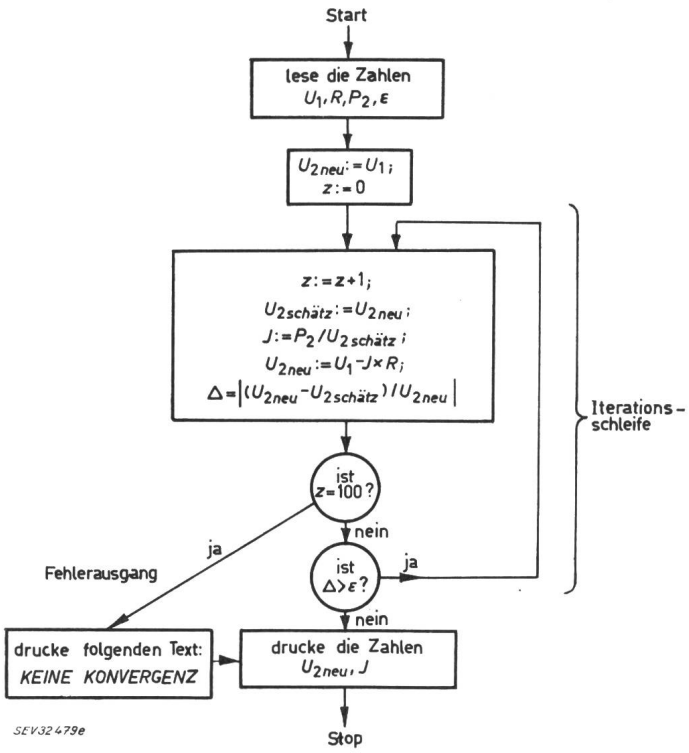


Fig. 8
Flussdiagramm für Beispiel *C*

der Iterationsschritte hängt ab von der Leistung P_2 und der gewünschten Genauigkeit ε. In Fig. 7 ist diese Abhängigkeit eingetragen für $R = 10\,\Omega$ und $U_1 = 400$ kV. Da bei Iterationsprozessen die Anzahl der Iterationsschritte oft schwer vorauszusagen ist und grundsätzlich der Fall eintreten kann, dass das Verfahren überhaupt nicht zur Lösung führt, empfiehlt sich immer eine Zählung der Iterationsschritte mit Beendigung des Rechenprozesses bei Erreichen einer festgelegten, maximalen Schrittzahl («Fehlerausgang» in Fig. 8). Im Beispiel C mit den Werten von Fig. 7 existiert für $P_2 > 4000$ MW keine Lösung mehr, die Lösung würde also hiefür nie erreicht. Wäre die Anzahl der durchlaufenen Schleifen dabei nicht nach oben durch eine maximale Schrittzahl ($= 100$ in Fig. 8) begrenzt, so würde der Digitalrechner theoretisch unendlich lange weiterrechnen.

Für den mit Digitalrechnern weniger vertrauten Leser mag die Verwendung eines Iterationsprozesses in Beispiel C überraschend erscheinen, da sich U_2 direkt berechnen lässt aus:

$$U_2 = \frac{U_1}{2} + \sqrt{\left(\frac{U_1}{2}\right)^2 - R\,P_2} \tag{3}$$

Man darf aber nicht vergessen, dass die Berechnung einer Quadratwurzel auf die 4 Grundrechnungsarten zurückgeführt werden muss. Meist verwendet man hiezu eine Tschebyscheffsche Polynomapproximation, deren Befehlsfolge dann im wesentlichen eine Induktionsschleife ist. Die direkte Lösung kann dadurch aufwendiger werden als das Iterationsverfahren. Für die direkte Lösung eines Falles von Beispiel C ergab sich an der PERM [6]) eine Rechenzeit von 14 ms, für 1 Schritt des Iterationsprozesses von 4 ms. Mit den Daten und Ergebnissen von Fig. 7 folgt daraus, dass für eine relative Genauigkeit von $\varepsilon = 10^{-2}$ der Iterationsprozess bis zu $P_2 = 2640$ MW schneller verläuft als das direkte Lösungsverfahren. Viele Probleme lassen sich überhaupt nicht direkt sondern nur mit Iterationsprozessen lösen.

[6]) Programmgesteuerte Elektronische Rechenanlage München.

Die Erstellung eines Rechenplans — z. B. in Form eines Flussdiagramms — hat unmittelbar mit dem Einsatz des Digitalrechners noch wenig zu tun. Ein solcher Rechenplan ist grundsätzlich auch bei Handrechnungen notwendig; er wird dabei nur selten vorher explizit ausgearbeitet vorliegen. Vielmehr entscheidet man meist erst im Laufe der Rechenarbeiten den weiteren Weg, falls eine Entscheidung notwendig wird. Anders ist die Situation bei Verwendung eines Digitalrechners, wo vorher schon alle Einzelheiten des Rechenablaufs genau festgelegt werden müssen. Hierin liegt die schöpferische Arbeit beim Einsatz von Digitalrechnern, und nicht etwa in der Niederschrift eines für den Digitalrechner verständlichen Programms.

5.2 Das Maschinenprogramm

Der Digitalrechner versteht nur die Befehle seines vereinbarten Befehlssystems. Er braucht deshalb eine Befehlsliste, wenn er die im Rechenplan festgelegten Operationen automatisch ausführen soll. Eine solche Befehlsliste heisst Maschinenprogramm. Die Art der Niederschrift muss dabei so sein, dass das Programm dem Digitalrechner eingegeben werden kann, also z. B. auf Lochstreifen oder Lochkarten.

Für das Beispiel *B* ist im folgenden nach dem Flussdiagramm von Fig. 6 das Maschinenprogramm aufgestellt unter Verwendung der in Tabelle III angegebenen Befehle:

Speicherzelle-Nr.	Befehl Operations-Teil	Adressteil
100:	LESE	
101:	ISETZ	
102:	BRING	115
103:	SPEICHER	116
104:	ISUB	1
105:	ISPRUNG	113
106:	LESE	
107:	SPEICHER	117
108:	LESE	
109:	MULT	117
110:	ADD	116
111:	SPEICHER	116
112:	SPRUNG	104
113:	DRUCK	
114:	STOP	
115:	+ 000000 + 00	
116:		
117:		

Ist dieses Maschinenprogramm einmal in die Zellen 100 bis 117 des Speichers eingelesen, so kann damit die Berechnung der Produktsumme

$$y = \sum_{i=1}^{n} a_i b_i$$

beliebig oft wiederholt werden. Es ist nur dafür zu sorgen, dass vorher immer die Zahlen am Eingabemedium (z. B. auf Lochstreifen) in folgender Reihenfolge bereitliegen:

$$n, a_1, b_1, a_2, b_2, \ldots a_n, b_n$$

5.2.1 *Erläuterung des Rechenablaufs*

1. Dem Digitalrechner wird mitgeteilt, dass das Programm ab Zelle 100 im Speicher steht durch Einstellen des Befehlszählregisters auf 100.

2. Nach Drücken der Starttaste holt das Steuerwerk den 1. Befehl «LESE» aus Zelle 100 und führt ihn aus. Es wird also die erste Zahl (= n) vom Lochstreifen gelesen und ins Akkumulatorregister — kurz AR genannt — gebracht. Hierauf holt das Steuerwerk den nächsten Befehl 101; seine Ausführung bewirkt, dass die Zahl n aus dem AR ins Indexregister gebracht wird.

3. Befehl 102 bringt den Wert 0 aus Zelle 115 ins AR, der anschliessend mit Befehl 103 nach Zelle 116 gespeichert wird. In Zelle 116 soll der jeweilige Wert von y stehen. Befehl 102 und 103 bewirken also die Anweisung $y := 0$.

4. Durch Befehl 104 wird der Inhalt des Indexregisters um 1 verringert. Hier beginnt die Induktionsschleife, die n-mal zu durchlaufen ist. Vom Wert n-1 ausgehend wird dadurch im Indexregister abwärts gezählt, bis beim n-ten Durchlauf der Wert —1 im Indexregister steht. Erst dann bewirkt Befehl 105, dass von der normalen Befehlsfolge abgewichen wird und die Rechnung auf Befehl 113 springt, wo zunächst das Ergebnis gedruckt und mit Befehl 114 die Maschine angehalten wird. Ansonsten führen

5. die Befehle 106 bis 111 die Anweisung $y := a_i \times b_i + y$ aus. Durch Befehl 106 wird die Zahl a_i ins AR gelesen und mit Befehl 107 nach Zelle 117 abgespeichert. Befehl 108 bewirkt das Einlesen der Zahl b_i ins AR, die anschliessend durch Befehl 109 mit a_i multipliziert wird. Zu diesem Produkt wird mit Befehl 110 der Wert für y aus Zelle 116 addiert und das Ergebnis wieder nach Zelle 116 gebracht. Danach beginnt die Schleife durch den Sprungbefehl 112 erneut mit Befehl 104 (Fortsetzung siehe 4.).

5.3 Die Formelsprachen

Das Codieren eines Maschinenprogramms erfordert viel Zeit und Sorgfalt. Es ist deshalb naheliegend, die mühsame und fehleranfällige Arbeit des Codierens dem Digitalrechner zu übertragen. Dazu sind 2 Voraussetzungen notwendig:

1. Es muss eine Formelsprache geben, in der sich der Rechenplan präzise und eindeutig ausdrücken lässt. Eine solche genormte Formelsprache ist z. B. ALGOL.

2. Für jeden Digitalrechner-Typ muss ein für allemal ein Übersetzungs-Maschinenprogramm angefertigt sein; mit Hilfe dieses «Übersetzers» werden dann alle in der Formelsprache formulierten Rechenpläne durch den Digitalrechner selbst in ein ihm verständliches Maschinenprogramm übersetzt.

Im technisch-wissenschaftlichen Bereich bedient man sich beim Programmieren immer mehr der Formelsprache ALGOL (ALGOrithmic Language); daneben wird auch die Formelsprache FORTRAN (FORmula TRANslation) vor allem in USA noch viel verwendet. Beide Formelsprachen sind im Grundsätzlichen sehr ähnlich. Für kaufmännische Zwecke wurde COBOL (COmmon Business Oriented Language) entwickelt, für nichtnumerische logische Aufgaben LOGALGOL. Die Formelsprachen APT (Automatic Programming of Tools) und AUTOPROMT (AUTOmatic PROgramming of Machine Tools) werden bei der numerischen Steuerung von Werkzeugmaschinen benutzt. Das Aufkommen der Formelsprachen — auch operative oder algorithmische Sprachen genannt — hat einen Begriffswandel der Bezeichnung «Programmieren» mit sich gebracht. Früher wurde sie oft gleichbedeutend mit Codieren gebraucht; heute bezieht sie sich auf das höhere Niveau der Niederschrift in einer Formelsprache [14].

6. ALGOL

Mit Hilfe der Formelsprache ALGOL lässt sich ein Flussdiagramm sehr einfach in vertrauter Schreibweise ausdrücken. Dazu benützt man ausser arithmetischen Formeln noch bestimmte Wortsymbole. ALGOL beruht auf internationalen Vereinbarungen und dient vorwiegend als Programmierungssprache; in zunehmendem Masse werden Rechenverfahren auch mittels ALGOL beschrieben und publiziert. Wie einfach das Programmieren mit ALGOL ist, lässt sich am besten an Hand konkreter Beispiele zeigen. Deshalb werden im folgenden die wichtigsten Vereinbarungen der Formelsprache ALGOL skizziert, soweit sie für das Verständnis der Beispiele notwendig sind. Derjenige Leser, der sich eingehend über ALGOL informieren will, sei auf das ALGOL-Manual der ALCOR-Gruppe [7]) verwiesen [10].

[7]) Zur Vereinheitlichung der ALGOL-Übersetzungsprogramme und zum Erfahrungsaustausch haben sich mehrere Institutionen 1959 zur ALCOR-Gruppe zusammengeschlossen (ALCOR: algol converter). Mitglieder der ALCOR-Gruppe sind Rechenzentren an Techn. Hochschulen, Universitäten, Forschungsinstituten und Firmen in der Schweiz, Deutschland, Österreich, Holland und den USA.

6.1 Zeichen

Bei Eingabe des ALGOL-Programms mittels 5-Kanal-Lochstreifen wird der internationale Fernschreibcode CCIT 2 benutzt mit

den Buchstaben A...Z

den Ziffern 0...9

den Schriftzeichen + — / . , : = ( ) '

und den zusätzlichen, vom CCIT 2-Code abweichenden Schriftzeichen × ; $_{10}$ [ ]

Bei Verwendung von Lochkarten stehen im wesentlichen die gleichen Zeichen zur Verfügung. Zwischenräume und Neubeginn einer Zeile sind in ALGOL im allgemeinen bedeutungslos; sie sollen aber ausgiebig benutzt werden, um die Übersichtlichkeit der Aufschreibung zu erhöhen.

6.2 Zahlen und Variable

6.2.1 *Zahlen* werden als Dezimalzahl mit (oder ohne) Vorzeichen und Skalenfaktor zur Basis 10 geschrieben, z. B. die Zahl 3,14 in der Form

$$+3.14 \quad 3.14 \quad +0.314_{10}+1 \quad 0.314_{10}1 \quad 314_{10}-2$$

6.2.2 *Variable* werden mit Namen bezeichnet, die aus der Aneinanderreihung von Buchstaben und Ziffern entstehen, z. B.

X AI A22 JNEU OMEGA1

(Das 1. Zeichen ist immer ein Buchstabe; 7. und mehr Zeichen sind bedeutungslos). Bei indizierten Variablen setzt man die Indizes in eckige Klammern (durch Komma getrennt), z. B.

A [I, K] für a_{ik}

U [R, S, MUE] für $U_{RS/\mu}$

Indizes sind natürlich nur dort sinnvoll, wo sie sich im Laufe der Rechnung ändern, z. B. bei Matrixelementen. Der gleichbleibenden Nennspannung U_n würde man deshalb den Namen UN geben; die Variablen $U_{2\,\text{schätz}}$ und $U_{2\,\text{neu}}$ im Flussdiagramm Fig. 8 könnten mit UALT und UNEU bezeichnet werden.

6.3 Einfache Anweisungen

6.3.1 Mit Hilfe der *Eingabeanweisung* werden Eingabedaten am Eingabemedium aufgerufen; sie hat die Form[8])

READ (*V*, ... *V*);

(*V* = beliebige Variable, Semikolon = Schlusszeichen der Anweisung!). Z. B. bewirkt die Eingabeanweisung

READ (U1, R, P2, EPS);

das Einlesen von 4 am Eingabegerät hintereinander stehenden Zahlen, deren Werte den Variablen U1,...EPS zugeordnet werden. (EPS = Name für die Variable ε).

6.3.2 Die *arithmetische Wertzuweisung* dient zur Berechnung des Zahlenwertes einer Variablen aus einer Formel und hat die Form

$V := E;$

(*V* = Variable, *E* = arithmetischer Ausdruck). Arithmetische Ausdrücke setzen sich aus Zahlen, Variablen und den Operationszeichen zusammen.

Beispiele:

Y := X;

X1 := (Y1 × A22 — Y2 × A12) / (A11 × A22 — A12 × A21);

UMFANG := 6.2831 × R;

Der Bruchstrich wird durch den Schrägstrich ersetzt. Die Klammern haben die übliche Bedeutung. Die artihmetische Wertzuweisung darf fest vereinbarte Standardfunktionen enthalten. Solche Standardfunktionen sind:

SQRT(*E*) für Quadratwurzel von *E*
SIN(*E*) für sinus von *E*
EXP(*E*) für Exponentialfunktion von *E*
ABS(*E*) für Absolutbetrag von *E*

und andere.

Beispiele mit Standardfunktionen:

Y := SIN (X);
U2 := U1/2 + SQRT (U1 × U1/4 — R × P2);

6.3.3 Mit Hilfe der *Ausgabeanweisung* werden Zahlenwerte der Variablen am Ausgabemedium gedruckt; sie hat die zur Eingabeanweisung analoge Form

PRINT (*V*, ... *V*);

6.3.4 Die *Schreibanweisung i*n der Form

WRITE "*beliebiger Text*");

bewirkt, dass der zwischen den Doppelapostrophen stehende Text am Ausgabemedium geschrieben wird. Sie findet Anwendung für Tabellenüberschriften und Erläuterungen des Rechenablaufs, z. B. in

WRITE ("DET=0");

6.4 Die bedingte Anweisung

Will man den Ablauf der Rechnung von einem Vergleich zweier Zahlen abhängig machen, so wird dazu die bedingte Anweisung benützt. Zum Vergleichen bedient man sich der 6 Vergleichszeichen in Form der Wortsymbole (zwischen Einfach-Apostrophen gesetzt):

'LESS'	für $<$
'NOT GREATER'	$\leqq$
'EQUAL'	$=$
'NOT LESS'	$\geqq$
'GREATER'	$>$
'NOT EQUAL'	$\neq$

Die einseitige Form der bedingten Anweisung lautet

'IF' $E_1\,\varrho\,E_2$ 'THEN' $S;$

(E_1, E_2 = arithmetische Ausdrücke, ϱ = eines der 6 Vergleichszeichen, *S* = Anweisung). Ist die Bedingung $E_1\,\varrho\,E_2$ erfüllt, so wird die Anweisung *S* ausgeführt. Bei Nichterfüllung läuft die Rechnung mit der auf *S* folgenden Anweisung weiter, d. h. *S* wird dann «übersprungen» (Fig. 9a). Beispiele:

'IF' D 'EQUAL' 0 'THEN' WRITE ("DET=0");
'IF' A+B 'GREATER' C—D 'THEN' G:=A×B + C×D;

Die Anweisung *S* kann auch aus mehreren einzelnen Anweisungen bestehen, wenn diese mit den Wortsymbolen 'BEGIN' und 'END' zu einer zusammengesetzten Anweisung «zusammengeklammert» werden, z. B.

[8]) Vereinbarungen über Formen werden durch Einrahmung hervorgehoben.

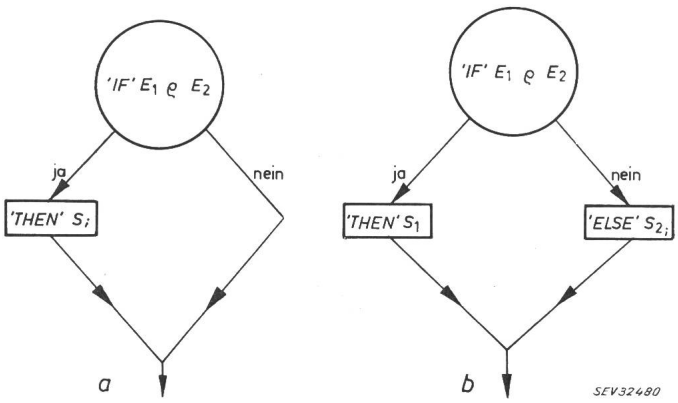


Fig. 9
Bedingte Anweisung im Flussdiagramm
a einseitige Form; *b* echte Alternative

```
'IF' D 'NOT EQUAL' 0 'THEN' 'BEGIN' X1:=C1/D;
X2:=C2/D;
PRINT (X1, X2)
'END';
```

Unter Anweisung sei im folgenden allgemein eine zusammengesetzte Anweisung verstanden. Diese darf sich aus allen Formen von Anweisungen zusammensetzen, z. B. aus einfachen und bedingten Anweisungen.

Die echte Alternative der bedingten Anweisung hat die Form

'IF' E_1 ϱ E_2 'THEN' S_1 'ELSE' S_2;

Ist die Bedingung erfüllt, dann wird nur die Anweisung S_1 ausgeführt, ansonsten nur die Anweisung S_2 (Fig. 9b). Beispiel:

```
'IF' D 'EQUAL' 0 'THEN' WRITE ("DET=0")
'ELSE' 'BEGIN' X1:=C1/D; X2:=C2/D; PRINT (X1, X2)
'END';
```

6.5 Die Laufanweisung

Wenn ein Programmteil mehrmals gerechnet und dabei die Variable V vom Wert des Ausdruckes E_1 ausgehend in Schritten von E_2 bis zum Wert E_3 verändert werden soll, benützt man die Laufanweisung

'FOR' V: = E_1 'STEP' E_2 'UNTIL' E_3 'DO' S;

(E = arithmetische Ausdrücke). Der mehrmals zu rechnende Programmteil ist die (zusammengesetzte) Anweisung S. Beispiel:

```
'FOR' P2:=PMIN 'STEP' PDELTA 'UNTIL' PMAX 'DO'
'BEGIN' U2:=U1/2 + SQRT (U1×U1/4 — R×P2);
PRINT (U2) 'END';
```

Hier wird also die Spannung U_2 des Beispiels C wiederholt berechnet mit einer Leistung P_2, die von P_{min} ausgehend in Schritten von P_{delta} bis zu P_{max} verändert wird. Mit Hilfe der Laufanweisung lassen sich Induktionsschleifen mit indizierten Variablen sehr einfach formulieren. Z. B. bewirkt die Laufanweisung:

```
'FOR' I:=1 'STEP' 1 'UNTIL' M 'DO' 'FOR' K:=1 'STEP' 1
'UNTIL' N 'DO' READ (A[I, K]);
```

dass die Elemente einer m-zeiligen und n-spaltigen Matrix zeilenweise gelesen und den indizierten Variablen a_{ik} zugeordnet werden. Hiebei ist die für den laufenden Index k sich wiederholende Anweisung selbst wieder eine Laufanweisung für den laufenden Index i.

6.6 Die Sprunganweisung

Einen von der normalen Folge abweichenden Rechenablauf erzielt man mit der Sprunganweisung

'GO TO' M;

(M = Marke). Sie bewirkt, dass die Rechnung mit derjenigen Anweisung S fortgesetzt wird, die mit der Marke M markiert ist in der Form

M:S;

Als Marken können Namen und ganze Zahlen verwendet werden. Häufig benützt man die Sprunganweisung als Teil einer bedingten Anweisung («bedingter Sprung»). Beispiel:

```
D:=A11×A22 — A12×A21;
'IF' D 'EQUAL' 0 'THEN' 'GO TO' SING;
X1:=C1/D; X2:=C2/D; PRINT (X1, X2); 'GO TO' 11111;
SING: WRITE ("DET=0");
11111: WRITE ("ENDE");
```

Ist $d = 0$, so würde der Text DET=0 gedruckt, ansonsten die Werte der Variablen x_1 und x_2 und danach stets der Text ENDE. Sprünge aus Laufanweisungen heraus sind erlaubt, Sprünge in Laufanweisungen hinein sind unerlaubt. Als Beispiel diene die Berechnung des Produktes $y = a_1\, a_2 \ldots a_n$:

```
Y:=1;
'FOR' I:=1 'STEP' 1 'UNTIL' N 'DO'
'BEGIN' 'IF' A[I] 'EQUAL' 0 'THEN' 'GO TO' DRUCK;
Y:=Y×A[I]
'END';
DRUCK: PRINT(Y);
```

6.7 Programmaufbau

Ein vollständiges Programm kann als eine einzige zusammengesetzte Anweisung betrachtet werden. Das Programm wird deshalb mit dem Wortsymbol 'BEGIN' eingeleitet und mit dem Wortsymbol 'END' abgeschlossen. Unmittelbar nach dem einleitenden 'BEGIN' muss für die im Programm vorkommenden Variablen vereinbart werden, ob sie als ganzzahlig, reell oder indiziert zu betrachten sind. Ganzzahlige Variable werden vereinbart in der Form

'INTEGER' V,...V;

und reelle Variable in der Form

'REAL' V,...V;

Beispiel:

```
'REAL' AI, BI, Y;  'INTEGER' I, N;
```

Für indizierte Variable werden Anzahl der Indizes und die Laufbereiche ihrer Zahlenwerte durch die Feldvereinbarung festgelegt mit dem Wortsymbol 'ARRAY'

Beispiel:

```
'ARRAY' A[1:M, L:30];
```

legt fest, dass A eine zweifach indizierte Variable a_{ik} ist, wobei der 1. Index i die Werte 1 bis M und der 2. Index k die Werte L bis 30 annehmen kann.

6.8 Programmbeispiele

Sehr einfach ist die Formulierung des ALGOL-Programms für Beispiel A (Abschnitt 5) mit dem Flussdiagramm in Fig. 5.

Man erkennt rasch, dass die Verzweigung im Flussdiagramm genau der echten Alternative der bedingten Anweisung von Fig. 9 entspricht. Das vollständige ALGOL-Programm für Beispiel *A* lautet:

```
'BEGIN' 'REAL' A11, A12, A21, A22, X1, X2, Y1, Y2, D;
READ (A11, A12, A21, A22, Y1, Y2);
D:=A11 × A22 — A12 × A21;
'IF' D 'EQUAL' 0 'THEN' WRITE ("DET=0")
'ELSE' 'BEGIN' X1:= (Y1 × A22 — Y2 × A12)/D;
X2:=(Y2 × A11 — Y1 × A21)/D;
PRINT (X1, X2)
'END'
'END'
```

Beispiel *B* mit Flussdiagramm in Fig. 6 enthält im wesentlichen eine Induktionsschleife. Hiefür verwendet man vorteilhaft die Laufanweisung. Das ALGOL-Programm wird dadurch sehr kurz:

```
'BEGIN' 'REAL' AI, BI, Y; 'INTEGER' I, N; READ (N); Y:=0;
'FOR' I:=1 'STEP' 1 'UNTIL' N 'DO'
'BEGIN' READ (AI, BI); Y:=AI × BI+Y 'END';
PRINT(Y)
'END'
```

Im Beispiel *C* mit Flussdiagramm in Fig. 8 sind Entscheidungen zu treffen, die sich einfach mit bedingten Sprüngen formulieren lassen. Das ALGOL-Programm für Beispiel *C* lautet:

```
'BEGIN' 'REAL' U1, UNEU, UALT, R, P2, I, EPS; 'INTEGER' Z;
READ (U1, R, P2, EPS); UNEU:=U1; Z:=0;
ITER: Z:=Z+1; UALT:=UNEU; I:=P2/UALT;
UNEU:=U1 — I × R;
'IF' Z 'EQUAL' 100 'THEN'
'BEGIN' WRITE ("KEINE KONVERGENZ");
'GO TO' SCHLUSS
'END';
'IF' ABS((UNEU — UALT)/UNEU) 'GREATER' EPS 'THEN'
'GO TO' ITER;
SCHLUSS: PRINT (UNEU, I)
'END'
```

Der Anfang der Iterationsschleife wird durch die Marke ITER markiert, auf die am Ende der Schleife (10. Zeile) zurückgesprungen wird, wenn die Genauigkeit noch nicht erreicht ist.

7. ALGOL-Programm für die Kurzschlussberechnung

Die Gedankengänge, die von der Formulierung eines Problems bis zur Erstellung des ALGOL-Programms führen, sollen skizziert werden am Beispiel einer automatischen Kurzschlussberechnung. Das gestellte Problem sei die Berechnung der dreipoligen, symmetrischen Stosskurzschluss-Wechselströme in einem beliebig aufgebauten Drehstromnetz. Der Kurzschlussort soll der Reihe nach an allen Sammelschienen des Netzes angenommen werden.

Zunächst gilt es, das Problem zu idealisieren. Dabei ist zu untersuchen, ob durch die Vereinfachung noch brauchbare Ergebnisse erzielt werden. So möge es hier genügen, bei Freileitungen und Kabeln nur die Reaktanzen zu berücksichtigen [9]).

[9]) Um das Programm anschaulich zu gestalten, wird auf die Einbeziehung von Transformatoren und die Berücksichtigung Ohmscher Widerstände verzichtet.

[10]) Anzahl der Spalten = Anzahl der Zeilen.

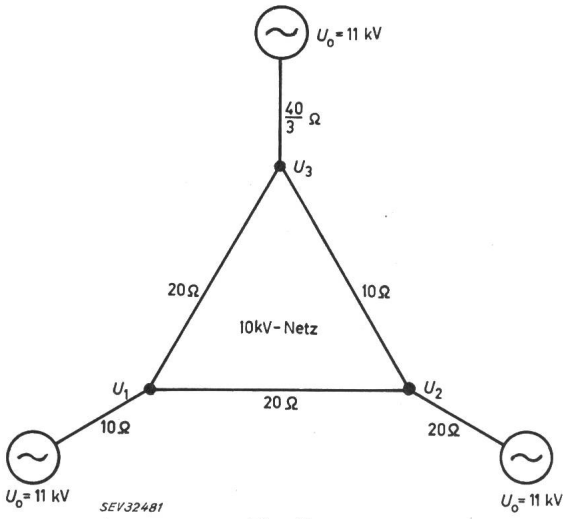


Fig. 10
Ersatzschaltung eines Netzes mit Reaktanzwerten und Polrad-EMK's

Generatoren sollen durch ihre Anfangsreaktanz und ihre Polrad-EMK charakterisiert sein; die Polrad-EMK U_0 sei konstant und überall gleich ($U_0 = 1{,}1\ U_N$). Verbraucher bleiben unberücksichtigt. Als Modell erhält man dann eine Ersatzschaltung, in der die Sammelschienen als Knotenpunkte erscheinen, die miteinander verbunden sind über Zweige mit den Reaktanzwerten der Leitungen, Kabel und Generatoren. Fig. 10 zeigt die Ersatzschaltung eines Netzes mit einem Leitungs-Dreieck, an dessen Eckpunkten Generatoren einspeisen.

Als mathematischer Ansatz für die Ersatzschaltung eines beliebigen Netzes werde das Knotenpunktsverfahren gewählt [1; 2]. Die n Knotenpunkte im Netz seien fortlaufend 1,...n numeriert, die Knotenpunkte der Polrad-EMK's mögen mit der Nummer 0 gekennzeichnet sein. Bei Kurzschluss z. B. im Knotenpunkt *1* lautet dann das System der Knotenpunktsgleichungen:

$$\begin{aligned}
&G_{10}\,U_0 + G_{11}\,U_1 + G_{12}U_2 + \ldots G_{1n}\,U_n = -\sqrt{3}\,I_1\\
&G_{20}\,U_0 + G_{21}\,U_1 + G_{22}\,U_2 + \ldots G_{2n}\,U_n = 0\\
&\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\\
&G_{n0}\,U_0 + G_{n1}\,U_1 + G_{n2}\,U_2 + \ldots G_{nn}\,U_n = 0
\end{aligned}$$

mit U = Leiterspannung und I = Kurzschlußstrom. Die mit U_0 multiplizierten, konstanten Anteile bringt man auf die rechte Seite und erhält dann ein System von n linearen Gleichungen mit n Unbekannten:

$$\begin{aligned}
&G_{11}\,U_1 + G_{12}\,U_2 + \ldots G_{1n}\,U_n = y_1\\
&G_{21}\,U_1 + G_{22}\,U_2 + \ldots G_{2n}\,U_n = y_2\\
&\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\\
&G_{n1}\,U_1 + G_{n2}\,U_2 + \ldots G_{nn}\,U_n = y_n
\end{aligned} \qquad (4)$$

Die Koeffizienten G_{ik} in Gl. (4) lassen sich übersichtlich zu einer quadratischen Matrix[10]) zusammenfassen:

$$G = \begin{pmatrix} G_{11}\,G_{12}\ldots G_{1n}\\ G_{21}\,G_{22}\ldots G_{2n}\\ \ldots\ldots\ldots\ldots\\ G_{n1}\,G_{n2}\ldots G_{nn}\end{pmatrix}$$

Ihre Elemente bestimmen sich sehr einfach aus den Kehrwerten der Reaktanzen:

Diagonalelement G_{ii} = Summe der Leitwerte der im Knotenpunkt i anliegenden Zweige,

nichtdiagonal-Element G_{ik} = negative Summe der Leitwerte der die Knotenpunkte i und k verbindenden Zweige.

Ein günstiges Lösungsverfahren ergibt sich, wenn Gl. (4) nach den Unbekannten $U_1,...U_n$ aufgelöst wird in der Form:

$$U_1 = Z_{11}\,y_1 + Z_{12}\,y_2 + ...Z_{1n}\,y_n$$
$$U_2 = Z_{21}\,y_1 + Z_{22}\,y_2 + ...Z_{2n}\,y_n$$
$$\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots$$
$$U_n = Z_{n1}\,y_1 + Z_{n2}\,y_2 + ...Z_{nn}\,y_n$$

mit der Koeffizientenmatrix

$$Z = \begin{pmatrix} Z_{11}\,Z_{12}\,...Z_{1n} \\ Z_{21}\,Z_{22}\,...Z_{2n} \\ \cdots\cdots\cdots\cdots \\ Z_{n1}\,Z_{n2}\,...Z_{nn} \end{pmatrix}$$

Die Matrix Z bezeichnet man als Kehrmatrix zu G; sie entspricht im Matrizenkalkül ungefähr dem, was der Kehrwert bei reellen Zahlen ist.

Für das Netz von Fig. 10 ist:

$$G = \begin{pmatrix} 0{,}200 & -0{,}050 & -0{,}050 \\ -0{,}050 & 0{,}200 & -0{,}100 \\ -0{,}050 & -0{,}100 & 0{,}225 \end{pmatrix} \text{ und } Z = \begin{pmatrix} \frac{560}{87} & \frac{260}{87} & \frac{240}{87} \\ \frac{260}{87} & \frac{680}{87} & \frac{360}{87} \\ \frac{240}{87} & \frac{360}{87} & \frac{600}{87} \end{pmatrix}$$

Die Werte dieser Kehrmatrix sind deshalb von Bedeutung, weil sich bei Kurzschluss im Knotenpunkt k der Kurzschlussstrom äusserst einfach aus dem k-ten Diagonalelement ergibt [2]:

$$I_k = \frac{U_0}{Z_{kk}} = \frac{1{,}1\,U_N}{Z_{kk}}$$

Die Hauptaufgabe liegt also im Lösungsprozess für die Berechnung der Kehrmatrix. Hiefür stehen an den Rechenzentren im allgemeinen fertige «Bibliotheksprogramme» zur Verfügung. Die Ermittlung der Kehrmatrix entspricht im wesentlichen der Auflösung eines Systems linearer Gleichungen und erfolgt meist nach der Gaußschen Eliminationsmethode (Tabelle I).

Nach der Wahl des Lösungsverfahrens ist noch zu überlegen, unter welchen Bedingungen eine Lösung möglich ist. Sicherlich darf erstens kein Zweig eine Reaktanz = 0 (Leitwert = ∞) haben. Dies lässt sich vermeiden durch Zusammenlegung der angrenzenden Knotenpunkte zu einem einzigen Knotenpunkt. Zweitens darf die Determinante der Matrix G nicht Null werden. Dies ist immer der Fall, wenn alle Zweige an die Polrad-EMK's direkt oder über andere Zweige angeschlossen sind («zusammenhängendes» Netz). Bei einer automatischen Berechnung ist es zweckmässig zu prüfen, ob einer dieser beiden Fälle eintritt.

Der Rechenplan für das Lösungsverfahren ist im Flussdiagramm von Fig. 11 skizziert. Als Zahlen sollen zunächst die Parameter n=Anzahl der Netz-Knotenpunkte, l=Anzahl der Zweige und U_N=Netzspannung in kV eingelesen werden, danach für die l Zweige jeweils die Nummern i und k der anliegenden Knotenpunkte und der Reaktanzwert X. Interessant ist die Berechnung der Matrix G. Da jeder Zweig (zwischen den Knotenpunkten i und k) mit seinem Leitwert $Y=1/X$ einen additiven Beitrag zu den Diagonalelementen G_{ii}, G_{kk} liefert und einen negativen Beitrag zu den nichtdiagonalen Elementen G_{ik}, G_{ki}, lässt sich die Matrix Zweig für Zweig «aufbauen». Zu

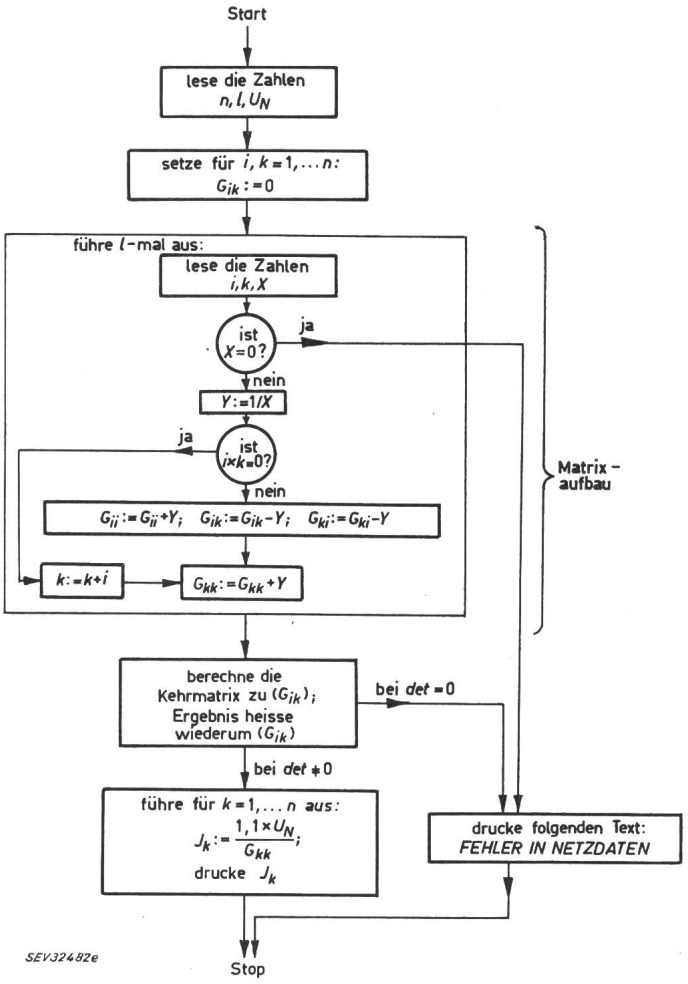


Fig. 11
Flussdiagramm für die Kurzschlussberechnung

Beginn müssen hiezu alle Elemente auf den Wert Null gesetzt werden. Ferner ist zu beachten, dass z. B. ein Zweig zwischen den Knotenpunkten 3 und 0 (=Polrad-EMK) nur einen Beitrag zu dem Diagonalelement G_{33} liefern darf, da die Elemente G_{00}, G_{03}, G_{30} in der Matrix überhaupt nicht existieren. Eine logische Entscheidung, ob der Fall i oder $k = 0$ eintritt, erreicht man einfach durch Prüfung des Produkts $i \cdot k$. Ist dies $i \cdot k = 0$, so erhält man diejenige Knotenpunktsnummer, die nicht Null ist, aus der Anweisung $k:=k+i$ und berechnet anschliessend die additive Erhöhung nur für G_{kk}.

Die Aufstellung des vollständigen ALGOL-Programms an Hand des Flussdiagramms ist dann sehr einfach. Es lautet:

```
'BEGIN' 'REAL' UN, X, Y; 'INTEGER' N, L, I, K, Z;
READ (N, L, UN);
'BEGIN' 'ARRAY' G[1:N, 1:N];
'FOR' I:=1 'STEP' 1 'UNTIL' N 'DO' 'FOR' K:=1 'STEP' 1
'UNTIL' N 'DO' G[I, K]:=0;
'FOR' Z:=1 'STEP' 1 'UNTIL' L 'DO'
'BEGIN' READ (I, K, X) ; 'IF' X 'EQUAL' 0 'THEN' 'GO TO'
FEHLER;
Y:=1/X;
'IF' I×K 'EQUAL' 0 'THEN' 'BEGIN' K:=K+I; 'GO TO'
GKK 'END';
G[I, I]:=G[I, I]+Y; G[I, K]:=G[I, K]—Y; G[K, I]:=G[K, I]—Y;
GKK: G[K, K]:=G[K, K]+Y
'END';
```

```
INVERS (G, N, FEHLER); UN:=UN×1.1;
'FOR' K:=1 'STEP' 1 'UNTIL' N 'DO' 'BEGIN' X: = UN/
G[K, K]; PRINT (K, X) 'END';
'GO TO' SCHLUSS;
FEHLER: WRITE ("FEHLER IN NETZDATEN");
SCHLUSS: WRITE ("ENDE RECHNUNG")
'END'
'END'
```

Dieses Programm enthält 2 Besonderheiten, die in Abschnitt 6 nicht behandelt wurden:

1. Die Feldvereinbarung **'ARRAY'** kann nicht bei den Vereinbarungen nach dem ersten **'BEGIN'** stehen, da erst nach der Eingabeanweisung **READ (N, L, UN)** der notwendige Wert für n bekannt ist. In solchen Fällen klammert man das Programm in ein zweites **'BEGIN'** und **'END'** ein, an dessen Anfang die Feldvereinbarung gesetzt wird (3. und vorletzte Zeile).

2. Für die Berechnung der Kehrmatrix soll ein fertiges «Bibliotheksprogramm» vorhanden sein, für das die Anweisung

INVERS (V_1, V_2, M);

gelten soll (V_1 = Name der Matrix, V_2 = Grad der Matrix und M = Marke für Sprungziel, wenn die Determinante Null ist). Diese Anweisung bewirkt, dass nach ihrer Ausführung die Kehrmatrix auf dem Platz steht, wo vorher die ursprüngliche Matrix stand. Im obigen Programm wäre also **INVERS (G, N, FEHLER);** zu schreiben; nach Ausführung ist dann z. B. die Variable G [1, 1] mit dem Wert des Elementes Z_{11} der Kehrmatrix besetzt.

Ist dieses Programm ein für allemal geschrieben und vom Digitalrechner in die Maschinensprache übersetzt und z. B. auf Lochstreifen vorhanden, so kann damit zu jedem späteren Zeitpunkt jedes beliebige Netz gerechnet werden. Die Vorbereitung des Digitalrechners besteht dann lediglich im Einlesen dieses Programm-Lochstreifens in den Speicher. Ausserdem müssen die Netzdaten auf Lochstreifen oder Lochkarten gestanzt werden, z. B. für das Netz von Fig. 10 in folgender Reihenfolge:

```
3                    (Anzahl der Netz-Knotenpunkte)
6                    (Anzahl der Zweige)
10                   (Netzspannung in kV)
1  2  20  \
1  3  20   \
2  3  10    } (i, k, X für die 6 Zweige)
1  0  10   /
2  0  20  /
3  0  13,333 /
```

Als Ergebnis würden dann für die 3 Knotenpunkte folgende Kurzschlußströme (in kA) ausgedrückt:

1	17,0893	3	15,9500
2	14,0735		

Zusammenfassung

Bei Verwendung eines Digitalrechners ist es vor allem notwendig, im voraus den Lösungsweg in allen Einzelheiten genau festzulegen. Dass die anschliessende Formulierung eines Programms dann nur noch eine einfache Aufgabe ist, versucht dieser Aufsatz zu zeigen.

Die Formelsprache ALGOL erleichtert dabei das Programmieren sehr wesentlich, da sie sich eng an die gewohnte Ausdrucksweise der Mathematik anlehnt. An Hand von Beispielen wird versucht, auch dem mit Digitalrechnern nicht vertrauten Leser das Unbehagen gegenüber der Programmierung zu nehmen.

Literatur

[1] *H. Prinz:* Elektronische Netzberechnung. Elektrizitätswirtschaft 57(1958), S. 524.

[2] *H. Dommel:* Digitale Rechenverfahren für elektrische Netze. Archiv f. Elektrotechnik 48(1963), S. 41 u. S. 118.

[3] *H. Frohne:* Rationalisierung beim Entwurf elektrischer Maschinen unter Verwendung digitaler Rechenautomaten. ETZ-A 84(1963), S. 49.

[4] *E. Kochendörfer:* Erfahrungen mit der elektronischen Berechnung von Transformatoren. Elektrizitätswirtschaft 62(1963), S. 158.

[5] *J. K. Dillard, H. K. Sels:* An Introduction to the Study of System Planning by Operational Gaming Models. Trans. AIEE III 78(1959), S. 1284.

[6] *J. Carpentier:* Contribution à l'étude du dispatching économique. Bull. Soc. Française des Electriciens 32(1962), S. 431.

[7] *W. Schneider:* Gesichtspunkte für die praktische Durchführung einer Netzbetriebs-Optimierung. Elektrizitätswirtschaft 62(1963), S. 152.

[8] *K.-J. Lesemann:* Prozess-Rechenanlage ermöglicht rationellen Betrieb von Kraftwerken. Elektronische Rechenanlagen 3(1961), S. 101.

[9] *R. Sauer:* Grossrechenanlagen und numerische Mathematik. Jahresbericht d. Deutschen Math.-Vereinigung 60(1957), S. 21.

[10] *R. Baumann:* ALGOL-Manual der ALCOR-Gruppe. Elektronische Rechenanlagen 3 (1961), 5/6 und 4 (1962), 2.

[11] *R. Zurmühl:* Matrizen. Springer Berlin, Göttingen, Heidelberg 1958.

[12] *W. Heimann:* Der Einsatz von Digital-Rechnern in Wissenschaft und Technik. Regelungstechnik 6(1958), S. 294.

[13] *W. Kämmerer:* Ziffernrechenautomaten. Akademie-Verlag Berlin 1960.

[14] *A. Walther:* Bedeutung der modernen Mathematik für Wissenschaft, Technik und Wirtschaft. Referat, gehalten am 12. 10. 1961 anlässlich der Jahreshauptversammlung der Arbeitsgem. Industrieller Forschungsvereinigungen in Bad Godesberg.

Adresse des Autors:
Dr.-Ing. *Hermann Dommel,* Institut für Hochspannungs- und Anlagentechnik, Technische Hochschule München, Arcisstrasse 21, München 2 (Deutschland).