STICHTING
MATHEMATISCH CENTRUM
2e BOERHAAVESTRAAT 49
AMSTERDAM

DR 23r

Ein ALGOL-60-Uebersetzer fuer die XI.

(Zeitschrift fuer moderne Rechentechnik und Automation
M.T.W., 8(1961), p 54-56, p 115-119).

E.W.Dijkstra.

1961

Zeitschrift für moderne Rechentechnik und Automation

23

MATHEMATIK TECHNIK WIRTSCHAFT

mtw

SONDERDRUCK
8. Jahrgang 1961, Heft 3

teil der Periode treten die Kombinationen, die eine gegensymmetrische Anordnung der Drucksituation angeben, stärker in Erscheinung, sie sagen jedoch nichts über das Verhalten des Zonalwindes aus (Kurve von r_{11}). Am Ende der Periode wird das Polynom g_{3j} gegenüber g_{2j} vorherrschend. Das positive Vorzeichen deutet auf Westwind zwischen ungefähr 50° N und 65° N, also in demselben Gebiet, in dem am Anfang der Periode Ostwind war.

Die Untersuchungen sollen nun nach dem Ende der vorbereitenden Arbeit in einigen Richtungen weitergeführt werden. Das nächste Ziel ist die Erarbeitung einer objektiven Klassifizierung der Großwetterlagen. Es gibt einige Klassifizierungen auf subjektiver Basis (z. B. Baur [1], Hess-Brezowsky [2] und Lauscher [4]), deren Grundlage die Lage der Druckzentren oder die Strömungsrichtungen sind. Dabei können Situationen auftreten, welche von verschiedenen Beurteilern verschiedenen Großwetterlagen beigezählt werden können, was bei statistischen Bearbeitungen zu Unhomogenitäten und verfälschten Resultaten führt.

Ein zweites Hauptziel ist die Verwendung der Polynomdarstellung für Zwecke der Wettervorhersage. Man könnte eine Hilfe für statistische Prognosen erhalten und man könnte vor allem bei den Prognosen Höhen- und Temperaturfelder miteinander verbinden. Bei der Herstellung von Mittelfristprognosen (Prognosenzeitraum 4—5 Tage) ist daran gedacht, die Konstruktion einer Vorhersagekarte auf dem Wege der Polynomdarstellung zu erleichtern und zu verbessern. Auch die Erfassung der Isoplethen des Zonalwindes und seine Vorhersage soll durch die Darstellung durch Orthogonalpolynome in Angriff genommen werden.

Es ist auch daran gedacht, die Orthogonalpolynomdarstellung der numerischen Prognose dienstbar zu machen. Es wäre schon eine Hilfe, durch die Polynome eine analytische Darstellung des Anfangsfeldes zu geben, dessen Darstellung bisher immer eine der Hauptschwierigkeiten der numerischen Prognose war. Es läßt sich auch die Vorticitygleichung selbst, mit der viele Methoden der numerischen Prognose arbeiten, durch Einführung der Orthogonalpolynome umformen, so daß eine analytische Behandlung in Frage käme. In etwas vereinfachter Form, die vor allem für großräumige Druckgebilde (Steuerungszentren) gültig ist, lautet die Vorticitygleichung:

$$\nabla^2 \frac{\partial H}{\partial t} + \frac{1}{f} \frac{\partial (H, \nabla^2 H)}{\partial (x, y)} + \beta \frac{\partial H}{\partial x} = 0 \qquad (6)$$

mit

t ... Zeit
H ... Höhe einer isobaren Fläche
f ... Coriolisparameter
β ... Änderung des Coriolisparameters in meridionaler Richtung

Durch Einsetzen der Orthogonalpolynome geht (6) über in die Form:

$$\frac{d}{dt} \sum_n \sum_k b_{nk} (f_{ni}'' g_{kj} + f_{ni} g_{kj}'') +$$
$$+ \frac{1}{f} \sum_n \sum_k \sum_r \sum_s b_{nk} b_{rs} \Big[f_{ni}' g_{kj} (f_{ri}'' g_{sj}' +$$
$$+ f_{ri} g_{sj}''') - f_{ni} g_{kj}' (f_{ri}''' g_{sj} +$$
$$+ f_{ri}' g_{sj}'') \Big] + \beta \sum_n \sum_k b_{nk} f_{ni}' g_{kj} = 0$$
$$n, r = 0, 1, 2, \ldots, m-1 \qquad k, s = 0, 1, 2, \ldots, m'-1 \qquad (7)$$

Die Striche bedeuten Ableitungen nach x bei den Polynomen f und nach y bei den Polynomen g. Von t abhängig sind nur die Koeffizienten b, daher ist (7) eine Differentialgleichung 1. Ordnung, 2. Grades in t mit den $m.m'$ unbekannten Funktionen $b_{nk}(t)$. Zur Zeit t_0 (Anfang des Prognosenzeitraumes) sind die Funktionswerte $b_{nk}(t_0)$ bekannt, das wären die Anfangsbedingungen. Gleichung (7) muß an allen Gitterpunkten gelten, man hat also auch $m.m'$ Gleichungen.

Endlich soll untersucht werden, ob nicht der Transport von Bewegungsgröße oder Wärme in fühlbarer oder latenter Form durch die Orthogonalpolynome leicht darstellbar oder aus der Darstellung des Höhenfeldes leicht gewonnen werden kann.

LITERATUR

[1] Baur F., Musterbeispiele europäischer Großwetterlagen. Wiesbaden 1947.
[2] Hess P. — H. Brezowsky, Katalog der Großwetterlagen Europas. Berichte des Deutschen Wetterdienstes in der US-Zone, Nr. 33, Bad Kissingen 1952.
[3] Jorgensen D. L., Prediction of hurricane motion with use of orthogonal polynomials. Journ. Meteor. 12. 428—435 (1955)
[4] Lauscher F., Dynamische Klimaskizze von Österreich in: Flohn H., Witterung und Klima in Mitteleuropa. Stuttgart 1954.
[5] Namias J. — P. F. Clapp, Observational studies of General Circulation patterns. Am. Met. Soc., Compendium of Meteorology, pp. 551—568, Boston 1951.

DR. E. W. DIJKSTRA, AMSTERDAM

Ein ALGOL-60-Übersetzer für die X 1, II. Teil

Prozeduren und Blöcke

Jede Prozedur hat die Eigenschaften eines Blocks; umgekehrt kann jeder Block, der gemäß dem ALGOL-Text keine Prozedur ist, betrachtet werden als eine (parameterlose) Prozedur, die nur an einer Stelle angerufen wird. Da unser Übersetzer letzteres tut, werden wir im Folgenden ohne Unterschied die Wörter „Prozedur" und „Block" durcheinander gebrauchen.

Wir haben oben beschrieben, wie die Arithmetik in dem Objektprogramm festgelegt wird. Es ist von Interesse hierzu zu bemerken, daß nicht nur die individuellen Akkumulatoren nicht explizit erwähnt werden, sondern daß auch nirgends explizit spezifiziert ist, wo der Akkumulatorenkeller sich im Speicher befindet. Dies hat seine guten Gründe: es bedeutet, daß das Objektprogramm anderswo die Pflicht hat von vornherein zu entscheiden, wo der Keller lokalisiert sein soll. Es hat jedoch nicht nur die Pflicht, sondern nun auch die Freiheit, diese Entscheidung im Laufe des Programms fortwährend zu ändern. Es wird diese Freiheit ausnützen, um den Speicher so vorteilhaft wie möglich zu benützen und um es möglich zu machen, daß Prozeduren einander oder sich selbst mehrmalig anrufen. Man achte darauf, daß diese Anzahl nur dynamisch bestimmt sein kann und also während der Übersetzung grundsätzlich unbekannt ist. Dieser Entscheidungsmechanismus tritt jedesmal in Wirkung, wenn eine Prozedur angerufen wird und es ist dieser Mechanismus, den wir nun beschreiben werden.

Der arithmetische Komplex wird gesteuert von vier administrativen „Zustandsgrößen", die in vier zu diesem Zweck reservierten Speicherplätzen beibehalten werden. Es sind:

AP = ACCUMULATOR POINTER
WP = WORKING SPACE POINTER
PP = PARAMETER POINTER
BN = BLOCK NUMBER

Die Größe *AP* spielt die Rolle des obengenannten Index k der v_k, das heißt *AP* ist gleich der Anfangsadresse des erstfreien Akkumulators. Weil jeder Akkumulator 4 Speicherplätze einnimmt, entspricht der oben gegebenen Erhöhung „$k := k + 1$" (während des Füllens des nächsten Akkumulators) im Unterprogrammkomplex die Operation „$AP := AP + 4$".

Wie die Beispiele zeigen, werden alle Akkumulatoren, die während der Ausführung einer Anweisung gefüllt werden, im Laufe dieser Ausführung auch wieder freigegeben, mit anderen Worten, nach Ablauf der Anweisung hat *AP* wieder denselben Wert wie am Anfang. In einer Reihe von Anweisungen eines selben Blocks nimmt also die Größe *AP* zwischen den Anweisungen einen konstanten Wert an: dieser konstante Wert ist außerdem festgelegt in der Größe *WP*, die also während der Ausführung eines Blocks den „Anfang des Arbeitsspeichers" angibt.

Die Ausarbeitung algebraischer Ausdrücke ist derart organisiert, daß, wenn ein komplizierter Teilausdruck ausgerechnet werden muß, der Wert dieses Teilausdrucks, wie kompliziert er auch war, am Ende immer in jenen Akkumulator geschrieben wird, der am Anfang der erste freie war. Für die Auswertung des Teilausdrucks ist in der Zwischenzeit eine Anzahl der nächsten Akkumulatoren zeitweilig benützt worden. Der Gedanke ist gewesen, dieselbe Technik anzuwenden in Fällen, wo in einem Ausdruck ein Teil durch eine (Funktions) Prozedur gegeben ist. Im Grunde ist letztere ebenfalls ein „komplizierter Teilausdruck": der einzige Unterschied ist, daß der Rechenvorgang für diesen Teilausdruck anderswo (und mit größerer Freiheit) definiert ist, nämlich in der Prozedurvereinbarung (Procedure declaration). Mit anderen Worten, jede Prozedur soll derart konstruiert sein, daß sie arbeitet in dem Teil des Kellers, der anfängt bei der Zelle, angewiesen von dem *AP*-Wert im Augenblick des Anrufes. Dieser Wert wird in der Größe *PP* festgelegt: der PARAMETER POINTER gibt also fortwährend die Stelle im Keller an, wo der nun aktive Block anfing zu arbeiten. (Der *PP*-Wert, der der einzigen Aktivierung des Hauptprogramms zugeordnet ist, ist belanglos.) Ebenso wie die Größe *WP* ist also auch *PP* während einer (bestimmten) Ausführung eines Blocks konstant.

Für die letzte administrative Größe *BN* gilt dies noch in höherem Maße: für einen bestimmten Block nimmt *BN* nämlich immer denselben Wert an. Für jeden Block wird der zugehörige Wert von *BN* während der Übersetzung aus rein lexikographischen Gründen ein für allemal festgelegt: *BN* gibt nämlich für jeden Block an, durch wieviel Blöcke er (lexikographisch) umfaßt wird. Demgemäß ist *BN* für das Hauptprogramm gleich Null. An dem (einzigen) Eingang jedes Blocks fügt der Übersetzer in das Objektprogramm die Befehle ein, die der Größe BN den dem Block zugehörigen Wert geben. Die Größe *BN* spielt eine Rolle bei Referenz nach non-lokalen Variablen (siehe unten).

Gesetzt den Fall, Block *a* enthält einen Ausdruck, worin eine (Funktions)Prozedur angerufen wird. Der Prozedur-Körper (procedure body) sei Block *b*, der lexikographisch unmittelbar umfaßt sei von Block *c*. Mit anderen Worten, die Aktivierung (procedure statement) von Block *b* steht innerhalb Block *a* und die Definition (procedure declaration) von Block *b* steht in Block *c*. (Man achte darauf, daß $a = b$ oder $a = c$ gestattet ist; $b = c$ ist selbstverständlich ausgeschlossen.) Die laufenden Werte der Zustandsgrößen im Moment, daß in Block *a* der Block *b* angerufen wird, bezeichnen wir mit *APa*, *WPa*, *PPa* und *BNa*. Wir geben nun ein Bild des Kellers, sobald der Übergang von Block *a* nach Block *b* gerade vollendet ist, und zwar unter der Voraussetzung, daß Block *b* eine Prozedur mit zwei formellen Parametern ist.

APa ⇒	0		Erster freier Akkumulator im Augenblick des Anrufes; hierin soll die Prozedur *b* ihr Ergebnis hinterlassen.
	1		
	2		
	3		
	4	*APa*	Angabe, ob ein Ergebnis gewünscht wird, und wenn ja, wo.
PPb ⇒	5	*PPc*	Rückkehrdaten.
	6	*WPa*	
	7	Rückkehradresse nach Block *a*	
	8	*PPa*	
	9	*BNa*	
	10		dynamische Charakterisierung (sogenanntes „PARD") des ersten aktuellen Parameters.
	11		
	12		dynamische Charakterisierung (sogenanntes „PARD") des zweiten aktuellen Parameters.
	13		
APb, *WPb* ⇒	14		

Oben ist das Kellerbild gegeben, das erzeugt wird von dem Anrufmechanismus *ETMR* (EXTRANSMARK RESULT). Erstens erhöht dieser *AP* um 4, um einen Akkumulator zu reservieren für das Ergebnis der Prozedur und in die nächste Zelle wird die Anfangsadresse dieses reservierten Akkumulators gespeichert.

ALGOL-60 gestattet jedoch ebenfalls, daß der Anruf dieser Prozedur außerhalb eines Ausdrucks, das heißt als selbständige Prozeduranweisung (procedure statement), vorkommt. In diesem Falle hat der anrufende Programmteil kein Interesse an dem Wert, den die Prozedur ihrer eigenen Bezeichnung (identifier) zuordnen will; die Erhöhung von *AP* um 4 wird unterdrückt und zur Angabe dieser Situation wird die nächste Zelle mit einer negativen Zahl ($= -0$) gefüllt. Solches geschieht in dem Prozeduraktivierungsmechanismus *ETMP* (EXTRANSMARK PROCEDURE), der folgende Besetzung der nächsten Zellen im Keller hervorruft:

APa ⇒	0	-0 Angabe, daß ein eventuelles Ergebnis nicht gewünscht wird.
PPb ⇒	1	*PPc*
	2	*WPa*
	3	Rückkehradresse nach Block *a*
	4	usw.
	5	. . . . .
	6	. . . . .

(*ETMP* und *ETMR* sind zwei verschiedene Eingänge desselben Aktivierungsprogramms *ETM*: nach einigen Befehlen fließen sie zusammen.) Für die Wertzuordnung zu der eigenen Prozedurbezeichnung (siehe [1], 5, 4, 4) steht dem Objektprogramm das spezielle Unterprogramm *STP* (STORE PROCEDURE VALUE) — oder das analoge Unterprogramm *STAP* (STORE ALSO PROCEDURE VALUE)— zur Verfügung. Der Mechanismus *STP* untersucht in diesem Falle den Inhalt der Zelle $PPb - 1$; ist $\{PPb - 1\}$ positiv, so gibt $\{PPb - 1\}$ die Anfangsadresse des für dieses Ergebnis reservierten Akkumulators an und der Inhalt des jüngst gefüllten Akkumulators wird dorthin transportiert; ist jedoch $\{PPb - 1\} = -0$, so wird dieser Transport unterdrückt, weil die Prozedur *b* offenbar durch *ETMP* angerufen wurde. (Daß ALGOL-60 eine Funktionsprozedur über *ETMP* zu aktivieren erlaubt, haben wir besonders für logistische Prozeduren (Boolean procedure) als nützlich empfunden.) Die nächste Zelle, die für *PPc* reserviert ist, wird von *ETM* offen gelassen; diese Zelle wird zu Anfang der Prozedur von dem Mechanismus *SCC* (siehe unten) gefüllt. Die nächsten vier Zellen werden gefüllt mit Daten, die sich auf Block *a*, dem Block, den wir zeitweilig verlassen, beziehen;

diese Daten stehen augenblicklich ohne weiteres zur Verfügung, sie machen es möglich, nach Vollendung der Prozedur *b* die Rechnung in Block *a* korrekt zu verfolgen. Ferner werden die statischen Charakterisierungen (sogenannten PORD's), wie sie im Programmtext von Block *a* vorkommen, in dynamische Charakterisierungen (sogenannten PARD'S) übersetzt. (Wenn ein aktueller Parameter eine einzelne Variable ist, so enthält das PARD die physische Adresse der Variablen. Wenn ein aktueller Parameter ein Ausdruck ist, wird dieser in der Form eines Unterprogramms festgelegt und das zugehörige PARD wird die Anfangsadresse dieses Unterprogramms enthalten. Ein aktueller Parameter kann primär gegeben sein als eine Adresse (wenn er ein „Ausgabeparameter" ist, d. h. als eine vielleicht indizierte Variable) oder als ein numerischer oder logistischer Wert (Ausdruck); dies und ähnliche Daten, die von Anruf zu Anruf wechseln können, werden durch *ETM* in den PARD's festgelegt.) Jedes PARD nimmt zwei Zellen im Keller und *ETM* erhöht *AP* und *WP* bis zu der ersten freien Stelle.)

Die Steuerung der *X 1* springt nach Vollendung von *ETM* nach der Anfangsadresse von Block *b*, weil die nächsten Handlungen wohl abhängig sind von der spezifischen Prozedur, die nun aktiviert wird. Falls in diesem Block lokale Größen vereinbart sind, werden die nächsten Zellen im Keller dafür reserviert. Sobald die Prozedur angerufen wird, ist bekannt, wieviel Speicher diese lokalen Größen diesmal erfordern, und die Prozedur fängt an mit der Erhöhung von *APb* und *WPb* um diesen Betrag, und zwar bevor die eigentliche Arithmetik beginnt. Dies bedeutet aber, daß die lokalen Größen von Block *b* in dem Text des Objektprogramms nur lokalisiert werden können (und sind) bezüglich des laufenden Wertes von *PPb*.

Vor der oben beschriebenen Erhöhung von *AP* und *WP* wird erst noch die Handlung *SCC* (Short Circuit) ausgeführt und zwar gesteuert von dem Werte *BNb* der Blocknummer, die der Übersetzer dem Block *b* gegeben hat. Die Handlung *SCC* macht die Zustandsgröße *BN* gleich der mitgegebenen Nummer (also in unserem Falle $BN := BNb$), und schreibt in die noch offengelassene Zelle, angezeigt vom laufenden Wert von *PP* (in unserem Falle *PPb*) den *PP*-Wert, welcher gehört zu dem ersten, Block *b* lexikographisch umfassenden Blocke (in unserem Falle wird also der Wert *PPc* in die Kellerzelle eingefüllt). Die Handlung *SCC* kann den Wert *PPc* finden, weil die Blocknummer von Block *c* ihr bekannt ist: tatsächlich ist $BNc = BNb - 1$. Mit Hilfe von *BNc* kann der gesuchte Wert *PPc* gefunden werden in der sogenannten Auslage (DISPLAY) (siehe unten); die Handlung *SCC* ist notwendig, um später garantieren zu können, daß die Auslage bei jeder Blockverspringung angepaßt werden kann.

Die Auslage

Während der Übersetzung werden alle lokalen Variablen, die ja irgendwo im Keller gespeichert werden sollen, lokalisiert bezüglich des *PP*-Wertes des Blocks, in welchem sie vereinbart sind. Der Wert dieses *PP* kann während der Übersetzung nicht bekannt sein, weil er während der Berechnung bei jeder Aktivierung des Blocks aufs neue festgestellt wird. Dagegen ist während der Übersetzung wohl die Blocknummer bekannt. Jede lokale Variable wird daher während der Übersetzung und in dem Text des Objektprogramms charakterisiert durch ihre Position *p* in bezug auf *PP* und die Blocknummer *n*, beide dem Block gehörend, in dem die Variable vereinbart ist. Während der Ausführung eines Blocks muß das arithmetische Komplex, um die lokalen und nicht lokalen Variablen im Keller finden zu können, verfügen können über die *PP*-Werte der jüngsten, noch nicht vollendeten Aktivierungen des Blocks selber, bzw. der dieses Block lexikographisch umfassenden Blöcke. Diese *PP*-Werte nun stehen nach Blocknummer geordnet in der sogenannten Auslage; diese ist eine Reihe von Speicherzellen, die die Rolle von Indexregistern erfüllen. Um die physische Adresse einer Variablen zu bestimmen, ist Referenz nach der Auslage notwendig: ist die Variable gekennzeichnet durch Position *p* und Blocknummer *n* (siehe oben), so wird die gewünschte Adresse gefunden durch Addieren des Inhaltes der Stelle *n* der Auslage zu *p*. Für jeden Block sind die nicht lokalen Größen vereinbart in lexikographisch umfassenden Blöcken, deren Blocknummern also niedriger sind. Für die korrekte Ausführung eines Blockes ist es im allgemeinen also notwendig, daß die Auslage korrekt eingefüllt ist bis einschließlich der Zelle, angegeben durch den laufenden Wert der Blocknummer. Jedes Mal, wo die korrekte Füllung der Auslage unsicher sein würde, wird bis an die laufende Blocknummer die Auslage angepaßt durch die Handlung *UDD* (UPDATE DISPLAY).

Wir können nun einsehen, daß die Handlung *SCC* bei nicht formellen Prozeduranweisungen immer möglich ist: wenn eine Prozedur angerufen wird, sind alle ihre nicht lokalen Variablen vereinbart in Blöcken, die nicht nur die Prozedurvereinbarung, sondern auch die Prozeduranweisung umfassen. Mit anderen Worten, wenn im obigen Falle Block *b* aktiviert wird, so kann die Handlung *SCC* den gesuchten Wert *PPc* in der Auslage finden und zwar an der Stelle $BNb - 1$; zweitens wird der neuerdings eingeführte Wert *PPb* in die Auslage an die Stelle *BNb* eingeschrieben.

Die Handlung *SCC* ist notwendig um später die Handlung *UDD*, die u. a. einen Teil ausmacht des RETURN-Mechanismus am Ende einer Prozedur, ausführen zu können. Die Steuerung kehrt dann zurück in einen Block und bekommt aus den Rückkehrdaten die Verfügung über *PP* und *BN* des Blocks, in den sie zurückkehrt; *BN* gibt dann an, wo in die Auslage dieser *PP*-Wert eingefüllt werden soll. Mit diesem *PP* als Anfangswert von *x* nimmt *x* durch wiederholte Ausführung von

$$x := \{x\}$$

die *PP*-Werte an, die geordnet nach ablaufender Blocknummer in die Auslage eingefüllt werden müssen.

Die Variablen, die in der Berechnung vorkommen, können nun in zwei Weisen lokalisiert sein: statisch oder dynamisch.

Statisch lokalisiert sind alle Größen, die in dem Hauptprogramm vereinbart sind; „own" Variablen werden ebenfalls statisch lokalisiert. Statische Lokalisierung heißt, daß der Übersetzer die physischen Adressen bestimmt, wo diese Variablen gespeichert werden, und daß demgemäß jede Referenz nach einer solchen Variablen in dem Text des Objektprogramms die zugeordnete Adresse enthält. (Die statische Lokalisierung von own Variablen ist die Ursache für die früher genannten Beschränkungen unseres Übersetzers. Übrigens ist es im ALGOL-Bericht [1] nicht klar genug beschrieben, welche die Konsequenzen des Begriffes own sein sollen im Falle von Rekursivem Gebrauch.)

Dynamische Lokalisierung ist die oben beschriebene Lokalisierung von Variablen im Keller in bezug auf den dem Block zugehörigen *PP*-Wert.

Dies hat zur Folge, daß die Operationen, die einer „Adresse" einer Variablen bedürfen, fünffach im arithmetischen Komplex aufgenommen sind. So haben wir beispielsweise für die Operation TAKE RESULT, die eine neue Zahl in den nächsten Akkumulator setzt, die folgenden fünf Versionen:

TRRD	TAKE REAL RESULT DYNAMIC
TRRS	TAKE REAL RESULT STATIC
TIRD	TAKE INTEGER RESULT DYNAMIC
TIRS	TAKE INTEGER RESULT STATIC
TFR	TAKE FORMAL RESULT

Wie gezeigt, besteht die Operation *TFR* nur einmal; der Unterschied zwischen statisch und dynamisch fällt nämlich weg, weil das *PARD* eines formellen Parameters immer im Keller steht, also dynamisch lokalisiert ist. Ebensowenig machen wir hier einen Unterschied zwischen real und integer: letzteres wird nämlich durch das Objektprogramm festgelegt in dem Moment, wo der aktuelle Parameter mitgegeben wird (siehe unten).

Aktuelle und formelle Parameter

Die Festlegung in dem Text des Objektprogramms eines aktuellen Parameters geschieht, falls möglich, in einem Wort. Dieses Wort (ein PORD genannt) besteht aus drei Teilen, $\underline{a}$ (15 Bits), $\underline{t}$ (2 Bits) und $\underline{Q}$ (2 Bits).

Die zwei Bits von $\underline{t}$ geben an, ob die 15 Bits von $\underline{a}$ interpretiert werden müssen als statische oder dynamische Adresse, und im letzteren Falle außerdem, ob der aktuelle Parameter, wie er in der Prozeduranweisung steht, da schon formell ist (das „Weitergeben" eines formellen Parameters). Für einen nicht formellen aktuellen Parameter hat $\underline{Q}$ folgende Bedeutung:

$\underline{Q} = 0$: $\underline{a}$ ist die Adresse einer Variablen von Typ real
$\underline{Q} = 1$: $\underline{a}$ ist die Adresse einer Variablen von Typ integer
$\underline{Q} = 2$: $\underline{a}$ ist die Anfangsadresse einer Prozedur (eines Unterprogramms) ohne oder mit einem numerischen Ergebnis.
$\underline{Q} = 3$: $\underline{a}$ ist die Anfangsadresse eines (impliziten) Unterprogramms mit einer Adresse als Ergebnis.

Sobald ein aktueller Parameter zu kompliziert ist, um durch ein Wort wie oben völlig charakterisiert zu werden, gibt dieser aktuelle Parameter in dem Objektprogramm Veranlassung zu einem sogenannten impliziten Unterprogramm: das PORD enthält dann die Anfangsadresse dieses impliziten Unterprogramms, zusammen mit der Angabe, ob das Ergebnis eine Adresse ist. Ist der aktuelle Parameter eine indizierte Variable, dann steht der entsprechende formelle Parameter innerhalb der Prozedur vielleicht an der linken Seite eines Ergibtzeichens und deshalb gibt dieses implizite Unterprogramm, das diesen aktuellen Parameter definiert, die Adresse und nicht den Wert der Variablen. Dieser Umstand wird im PORD durch $\underline{Q} = 3$ angezeigt; allen übrigen impliziten Unterprogrammen entspricht ein PORD mit $\underline{Q} = 2$.

Durch den Anrufsmechanismus (TRANSMARK) werden die PORD's übersetzt — u. a. werden dabei dynamische Adressen umgerechnet in physische — in sog. PARD's. Die PARD's, die je zwei Worte belegen, werden anschließend an die Rückkehrdaten in den Keller gespeichert (also an die Stellen *PP* + 5, *PP* + 7 u.s.w.). Für die Prozedur spielen die PARD's die Rolle der formellen Parameter: wenn z. B. eine Prozedur den zweiten Parameter benützen will, so steht im Text des Objektprogramms eine Referenz nach dem PARD mit dynamischer Adresse *PP* + 7. Das erste Wort von einem PARD wird aus dem PORD hergeleitet, das zweite enthält den *PP*-Wert und die Blocknummer, gehörend zu dem Block, worin die entsprechende Prozeduranweisung gegeben ist; nur wenn das PORD angibt, daß ein schon formeller Parameter weitergegeben wird, werden durch TRANSMARK die beiden Worte des entsprechenden PARD transportiert. Das zweite PARD-Wort ist ohne Bedeutung in den Fällen $\underline{Q} = 0$ oder *1*; ist $\underline{Q} = 2$ oder *3*, dann werden *PP* und *BN* aus dem zweiten PARD-Wort als Anfangswerte für den Prozess *UDD* benützt, jedes Mal, wo die Prozedur die Ausführung des Unterprogramms erfordert. Um die Auswertung eines solchen komplizierten Parameters oder einer formellen Prozedur garantieren zu können, muß die Auslage für die niedrigeren Blocknummern dieselben *PP*-Werte enthalten wie im Moment des Anrufes; diese Anpassung der Auslage wird durch *UDD* ausgeführt unter Steuerung des zweiten PARD-Wortes.

Die Analyse von einem PARD findet statt in den folgenden Operationen:

TFA	TAKE FORMAL ADDRESS
TFR	TAKE FORMAL RESULT
ADF	ADD FORMAL RESULT
SUF	SUBTRACT FORMAL RESULT
MUF	MULTIPLY FORMAL RESULT
DIF	DIVIDE FORMAL RESULT

Wenn diese Analyse findet, daß $\underline{Q} = 0$ oder *1* ist, ist sie bald beendigt. Findet die Analyse jedoch $\underline{Q} = 2$ oder *3*, dann muß zur Erhaltung des gewünschten Ergebnisses ein Ausdruck oder eine Prozedur ausgewertet werden, und zwar grundsätzlich von unbeschränkter Allgemeinheit. Dies bedeutet aber, daß für die oben genannten sechs Operationen rekursive Aktivierung gestattet sein muß. Zu obigen sechs kommen noch zwei Mechanismen hinzu zur Aktivierung einer formellen Prozedur, nämlich *FTMP* (FORM TRANSMARK PROCEDURE), wenn kein Ergebnis gewünscht wird, und *FTMR* (FORM TRANSMARK RESULT), wenn dies wohl der Fall ist. Die mögliche Rekursivität dieser Mechanismen erfordert im Keller drei Stellen mehr als *ETMP* bzw. *ETMR*.

Schlußbemerkungen

Im vorhergehenden ist ein Überblick gegeben über die Struktur des Objektprogramms, oder eher ein Gesamteindruck der Operationen, mit Hilfe welcher der Übersetzer das Objektprogramm formulieren soll. Der Unterprogrammkomplex, der diese Operation ausführt, unterscheidet sich nicht wesentlich hinsichtlich Speicherbedürfnis und Geschwindigkeit von einem normalen Komplex für Gleitkommaoperationen. Auch die Struktur dieses Komplexes ist großenteils konventionell: nur die Aufgabe, neue Blöcke zu aktivieren oder ihre Aktivität zu beenden, hat zu Programmteilen Veranlassung gegeben, die, sogar nach unserem heutigen Maßstab, ziemlich verwickelt sind.

Die oben gegebenen Einzelheiten sind in erster Linie beschrieben für diejenigen unter den Lesern, die sich mehr oder weniger intensiv mit der Konstruktion eines Übersetzers beschäftigt haben. Aber auch dem Leser mit allgemeinerem Interesse wird es deutlich, daß die Verfertigung eines ALGOL-Übersetzers eine relativ einfache Arbeit ist, wenn der Übersetzer das Objektprogramm formulieren darf in dem Problem derart zugeschnittenen Operationen. Dadurch wurde es möglich, daß der Übersetzer, der etwa 2500 Befehle enthält, von zwei Personen — nämlich von J. A. Zonneveld und dem Autor — in einigen Monaten verfaßt werden konnte. Einige spezielle Eigenschaften des Übersetzers will ich nicht unerwähnt lassen.

Erstens ist der Übersetzer in hohem Maße unabhängig von der gewählten Methode für die Darstellung des ALGOL-Textes (hardware representation). Jedes Mal, wo der Übersetzer das nächste ALGOL-Symbol benötigt, ruft er ein Unterprogramm an, das das nächste ALGOL-Symbol in eine feste, interne Darstellung abliefern muß. Für dieses Unterprogramm werden ebensoviele verschiedene Versionen bestehen müssen wie zu verarbeitende Darstellungen. Auf der Hand liegen Lochstreifen mit 7 oder 5 Kanälen.

Der Kode, in dem das Objektprogramm gelocht wird, zeigt dieselbe Form von Flexibilität. Alle Referenzen nach dem Unterprogrammkomplex, der dem Objektprogramm zur Verfügung stehen soll, sind numeriert und der Übersetzer locht in diesen Fällen nur die Nummer. Der Lochstreifen mit dem Objektprogramm muß eingelesen werden durch ein spezielles einfaches Einleseprogramm, das zuvor die Daten zur Verfügung bekommt, die es an Stelle der Nummern einfüllen muß. Augenblicklich sind verschiedene Unterprogrammkomplexe schon entwickelt, arbeitend in verschiedener Genauigkeit. Weiter ist beabsichtigt, außerdem Komplexe zu entwickeln, welche wohl etwas langsamer arbeiten, dafür aber die Möglichkeit einer automatischen Programmprüfung bieten. Alle diese Komplexe können bespielt werden von demselben Objektprogrammstreifen: man braucht erst bei dem Einlesen des Objektprogramms den gewünschten Komplex zu spezifizieren.

Der wichtigste Punkt unserer Übersetzungsmethode ist vielleicht, daß wir nicht entscheiden auf die Kombination von zwei aufeinander folgenden Begrenzern (wie z. B. beschrieben in [2]), aber auf jeden einzelnen Begrenzer. Die Tatsache, daß wir nicht eine sog. Übergangsmatrix haben, aber einen Entscheidungsvektor, hat wahrscheinlich nicht unwesentlich dazu beigetragen, den Umfang des Übersetzers zu reduzieren.

Die Namenliste ist als Keller organisiert. Bei dem Anfang eines Blockes werden dessen lokale Namen der Namenliste zugefügt und, sobald die Übersetzung des Blockes vollendet ist, werden diese Namen wieder aus der Namenliste gestrichen (durch passende Erniedrigung eines Anzeigers). Dank der vollständigen Klammerstruktur der ALGOL-Sprache ist es unnötig, daneben mehr als einen Keller zu introduzieren. Algebraische Ausdrücke, geklammerte konditionelle Ausdrücke und/oder Anweisungen, geklammerte for-Anweisungen und Prozedurvereinbarungen können alle mit diesem letzten universellen Keller übersetzt werden. Die Verarbeitung der for-Anweisung, die ohne jede Beschränkung zugelassen ist, wurde erheblich erleichtert, indem das Gebiet der for-Anweisung als Block betrachtet wurde (es ist z. B. verboten, daß eine goto-Anweisung von außen hineinleitet).

Die verschiedenen Listen, die der Übersetzer während seiner Arbeit aufbaut, werden hintereinander in den Arbeitsspeichern gelegt: wenn eine der vordersten Listen zuviel wächst, werden die folgenden Listen aufgeschoben. Erst wenn der Gesamtumfang des Arbeitsspeichers nicht hinreichen würde, stoppt die Übersetzung wegen Speichermangels. So wie das Objektprogramm benützt also auch der Übersetzer den Speicher so zweckmäßig als möglich.

Um die Übersetzung so schnell wie möglich verlaufen zu lassen, wird die Eingriffsmöglichkeit völlig ausgenützt und finden Ein- und Ausgabe parallel mit der Übersetzung statt: Datentransport von der Eingabe zum Übersetzer und vom Übersetzer zur Ausgabe geschieht über zyklisch angeordnete Puffer, welche die Schwankungen in den Verarbeitungs- und Produktionsgeschwindigkeiten auffangen.

Den größten Dank bin ich schuldig erstens meinem Kollegen Herrn J. A. Zonneveld, der vom Anfang bis zum Ende die intensive Zusammenarbeit zu einem fruchtbaren Vergnügen gemacht hat, und zweitens Herrn Prof. van Wijngaarden, der in den ersten Monaten, als wir drei die zu befolgende Taktik bestimmen mußten, viele konstruktive Beiträge geliefert hat. Ein weiteres Wort großer Erkenntlichkeit gehört fast allen Mitarbeitern der Rechenabteilung des „Mathematisch Centrum" für die umfangreiche und pünktliche Arbeit, welche sie geleistet haben.

HOCHSCH.-DOZ. DR. W. EBERL — DKFM. DR. P. SWOBODA, WIEN

Die Anwendung der Probentheorie auf die Prüfung wirtschaftlicher Sachverhalte und Vorgänge, II. Teil

3. Verläßlichkeit und Kosten eines Probenplanes

Mit dem Probenumfang steigen und fallen sowohl die Verläßlichkeit als auch die Kosten einer Prüfung durch Proben. Soll nun eine bestimmte Gesamtheit durch Proben geprüft werden, so handelt es sich vor allem um die Ermittlung von Probenplänen, die ein bestimmtes aus praktischen Gründen erforderliches Mindestmaß an Verläßlichkeit mit einem möglichst geringen Aufwand an Prüfkosten verbinden. Zur Lösung dieser Aufgaben müssen beide Begriffe, Verläßlichkeit und Kosten eines Probenplanes, mathematisch erfaßt werden. Das geschieht mit Hilfe der Wahrscheinlichkeitstheorie.

3,1 Sicherheit und Schärfe.

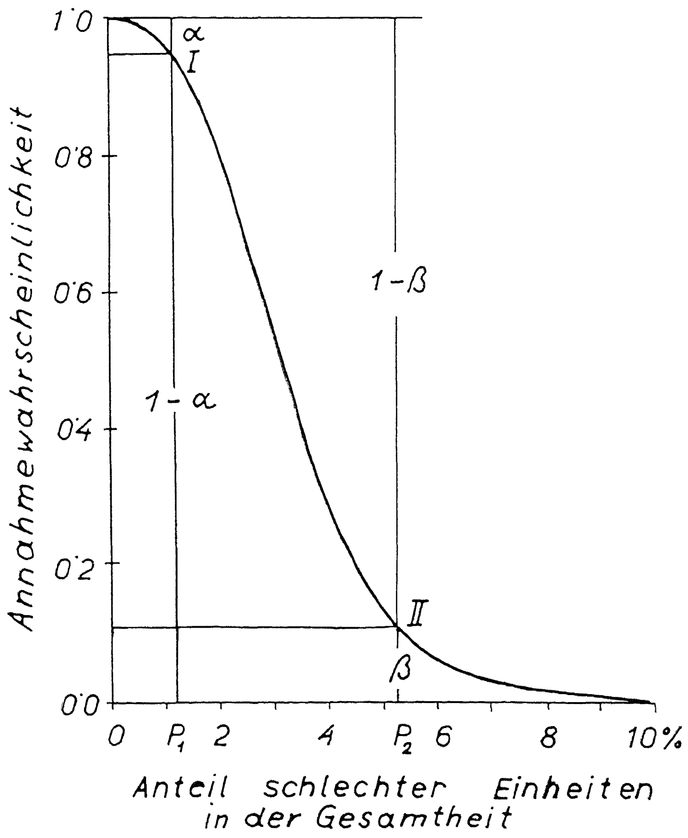
Im folgenden ist von einer Gesamtheit die Rede, deren Einheiten alternativ beurteilt werden. Im Falle eines quantitativen Befundes liegen die Dinge zwar ähnlich, sind aber nicht ganz so einfach.

Bezeichnet man mit $P\ (0 \leq P \leq 1)$ den Bruchteil schlechter Einheiten einer Gesamtheit, so läßt sich bei Vorliegen eines bestimmten Probenplanes mit den Mitteln der Wahrscheinlichkeitstheorie zu jedem P die zugehörige *Annahmewahrscheinlichkeit* $W(P)$ der Gesamtheit ausrechnen. Über den Verlauf der Bildkurve dieser Funktion $W = W(P)$ lassen sich folgende allgemeine Aussagen machen: Ist $P = 0$, enthält die Gesamtheit also keine schlechten Einheiten, so ist $W(P) = 1$, da die Gesamtheit auf jeden Fall angenommen wird. Ist jedoch $P = 1$, besteht die Gesamtheit nur aus schlechten Einheiten, so ist $W(P) = 0$, da die Gesamtheit sicher abgelehnt wird. Im übrigen wird $W(P)$ bei jedem „vernünftigen" Probenplan umso kleiner ausfallen, je größer P ist. Die Bildkurve von $W = W(P)$ wird also, wie Abb. 4 für den Plan einfacher Stichproben mit $N = 3000$, $n = 150$, $c = 4$ zeigt, vom Punkt $(0,1)$ bis zum Punkt $(1,0)$ fallen. Da diese Kurve die Wirkung des Probenplanes auf die verschiedenen Gesamtheiten kennzeichnet, heißt sie *Kennkurve* des Probenplanes. Ihre Ordinaten bezüglich der Geraden $W(P) = 1$ geben die *Ablehnungswahrscheinlichkeiten* $1 - W(P)$ an.

Nun seien α und β zwei kleine Zahlen, etwa $\alpha = 0{,}05$ und $\beta = 0{,}10$. Weiter seien P_1 und P_2 jene Werte von P, denen die Annahmewahrscheinlichkeiten $1 - \alpha$ bzw. β entsprechen:

$$W(P_1) = 1 - \alpha, \quad W(P_2) = \beta.$$

Abb. 4.



Offenbar wird dann jede Gesamtheit, für die $P \leq P_1$ ist, mindestens mit der (großen) Wahrscheinlichkeit $1 - \alpha$ angenommen, und jede Gesamtheit, für die $P \geq P_2$ ist, mindestens mit der (großen) Wahrscheinlichkeit $1 - \beta$ abgelehnt. P_1 und P_2 heißen daher *Annahme-* bzw. *Ablehnungsschwelle* des Probenplanes.

Bezeichnet man Gesamtheiten als *gut*, solange $P \leq P_1$, und als *schlecht*, sobald $P \geq P_2$ ist, so ergibt sich folgende Deutung der Größen α und β : α ist die Höchstwahrscheinlichkeit für die Ablehnung einer guten, und β die Höchstwahrscheinlichkeit für die Annahme einer schlechten Gesamtheit. α heißt daher das *Prüflingsrisiko* und β das *Prüferrisiko*. Beide sind bei brauchbaren Probenplänen klein. Dagegen ist $1 - \alpha$ die Mindestwahrscheinlichkeit für die Annahme einer guten, und $1 - \beta$ die Mindeswahrscheinlichkeit für die Ablehnung einer schlechten Gesamtheit. $1 - \alpha$ heißt daher *Sicherheit* und $1 - \beta$ *Schärfe* des Probenplanes. Beide zusammen machen dessen *Verläßlichkeit* aus.

Die Auswahl eines Probenplanes durch den Prüfer entsprechend den praktischen Erfordernissen eines Prüfauftrages spielt sich im Falle einfacher Stichproben und eines alternativen Befundes etwa so ab: Man wählt zunächst unter Berücksichtigung allgemein üblicher Werte zwei kleine Zahlen α und β. Dann überlegt man sich die Werte von Annahme- und Ablehnungsschwelle, d. h. den Höchstanteil P_1 an schlechten Einheiten, den man bereit ist, mit der Wahrscheinlichkeit $1 - \alpha$ anzunehmen und den Mindestanteil P_2, der mit der Wahrscheinlichkeit $1 - \beta$ abgelehnt werden soll. Mit Hilfe der 4 Zahlen α, β, P_1, P_2, also der beiden Punkte $I = (P_1, 1 - \alpha)$ und $II = (P_2, \beta)$ (Abb. 4) kann man dann die Zahlen n und c bestimmen.

Beispiel: Eine Gesamtheit von 5000 Fakturen soll mit Hilfe einer einfachen Stichprobe alternativ auf Richtigkeit oder Falschheit geprüft werden. Bei einem Prüflingsrisiko $\alpha = 0{,}05$ und einem Prüferrisiko $\beta = 0{,}10$ sei die Annahmeschwelle $P_1 = 0{,}01 = 1\,\%$ und die Ablehnungsschwelle $P_2 = 0{,}03 = 3\,\%$. Probenumfang n und Annahmezahl c sind zu bestimmen.

Die Lösung dieser Aufgabe wird entweder einer Tabelle, z. B. (2), entnommen oder mit Methoden der Probentheorie bestimmt. Man findet: $n = 393$, $c = 7$. Es müssen also 393 Fakturen geprüft werden. Sind höchstens 7 falsch, so wird angenommen, sonst abgelehnt.

Beispiel: Die Tabelle 2 enthält eine Zusammenstellung von 9 Plänen einfacher Stichproben für Gesamtheiten mit alternativem Befund. Die Annahme- und Ablehnungsschwellen sind bei allen Plänen $P_1 = 0{,}02 = 2\,\%$ bzw. $P_2 = 0{,}06 = 6\,\%$, Prüflingsrisiko α und Prüferrisiko β sind variabel. $100\, n/N$ ist der in Prozenten des Gesamtumfangs ausgedrückte Probenumfang, c ist die jeweilige Annahmezahl.

N	n	$\frac{100 \cdot n}{N}$ %	c	α	β
400	50	12,5	2	0,08	0,42
750	75	10,0	3	0,07	0,34
1375	110	8,0	4	0,07	0,21
2665	160	6,0	6	0,04	0,16
5625	225	4,0	8	0,04	0,08
15000	300	2,0	10	0,04	0,03
56250	450	0,8	14	0,04	0,005
375000	750	0,2	22	0,03	0,000
1,500000	1500	0,1	44	0,006	0,000

Dieses Beispiel illustriert sehr deutlich eine wichtige Einsicht der Probentheorie: Obwohl Hand in Hand mit dem Wachsen der Gesamtheiten der prozentuelle Probenumfang von 12,5% auf 0,1% fällt, nimmt die Verläßlichkeit der Probenpläne ständig zu, wie man an der Abnahme von α und β erkennt. Um also eine Folge von Gesamtheiten von immer größer werdenden Umfängen mit gleichbleibender Verläßlichkeit zu prüfen, hat man Proben zu prüfen, deren Umfänge prozentuell immer kleiner werden. Die Prüfung durch Proben ist daher umso wirtschaftlicher, je größer die zu prüfenden Gesamtheiten sind.

3,2 Mittlere Prüfzahl

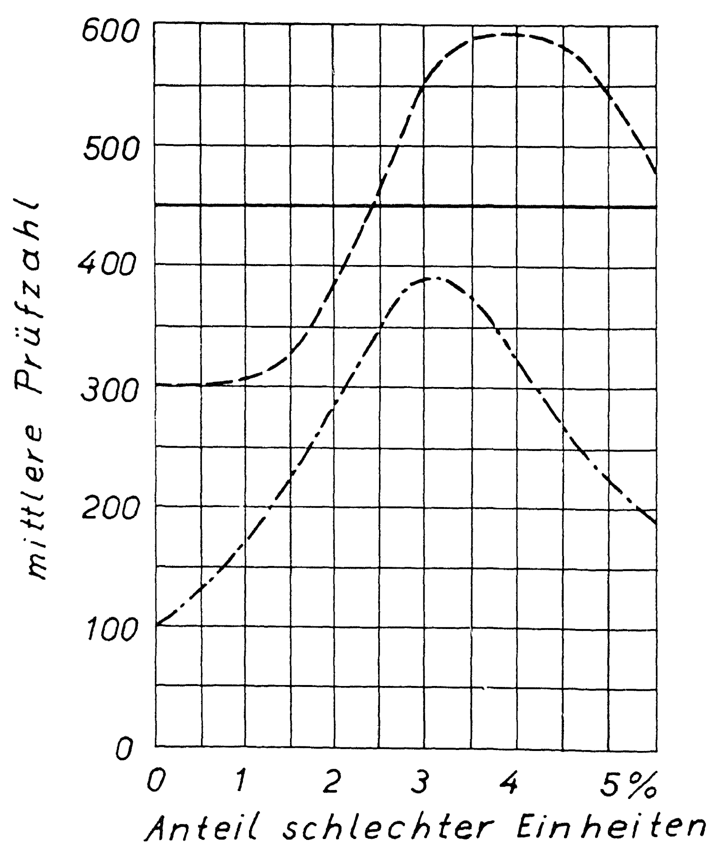
Die Prüfkosten werden in vielen Fällen proportional der Zahl von Einheiten sein, die bis zur Erreichung einer Entscheidung geprüft werden müssen. Bei einfachen Proben steht diese Zahl von vornherein fest, jedoch nicht bei doppelten, mehrfachen und Folgeproben, da bei diesen die Entscheidung nicht nach der ersten Probe erfolgen muß, sondern oft erst nach einer der folgenden eintreten wird. Legt man nun wieder einen bestimmten Probenplan zugrunde, so lassen sich die Wahrscheinlichkeiten $p_k = p_k(P)$ ausrechnen, daß für eine Gesamtheit mit dem Anteil P an schlechten Einheiten die Entscheidung unmittelbar nach der k-ten Probe eintritt, $k = 1, 2, \ldots$. Ist n_k der Umfang der k-ten Probe, so ist das Mittel der je Gesamtheit zu prüfenden Einheiten, die *mittlere Prüfzahl*,

$$\bar{n} = \sum_k \Big(\sum_{i=1}^{k} n_i\Big) p_k = \bar{n}(P)$$

ebenfalls eine Funktion von P. Man erhält das Bild dieser Funktion, indem man auf der waagrechten Achse wieder P und senkrecht dazu $\bar{n}(P)$ aufträgt.

Eine bestimmte Verläßlichkeit läßt sich nun sowohl durch einfache als auch durch doppelte, mehrfache oder Folgeproben erzielen. Die mittleren Prüfzahlen dieser gleich verläßlichen Pläne werden jedoch im allgemeinen verschieden ausfallen und geben dem Prüfer die Möglichkeit, sich den wirtschaftlichsten Plan, d. h. den mit der kleinsten mittleren Prüfzahl, auszusuchen.

Abb. 5.



MATHEMATIK-TECHNIK-WIRTSCHAFT

Zeitschrift für moderne Rechentechnik und Automation

Einzelheft öS 45.—, DM 8.—, sFr. 8.—, Jahresabonnement öS 165.—, DM 30.—, sFr. 30.—

Bisher erschienen als „MTW-Mitteilungen“ mit dem Untertitel „Zeitschrift zur Pflege der Beziehungen zwischen Mathematik-Technik-Wirtschaft“, herausgegeben vom Mathematischen Labor der Technischen Hochschule Wien als Fortsetzung der im Jahre 1954 gegründeten „Mitteilungen des Mathematischen Labors“.

Herausgeber: Prof. Dr. R. Inzinger und Prof. Dr. E. Bukovics, beide Mathematisches Labor, Technische Hochschule, Wien IV, Karlsplatz 13, Tel. 65-76-41. Eigentümer, Verleger und Anzeigenverwaltung: Stiasny-Verlag G. m. b. H.; nach dem Pressegesetz für den Inhalt verantwortlich: Wilhelm Setje-Eilers; Druck: Buchdruckerei Heinrich Stiasny's Söhne, alle Graz, Annenstraße 65; Printed in Austria. In Deutschland: Anzeigenannahme: Carl Gabler G. m. b. H., München 2, Karlsplatz 13; Bezugsnachweis: Hermann Drewes, Regensburg, Luitpoldstraße 18