KERNFORSCHUNGSZENTRUM

KARLSRUHE

Oktober 1972 KFK 1527

Institut für Datenverarbeitung in der Technik

Eine schnelle Fouriertransformation für den Realzeit-Betrieb

J. L. Krug
R. Mache
K. Rietzschel

GESELLSCHAFT FÜR KERNFORSCHUNG M. B. H.

KARLSRUHE

GESELLSCHAFT FÜR KERNFORSCHUNG M. B. H.
KARLSRUHE

KERNFORSCHUNGSZENTRUM KARLSRUHE

Juni 1972 KFK 1527

Institut für Datenverarbeitung in der Technik

Eine schnelle Fouriertransformation für den Realzeit-Betrieb

J.L. Krug
R. Mache
K. Rietzschel

Gesellschaft für Kernforschung m.b.H., Karlsruhe

Zusammenfassung

Für die on-line Fouriertransformation mit mittleren und kleinen Prozeßrechnern wurden zwei alternative Programmodule entwickelt. Die mathematischen Grundlagen werden kurz erläutert, die Programme anhand von detaillierten, allgemeingültigen Flußdiagrammen beschrieben und bezüglich Ausführungszeit und Speicherbedarf miteinander verglichen.

Abstract

Two alternative software-modules were developed for on-line Fourier-Transform on medium-scale and minicomputers. The basic mathematical features are briefly explained, the structures are described by means of detailed, universally applicable flow-charts and a comparison is made regarding execution- times and storage-requirements.

Inhaltsverzeichnis

1. Einleitung

Die Fouriertransformation ist in der Meßtechnik ein wertvolles und vielfach anwendbares Hilfsmittel. Charakteristische Beispiele sind die Impulsspektroskopie und die Kurventrennung, bei der Impuls- in Frequenzspektren umgewandelt bzw. überlappende Kurven durch Entfaltung getrennt werden müssen. Der wachsende Rechnereinsatz für die Meßwerterfassung und -verarbeitung verlangt nach einer schnellen und speicherplatzsparenden Fouriertransformation für den on-line-Betrieb. Es sind sowohl hardwaremäßige als auch softwaremäßige Lösungen möglich.

In der vorliegenden Veröffentlichung werden zwei on-line Programm-Module für die schnelle Fouriertransformation reeller Spektren beschrieben, die auf mittlere und kleinere Prozeßrechner implementiert werden können. Die Programme selbst wurden an einem TELEFUNKEN-Rechner TR86 entwickelt und getestet und für die NMR-Impulsspektrometrie eingesetzt.

2. Algorithmus für die schnelle Fouriertransformation

Die Reihenentwicklung einer Funktion nach Kreisfunktionen wird allgemein als Fourierreihe bezeichnet. Für eine Funktion X(k) aus N diskreten und komplexen Punkten gilt:

$$X(k) = \sum_{n=0}^{N-1} A(n) \cdot \exp(2\cdot\pi\cdot i\cdot k\cdot n/N), \quad (1)$$

A(n) sind dabei die komplexen Fourierkoeffizienten und i die imaginäre Einheit. Die Bestimmung dieser Fourierkoeffizienten ist die eigentliche Fouriertransformation, und die Glieder A(n) als Ganzes stellen die Fouriertransformierte von X(k) dar.

Nach den klassischen Verfahren ergeben sich die Koeffizienten aus einer N-fachen Summe:

$$A(n) = \frac{1}{N} \sum_{k=0}^{N-1} X(k) \exp(-2\cdot\pi\cdot i\cdot k\cdot n/N); \quad (2)$$

Für die Berechnung der Gesamtheit der N Fourierkoeffizienten sind danach N^2 komplexe Multiplikationen und N^2 komplexe Additionen erforderlich.

Nach einem Algorithmus von Cooley und Tukey[1] wird die Zahl der komplexen Rechnungen auf $2N \cdot {}_2\log N$ reduziert. Die Voraussetzung dafür ist, das Spektrum besteht aus $N=2^M$ (M = ganzzahlig) Punkten. Im allgemeinen läßt sich diese Bedingung experimentell durch geeignete Kurvenabtastung bzw. Kurvenbegrenzung oder -erweiterung erfüllen.

Die Grundlage der schnellen Fouriertransformation nach Cooley und Tukey[1] (FFT = Fast-Fourier-Transform) besteht im Übergang von der Einfachindizierung der Punkte X(k) bzw. Fourierkoeffizienten A(n) zu der Mehrfachindizierung $X(k_{M-1}, \ldots\ldots k_0)$ bzw. $A(n_{M-1}, \ldots\ldots n_0)$.

$$k = \sum_{j=0}^{M-1} 2^j k_j; \quad (3a) \qquad n = \sum_{j=0}^{M-1} 2^j n_j; \quad (3b)$$

mit $k_j = 0,1$ und $n_j = 0,1$

Die Darstellung der Indizes k und n in Potenzen von 2 ist für die Implementierung auf Digitalrechnern recht günstig.

Die Fourierkoeffizienten werden rekursiv in mehreren aufeinanderfolgenden Schritten berechnet. Die Reihenfolge der Glieder wird dabei verändert, so daß am Ende der Transformation durch einfaches Umsortieren (Bit-Spiegelung) die Fourierkoeffizienten geordnet werden müssen.

Die Rekursionsformel für den Cooley-Tukey-Algorithmus lautet:

$$A_L(k_{M-1},\ldots.k_{M-L+1},a,k_{M-L-1},\ldots.k_0) =$$

$$= \sum_{u=0}^{1} A_{L-1}(k_{M-1},\ldots.k_{M-L+1},u,k_{M-L-1}\ldots.k_0)\cdot(-1)^{a\cdot u}\cdot\exp\left(-2\pi\cdot i\cdot u\cdot 2^{-L}\cdot\sum_{i=0}^{L-2} k_{M-1-i}\,2^i\right)$$

$$a = 0,1,\ L = 1,2,\ldots.M; \quad (4)$$

Die Anfangswerte sind:

$$A_0(k_{M-1},\ldots k_0) = X(k_{M-1},\ldots k_0); \quad (5)$$

Für das Umsortieren gilt:

$$A(n_{M-1},\ldots.n_1,n_0) = A_M(k_0,k_1\ldots.k_{M-1}); \quad (6)$$

Für die Transformation reeller Spektren erweist sich der Algorithmus von Cooley und Tukey in der ursprünglichen Form als nicht vollkommen befriedigend. Die Gründe dafür sind:

a) Das reelle Spektrum muß in ein komplexes Spektrum umgewandelt werden; das bedeutet einen hohen Speicherplatzbedarf.

b) Es werden viele redundante Werte berechnet; von den N-komplexen Fourierkoeffizienten sind nur $\frac{N}{2} + 1$ voneinander unabhängig[+].

[+] Aus Gleichung (2) folgt für X(k) reell

$$A(N-n) = \overline{A(n)}; \quad (7)$$

Bei Zusammenfassen der Summanden mit n und N-n ergibt sich über Gleichung (1)

$$A(n)\cdot\exp(2\pi\cdot i\cdot k\cdot n/N) + A(N-n)\cdot\exp(2\cdot\pi\cdot i\cdot k\cdot(N-n)/N) =$$

$$= A(n)\cdot\exp(2\cdot\pi\cdot i\cdot k\cdot n/N) + \overline{A(n)}\cdot\overline{\exp(2\pi\cdot i\cdot k\cdot n/N)} =$$

$$= 2\cdot\mathcal{Re}(A(n)\cdot\exp(2\cdot\pi\cdot i\cdot k\cdot n/N)); \ n = 0,1,\ldots.\frac{N}{2}; \quad (8)$$

und damit die Behauptung, daß es für reelle Spektren nur $\frac{N}{2} - 1$ komplexe und 2 reelle voneinander unabhängige Fourierkoeffizienten gibt.

Basierend auf dem FFT-Algorithmus sind 2 wesentlich verschiedene Verfahren für die Transformation reeller Spektren[+)] entwickelt worden, die diese Nachteile nicht mehr besitzen.

Das von Brenner[2)], Cooley, Lewis und Welch[3)] entwickelte Verfahren - in der Folge kurz Brenner-Algorithmus genannt - interpretiert das reelle Spektrum von N Punkten als komplexes Spektrum von N/2-Punktepaaren. Es wendet auf dieses das Cooley-Tukey-Rechenschema an und führt in einem abschließenden Transformationsschritt das vorläufige Ergebnis in die Fourierkoeffizienten des eigentlichen Spektrums über.

Bei dem Algorithmus von Bergland[4)] wird im Gegensatz dazu der ursprüngliche Cooley-Tukey-Algorithmus abgeändert, so daß keine redundanten Elemente berechnet werden. Das einfache Rechenschema bleibt dabei erhalten, die Ermittlung der Argumente für die Kreisfunktionen und die Identifikation der errechneten Fourierkoeffizienten sind jedoch komplizierter.

Beide Verfahren scheinen für die on-line Berechnung geeignet zu sein, da sie unnütze Rechenschritte vermeiden und wegen der Überspeicherung des Ausgangsspektrums mit Zwischenergebnissen und Fourierkoeffizienten wenig Speicherplatz benötigen (abgesehen von eventuellen Listen für die Kreisfunktionen und das Umsortieren). Eine Entscheidung zwischen den Algorithmen konnte im voraus nicht getroffen werden, so daß für beide Software-Programme erstellt wurden.

+) Die Fourierentwicklung für reelle Spektren wird häufig in der folgenden Form angegeben:

$$X(h) = \frac{1}{2}a(0) + \sum_{n=1}^{\frac{N}{2}-1} \left(a(n)\cdot\cos\frac{2\cdot\pi\cdot n\cdot h}{N} + b(n)\cdot\sin\frac{2\cdot\pi\cdot n\cdot h}{N}\right) + (-1)^k\cdot\frac{a(\frac{N}{2})}{2} \quad ; \quad (9)$$

Der Zusammenhang mit den komplexen Fourierkoeffizienten A(n) ist gegeben durch

$$a(n) = \mathrm{Re}(A(n)); \quad b(n) = -\mathrm{Im}(A(N)); \quad n = 0,1,\ldots\ldots \frac{N}{2} \quad ; \quad (10)$$

3. Fouriertransformation nach Brenner[2)3)]

a) Mathematische Grundlagen

Durch Zusammenfassen je zwei aufeinander folgender Punkte des reellen Spektrums zu einem komplexen Punktepaar X(2l) + iX(2l+1) entsteht ein komplexes Spektrum. Entsprechend Gleichung (2) gilt für die Fourierkoeffizienten B(n) dieses Spektrums:

$$B(n) = \frac{2}{N} \sum_{l=0}^{\frac{N}{2}-1} \left[X(2l) + iX(2l+1)\right] \cdot \exp(-2\cdot\pi\cdot i\cdot n\cdot l/N); \quad (11)$$

Nach Berechnen aller B(n) mit dem Verfahren von Cooley und Tukey[1)] werden diese mit Hilfe der folgenden Gleichung in die Fourierkoeffizienten A(n) des ursprünglichen Spektrums überführt.

$$A(n) = \frac{1}{2} \left[B(n) + B(\tfrac{N}{2}-n) - i(B(n) - B(\tfrac{N}{2}-n))\cdot \right.$$

$$\left. \cdot \exp(-2\cdot\pi\cdot i\cdot m/N)\right]; \quad (12)$$

mit $B(\frac{N}{2}) = B(0)$ und $n = 0,1,\ldots\ldots \frac{N}{2}$;

b) Programmbeschreibung

Ein möglicher Programmablauf für eine rationelle Fouriertransformation nach dem Brenner-Algorithmus ist in Abb.2 durch ein Flußdiagramm wiedergegeben. Das Flußdiagramm als Beschreibungsform wurde gewählt, um von der Assembler-Sprache eines bestimmten Rechners frei zu sein. Es zeigt alle Einzelheiten, so daß die Codierung für jede beliebige Assembler-Sprache keine grösseren Schwierigkeiten bereiten sollte.

Der Brenner-Algorithmus zerfällt grob in 2 Hauptteile:
Die Fouriertransformation nach Cooley-Tukey und die Umformung der berechneten Fourierkoeffizienten in die des ursprünglichen Spektrums. Der erste Schritt läßt sich seinerseits unterteilen in das eigentliche Rekursionsschema und das Umsortieren der Fourierkoeffizienten.

Abb.1 veranschaulicht schematisch das Prinzip der rekursiven Berechnung.

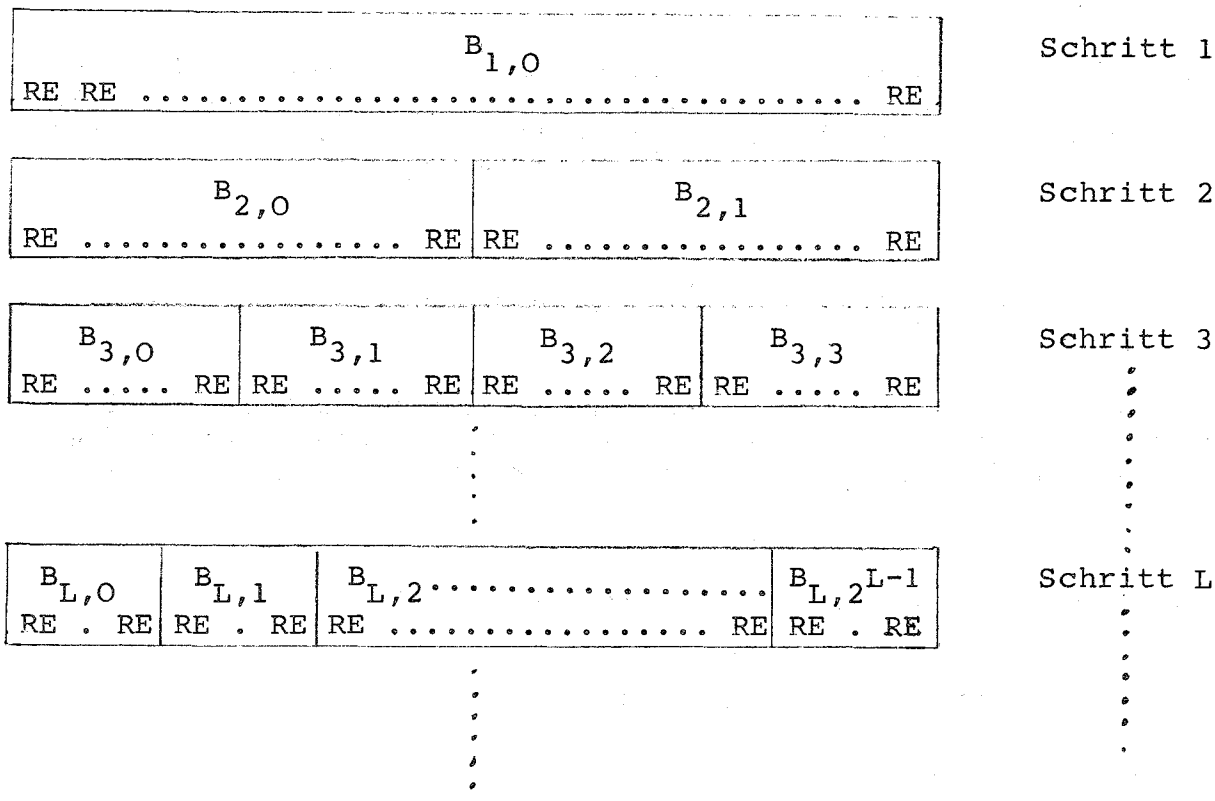


Abb.1: Schematische Darstellung des Rekursionsschemas

RE: komplexe Recheneinheit
$R_{L,I}$: I-ter Block des L-ten Rekursionsschritts
Schritt L: Rekursionsschritt L

Das Rekursionsschema enthält 3 verschiedene Strukturelemente: die komplexe Recheneinheit, den Block und den Rekursionsschritt. Die komplexe Recheneinheit ist die kleinste rationelle Grundrecheneinheit, die nach Gleichung (4) aus der gleichzeitigen Berechnung 2 komplexer Elemente eines Rekursionsschrittes besteht. Die Blöcke werden von allen komplexen Recheneinheiten eines Rekursionsschrittes gebildet, die den gleichen Exponentialfaktor (Kreisfunktionen) besitzen. Ein oder mehrere Blöcke stellen den Rekursionsschritt

dar, innerhalb dem der Übergang aller Elemente vom vorausgehenden zum momentanen Rekursionsschritt durchgeführt wird. Entsprechend diesen 3 Strukturelementen besteht das Rechenschema (im Flußdiagramm mit CT-SCHEMA bezeichnet) aus 3 ineinandergeschachtelten Schleifen.

Zur Steigerung der Rechengeschwindigkeit ist es vorteilhaft, 2 Arten von komplexen Recheneinheiten einzuführen. Während bei den normalen Elementen (GLIEDER B) eine Multiplikation mit dem komplexen Exponentialfaktor durchgeführt werden muß, entfällt diese für die Elemente (GLIEDER A), bei denen der Exponentialfaktor gleich 1 ist.

Auf das Rekursionsschema folgt das Sortieren (VERT.) der berechneten Fourierkoeffizienten. Da die Indizes der Fourierkoeffizienten vor und nach dem Umordnen durch einfache Bitspiegelung ineinander übergehen[+)], besteht der Rechenschritt lediglich aus paarweisen Vertauschungen von 2 zusammengehörigen Werten.

Die Transformation (TRANSFORM.) der Fourierkoeffizienten in die des Ausgangsspektrums ähnelt einem Rekursionsschritt des Cooley-Tukey-Algorithmus und ist annähernd gleich rechenintensiv wie dieser. Nacheinander werden paarweise Fourierkoeffizienten verknüpft und ergeben durch Multiplikation mit Exponentialfaktoren die neuen Fourierkoeffizienten[++)].

Die benötigten Kreisfunktionen werden bei diesem Programm jeweils bei Bedarf ausgerechnet. Es wird dadurch im Mittel jede Funktion dreimal berechnet, dafür kann eine umfangreiche Liste mit der Länge vom halben Spektrum eingespart werden. Insgesamt ist der Algorithmus sehr speicherplatzsparend, daß außer dem Spektrum selbst und einigen Hilfszellen kein Arbeitsspeicher benötigt wird.

[+)] siehe Gleichung (6)

[++)] siehe Gleichung (12)

4. Fouriertransformation nach Bergland[4)]

a) Mathematische Grundlagen

Die rekursive Berechnung der Fourierkoeffizienten unterscheidet sich von der des Cooley-Tukey-Algorithmus durch eine andere Kombination der Näherungswerte und durch eine kompliziertere Abhängigkeit der Exponentialfaktoren von den Indizes k_j. Außerdem treten innerhalb des Rekursionsschemas 2 unterschiedliche Rekursionsformeln auf.

$$\sum_{b=0}^{1} i^b \, B_L(k_{M-1},\ldots.k_{M-L+1},a,k_{M-L-2},\ldots.k_0) =$$

$$= \begin{cases} \sum_{u=0}^{1} (-1)^{a\cdot u} \left[\sum_{v=0}^{1} (-1)^{a\cdot v} \cdot i^{v\cdot} \right. \\ \quad \left. \cdot B_{L-1}(k_{M-1},\ldots.k_{M-L+1},u,v,k_{M-L-2},\ldots.k_0) \right] \\ \text{für } \sum_{i=1}^{L-1} k_{M-i} = 0; \quad (13) \\ \\ \sum_{u=0}^{1} (-1)^{a\cdot u} \cdot \exp(-2\cdot\pi\cdot i\cdot n\cdot 2^{-L}\cdot l(k_{M-1},\ldots.k_{M-L+1}))\cdot \\ \quad \left[\sum_{v=0}^{1} (-1)^{a\cdot v} \cdot i^{v} \cdot B_{L-1}(k_{M-1},\ldots.k_{M-L+1},v,u,k_{M-L-2},\ldots.k_0) \right] \\ \text{für } \sum_{i=1}^{L-1} k_{M-i} = 0 \; ; \quad (14) \end{cases}$$

$a = 0,1; \quad L = 1,2,\ldots M-1; \quad B_L(k_{M-1},\ldots k_0)$ reell.

Die Anfangswerte sind

$$B_0(k_{M-1},\ldots.k_0) = X(k_{M-1},\ldots.k_0) \; ; \quad (15)$$

Der Koeffizient $l(m_{p-1},\ldots m_0)$ in Gleichung (14) ist eine Funktion der Indizes m_j und gegeben durch:

$$l(m_{p-1},\ldots m_0) = \sum_{j=0}^{p-1} (2^{M-j-1} \cdot m_j \cdot (-1)^{1+\sum_{h=0}^{j} m_h}) +$$

$$+ (-1)^{\sum_{h=0}^{p-1} m_h} \cdot 2^{M-2-a} \quad ; \quad (16)$$

mit p ganzzahlig und kleiner M und a gleich "Sub"-index des höchsten nicht verschwindenden Index m_j.

Das Umsortieren der Fourierkoeffizienten erfolgt nach der folgenden Gleichung

$$A(l(k_{M-2},\ldots k_0)) = \sum_{k_0=0}^{l} i^{k_0} \cdot B_{M-1}(k_{M-1},\ldots k_0) \quad ; \quad (17),\ l \neq 0$$

und

$$A(0) = \sum_{k_0=0}^{l} i^{k_0} (B_{M-1}(0,\ldots 0) + (-1)^{k_0} \cdot B_{M-1}(0,\ldots 0,1)) \quad ; \quad (18)$$

b) Programmbeschreibung

Die wesentlichen Bestandteile des Verfahrens nach Bergland sind wie beim Cooley-Tukey-Algorithmus das Rekursionsschema und das Umsortieren der Fourierkoeffizienten. Schwierigkeiten bereiten dabei die Bestimmung der Argumente von den Exponentialfunktionen (Kreisfunktionen) und die Identifikation der Fourierkoeffizienten. In beiden Fällen tritt in den Gleichungen ein Koeffizient l auf, der in komplizierter Form von den Indizes k_i abhängt[+]. Für die Berechnung dieser Koeffizienten gibt es grundsätzlich 2 Möglichkeiten: Entweder Einzelberechnung bei Bedarf oder Be-

---

[+] siehe dazu Gleichung (16)

rechnung aller Koeffizienten über eine Tabelle. Der erste Weg ist zwar speicherplatzsparend, jedoch in der Ausführung sehr rechenintensiv. Beim alternativen Verfahren ist der Zeitaufwand wesentlich geringer, dafür wird aber ein Hilfsspeicher von der halben Länge des reellen Spektrums benötigt.

Im vorliegenden Programm wurde zur Erhöhung der Ausführungsgeschwindigkeit die Bestimmung der Koeffizienten über eine Tabelle gewählt, zumal in vielen Fällen der zusätzliche Speicherplatzbedarf in Kauf genommen und teilweise durch Overlay-Technik mit anderen Programmteilen gewonnen werden kann. Der Programmaufbau ist wiederum mit allen Einzelheiten durch ein Flußdiagramm in Abb.3 dargestellt.

Zu Programmbeginn wird die Koeffiziententabelle generiert (TABEL). Die meisten der N/2 Werte ergeben sich dabei durch eine einzige Rechenoperation aus bereits vorher berechneten Elementen der Tabelle[+)]. Mit Hilfe dieser Koeffiziententabelle werden alle im Rekursionsschema benötigten sin- und cos-Werte (SIN-COS) berechnet. Wegen Platzersparnis werden die ursprünglichen Koeffizienten durch die Kreisfunktionen überspeichert.

Das Rekursionsschema (REKURSION) ist grundsätzlich dasselbe wie in Abb.1 und unterscheidet sich bis auf die Rekursionsformel selbst kaum von dem beim Brenner-Algorithmus beschriebenen. Die Unterschiede in den entsprechenden Teilen der Flußdiagramme für beide Algorithmen ergeben sich durch Anwendung einer anderen Programmiertechnik.

Für das Umsortieren der Fourierkoeffizienten ist die Identifikation der berechneten Fourierkoeffizienten über die Koeffizienten l[++)] und der Ringtausch zwischen einer variablen Zahl von Fourierkoeffizienten notwendig. Da für letzteren keine brauchbare Methode gefunden werden konnte, wird die eigentliche

---

[+)] siehe dazu Gleichung (16) und die Originalarbeit von Brenner[4)]

[++)] siehe Gleichungen (16) und (17)

Umordnung über Zwischenspeicherung in einem Hilfsfeld durchgeführt. Der notwendige Zusatzspeicherplatz wird gewonnen, in dem die Tabelle für die 1-Koeffizienten zur Hälfte für die Identifikation und zur Hälfte für die Zwischenspeicherung der Fourierkoeffizienten verwendet wird. Die Folge davon ist, daß die Umordnung in 4 Teilschritten erfolgen muß.

Für die Durchführung des Umsortierens ergibt sich damit folgendes Schema:

a) Identifikation aller Fourierkoeffizienten mit geraden Indizes durch eine neu generierte Tabelle der 1-Koeffizienten mit der Länge N/4 (KOEFF.-TAB.)

b) Umordnen der Realteile der Fourierkoeffizienten mit geraden Indizes durch Zwischenspeicherung in die 2. Hälfte des Zusatzspeicherfeldes und anschließendes gespreiztes Rückspeichern (Faktor 2) in das Spektrenfeld[+)] (UMSORT1, UMSORT2)

c) Umordnen der Imaginärteile der Fourierkoeffizienten mit geraden Indizes analog zu b)

d) Abändern der Identifikationstabelle für die Identifikationen der Fourierkoeffizienten mit ungeraden Indizes (UMFORM TAB)

e) Umordnen der Realteile der Fourierkoeffizienten mit ungeraden Indizes durch Zwischenspeichern im Zusatzspeicher, Umspeichern der Realteile der Fourierkoeffizienten mit geraden Indizes im Spektrum selbst und Rückspeichern der Werte vom Zusatzspeicher in den Spektrenbereich (UMSORT1, UMSORT3, UMSORT2)

f) Umordnen der Imaginärteile der Fourierkoeffizienten mit ungeraden Indizes analog zu e)

[+)] Die gespreizte Rückspeicherung ist wegen Vermeidung der Überspeicherung noch nicht verwendeter Fourierkoeffizienten notwendig.

5. Implementierung

Für die Anwendung der Fouriertransformation in der Meßtechnik genügt meist, nur die relative Größe der Fourierkoeffizienten zu ermitteln und diese mit einer Genauigkeit von ca. 1% zu berechnen. Diese Tatsachen wurden bei der Implementierung beider Verfahren auf einem TELEFUNKEN-Rechner in der TR86-Assemblersprache ausgenutzt. Nach Überführen des Ausgangsspektrums durch Multiplikationen mit einer Potenz von 2 in ein Spektrum mit Maximalamplitude von 10^5 - 10^6 wurden alle Berechnungen bis auf die sin- und cos-Funktionen mit Integer-Größen durchgeführt.

Der Aufbau der Programmodule entspricht den Flußdiagrammen in Abb.2 bzw. 3. Die speziellen Gegebenheiten für die Entwicklung der Programmodule waren

- die feste Wortlänge des Rechners von 24 Bit
- das Fehlen von Indexregistern
- die mögliche Interpretation von Bitmustern
  als Integerzahlen von $-2^{23} < x < 2^{23}$ oder
  als Festkommazahlen von $-0{,}9999999 \leq x \leq 0{,}9999999$
- die Hardwaremultiplikation

Die Assemblerprogramme besitzen für den Brenner-Algorithmus eine Länge von 227 Programmzellen und 2^m Zellen für das Spektrum bzw. für den Algorithmus nach Bergland 320 Programmzellen, 2^m Zellen für das Spektrum und ein Hilfsfeld von 2^{m-1} Zellen. Hinzu kommen noch 53 Zellen für das Unterprogramm, in dem gleichzeitig die sin- und cos-Funktionen berechnet werden.

Als Ausführungszeiten für die on-line-Fouriertransformation ergeben sich die in Tabelle 2 wiedergegebenen Werte, wie sie für verschiedene Spektrenlängen im Echtzeitlauf gemessen wurden.

Tabelle 1

Theoretische Werte für die Zahl der arithmetischen Operationen bei der Fouriertransformation nach Brenner bzw. Bergland ($m = {}_2\log N$)

Algorithmus	Additionen	Multiplikationen	sin-/cos-Funktionen
Brenner	$3m \cdot 2^{m-1}$	$(m-2) \cdot 2^m$	$3 \cdot 2^{m-2} - m$
Bergland	$(3m-5) \cdot 2^{m-1}+2$	$(m-3) \cdot 2^m$	$2^{m-2} - 1$

Tabelle 2

Ausführungszeiten für die Fouriertransformation nach Brenner bzw. Bergland

Spektrenlänge	Brenner-Algorithmus	Bergland-Algorithmus
	sec	sec
512	0,22	0,19
1024	0,46	0,41
2048	1,00	0,89
4096	2,10	1,82
8192	4,35	4,00
16384	9,20	8,50

6. Vergleich der beiden Programmodule

Welches Verfahren für die spezielle Anwendung am besten geeignet ist, läßt sich durch Vergleich der Ausführungszeiten, des Speicherplatzbedarfs und der Komplexität der Programme entscheiden.

In Tabelle 1 ist die theoretische Anzahl der arithmetischen Rechenoperationen, aufgeteilt in reelle Additionen, Multiplikationen und Berechnung von Kreisfunktionen, für die beiden Algorithmen zusammengestellt. Der Vergleich zeigt unmittelbar, daß bei dem Verfahren nach Bergland weniger Rechenoperationen notwendig sind. Berücksichtigt man jedoch dabei, daß beim Brenner-Algorithmus entsprechend dem Bergland-Verfahren auch die Kreisfunktionen in einer Liste abgespeichert werden können, so ergibt sich für die Transformation eines Spektrums von 1024 Werten nur 1/7 an Zeitgewinn[+)] beim Bergland-Algorithmus gegenüber dem Brenner-Algorithmus. Diese Zeitdifferenz nimmt übrigens mit zunehmender Länge des Spektrums ab.

Die experimentellen Ausführungszeiten der Tabelle 2 bieten jedoch eine realistischere Vergleichsmöglichkeit, weil in ihnen der Zeitaufwand für die Adressenberechnung, Umsortierung und Umspeicherung enthalten sind. Entsprechend den implementierten Versionen ist der Bergland-Algorithmus mit ca. 10-12% kürzerer Ausführungszeit der schnellere Algorithmus. Korrigiert man die Zeiten für den Brenner-Algorithmus um die nicht unbedingt notwendige 3-fache Berechnung der Kreisfunktionen[++)], so wäre bei gleichem Speicherplatzbedarf die Fouriertransformation nach Brenner sogar etwas schneller als nach Bergland.

Während nach obigen Ausführungen der Arbeitsspeicher beim Bergland-Algorithmus mindestens das 1,5-fache der Spektrenlänge beträgt,

---

[+)] Es wurde angenommen, daß sich die Ausführungszeiten für eine Addition und Multiplikation wie 1 : 4 verhalten.

[++)] Das bei den Telefunken-Rechnern benutzte Unterprogramm benötigt für die gleichzeitige Berechnung eines sin- und cos-Wertes ca. 150 /usec.

ist beim Brenner-Algorithmus die einfache Spektrenlänge ausreichend. Hinsichtlich der Aufbaus der Programme zeigen bereits die Flußdiagramme der Abb. 2 und 3, daß der Bergland-Algorithmus wesentlich schwieriger in der Implementierung ist.

Insgesamt läßt sich aus dem Vergleich schließen, daß sich der Brenner-Algorithmus für die meisten Anwendungen zur Implementierung besser eignet und mit dem Bergland-Algorithmus bei hinreichendem Speicherplatzangebot[+)] eine optimale Geschwindigkeit für die Transformation erreicht werden kann.

[+)] Zur Vermeidung der mehrfachen Umspeicherung beim Bergland-Verfahren wäre ein Hilfsspeicherfeld von der Länge des zu transformierenden Spektrums ausreichend.

7. Literaturverzeichnis

[1] Cooley, J.W., Tukey, J.W.
An Algorithm for the Machine Calculation of
Complex Fourier Series
Math. Comput. 19 (1965), 297-301

[2] Brenner, N.M.
Three Fortran Programs that Perform the Cooley-
Tukey Fourier Transform
Lincoln Laboratory, Massachusetts Institute of
Technology, Lexington, Technical Note ESD-TR67-462, 1967

[3] Cooley, J.W., Lewis, P.A.W., Welch, P.D.
The Fast Fourier Transform Algorithm and its Applications
IBM Research, RC 1743, February 9, 1967, pp. 15-33

[4] Bergland, G.D.
A Fast Fourier Transform Algorithm for Real-Valued Series
Communications of the ACM 11 (1968), S. 703-710

FLUSSDIAGRAMM ZUR FFT NACH BRENNER

PAGE 001
MAY 15, 1972

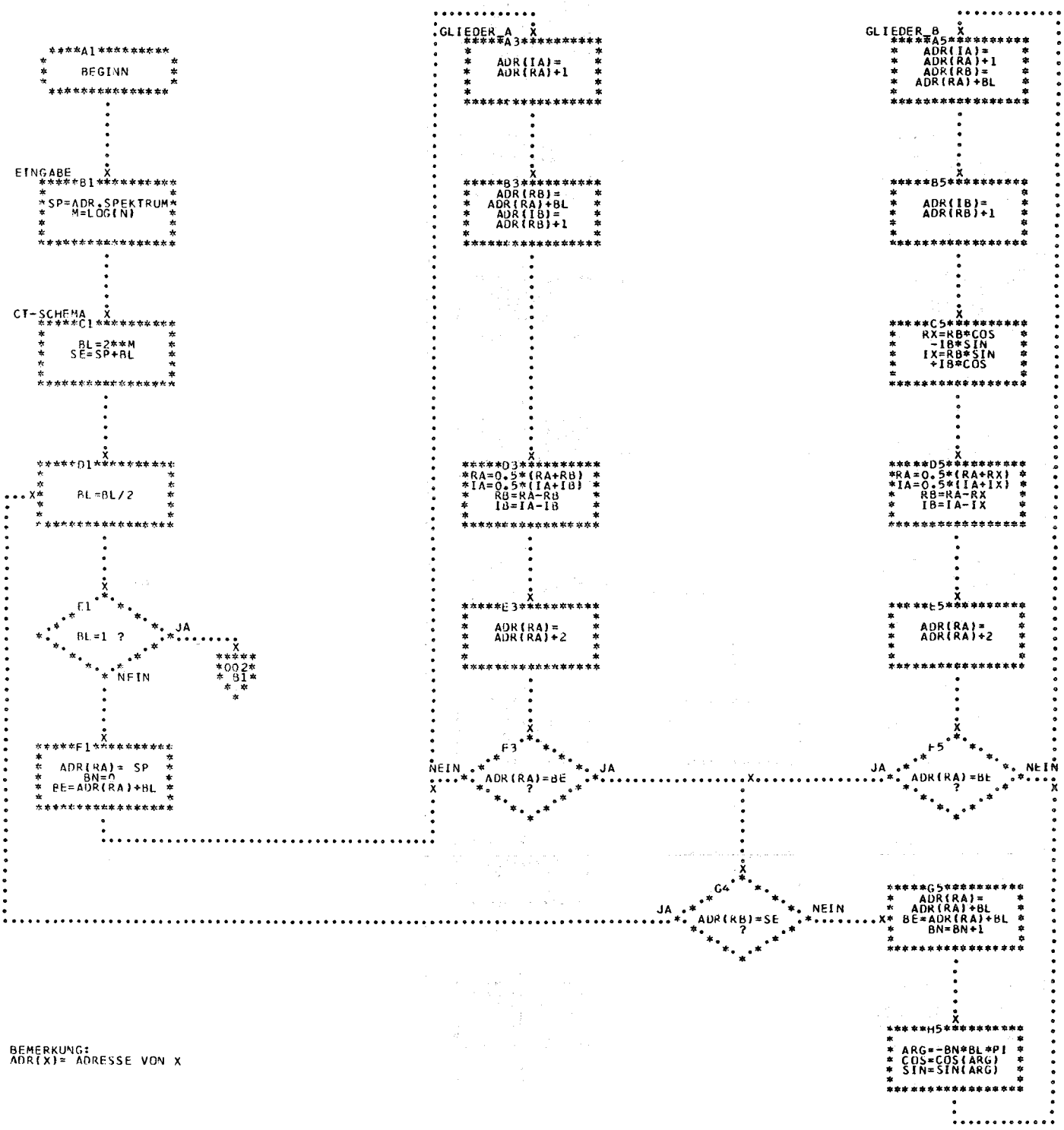


Abb. 2a

FORTSETZUNG DES FLUSSDIAGRAMMS ZUR FFT NACH BRENNER
VERTAUSCHUNG UND TRANSFORMATION COMPLEX-REELL

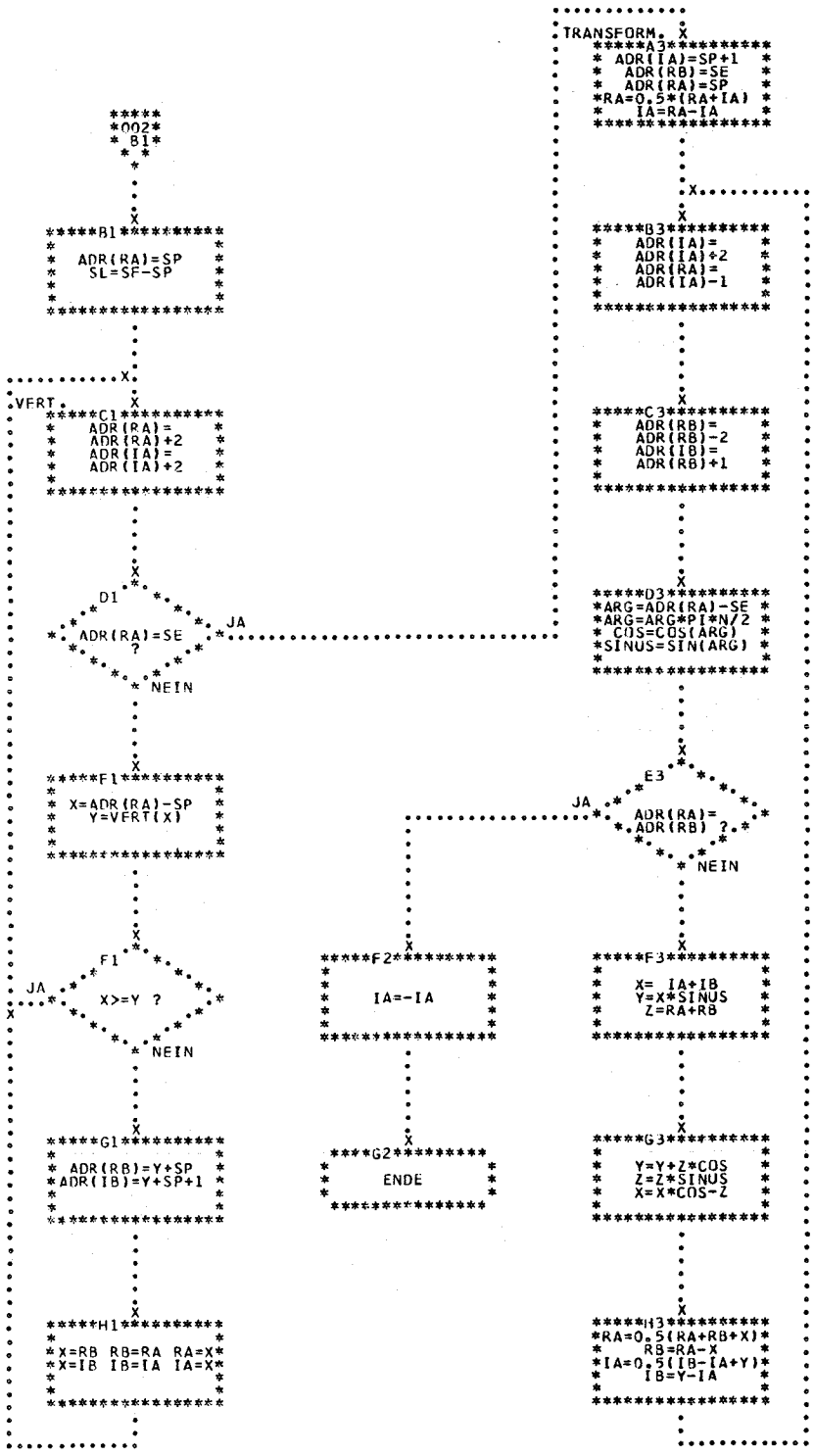
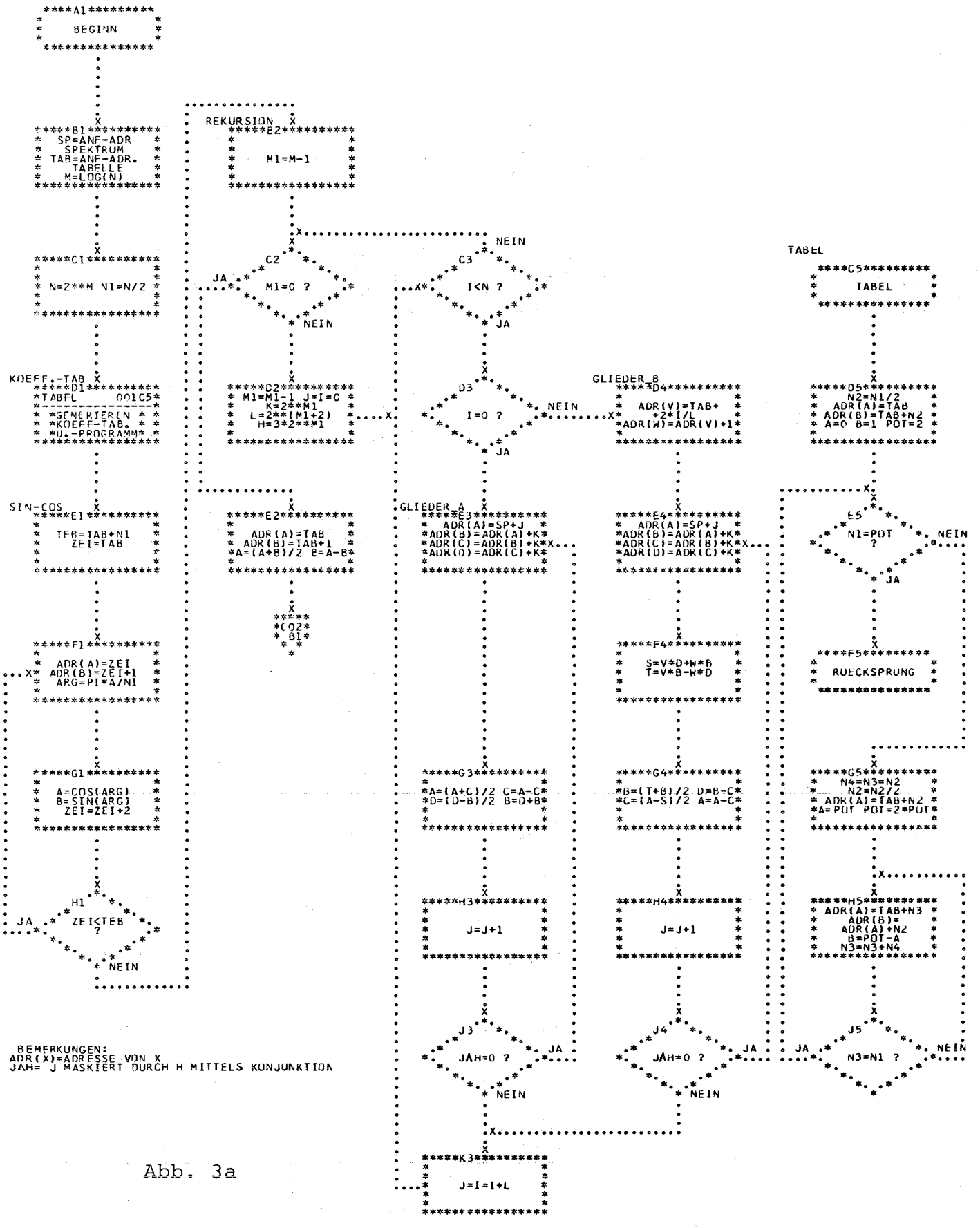


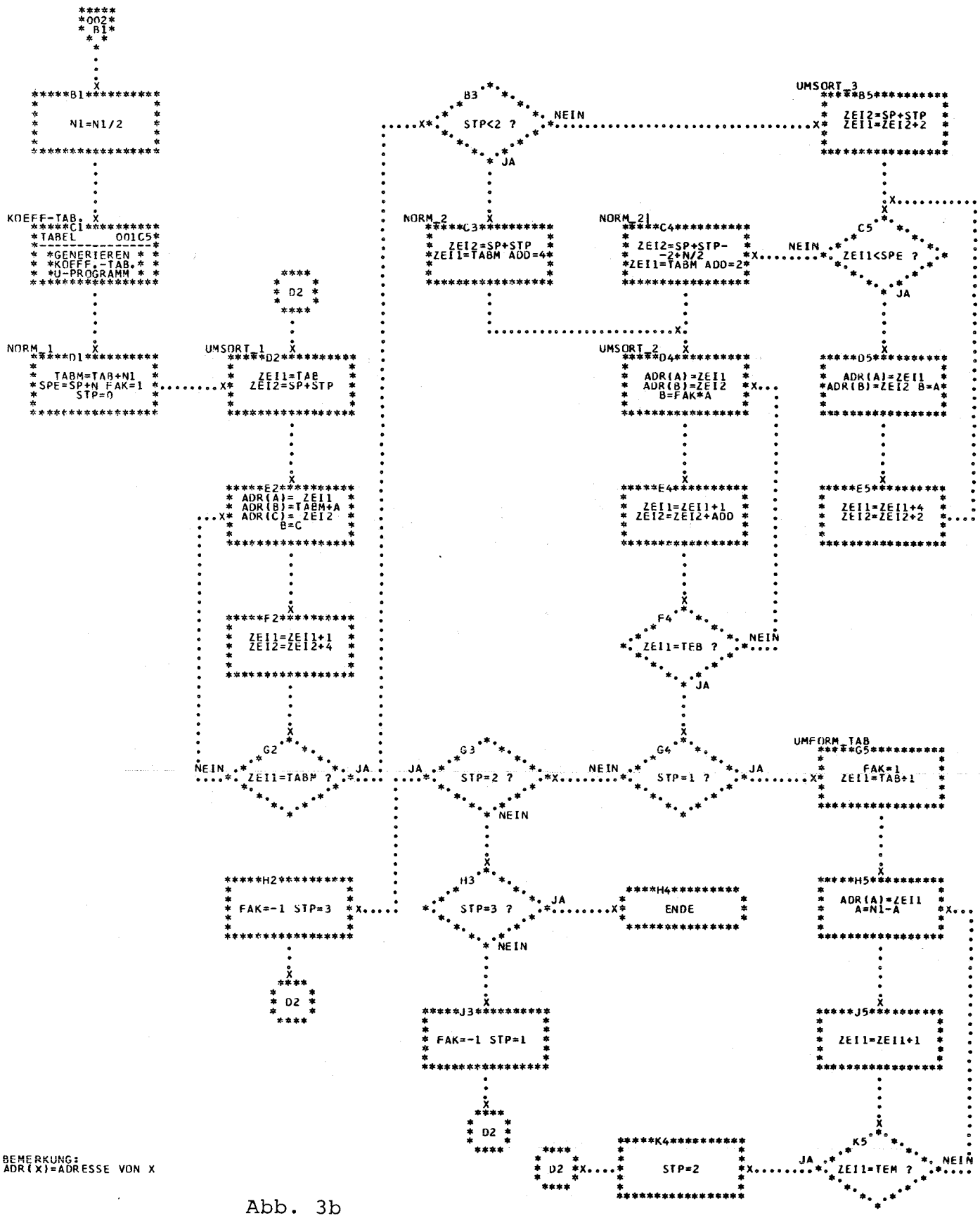
Abb. 2b

BEMERKUNGEN:
ADR(X)= ADRESSE VON X
VERT.(X)= BIT- SPIEGELUNG VON X



BEMERKUNGEN:
ADR(X)=ADRESSE VON X
JAH= J MASKIERT DURCH H MITTELS KONJUNKTION

Abb. 3a



BEMERKUNG:
ADR(X)=ADRESSE VON X

Abb. 3b