Institut für Übertragungstechnik
und Elektroakustik
Fachbereich Elektrische Nachrichtentechnik

D-6100 Darmstadt, Merckstr. 25
Tel. (06151) 16 21 69 / 16 28 69

Technische Hochschule
Darmstadt

01. Mai 1984

Diplomarbeit

Untersuchung von Vektorquantisierern in Baumstruktur für die Quantisierung von Sprach- und Residualsignalen.

W. Endres

Bearbeiter:	Meyer, Jochen	D 2065 EA
Betreuer:	Dr. Braun, FTZ Prof.Dr.rer.nat.Endres	
ausgestellt am:	01. Mai 1984	
Einlieferungstermin:	31. Juli 1984	

Erklärung

==========

Ich versichere, daß ich die vorliegende Diplomarbeit ohne fremde Hilfe und nur unter Verwendung der im Verzeichnis angegebenen Literatur angefertigt habe.

Darmstadt, im Juli 1984

Jochen Meyer

Hiermit möchte ich Dr. Braun meinen herzlichsten Dank für die freundliche Betreuung meiner Diplomarbeit aussprechen.

Ein Danke-Schön auch an Prof. Endres, der es mir ermöglichte, die Arbeit im angenehmen Betriebsklima des Forschungsinstituts der Deutschen Bundespost anzufertigen.

Mein Dank gilt auch allen Mitarbeitern der Abteilung, die mir bei vielen Problemen mit Rat und Tat zur Seite standen.

Nicht zuletzt möchte ich mich noch bei der VAX bedanken, die über vier Wochen Tag und Nacht und sogar an den Wochenenden mehr als 150 Codebücher errechnete.

Darmstadt, im Juli 1984

Jochen Meyer

1 Einleitung

1.1 Vektorquantisierung

Bei der VEKTORQUANTISIERUNG (VQ) handelt es sich um ein Verfahren zur REDUNDANZREDUKTION.

Ein Beispiel soll dies verdeutlichen: Für die digitale Übertragung von Sprache wird üblicherweise die Pulscodemodulation (PCM) verwendet. Man benötigt für eine ausreichende Sprachqualität eine Abtastfrequenz von 8 kHz und bei linearer Quantisierung eine Amplitudenauflösung von 12 Bit (CCITT-Empfehlungen). Daraus errechnet sich eine Datenrate von 96 kBit/s, ein Vielfaches derjenigen Information, die tatsächlich mit der Sprache übertragen wird.

Die Datenrate läßt sich auf folgende Weise senken: Man faßt k aufeinanderfolgende Abtastwerte (Samples) $x_1, x_2, x_3, \ldots, x_k$ zu einem Vektor $\vec{X}_1 = (x_1, x_2, x_3, \ldots, x_k)$ zusammen. Eine gegebene Folge von Abtastwerten $x_1, x_2, x_3, \ldots$ entspricht einer Folge von Vektoren $\vec{X}_1, \vec{X}_2, \vec{X}_3, \ldots$ mit:

$$\vec{X}_i = (x_{k(i-1)+1}, x_{k(i-1)+2}, x_{k(i-1)+3}, \ldots, x_{k(i-1)+k}),$$

$$i = 1, 2, 3, \ldots \qquad (1.1, 1)$$

Die Vektoren können im k-dimensionalen Raum dargestellt werden. Da aufeinanderfolgende Abtastwerte nicht statistisch unabhängig voneinander sind, sondern innere Bindungen aufweisen, sind die Endpunkte der Vektoren nicht gleichmäßig im Raum verteilt. Es gibt Häufungspunkte, die auch Wolken oder Cluster genannt werden.

Wenn die Vektoren in N Wolken aufgeteilt werden, können alle n_j Vektoren der Wolke j $\vec{X}_{j1}$, $\vec{X}_{j2}$, $\vec{X}_{j3}$,..., $\vec{X}_{jn_j}$ durch ihren gemeinsamen Mittelwert

$$\vec{Y}_j = \frac{1}{n_j} \sum_{i=1}^{n_j} \vec{X}_{ji}, \quad j = 1, 2, 3, \ldots, N \qquad (1.1, 2)$$

ersetzt werden. Dadurch wird natürlich das ursprüngliche Sprachsignal verfälscht.

Für jede Wolke existiert genau ein repräsentativer Vektor. Die Liste mit allen N Mittelwertvektoren ist das CODEBUCH $\vec{A}$. Es muß vor der Übertragung aus einem Trainingssignal berechnet werden:

$$\vec{A} = (\vec{Y}_1, \vec{Y}_2, \vec{Y}_3, \ldots, \vec{Y}_N) \qquad (1.1, 3)$$

Statt dem Vektor $\vec{Y}_j$ wird nur noch seine Nummer j übertragen. Der Empfänger setzt dann mit Hilfe des Codebuchs die Zeitfunktion zusammen. Für die Übertragung der Codebuchvektor-Adresse werden

$$R = \mathrm{ld}\ N \ \mathrm{Bit} \qquad (1.1, 4)$$

benötigt. Anstelle des Abtastwertes müssen

$$r = \frac{R}{k} \ \mathrm{Bit} \qquad (1.1, 5)$$

übertragen werden. Bei einer Vektorlänge von k = 8 Samples/Vektor und einer Codebuchgröße von N = 256 errechnet sich eine Datenrate von r = 1 Bit/Sample. Das entspricht bei einer Abtastfrequenz von 8 kHz einem Informationsfluß von 8 kBit/s, dem zwölften Teil der anfangs erwähnten Datenrate.

Leider versagt das menschliche Vorstellungsvermögen bei k-dimensionalen Räumen für $k > 3$. Deshalb ist eine andere Darstellung der Vektoren günstiger, nämlich als Folge von Abtastwerten. Bild 1.1, 1 zeigt für eine Vektorlänge von $k = 8$ Samples/Vektor ein Codebuch der Größe $N = 16$.

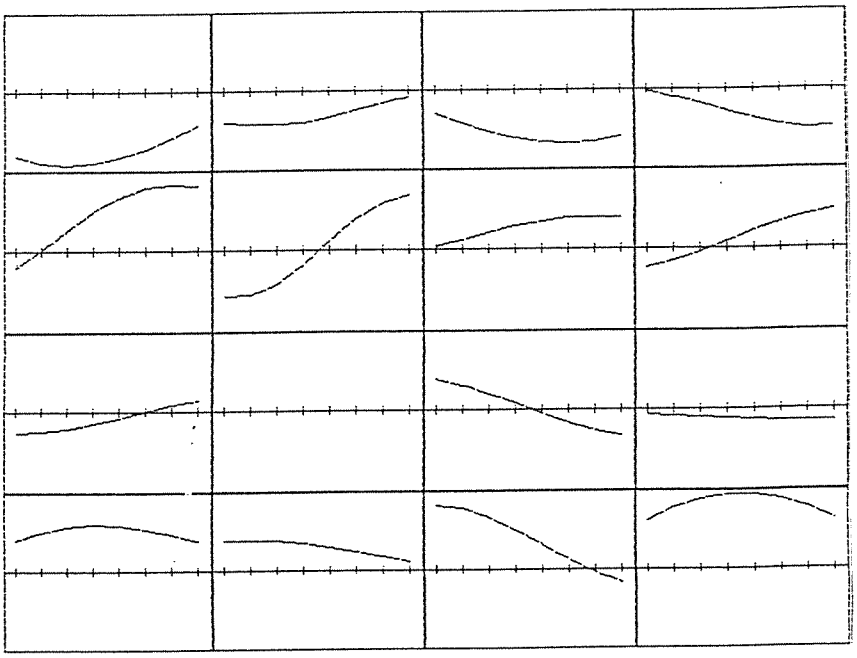

Bild 1.1, 1 Codebuch der Größe 16,
Vektorlänge: 8 Samples/Vektor

Die Quantisierung einer Zeitfunktion (Bild 1.1, 2) mit diesem Sortiment von 16 Abtastwertfolgen zeigt Bild 1.1, 3. Man erkennt deutliche Verfälschungen des ursprünglichen Signals.
Bild 1.1, 4 zeigt die entsprechende Quantisierung mit einem Codebuch der Größe 1024. Die Zeitfunktion wird erheblich besser angenähert.

Bild 1.1, 2 Ausschnitt von 10 ms aus einer Sprachprobe

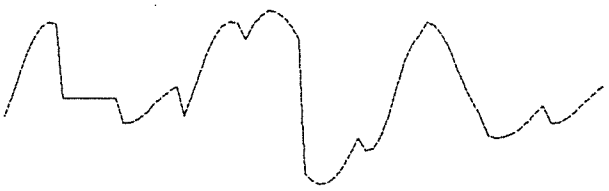

Bild 1.1, 3 Vektorquantisierung des obigen Ausschnittes,
Vektorlänge : 8 Samples/Vektor,
Codebuchgröße: 16

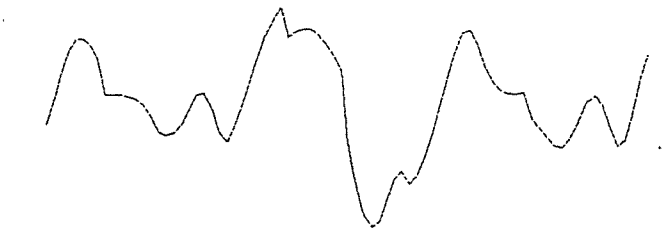

Bild 1.1, 4 Vektorquantisierung des obigen Ausschnittes,
Vektorlänge : 8 Samples/Vektor,
Codebuchgröße: 1024

Allgemein läßt sich über die Vektorquantisierung folgendes sagen, wenn man anstelle von Abtastwertfolgen beliebig vorgegebene Muster quantisieren will:

> Das Grundproblem der Vektorquantisierung ist,
> mit möglichst wenigen repräsentativen Mustern
> gegebene Muster möglichst gut anzunähern.

Dazu müssen optimale Codebücher vorhanden sein.

1.2 Vektorquantisierung in Baumstruktur

Im Codierteil des Vektorquantisierersystems wird einem gegebenen Vektor $\vec{X}_i$ derjenige Codebuchvektor $\vec{Y}_j$ zugeordnet, der ihn am besten annähert:

$$\vec{Y}_j = VQ(\vec{X}_i) \qquad (1.2, 1)$$

Dazu muß der Abstand der Vektoren $\vec{X}_i$ und $\vec{Y}_j$

$$d(\vec{X}_i, \vec{Y}_j) = \left| \vec{X}_i - \vec{Y}_j \right| = \sqrt{\sum_{l=1}^{k} (x_{il} - y_{jl})^2} \qquad (1.2, 2)$$

minimal sein:

$$d(\vec{X}_i, \vec{Y}_j) \leq d(\vec{X}_i, \vec{Y}_n), \; n = 1, 2, 3, \ldots, N \qquad (1.2, 3)$$

Der Repräsentant $\vec{Y}_j$ ist erst dann bekannt, wenn alle N Abstände berechnet und verglichen wurden.

Der Rechenaufwand bei der Quantisierung kann stark verringert werden, wenn nicht alle N Abstände berechnet werden müssen (Full-Search, FS), sondern ein gezielter Suchbaum verfolgt wird (Tree-Search, TS). Bild 1.2, 1 zeigt einen möglichen Baum für ein Codebuch der Größe 16. Auf der ersten Knotenebene müssen vier Abstände berechnet werden, auf der zweiten ebenfalls vier. Insgesamt werden hier nur acht Abstände errechnet, halb so viele wie bei einem Full-Search-Codebuch gleicher Größe.

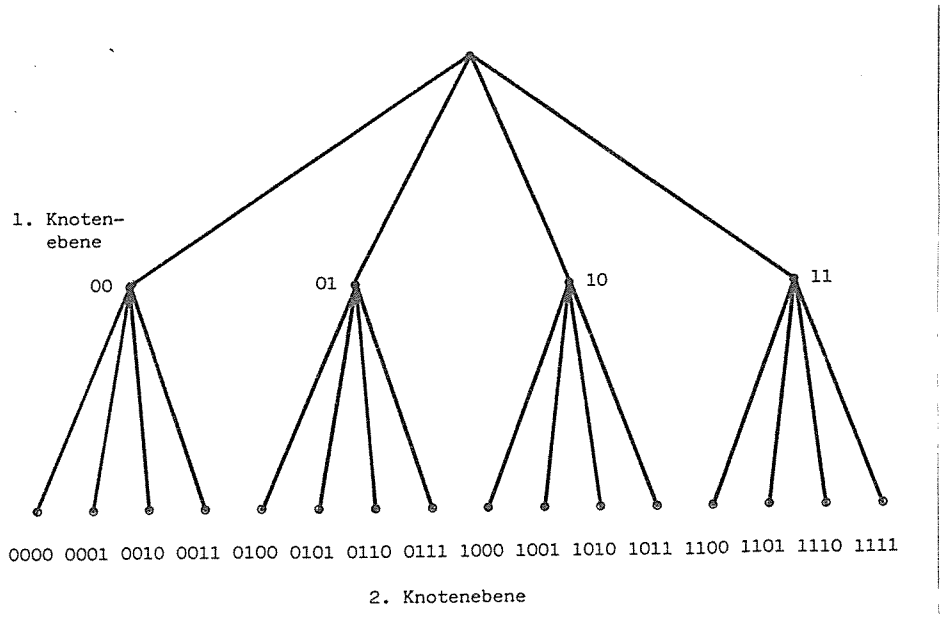

Bild 1.2, 1 Eine mögliche Baumstruktur
für eine Codebuchgröße von 16

Die Zahl der möglichen Baumstrukturen ist sehr hoch. Ich habe mich auf solche Bäume beschränkt, bei denen sich alle Knoten einer Knotenebene in gleich viele Knoten aufspalten und die Zahl der Aufspaltungen eine Zweierpotenz ist. Mit diesen Nebenbedingungen lassen sich alle Baumstrukturen mit m Knotenebenen leicht beschreiben: N_i' sei die Zahl der Aufspaltungen eines Knotens der Ebene (i-1). Dann ist

$$R_i' = \mathrm{ld}\ N_i'\ \mathrm{Bit},\ i = 1, 2, 3, \ldots, m \qquad (1.2, 4)$$

die Zahl der binären Entscheidungen, die getroffen werden müssen, um von einem Knoten der Ebene (i-1) zu einem bestimmten Knoten der Ebene i zu gelangen. Die Ebene i = 0 ist der Ursprungsknoten. Der Vektor

$$\vec{B} = (R_1', R_2', R_3', \ldots, R_m') \qquad (1.2, 5)$$

beschreibt jede zugelassene Baumstruktur eindeutig. Beispielsweise hat der Baum von Bild 1.2, 1 die Struktur $\vec{B} = (2, 2)$.

Insgesamt müssen

$$R_i = \sum_{j=1}^{i} R'_j, \quad i = 1, 2, 3, \ldots, m \qquad (1.2, 6)$$

binäre Entscheidungen getroffen werden, um auf die i-te Ebene zu gelangen. Die Knotenebene i umfaßt

$$N_i = 2^{R_i}, \quad i = 1, 2, 3, \ldots, m \qquad (1.2, 7)$$

Knoten. Die letzte Ebene i = m gibt die Codebuchgröße an:

$$N = N_m \qquad (1.2, 8)$$

Es werden insgesamt

$$R = R_m \qquad (1.2, 9)$$

binäre Entscheidungen getroffen. Alle Vektoren der Ebene i werden zu dem Codebuch

$$\vec{A}_i = (\vec{Y}_1, \vec{Y}_2, \vec{Y}_3, \ldots, \vec{Y}_{N_i}), \quad i = 1, 2, 3, \ldots, m \qquad (1.2, 10)$$

zusammengefaßt. Die Codebücher aller Ebenen sind im baumstrukturierten Codebuch

$$\vec{C} = (\vec{A}_1, \vec{A}_2, \vec{A}_3, \ldots, \vec{A}_m) \qquad (1.2, 11)$$

enthalten.

Nach diesen Definitionen kann nun allgemein die Anzahl der zu berechnenden Abstände bestimmt werden:

$$n_d = \sum_{i=1}^{m} N_i' \qquad (1.2, 12)$$

Im Codebuchspeicher sind

$$M = \sum_{i=1}^{m} N_i \qquad (1.2, 13)$$

Vektoren enthalten.

Die Bilder 1.2, 2 und 1.2, 3 zeigen ein Codebuch mit der Baumstruktur von Bild 1.2, 1. Die Vektoren sind der Reihe nach mit aufsteigenden Nummern von links nach rechts und von oben nach unten sortiert. Auf der ersten Ebene werden die Sprachsignalvektoren grob vorsortiert, danach erfolgt die Bestimmung des Repräsentanten. Soll beispielsweise der Vektor

$$\vec{X}_1 = (0, 0, 0, 0, 0, 0, 0, 0)$$

quantisiert werden, wird zuerst der passendste Vektor der ersten Ebene ausgesucht. Der linke untere Vektor $\vec{Y}_3$ in Bild 1.2, 2 ist der gesuchte und entspricht in Bild 1.2, 1 dem Knoten "10". Danach stehen in Bild 1.2, 3 alle Vektoren der dritten Zeile zur Auswahl. $\vec{Y}_{10}$ ist der gesuchte optimale Vektor (der zweite in der dritten Zeile), der dem Knoten "1001" entspricht.

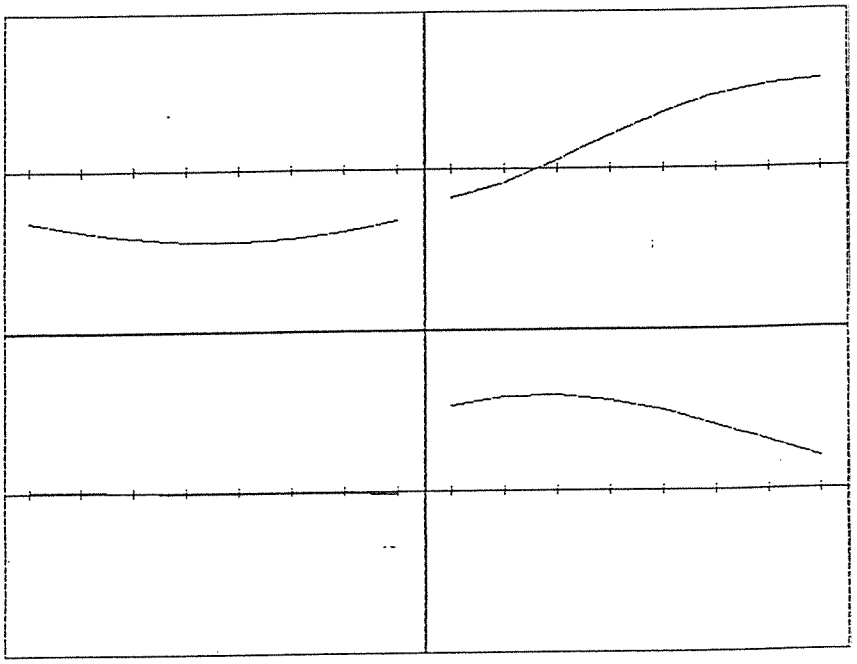

Bild 1.2, 2 Baumstrukturiertes Codebuch der Größe 16, Baumstruktur von Bild 1.2, 1, erste Knotenebene

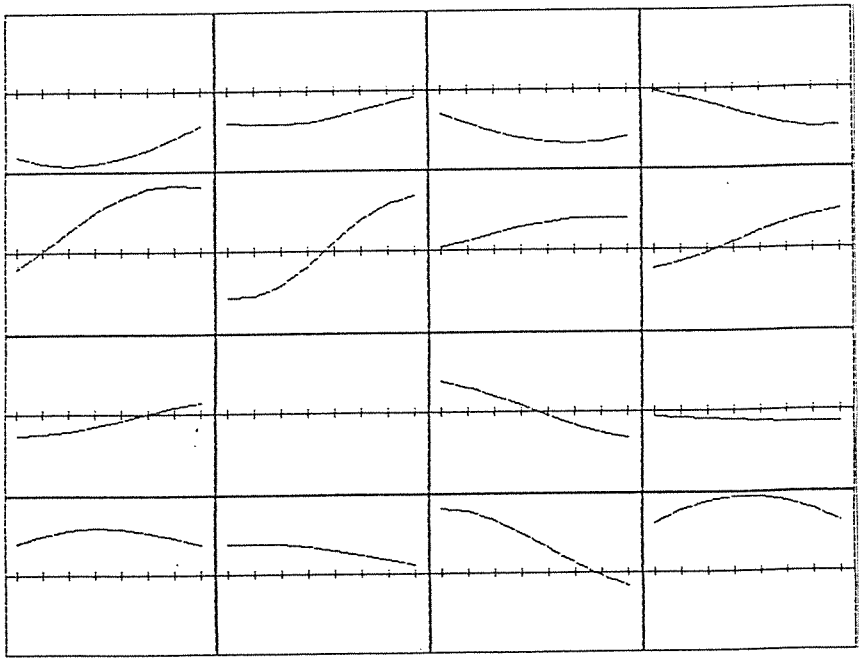

Bild 1.2, 3 Baumstrukturiertes Codebuch der Größe 16, Baumstruktur von Bild 1.2, 1, zweite Knotenebene

1.3 Anwendungen der Vektorquantisierung

Bestehende digitale Telefon-Übertragungssysteme könnten besser genutzt werden, wenn vektorquantisierende Codierer und Decodierer vor- und nachgeschaltet würden. Dadurch wird weniger redundante Information übertragen. Allerdings ist es unwahrscheinlich, daß sich reine Vektorquantisierer durchsetzen. Der technische Aufwand für einen akzeptablen Störabstand ist wegen der umfangreichen Codebücher zu groß. Momentan scheint eine Kombination von linearer Prädiktion und Vektorquantisierung (RELP-VQ-Verfahren) ein vernünftiger Kompromiß zwischen erzielter Redundanzreduktion, technischem Aufwand und Übertragungsgüte zu sein /4/.

Durch die Vektorquantisierung wird dem Sprachsignal ein Störsignal überlagert, das im wesentlichen weißes Rauschen ist. Das Spektrum der Sprache dagegen ist gefärbt. Störungen in energiereichen Frequenzbereichen fallen viel weniger auf als in energiearmen. Dieser psychoakustische Effekt der Überdeckung wird beim RELP-VQ-Verfahren ausgenutzt, indem die Störenergie spektral so umverteilt wird, daß sie unter energiereichen Komponenten des Sprachsignals liegt und somit weitgehend verdeckt wird. Dadurch benötigt man für die gleiche Übertragungsgüte kleinere Codebücher als bei reiner Vektorquantisierung.

Bei der linearen Prädiktion werden blockweise die Prädiktionskoeffizienten ermittelt, indem die Differenz zwischen dem Sprach- und dem vorhergesagten Signal, das sogenannte RESIDUALSIGNAL, minimiert wird. Seine spektrale Energieverteilung ist im wesentlichen weiß. Im Decodierer wird aus den Prädiktionskoeffizienten und dem Residualsignal das ursprüngliche Signal wiedergewonnen.

Beim RELP-VQ-Verfahren werden die Prädiktionskoeffizienten und das Residualsignal vektorquantisiert übertragen. Dem Residualsignal wird dadurch ein Störsignal überlagert. Da die Spektren beider Signale weiß sind, werden alle Frequenzanteile gleichmäßig gestört. Nach dem Decodieren ist in energiereichen Frequenzbereichen auch die größte Störenergie.

Auch auf anderen Gebieten ist die Einführung der Vektorquantisierung denkbar. Beispielsweise könnte beim digitalen Fernsehen das Bild aus Grundmustern zusammengesetzt werden, wobei hier zweidimensionale Zahlenfelder (Matrizen) quantisiert werden müßten.

1.4 Vorangegangene Untersuchungen

Gray und Linde haben die Vektorquantisierung in Baumstruktur mit Signalen im Zeitbereich untersucht /1/. Allerdings verwendeten sie für ihre Experimente ein Trainingssignal aus einer Gauss-Markov-Quelle und eine höhere Abtastfrequenz als 8 kHz, so daß ein Vergleich der Ergebnisse nicht ohne weiteres möglich ist. Eine systematische Untersuchung der möglichen Baumstrukturen wurde nicht durchgeführt.

Wong, Juang und Gray dagegen haben einige Baumstrukturen untersucht /3/. Allerdings quantisierten sie Spektren. Abtastwertfolgen haben andere statistische Verteilungen im k-dimensionalen Raum als entsprechende Spektren. Deshalb unterscheiden sich die Untersuchungsergebnisse.

Halaski und Fasshauer untersuchten Full-Search-Codebücher /4/. Ein Vergleich ihrer Ergebnisse mit meinen eigenen ist möglich, da die Nebenbedingungen der Experimente (Abtastrate und Trainingssignal) übereinstimmen.

2 Algorithmen zur Erzeugung von Codebüchern

Bei der Vektorquantisierung werden optimale Codebücher benötigt. Es sind einige Algorithmen bekannt, mit denen nach Vorgabe der Vektorlänge k und der Codebuchgröße N aus gegebenen Trainingsvektoren $\vec{x}_1$, $\vec{x}_2$, $\vec{x}_3$,..., $\vec{x}_L$, die nach (1.1, 1) einer Folge von Samples x_1, x_2, x_3,..., x_{kL} entsprechen, Codebücher berechnet werden können. Die Algorithmen unterscheiden sich im Programmieraufwand, der benötigten Rechenzeit und der Güte der erzeugten Codebücher.

Die Güte eines Codebuchs wird durch das Signal/Rauschverhältnis SNR bestimmt. Dazu muß vorher der Gleichspannungsanteil

$$\bar{x} = \frac{1}{kL} \sum_{i=1}^{kL} x_i \qquad (2, 1)$$

und die Energie des Signals bekannt sein:

$$E_{sig} = \sum_{i=1}^{kL} (x_i - \bar{x})^2 \qquad (2, 2)$$

Nach Vergleich mit (1.2, 2) läßt sich die Signalenergie mit $\vec{X}_{=} = (\bar{x}, \bar{x}, \bar{x}, \ldots, \bar{x})$ auch noch anders ausdrücken:

$$E_{sig} = \sum_{i=1}^{L} d^2(\vec{X}_i, \vec{X}_{=}) \qquad (2, 3)$$

Zur Berechnung der Störenergie E_{dif} muß statt dem Gleichspannungsanteil das quantisierte Signal subtrahiert werden. Analog zu (2, 2) und (2, 3) ergibt sich mit (1.2, 1):

$$E_{dif} = \sum_{i=1}^{L} d^2(\vec{X}_i, VQ(\vec{X}_i)) \qquad (2, 4)$$

Das Signal/Rauschverhältnis ist hiernach bekannt:

$$SNR = 10 \lg \frac{E_{sig}}{E_{dif}} \text{ dB} \qquad (2, 5)$$

Ist ein Algorithmus zur Erzeugung von Full-Search-Codebüchern gegeben, kann nach dem Schema von Bild 2, 1 ein Tree-Search-Codebuch berechnet werden, wenn die Baumstruktur $\vec{B}$ vorgegeben ist.

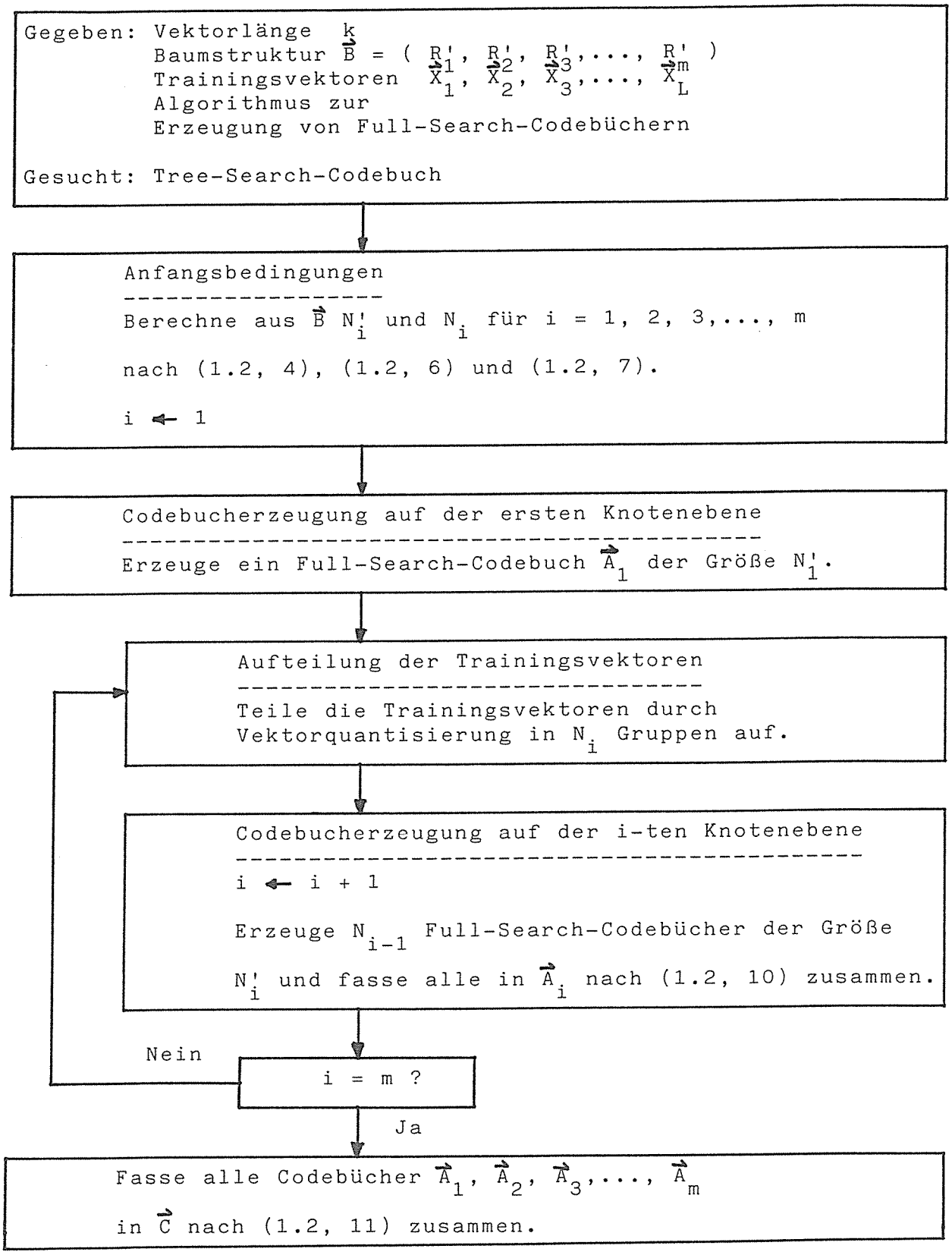

Bild 2, 1 Algorithmus zur Erzeugung eines baumstrukturierten Codebuchs

2.1 Braun/Lochschmidt-Algorithmus*)

Halaski und Fasshauer benutzten in ihrer Arbeit folgenden Algorithmus, der Full-Search-Codebücher in zwei Stufen berechnet /4/: Zuerst wird aus den Trainingsdaten eine Anfangsnäherung ermittelt, die dann durch ein Iterationsverfahren schrittweise verbessert wird.

Gegeben sind die Vektorlänge k, die Codebuchgröße N und die Trainingsvektoren $\vec{X}_1, \vec{X}_2, \vec{X}_3, \ldots, \vec{X}_L$.

1. Schritt: Der erste Codebucheintrag ist der Schwerpunktsvektor:

$$\vec{Y}_1 = \frac{1}{L} \sum_{i=1}^{L} \vec{X}_i \qquad (2.1, 1)$$

2. Schritt: Danach wird derjenige Trainingsvektor gesucht, der den größten Abstand zu $\vec{Y}_1$ hat. Er wird an die zweite Stelle des Codebuchs geschrieben.

3. Schritt: Das Trainingssignal wird mit dem vorhandenen Codebuch vektorquantisiert.

4. Schritt: Derjenige Trainingsvektor mit dem größten auftretenden Abstand ist der nächste Codebucheintrag.

Die Schritte 3 und 4 werden solange wiederholt, bis das Codebuch die gewünschte Größe hat.

Das Iterationsverfahren geht vom vollständigen Codebuch der Größe N aus. Wie aus Bild 2.1, 1 ersichtlich ist, werden abwechselnd die Trainingsvektoren quantisiert und die Codebuchvektoren zentriert. Dadurch wandern die Codebuchvektoren in die Schwerpunkte der Cluster. Die Iteration bricht dann ab, wenn die relative Senkung der Störenergie unter einem Schwellwert ε liegt.

*) erstmals von Braun/Lochschmidt (FTZ) verwendet.

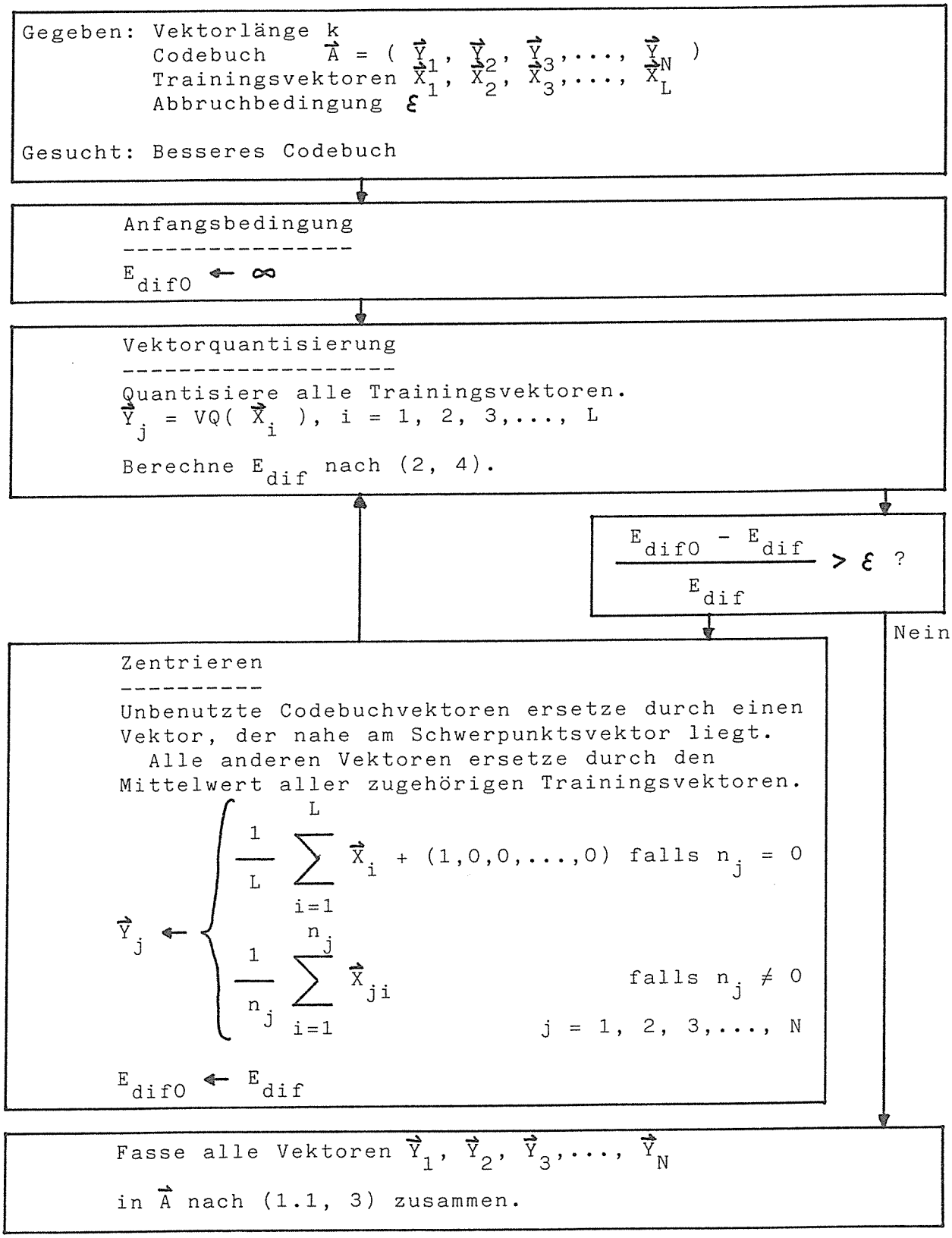

Bild 2.1, 1 Iterationsverfahren

2.2 Linde/Buzo/Gray-Algorithmus

Gray und Linde geben einen Algorithmus an /1/, der sehr einfach zu programmieren ist. Ausgehend vom Schwerpunktsvektor der Trainingssequenz wird abwechselnd das Codebuch verdoppelt und nach dem in Bild 2.1, 1 erläuterten Verfahren iteriert (siehe Bild 2.2, 1). Die Codebuchgröße wird dadurch verdoppelt, daß alle vorhandenen Codebuchvektoren in jeweils zwei dicht beieinanderliegende aufgespalten werden. Dadurch sind große Codebücher sehr schnell generiert. Die Größe N des Full-Search-Codebuchs ist stets eine Zweierpotenz.

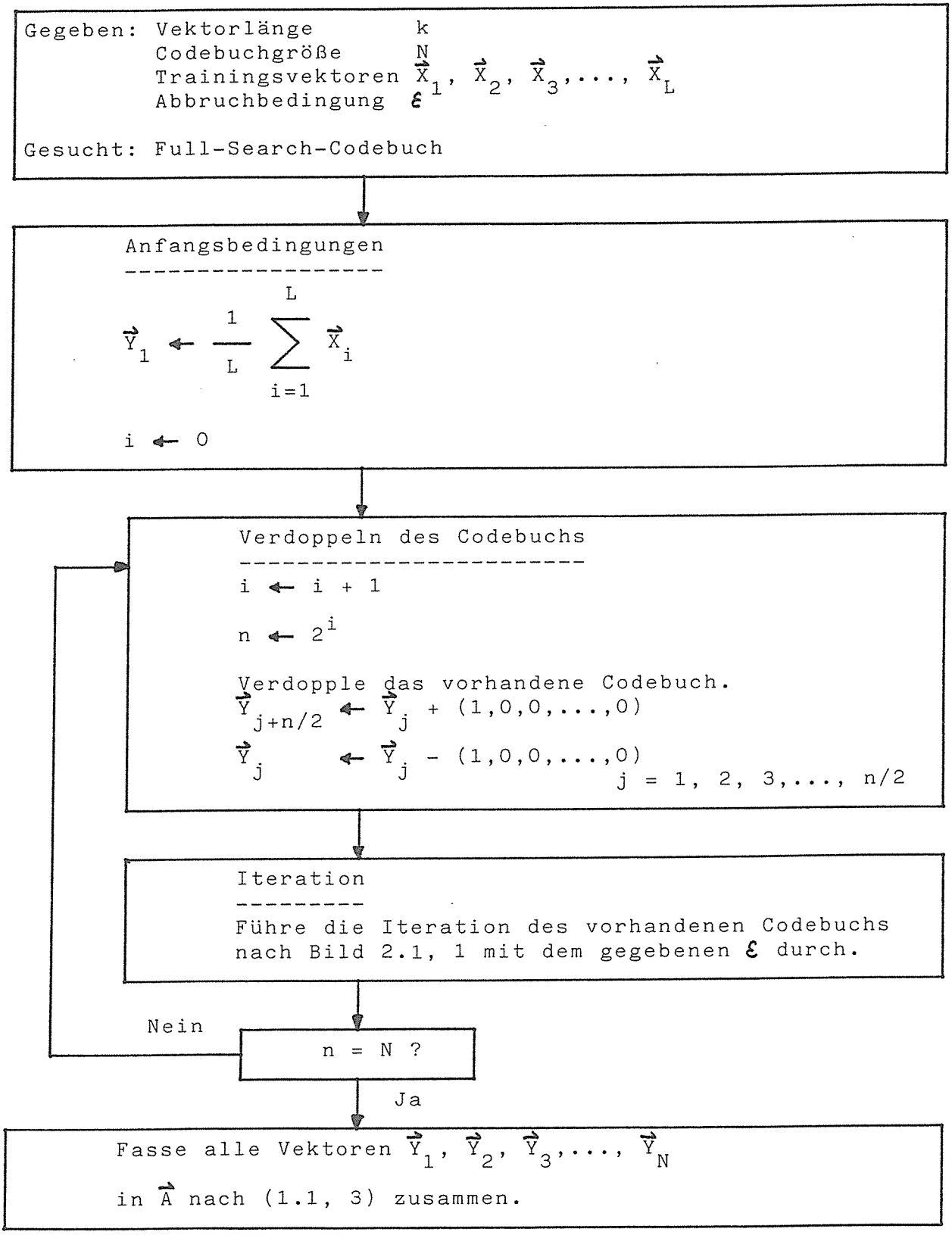

Bild 2.2, 1 Linde/Buzo/Gray-Algorithmus

3 Experimente

Für die Untersuchungen stand mir eine VAX 11/780, ein schneller Rechner mit großer Speicherkapazität zur Verfügung.

Als Trainingssignal wurde eine Sprachprobe benutzt, die einen Satz enthält, dessen Häufigkeitsverteilung der Laute mit derjenigen der deutschen Sprache übereinstimmt. Der Satz lautet:

"Hallo, hören Sie? Ich bin Rudolf Ranick hier, vom FTZ. Prüfen Sie bei Kurt Meyer in der Burgstraße den Leitungsanschluß und auch das Geräusch aus den Kapseln!"

Dieser Satz wurde von zwei männlichen und zwei weiblichen Sprechern über Telefon aufgenommen. Die Länge der Sprachprobe beträgt etwa 54 Sekunden. Dies entspricht 108.000 Vektoren der Länge k = 4 und reicht für die Untersuchungen völlig aus, da optimale Baumstrukturen gesucht sind. Für optimale Codebücher dagegen müßte das Trainingssignal ungefähr eine Dauer von 30 Minuten haben. Der dadurch entstehende Aufwand an Speicherkapazität und Rechenzeit ist selbstverständlich erst dann tragbar, wenn die optimale Baumstruktur bekannt ist.

Das Programm "Codebuchsuche" (siehe 8.1, Programm COBUS) verwendet den Linde/Buzo/Gray-Algorithmus (siehe 2.2) und erzeugt Codebücher mit beliebig vorgebbarer Baumstruktur (siehe Bild 2, 1). Die maximal zugelassene Codebuchgröße ist N = 1024, weil umfangreichere Codebücher einen nicht mehr tragbaren Aufwand an Speicherkapazität in Vektorquantisierersystemen erfordern. Damit ist auch die maximale Vektorlänge mit k = 8 festgelegt, da größere Vektorlängen umfangreichere Codebücher als zugelassen erzwingen.

Die von COBUS erzeugten Codebücher werden unter dem Filenamen CBUCH.DAT abgelegt. In die ersten vier Zeilen werden die Vektorlänge k, die Baumstruktur $\vec{B}$, der verwendete Algorithmus und das errechnete Signal/Rauschverhältnis SNR geschrieben.

k : 8
$\vec{B}$: 2 2 0 0 0 0 0 0 0 0
Algorithmus: 1
SNR : 8.0214

Bild 3, 1 Kopf eines Codebuchs

Das Beispiel in Bild 3, 1 zeigt den Kopf des Codebuchs der Bilder 1.2, 2 und 1.2, 3. Das Signal/Rauschverhältnis beträgt rund SNR = 8 dB. Die "1" gibt an, daß der Linde/Buzo/Gray-Algorithmus verwendet wurde. Eine Abwandlung dieses Algorithmus, die in Kapitel 3.4 besprochen wird, wird mit einer "2" gekennzeichnet.

3.1 Voruntersuchungen

Bei gegebenem ε bricht die Iteration dann ab, wenn sich das Signal/Rauschverhältnis pro Iterationsschritt um weniger als

$$\Delta \text{ SNR} = 10 \lg (1 + \varepsilon) \text{ dB} \qquad (3.1, 1)$$

verbessert (siehe Bild 2.1, 1). Das Codebuch wird umso besser, je kleiner ε gewählt wird. Allerdings dauert dadurch die Berechnung länger. Nach einigen Tests wurde $\varepsilon = 10^{-3}$ gewählt. Das Signal/Rauschverhältnis verbessert sich damit höchstens noch in der zweiten Nachkommastelle (Δ SNR = 0,00434 dB), was an einem Full-Search-Codebuch gezeigt werden konnte. Dagegen konnte an einem Binary-Tree-Search-Codebuch (BTS-Codebuch, Anzahl der Knoten verdoppelt sich mit jeder Knotenebene: $\vec{B}$ = (1, 1, 1,..., 1)) folgende Unregelmäßigkeit nachgewiesen werden, die in Kapitel 3.4 noch genauer untersucht wird: Das Signal/Rauschverhältnis wird mit kleiner werdendem ε geringfügig schlechter.

Zuerst wurden für Vektorlängen k = 1, 2, 3,..., 8 alle Full-Search- und Binary-Tree-Search-Codebücher mit R = k berechnet (siehe 8.2, "Vektorlänge 1-8, 1 Bit/Sample"). Die Datenrate

beträgt nach (1.1, 5) r = 1 Bit/Sample. Wie man in Bild 3.1, 1 erkennt, werden bei konstanter Datenrate mit größeren Vektorlängen bessere Ergebnisse erzielt. Full-Search-Codebücher sind um etwa 1 dB besser als Binary-Tree-Search-Codebücher gleicher Größe.

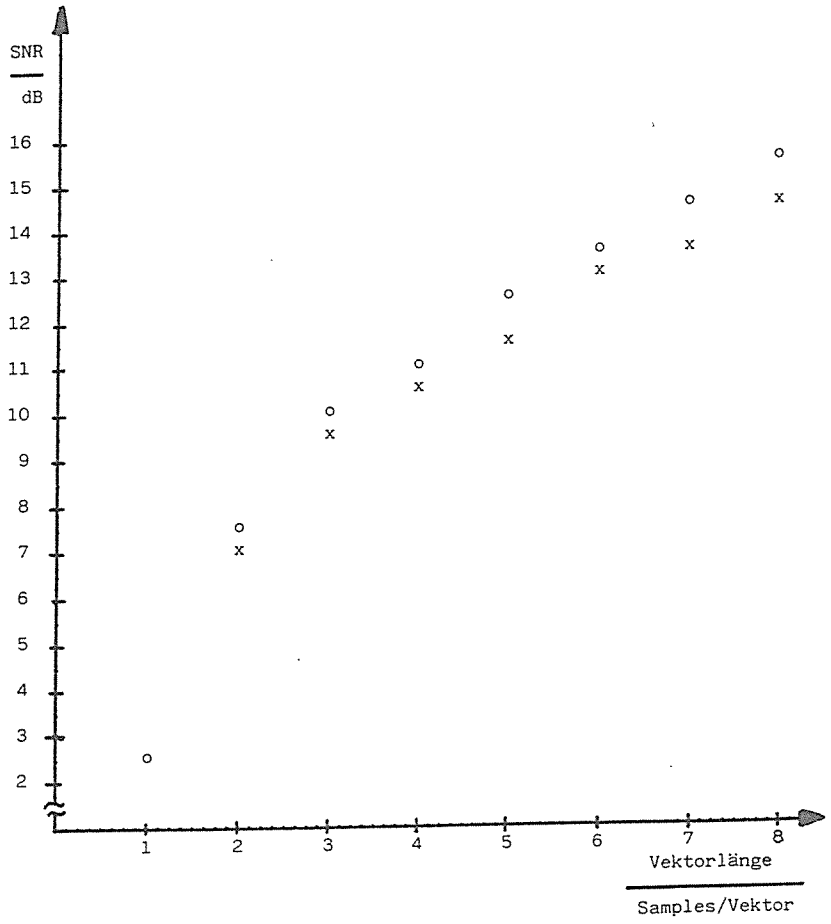

Bild 3.1, 1 Signal/Rauschverhältnis in Abhängigkeit von der Vektorlänge,
Datenrate: 1 Bit/Sample,
o FS-Codebuch
x BTS-Codebuch

Mit einer Datenrate von r = 2 Bit/Sample wurden für k = 1, 2, 3, 4 entsprechende Untersuchungen durchgeführt (siehe 8.2, "Vektorlänge 1-4, 2 Bit/Sample"). Obige Feststellungen bestätigten sich (siehe Bild 3.1, 2).

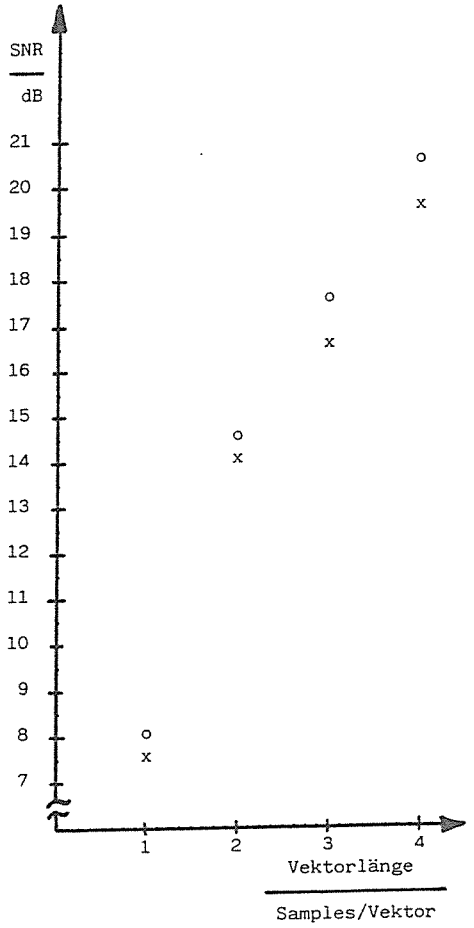

Bild 3.1, 2 Signal/Rauschverhältnis in Abhängigkeit von der Vektorlänge,
Datenrate: 2 Bit/Sample,
o FS-Codebuch
x BTS-Codebuch

3.2 Untersuchung aller 16er Codebücher

Für die Codebuchgröße N = 16 existieren genau acht mögliche Baumstrukturen:

$$\vec{B}_1 = (4)$$
$$\vec{B}_2 = (3, 1)$$
$$\vec{B}_3 = (2, 2)$$
$$\vec{B}_4 = (2, 1, 1)$$
$$\vec{B}_5 = (1, 3)$$
$$\vec{B}_6 = (1, 2, 1)$$
$$\vec{B}_7 = (1, 1, 2)$$
$$\vec{B}_8 = (1, 1, 1, 1)$$

Nach (1.2, 8), (1.2, 7), (1.2, 6) und (1.2, 9) muß die Summe aller Baumstrukturkomponenten R = 4 ergeben. Es gibt acht Möglichkeiten, diese Bedingung mit positiven natürlichen Zahlen zu erfüllen. Man kann zeigen, daß die Zahl der möglichen Baumstrukturen

$$n_B = \frac{N}{2} \qquad (3.2, 1)$$

ist, wenn N eine Zweierpotenz ist.

Weil die Hardware-Realisierung für Vektorlängen, die Zweierpotenzen sind, besonders einfach ist, wurden alle 16er Codebücher für k = 4 und k = 8 erzeugt. Für beide Vektorlängen ergeben sich unterschiedliche Rangfolgen der Baumstrukturen (siehe 8.2, "Alle Bäume"). Trotzdem läßt sich über "regelmäßige" Baumstrukturen, für die

$$R'_1 = R'_2 = R'_3 = \ldots = R'_m = R', \text{ mit } mR' = R \qquad (3.2, 2)$$

gilt, folgendes sagen: Das Full-Search-Codebuch ist jeweils am besten ($\vec{B}$ = (4)). Es folgt der quaternäre Baum ($\vec{B}$ = (2, 2)).

Das Binary-Tree-Search-Codebuch ist das schlechteste ($\vec{B}$ = (1, 1, 1, 1)). Offenbar ist das Signal/Rauschverhältnis umso besser, je weniger Knotenebenen m die regelmäßige Baumstruktur hat.

Wenn die Anzahl der zu berechnenden Abstände n_d minimal sein soll, ist für einen Baum mit m Ebenen die regelmäßige Struktur die beste. Dies gilt bis auf eine Ausnahme: Gemischt binär-quaternäre Baumstrukturen (z. B. $\vec{B}$ = (1, 2, 1,..., 2)) haben nach (1.2, 12) den gleichen Suchaufwand wie binäre und quaternäre Bäume:

$$n_d = 2 \text{ ld } N \qquad (3.2, 3)$$

3.3 Untersuchung größerer Codebücher

Im Folgenden werden nur Codebücher mit Vektorlänge k = 8 untersucht, da die Rangfolge der regelmäßigen Baumstrukturen offensichtlich unabhängig von der Vektorlänge ist.

Für die Codebuchgröße N = 1024 existieren nach (3.2, 1) n_B = 512 Baumstrukturen. Es ist unsinnig, alle zu untersuchen, da nur Strukturen mit minimalem Suchaufwand interessieren.

Es wurden alle 32er, 64er, 128er, 256er, 512er und 1024er Codebücher mit regelmäßiger Baumstruktur nach (3.2, 2) erzeugt. Außerdem wurden noch Baumstrukturen zugelassen, deren Komponenten sich um 1, höchstens um 2 unterscheiden (siehe 8.2, "Ausgewählte Bäume"). Die regelmäßigen Strukturen sind wieder nach der Anzahl ihrer Knotenebenen sortiert. Bäume, die in den ersten Ebenen eine binäre, in den letzten eine quaternäre Struktur haben, sind günstig.

Das 512er und das 1024er Full-Search-Codebuch wurde nicht generiert, weil die Berechnung zu viel Rechenzeit gekostet und keine neuen Erkenntnisse gebracht hätte.

3.4 Variation des Algorithmus

Ein Code ist genau dann redundanzfrei, wenn alle Symbole mit gleicher Häufigkeit auftreten. Codebücher, die mit dem Linde/Buzo/Gray-Algorithmus erzeugt sind, erfüllen diese Bedingung nicht. Bei einer binären Baumstruktur sollten auf der ersten Knotenebene die Trainingsvektoren nach positivem und negativem Mittelwert geordnet sein. Nach dem ersten Iterationsschritt ist dies auch der Fall. Wenn aber die Verteilung der Trainingsvektoren nicht exakt symmetrisch ist, weil beispielsweise der Mittelwert der positiven Vektoren größer ist als derjenige der negativen, werden im nächsten Iterationsschritt Trainingsvektoren mit kleinem positiven Mittelwert dem negativen Cluster zugeordnet. Dadurch werden die Trainingsvektoren ungleichmäßig aufgeteilt.

Das Programm COBUS enthält eine Abwandlung des Linde/Buzo/Gray-Algorithmus (siehe 8.1, Programm COBUS). Bei Baumstrukturen mit vielen Knotenebenen wird die Iteration schon nach dem ersten Schritt abgebrochen. Dadurch werden die Trainingsvektoren gleichmäßig aufgeteilt. Lediglich die letzten beiden Knotenebenen werden voll durchiteriert.

Bei kleinen Codebüchern läßt sich damit eine leichte Verbesserung des Signal/Rauschverhältnisses in der Größenordnung von 0,1 dB erzielen (siehe 8.2, "Neue Strategie"). Im Mittel sind die Ergebnisse aber schlechter. Wahrscheinlich wäre eine stetige Änderung des Abbruchkriteriums in Abhängigkeit von der Knotenebene $\varepsilon = \varepsilon(m)$ vorteilhaft.

Die bei den Voruntersuchungen (siehe 3.1) auftretende Anomalie kann damit erklärt werden, daß bei zu kleinem ε auf den ersten Knotenebenen die Trainingsvektoren ungleichmäßig aufgeteilt werden. Die daraus resultierenden Codebücher sind schlechter als solche mit etwas größerem ε.

Die mit dem Braun/Lochschmidt-Algorithmus erzielten Ergebnisse /4/ sind im Mittel schlechter als diejenigen des Linde/Buzo/Gray-Algorithmus, so daß es sich erübrigt, Baumstrukturen mit dem anderen Algorithmus zu untersuchen.

3.5 Optimale Bäume

Nach der Erzeugung aller Codebücher mit der Vektorlänge k = 4, die entweder eine regelmäßige oder eine binär-quaternäre Baumstruktur haben, läßt sich folgendes über optimale Bäume sagen, wenn die Zahl der zu berechnenden Abstände n_d möglichst klein sein soll:

> Baumstrukturen, bei denen sich jeder Knoten in vier weitere Knoten aufspaltet (quaternäre Bäume), sind optimal. Die Codebuchgröße N muß allerdings eine Potenz von vier sein. Ist sie eine ungerade Potenz von zwei, dann ist derjenige Baum der beste, dessen Ursprungsknoten sich zweifach, alle anderen Knoten sich vierfach aufspalten (binär-quaternärer Baum). Bei beiden Baumstrukturen müssen $n_d = 2 \text{ ld } N$ Abstände berechnet werden.

Codebücher mit den optimalen Baumstrukturen sind um etwa 0,5 dB schlechter als die zugehörigen Full-Search-Codebücher (siehe 8.2, "Optimale Bäume").

4 Untersuchungen mit Residualsignalen

Bei den weiteren Experimenten wurde das Residualsignal der in Kapitel 3 beschriebenen Sprachprobe als Trainingssequenz benutzt. Die Ergebnisse der Baumuntersuchungen stimmen prinzipiell mit den bereits diskutierten überein (siehe 8.2, "Residual-Codebücher"). Da das Residualsignal eine geringere Energie hat als das Originalsignal, die Störenergien aber in der gleichen Größenordnung liegen, sind die Signal/Rauschverhältnisse der Residual-Codebücher deutlich schlechter. Trotzdem schneidet das RELP-VQ-Verfahren im Vergleich mit der reinen Vektorquantisierung besser ab. Das "objektive" Signal/Rauschverhältnis erfaßt den Überdeckungseffekt /6/ nicht. Die beiden Verfahren müssen in akustischen Verständlichkeitstests (z. B. Logatomtest, siehe /5/) verglichen werden.

In den Bildern 4, 1 und 4, 2 sind für die konstanten Vektorlängen k = 4 und k = 8 jeweils die Signal/Rauschverhältnisse der Full-Search- und der Binary-Tree-Search-Codebücher in Abhängigkeit von der Datenrate aufgetragen. Die oberen Meßreihen zeigen die Ergebnisse der Original-, die unteren jene der Residual-Codebücher. Wie erwartet, sind mit einer höheren Datenrate, d. h. mit einem größeren Codebuch, bessere Ergebnisse zu erzielen.

Mit dem Programm "Tree-Search-Vector-Quantizer" (siehe 8.1, Programm TSV) kann die Vektorquantisierung einer gegebenen Sprachprobe mit dem Codebuch CBUCH.DAT durchgeführt werden. Das berechnete Signal/Rauschverhältnis wird ausgegeben.

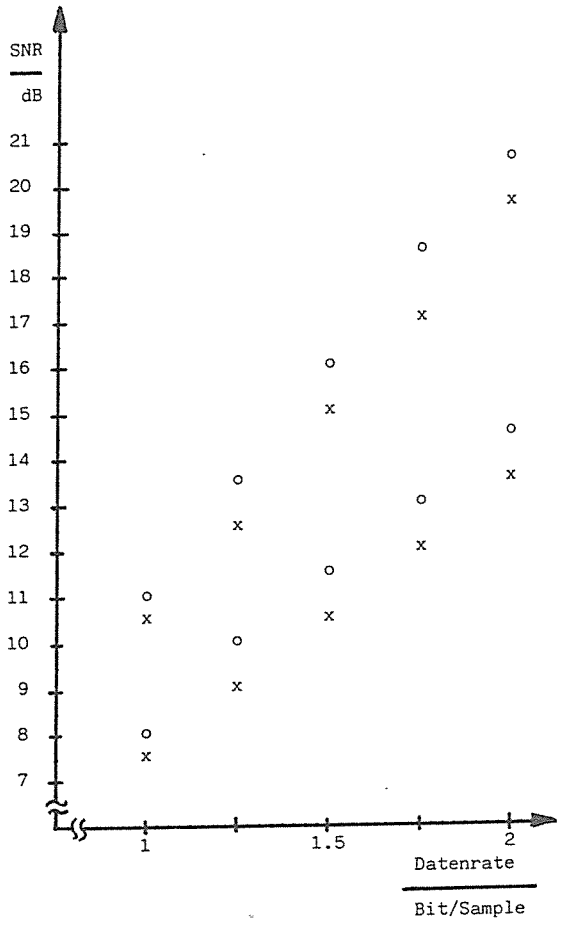

Bild 4, 1 Signal/Rauschverhältnis in Abhängigkeit von der Datenrate,
Vektorlänge : 4 Samples/Vektor,
obere Meßreihe : Original-Codebücher
untere Meßreihe: Residual-Codebücher
o FS-Codebuch
x BTS-Codebuch

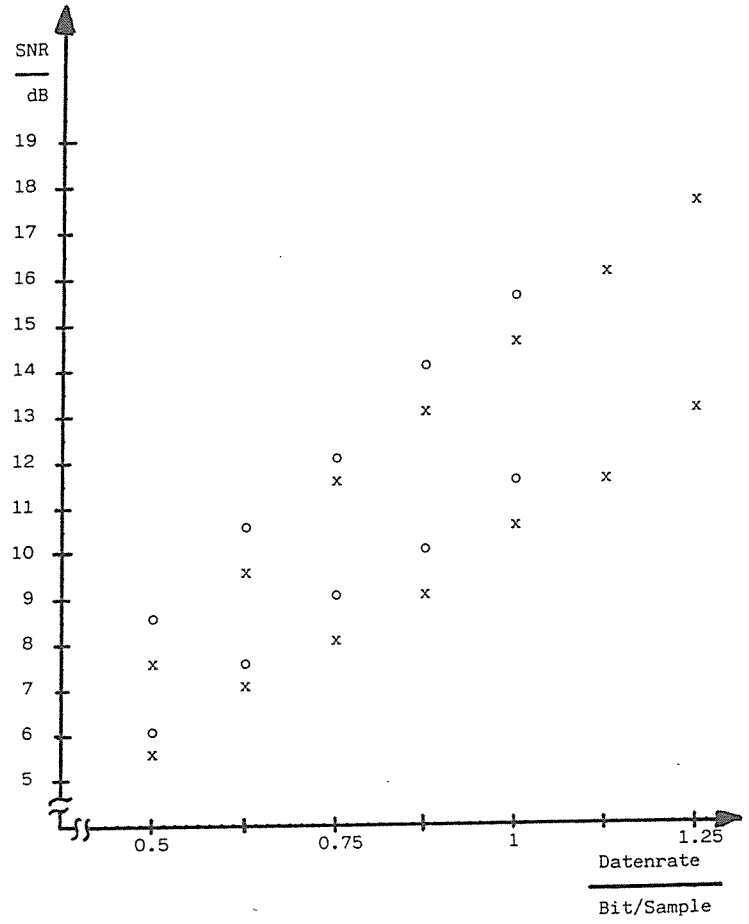

Bild 4, 2 Signal/Rauschverhältnis in Abhängigkeit von der Datenrate,
Vektorlänge : 8 Samples/Vektor,
obere Meßreihe : Original-Codebücher
untere Meßreihe: Residual-Codebücher
o FS-Codebuch
x BTS-Codebuch

5 Aufwandsbetrachtungen

Die Suche des kleinsten Abstandes $d(\vec{X}_i, \vec{Y}_j)$ läßt sich auf folgende Weise vereinfachen: Da die Wurzelfunktion monoton ist, reicht es aus, $d^2(\vec{X}_i, \vec{Y}_j)$ nach (1.2, 2) zu berechnen:

$$d^2(\vec{X}_i, \vec{Y}_j) = \sum_{l=1}^{k} (x_{il} - y_{jl})^2 = K_1 - 2 \sum_{l=1}^{k} x_{il} y_{jl} + K_2 \quad (5, 1)$$

K_1 ist für einen gegebenen Sprachsignalvektor bei allen zu berechnenden Abständen eine Konstante. Bei der Suche nach dem kleinsten Abstand kann sie wegfallen.

K_2 ist die Energie des jeweiligen Codebuchvektors. Sie kann vor der Quantisierung berechnet und abgespeichert werden.

Pro Abstandsberechnung müssen also k Multiplikationen und k Additionen durchgeführt werden, wenn statt der Multiplikation der Summe mit dem Faktor 2 der Ausdruck $K_2/2$ abgespeichert wird. Anschließend folgt ein Vergleich mit dem bisher optimalen Abstand. Zur Aufwandsberechnung wird der Vergleich zu den Additionen gezählt. Für die Berechnung von n_d Abständen müssen

$$n_x = n_d k \quad (5, 2)$$

Multiplikationen und

$$n_+ = n_d (k + 1) \quad (5, 3)$$

Additionen durchgeführt werden.

Bei quaternären und binär-quaternären Bäumen entspricht das

$$n_x = 2\ k\ \mathrm{ld}\ N \qquad (5,\ 4)$$

Multiplikationen und

$$n_+ = 2\ (\ k + 1\ )\ \mathrm{ld}\ N \qquad (5,\ 5)$$

Additionen.

Im Speicher müssen pro Codebuchvektor k Komponenten und die Energie enthalten sein. Es müssen mit (1.2, 13)

$$S = (\ k + 1\ )\ M \qquad (5,\ 6)$$

Vektorkomponenten gespeichert werden. Nach Abschätzung von

$$M = \sum_{i=1}^{m} 2^i < 2^{m+1} = 2\ N \qquad (5,\ 7)$$

kann eine Obergrenze angegeben werden: Im Speicher sind maximal

$$S < 2\ (\ k + 1\ )\ N \qquad (5,\ 8)$$

Vektorkomponenten enthalten, d. h. höchstens doppelt so viele wie im Full-Search-Codebuch gleicher Größe.

Das Diagramm in Bild 5, 1 zeigt den Rechenaufwand und die Speichergröße in Abhängigkeit von der Baumstruktur (QTS-Codebuch = Quaternary-Tree-Search-Codebuch).

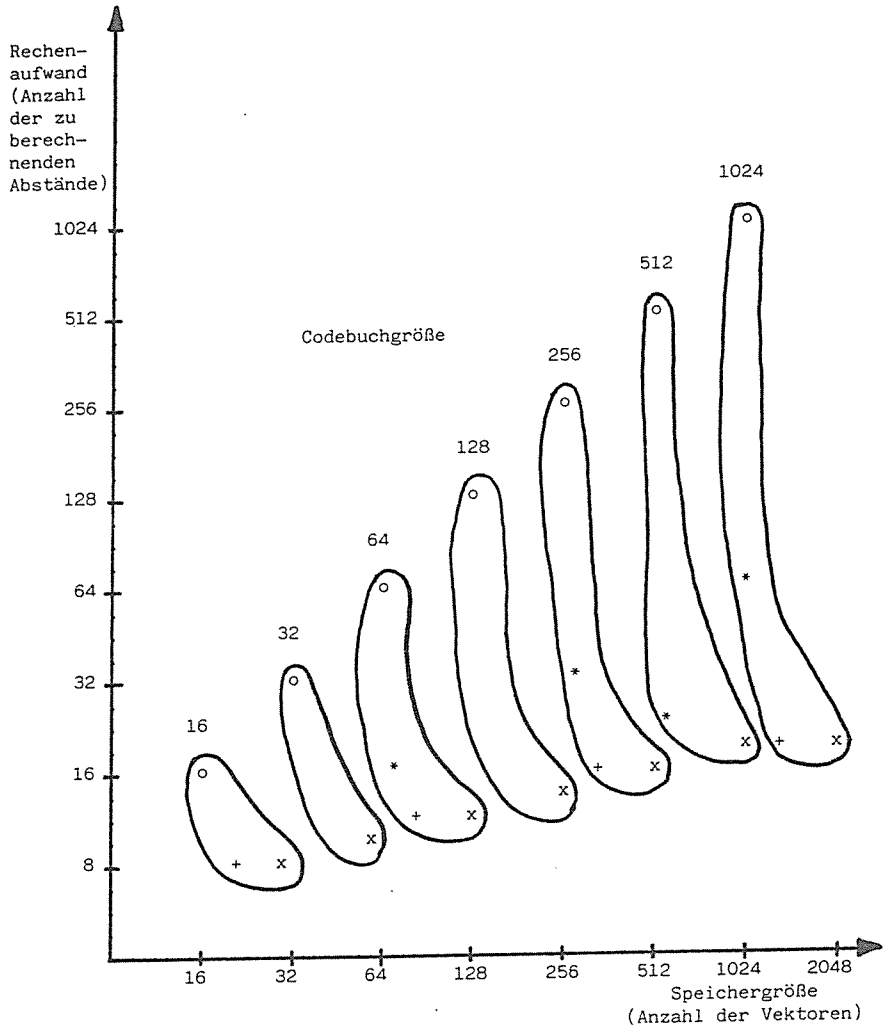

Bild 5, 1 Rechenaufwand und Speichergröße in Abhängigkeit von der Baumstruktur,

o FS-Codebuch

\+ QTS-Codebuch

x BTS-Codebuch

* N = 64 : $\vec{B}$ = (3, 3)

* N = 256 : $\vec{B}$ = (4, 4)

* N = 512 : $\vec{B}$ = (3, 3 ,3)

* N = 1024: $\vec{B}$ = (5, 5)

6 Zusammenfassung und Ausblick

Die Vektorquantisierung ist eine Methode zur Redundanzreduktion bei der digitalen Übertragung von Sprache. Dabei werden optimale Codebücher benötigt, deren Dimension sowohl hinsichtlich der Zahl der Vektoren als auch hinsichtlich der Vektorlänge variabel ist. Der Rechenaufwand bei der Quantisierung kann stark verringert werden, wenn nicht jeder Codebucheintrag abgesucht werden muß (Full-Search), sondern ein gezielter Suchbaum verfolgt wird (Tree-Search).

Im Rahmen dieser Arbeit konnte gezeigt werden, daß mit quaternären und binär-quaternären Baumstrukturen der Suchaufwand bei der Vektorquantisierung von Abtastwerten minimal wird. Beispielsweise müssen bei einem 1024er Codebuch nur 20 Abstände berechnet werden. Wie die Untersuchungen mit den Vektorlängen $k = 4$ und $k = 8$ Samples/Vektor ergaben, sind die Signal/Rausch-abstände der Codebücher mit optimalen Baumstrukturen nur geringfügig schlechter als die der Full-Search-Codebücher.

Interessant wird die Vektorquantisierung allerdings erst für große Vektorlängen. Wahrscheinlich könnte für $r = 1$ Bit/Sample die CCITT-Norm mit der Vektorlänge $k = 32$ Samples/Vektor erfüllt werden. Der Suchaufwand ist mit $n_d = 64$ relativ gering. Dagegen müßte der Speicher die utopische Größe von etwa 200 GByte haben. Mit folgendem Trick käme man mit kleineren Speichern aus: Statt ein 2^{32}er Codebuch zu verwenden, könnte man ebensogut mit vier 2^{8}er Codebüchern arbeiten. Mit dem ersten Codebuch wird das Sprachsignal quantisiert, mit dem zweiten das dabei entstehende Differenzsignal und mit dem dritten und vierten jeweils die Differenzsignale der vorhergehenden Stufen. Der technische Aufwand für dieses Verfahren dürfte noch tragbar sein.

Eine weitere Verbesserung wäre dadurch zu erzielen, daß auf der ersten Knotenebene nach stimmhaften und stimmlosen Lauten sortiert wird. Die beiden Untercodebücher wären zusammengenommen kleiner als ein universelles Codebuch, da sie der Lautstruktur besser angepaßt sind.

7 Literaturverzeichnis

/1/ Gray, R.M., Linde, Y.:
Vector Quantizers and Predictive Quantizers
for Gauss-Markov Sources.
IEEE Trans. Commun., vol. COM-30, No. 2 (1982),
S. 381-389.

/2/ Abut, H., Gray, R.M., Rebolledo, G.:
Vector Quantization of Speech and Speech-Like Waveforms.
IEEE Trans. Acoust., Speech, Signal Processing,
vol. ASSP-30, No. 3 (1982), S. 423-435.

/3/ Wong, D.Y., Juang, B.H., Gray, A.H.Jr.:
An 800 bit/s Vector Quantization LPC Vocoder.
IEEE Trans. Acoust., Speech, Signal Processing,
vol. ASSP-30, No. 5 (1982), S. 770-780.

/4/ Halaski, R., Fasshauer, W.:
Untersuchung von Verfahren zur Reduktion des
Datenflusses bei residualerregten LPC-Verfahren.
St 1985 EA (1984).

/5/ Endres, W.:
Nachrichtenübertragung durch Sprache I+II.
Skripte zur Vorlesung (1982).

/6/ Sessler, G. M.:
Akustik I. Skript zur Vorlesung (1980).

8 Anhang

8.1 Programme

```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C       PROGRAMM ZUR UNTERSUCHUNG VON VEKTORQUANTISIERERN
C            IN BAUMSTRUKTUR FUER DIE QUANTISIERUNG
C               VON SPRACH- UND RESIDUALSIGNALEN
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C PROGRAMM : CODEBUCHSUCHE
C
C DATUM    : 15. 5.84
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
        PROGRAM COBUS
        INTEGER BAUM(10,4),XMAX,YMAX(10),ZMAX,ALGO,VEKT,ANFA(8),
     1  MERK,SINN,FIND,SUCH,ZAHL(1024),MITT(1024,8),I,J,K,L,M,N
        INTEGER*2 WD(433408),BUCH(10,1024,8)
        REAL EPSI,GLEI,ESIG,EDIF,SNDB,DIFF,SUMM,QUAD
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C VERWALTUNGSKREMPEL
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C LESEN DER VEKTORLAENGE
C
        ZMAX=0
        DO WHILE(ZMAX.LT.1.OR.ZMAX.GT.8)
                TYPE *,'SAMPLES/VEKTOR ?'
                ACCEPT 1,ZMAX
        END DO
C
C LESEN DER BAUMSTRUKTUR
C
        DO I=1,10
                YMAX(I)=0
                DO J=1,4
                        BAUM(I,J)=0
                END DO
        END DO
        DO WHILE(BAUM(1,1).LT.1.OR.BAUM(1,1).GT.10)
                TYPE *,'BAUMSTRUKTUR ?'
                ACCEPT 1,BAUM(1,1)
        END DO
        XMAX=1
        BAUM(1,2)=BAUM(1,1)
        DO WHILE(BAUM(XMAX,1).NE.0.AND.BAUM(XMAX,2).LT.10
     1  .AND.XMAX.LT.10)
                ACCEPT 2,BAUM(XMAX+1,1)
                BAUM(XMAX+1,2)=BAUM(XMAX,2)+BAUM(XMAX+1,1)
                IF(BAUM(XMAX+1,2).GT.10)THEN
                        TYPE *,'LETZTE EINGABE KORRIGIEREN !'
                ELSE
                        XMAX=XMAX+1
                END IF
        END DO
        IF(BAUM(XMAX,1).EQ.0)THEN
                BAUM(XMAX,2)=0
                XMAX=XMAX-1
        END IF
        DO I=1,2
                DO J=1,XMAX
                        BAUM(J,I+2)=2**BAUM(J,I)
                END DO
        END DO
```

```
C
C WAHL DES ALGORITHMUS
C
        ALGO=0
        DO WHILE(ALGO.LT.1.OR.ALGO.GT.2)
                TYPE *,'ALGORITHMUS ?'
                ACCEPT 2,ALGO
        END DO
C
C KONTROLLE DER EINGABEN
C
        TYPE 3,ZMAX
        TYPE 4,BAUM(XMAX,2)
        TYPE 5
        DO I=1,XMAX
                TYPE 6,BAUM(I,1)
        END DO
        TYPE 7,ALGO
1       FORMAT(I2)
2       FORMAT(I1)
3       FORMAT(' ','SAMPLES/VEKTOR : ',I4)
4       FORMAT(' ','BIT/VEKTOR     : ',I4)
5       FORMAT(' ','BAUMSTRUKTUR   : ',$)
6       FORMAT('+',I4,$)
7       FORMAT(' ','ALGORITHMUS    : ',I4)
C
C LESEN DER TRAININGSSEQUENZ
C
        OPEN(UNIT=10,NAME='WD4711.;1',TYPE='OLD',
     1  FORM='UNFORMATTED',RECL=128,ACCESS='DIRECT')
                DO I=2,1694
                        READ(10'I)(WD(256*I+J-512),J=1,256)
                END DO
        CLOSE(UNIT=10)
C
C ANZAHL DER TRAININGSVEKTOREN
C
        VEKT=433408/ZMAX
C
C SCHWERPUNKTSVEKTOR UND GLEICHSPANNUNGSANTEIL
C
        DO I=1,8
                ANFA(I)=0
        END DO
        GLEI=0.
        DO I=1,ZMAX
                DO J=0,VEKT-1
                        ANFA(I)=ANFA(I)+WD(J*ZMAX+I)
                END DO
                GLEI=GLEI+FLOAT(ANFA(I))
                J=10*ANFA(I)/VEKT
                IF(J.EQ.0)J=1
                ANFA(I)=(J+IABS(J)/J*5)/10
        END DO
        GLEI=GLEI/FLOAT(VEKT*ZMAX)
C
C BERECHNUNG DER SIGNALENERGIE
C
        ESIG=0.
        DO I=1,VEKT*ZMAX
                ESIG=ESIG+(FLOAT(WD(I))-GLEI)**2.
        END DO
C
C LOESCHEN DES CODEBUCHS
C
        DO I=1,8
                DO J=1,10
                        DO K=1,1024
                                BUCH(J,K,I)=0
                        END DO
                END DO
        END DO
```

```
C
C ABBRUCHBEDINGUNG
C
      EPSI=0.001
C
C SPRUNG ZUM GEWAEHLTEN ALGORITHMUS
C
      GOTO(100,200),ALGO
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C LINDE/BUZO/GRAY-ALGORITHMUS
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C KNOTENEBENEN-SCHLEIFE
C
100   DO I=1,XMAX
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                       C
C ANFANGSBEDINGUNGEN                                                    C
C                                                                       C
      IF(I.EQ.1)THEN
            DO J=1,ZMAX
                  BUCH(1,1,J)=ANFA(J)
            END DO
      ELSE
            DO J=1,BAUM(I-1,4)
                  MERK=(J-1)*BAUM(I,3)+1
                  DO K=1,ZMAX
                        BUCH(I,MERK,K)=BUCH(I-1,J,K)
                  END DO
            END DO
      END IF
C                                                                       C
C ANFANGSBEDINGUNGEN UEBERTRAGEN                                        C
C                                                                       C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C CODEBUCHERSTELLUNG AUF DER I-TEN EBENE
C
            DO J=1,BAUM(I,1)
                  YMAX(I)=2**J
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                       C
C VERDOPPELN DES CODEBUCHS                                              C
C                                                                       C
      DO K=1,BAUM(I,4)/BAUM(I,3)
            MERK=(K-1)*BAUM(I,3)
            SINN=MERK+YMAX(I)/2
            DO L=1,YMAX(I)/2
                  BUCH(I,SINN+L,1)=BUCH(I,MERK+L,1)+1
                  BUCH(I,MERK+L,1)=BUCH(I,MERK+L,1)-1
                  DO M=2,ZMAX
                        BUCH(I,SINN+L,M)=BUCH(I,MERK+L,M)
                  END DO
            END DO
      END DO
C                                                                       C
C CODEBUCH VERDOPPELT                                                   C
C                                                                       C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C ITERATION
C
                  DIFF=1.E32
                  EDIF=1.E30
                  DO K=1,1024
                        ZAHL(K)=0
                        DO L=1,8
                              MITT(K,L)=0
                        END DO
                  END DO
                  DO WHILE((DIFF/EDIF-1.).GT.EPSI)
                        DIFF=EDIF
                        EDIF=0.
```

```
                                    DO K=1,BAUM(I,4)
                                            ZAHL(K)=0
                                            DO L=1,ZMAX
                                                    MITT(K,L)=0
                                            END DO
                                    END DO
C
C BERECHNUNG DER RAUSCHENERGIE
C
                                    DO K=0,VEKT-1
                                            SINN=K*ZMAX
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                              C
C VEKTORQUANTISIERUNG                                                          C
C                                                                              C
        FIND=1
        DO L=1,I
                SUMM=1.E30
                MERK=(FIND-1)*BAUM(L,3)
                DO M=1,YMAX(L)
                        QUAD=0.
                        DO N=1,ZMAX
                                QUAD=QUAD+(FLOAT(WD(SINN+N))-
     1                          FLOAT(BUCH(L,MERK+M,N)))**2.
                        END DO
                        IF(QUAD.LT.SUMM)THEN
                                SUCH=M
                                SUMM=QUAD
                        END IF
                END DO
                FIND=MERK+SUCH
        END DO
C                                                                              C
C ENDE DER VEKTORQUANTISIERUNG                                                 C
C                                                                              C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
                                            EDIF=EDIF+SUMM
                                            ZAHL(FIND)=ZAHL(FIND)+1
                                            DO L=1,ZMAX
                                                    MITT(FIND,L)=
     1                                              MITT(FIND,L)+
     1                                              WD(SINN+L)
                                            END DO
                                    END DO
                                    IF(DIFF.GT.EDIF)THEN
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                              C
C ZENTRIEREN                                                                   C
C                                                                              C
        DO K=1,BAUM(I,4)/BAUM(I,3)
                MERK=(K-1)*BAUM(I,3)
                DO L=1,YMAX(I)
                        SINN=MERK+L
                        IF(ZAHL(SINN).NE.0)THEN
                                DO M=1,ZMAX
                                        N=10*MITT(SINN,M)/
     1                                  ZAHL(SINN)
                                        IF(N.EQ.0)N=1
                                        BUCH(I,SINN,M)=
     1                                  (N+IABS(N)/N*5)/10
                                END DO
                        ELSE
                                IF(I.EQ.1)THEN
                                        BUCH(1,L,1)=ANFA(1)+1
                                        DO M=2,ZMAX
                                                BUCH(1,L,M)=
     1                                          ANFA(M)
                                        END DO
                                ELSE
                                        BUCH(I,SINN,1)=
     1                                  BUCH(I-1,K,1)+1
                                        DO M=2,ZMAX
                                                BUCH(I,SINN,M)=
     1                                          BUCH(I-1,K,M)
                                        END DO
                                END IF
```

```
                        END IF
                END DO
        END DO
C                                                                       C
C ENDE DES ZENTRIERENS                                                  C
C                                                                       C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
                                ELSE
                                        EDIF=DIFF
                                END IF
                        END DO
                END DO
        END DO
C
C ENDE DES LINDE/BUZO/GRAY-ALGORITHMUS
C
        GOTO 1000
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C LINDE/BUZO/GRAY/MEYER-ALGORITHMUS
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C KNOTENEBENEN-SCHLEIFE
C
200     DO I=1,XMAX
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                       C
C ANFANGSBEDINGUNGEN                                                    C
C                                                                       C
        IF(I.EQ.1)THEN
                DO J=1,ZMAX
                        BUCH(1,1,J)=ANFA(J)
                END DO
        ELSE
                DO J=1,BAUM(I-1,4)
                        MERK=(J-1)*BAUM(I,3)+1
                        DO K=1,ZMAX
                                BUCH(I,MERK,K)=BUCH(I-1,J,K)
                        END DO
                END DO
        END IF
C                                                                       C
C ANFANGSBEDINGUNGEN UEBERTRAGEN                                        C
C                                                                       C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C CODEBUCHERSTELLUNG AUF DER I-TEN EBENE
C
                DO J=1,BAUM(I,1)
                        YMAX(I)=2**J
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                       C
C VERDOPPELN DES CODEBUCHS                                              C
C                                                                       C
        DO K=1,BAUM(I,4)/BAUM(I,3)
                MERK=(K-1)*BAUM(I,3)
                SINN=MERK+YMAX(I)/2
                DO L=1,YMAX(I)/2
                        BUCH(I,SINN+L,1)=BUCH(I,MERK+L,1)+1
                        BUCH(I,MERK+L,1)=BUCH(I,MERK+L,1)-1
                        DO M=2,ZMAX
                                BUCH(I,SINN+L,M)=BUCH(I,MERK+L,M)
                        END DO
                END DO
        END DO
C                                                                       C
C CODEBUCH VERDOPPELT                                                   C
C                                                                       C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C ITERATION
C
                        DIFF=1.E32
                        EDIF=1.E30
```

```
                        DO K=1,1024
                                ZAHL(K)=0
                                DO L=1,8
                                        MITT(K,L)=0
                                END DO
                        END DO
                        DO WHILE((DIFF/EDIF-1.).GT.EPSI)
                                DIFF=EDIF
                                EDIF=0.
                                DO K=1,BAUM(I,4)
                                        ZAHL(K)=0
                                        DO L=1,ZMAX
                                                MITT(K,L)=0
                                        END DO
                                END DO
C
C BERECHNUNG DER RAUSCHENERGIE
C
                                DO K=0,VEKT-1
                                        SINN=K*ZMAX
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                           C
C VEKTORQUANTISIERUNG                                                       C
C                                                                           C
        FIND=1
        DO L=1,I
                SUMM=1.E30
                MERK=(FIND-1)*BAUM(L,3)
                DO M=1,YMAX(L)
                        QUAD=0.
                        DO N=1,ZMAX
                                QUAD=QUAD+(FLOAT(WD(SINN+N))-
     1                          FLOAT(BUCH(L,MERK+M,N)))**2.
                        END DO
                        IF(QUAD.LT.SUMM)THEN
                                SUCH=M
                                SUMM=QUAD
                        END IF
                END DO
                FIND=MERK+SUCH
        END DO
C                                                                           C
C ENDE DER VEKTORQUANTISIERUNG                                              C
C                                                                           C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
                                EDIF=EDIF+SUMM
                                ZAHL(FIND)=ZAHL(FIND)+1
                                DO L=1,ZMAX
                                        MITT(FIND,L)=
     1                                  MITT(FIND,L)+
     1                                  WD(SINN+L)
                                END DO
                        END DO
                        IF(DIFF.GT.EDIF)THEN
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                           C
C ZENTRIEREN                                                                C
C                                                                           C
        DO K=1,BAUM(I,4)/BAUM(I,3)
                MERK=(K-1)*BAUM(I,3)
                DO L=1,YMAX(I)
                        SINN=MERK+L
                        IF(ZAHL(SINN).NE.0)THEN
                                DO M=1,ZMAX
                                        N=10*MITT(SINN,M)/
     1                                  ZAHL(SINN)
                                        IF(N.EQ.0)N=1
                                        BUCH(I,SINN,M)=
     1                                  (N+IABS(N)/N*5)/10
                                END DO
```

```
                        ELSE
                                IF(I.EQ.1)THEN
                                        BUCH(1,L,1)=ANFA(1)+1
                                        DO M=2,ZMAX
                                                BUCH(1,L,M)=
     1                                          ANFA(M)
                                        END DO
                                ELSE
                                        BUCH(I,SINN,1)=
     1                                  BUCH(I-1,K,1)+1
                                        DO M=2,ZMAX
                                                BUCH(I,SINN,M)=
     1                                          BUCH(I-1,K,M)
                                        END DO
                                END IF
                        END IF
                END DO
        END DO
C                                                                       C
C ENDE DES ZENTRIERENS                                                  C
C                                                                       C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
                                ELSE
                                        EDIF=DIFF
                                END IF
C
C ABBRUCH NACH ERSTER ITERATION FUER I < XMAX-1
C
                                IF(I.LT.XMAX-1)DIFF=EDIF
                        END DO
                END DO
        END DO
C
C ENDE DES LINDE/BUZO/GRAY/MEYER-ALGORITHMUS
C
        GOTO 1000
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C ABSPEICHERN DER DATEN
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C BERECHNUNG DES SIGNAL/RAUSCH-VERHAELTNISSES
C
1000    SNDB=10*ALOG10(ESIG/EDIF)
C
C ABLEGEN DES CODEBUCHS
C
        OPEN(UNIT=10,NAME='CBUCH.DAT',TYPE='NEW')
                WRITE(10,10)ZMAX
                WRITE(10,11)(BAUM(I,1),I=1,10)
                WRITE(10,10)ALGO
                WRITE(10,12)SNDB
                DO I=1,XMAX
                        DO J=1,BAUM(I,4)
                                WRITE(10,13)(BUCH(I,J,K),K=1,8)
                        END DO
                END DO
        CLOSE(UNIT=10)
10      FORMAT(I4)
11      FORMAT(10I4)
12      FORMAT(F10.4)
13      FORMAT(8I6)
        STOP'HIER ENDET DAS PROGRAMM !'
        END
```

```
C...            INTERACTIVE LABORATORY SYSTEM
C...
C...            ILS COMMAND PROGRAM  ** TSV **
C...
C...            COPYRIGHT - JUNE 1984
C...            JOCHEN MEYER
C...            TECHNISCHE HOCHSCHULE DARMSTADT
C...
      PROGRAM TSV
C...
      IMPLICIT INTEGER (I-N)
C...
C...  START OF DOCUMENTATION
C...
C...  TREE SEARCH VECTOR QUANTIZER
C...
C...  COMMAND FORMAT
C...
C...  TSV N1,N2
C...
C...  ALPHABETIC ARGUMENTS
C...
C...  NONE
C...
C...  NUMERIC ARGUMENTS
C...
C...  N1      STARTING FRAME (DEFAULT='N1SAV')
C...  N2      NUMBER OF FRAMES (DEFAULT='N2SAV')
C...
C...  END OF DOCUMENTATION
C...
      COMMON /CLBF/ INSFLG,LCLBF,ICLBF(40)
      COMMON /ILSA/ NBCW,NCWBK,NDPBK,NDPF,NBDP,NCWFH,
     1              KBU,KBUIN,LPU,LUGI,LUGO,NSC,CWSC,
     2              NBA2D,MIDA2D,ICTIM(4),ICDAT(6)
      COMMON /ILSB/ IA(4),N1,N2,N3,N4,N5,N6,N7,N8,N9,N10,N11,N12
      COMMON /ILSC/ IASAV(4),N1SAV,N2SAV,N3SAV,N4SAV,N5SAV,N6SAV,
     1              N7SAV,N8SAV,N9SAV,N10SAV,N11SAV,N12SAV
      COMMON /ILSE/ IFLPA(16),LENPA,LFILPA,NFLPA,IDKPA,IDKDPA,IDPA
      COMMON /ILSF/ IFLSA(16),LENSA,LFILSA,NFLSA,IDKSA,IDKDSA,IDSA
      COMMON /ILSH/ FS,M,MP1,MO2,N,NSPBK,NSHFT,ICON,ISF,IHAM,LAN
      COMMON /ILSI/ LRH,IEX,ISTAN,NAN,IDF(5),NP,IVL,IC
C...
      INTEGER BAUM(10,4),XMAX,YMAX(10),ZMAX,ALGO,VEKT,
     1        MERK,SINN,FIND,SUCH
      INTEGER*2 BUCH(2046,8)
      REAL ESIG,EDIF,SNDB,SUMM,QUAD
C...
      DIMENSION IR(1024),IS(1024),IY(512)
C...
      DATA I1,I5/1,5/
      DATA IIBR,IIBW/2*0/
      DATA LIY,LIR,LIS/512,2*1024/
      DATA ICHNP,ICHS/2HNP,2HS /
C...
      CALL RCOMM
C...
C...  INITIALISIERT PRIMARY-FILE
C...
      CALL CHKFL(NFLPA,IDKPA,IFLPA,LENPA,LFILPA,IY,ICHS,IERR)
      IF(IERR.NE.0) GOTO 170
      NSPBK=IY(6)
      CALL ANCHK(NSC,N1,N2,N1SAV,N2SAV,NSCA,ISTAN,NAN,NDPF,
     1           NDPF,NCWFH,NSPBK,LFILPA,NDPBK,CWSC,KBU)
      NPTS=NDPF
      IF(NPTS.LE.LIY) GOTO 120
      WRITE(KBU,110)
110   FORMAT(17H DECREASE CONTEXT)
      GOTO 170
C...
C...  ERZEUGT SECONDARY-FILE
C...
120   ITYPE=-1
      CALL CHKFL(NFLSA,IDKSA,IFLSA,LENSA,LFILSA,IS,ITYPE,IERR)
      ISTOP=N1SAV+N2SAV-1
```

```
      CALL SETUP(IFLSA,LENSA,LFILSA,ISTOP,NPTS,NCWFH,IS,ICHNP)
      IF(ISTOP.LT.0) GOTO 170
      IY(8)=ISTAN
      IY(9)=NAN
      IY(10)=NSC
      IY(19)=IDKPA
      IY(20)=NFLPA
      IY(63)=-32000
      CALL WHEAD(IY,IFLSA,LENSA)
      NCWFH1=NCWFH
      DO 130 JKL=1,512
130   IY(JKL)=0
      DO 140 JKL=1,ISTOP
      CALL WRITD(NSC,I1,ISDB1,NPTS,IS,LIS,IY,IIBW,
     1           IFLSA,LENSA,LFILSA,NCWFH,NDPBK,CWSC)
      IF(IIBW.GE.0) GOTO 140
      CALL MESG(I5,IFLSA)
      GOTO 170
140   CONTINUE
      IIBW=-IIBW
      CALL WRITD(NSC,I1,ISDB1,NPTS,IS,LIS,IY,IIBW,
     1           IFLSA,LENSA,LFILSA,NCWFH,NDPBK,CWSC)
      IIBW=0
C...
C...  LESEN DES CODEBUCH
C...
      OPEN(UNIT=10,NAME='[MEYER.ERNST]CBUCH.DAT;1',TYPE='OLD')
            READ(10,1)ZMAX
            READ(10,2)(BAUM(I,1),I=1,10)
            XMAX=1
            DO WHILE(BAUM(XMAX+1,1).NE.0.AND.XMAX.LT.10)
                  XMAX=XMAX+1
            END DO
            BAUM(1,3)=BAUM(1,1)
            DO I=2,XMAX
                  BAUM(I,3)=BAUM(I-1,3)+BAUM(I,1)
            END DO
            DO I=1,XMAX
                  BAUM(I,2)=2**BAUM(I,1)
                  BAUM(I,3)=2**BAUM(I,3)
            END DO
            BAUM(1,4)=0
            DO I=2,XMAX
                  BAUM(I,4)=BAUM(I-1,4)+BAUM(I-1,3)
            END DO
            READ(10,1)ALGO
            READ(10,3)SNDB
            DO I=1,BAUM(XMAX,3)+BAUM(XMAX,4)
                  READ(10,4)(BUCH(I,J),J=1,8)
            END DO
      CLOSE(UNIT=10)
C...
C...  DATENKONTROLLE
C...
      TYPE 5,ZMAX
      TYPE 6
      DO I=1,XMAX
            TYPE 7,BAUM(I,1)
      END DO
      TYPE 8,SNDB
C...
C...  BERECHNUNG EINIGER KONSTANTEN
C...
      VEKT=NPTS/ZMAX
      NPTS1=VEKT*ZMAX
      ISTAN1=INT(FLOAT(ISTAN)*FLOAT(NPTS)/FLOAT(NPTS1))
      NAN1=INT(FLOAT(NAN)*FLOAT(NPTS)/FLOAT(NPTS1))
      ESIG=0.
      EDIF=0.
C...
C...  LIEST EIN FRAME DES PRIMARY-FILES
C...
      DO 160 JKL=1,NAN1
      CALL GETD(NSC,ISTAN1,NPTS1,NPTS1,IR,LIR,IY,IIBR,
     1          IFLPA,LENPA,LFILPA,NCWFH1,NDPBK,CWSC)
```

```
      IF(IIBR.GE.0) GOTO 150
      CALL MESG(I5,IFLFA)
      GOTO 170
C...
C...  BERECHNUNG DER SIGNALENERGIE
C...
150   DO I=1,VEKT*ZMAX
            ESIG=ESIG+FLOAT(IY(I))**2.
      END DO
C...
C...  VEKTORQUANTISIERUNG
C...
      DO I=0,VEKT-1
            SINN=I*ZMAX
            FIND=1
            DO J=1,XMAX
                  SUMM=1.E30
                  MERK=(FIND-1)*BAUM(J,2)+BAUM(J,4)
                  DO K=1,BAUM(J,2)
                        QUAD=0.
                        M=MERK+K
                        DO L=1,ZMAX
                              QUAD=QUAD+
     1                        (FLOAT(IY(SINN+L))-
     2                        FLOAT(BUCH(M,L)))**2.
                        END DO
                        IF(QUAD.LT.SUMM)THEN
                              SUCH=K
                              SUMM=QUAD
                        END IF
                  END DO
                  FIND=MERK+SUCH-BAUM(J,4)
            END DO
            DO J=1,ZMAX
                  IY(SINN+J)=BUCH(BAUM(XMAX,4)+FIND,J)
            END DO
            EDIF=EDIF+SUMM
      END DO
      CALL WRITD(NSC,ISTAN1,ISDB1,NPTS1,IS,LIS,IY,IIBW,
     1           IFLSA,LENSA,LFILSA,NCWFH,NDPBK,CWSC)
      IF(IIBW.GE.0) GOTO 160
      CALL MESG(I5,IFLSA)
      GOTO 170
160   CONTINUE
      IIBW=-IIBW
      CALL WRITD(NSC,ISTAN1,ISDB1,NPTS1,IS,LIS,IY,IIBW,
     1           IFLSA,LENSA,LFILSA,NCWFH,NDPBK,CWSC)
C...
C...  BERECHNUNG DES SIGNAL/RAUSCH-VERHAELTNISSES
C...
      IF(EDIF.EQ.0.)THEN
            TYPE *,'Optimale Quantisierung !'
      ELSE
            IF(ESIG.NE.0.)THEN
                  SNDB=10*ALOG10(ESIG/EDIF)
                  TYPE 9,SNDB
            ELSE
                  TYPE *,'Testsignal ist NULL !'
            END IF
      END IF
C...
170   CALL WCOMM
      CALL EXILS
      STOP
1     FORMAT(I4)
2     FORMAT(10I4)
3     FORMAT(F10.4)
4     FORMAT(8I6)
5     FORMAT(' ','Samples/Vektor                  : ',I4)
6     FORMAT(' ','Baumstruktur                    : ',$)
7     FORMAT('+',I4,$)
8     FORMAT(' ','Stoerabstand (Traininssdaten) : ',F9.4,' dB')
9     FORMAT(' ','Stoerabstand (Testdaten)      : ',F9.4,' dB')
      END
```

8.2 Ergebnisse der Untersuchungen

VEKTORLAENGE 1-8, 1 BIT/SAMPLE
==========
FULL-SEARCH

```
   1
   1   0   0   0   0   0   0   0   0   0
   1
    2.5952
   2
   2   0   0   0   0   0   0   0   0   0
   1
    7.4947
   3
   3   0   0   0   0   0   0   0   0   0
   1
   10.1886
   4
   4   0   0   0   0   0   0   0   0   0
   1
   11.2450
   5
   5   0   0   0   0   0   0   0   0   0
   1
   12.5864
   6
   6   0   0   0   0   0   0   0   0   0
   1
   13.7458
   7
   7   0   0   0   0   0   0   0   0   0
   1
   14.7298
   8
   8   0   0   0   0   0   0   0   0   0
   1
   15.5923
```

VEKTORLAENGE 1-8, 1 BIT/SAMPLE
==========
BINARY-TREE-SEARCH

```
   1
   1   0   0   0   0   0   0   0   0   0
   1
    2.5952
   2
   1   1   0   0   0   0   0   0   0   0
   1
    7.3221
   3
   1   1   1   0   0   0   0   0   0   0
   1
    9.5284
   4
   1   1   1   1   0   0   0   0   0   0
   1
   10.4713
   5
   1   1   1   1   1   0   0   0   0   0
   1
   11.5778
   6
   1   1   1   1   1   1   0   0   0   0
   1
   12.9172
   7
   1   1   1   1   1   1   1   0   0   0
   1
   13.6192
   8
   1   1   1   1   1   1   1   1   0   0
   1
   14.4216
```

VEKTORLAENGE 1-4, 2 BIT/SAMPLE
==========
FULL-SEARCH

```
   1
   2   0   0   0   0   0   0   0   0   0
   1
    7.9403
   2
   4   0   0   0   0   0   0   0   0   0
   1
   14.6507
   3
   6   0   0   0   0   0   0   0   0   0
   1
   17.5665
   4
   8   0   0   0   0   0   0   0   0   0
   1
   20.3634
```

VEKTORLAENGE 1-4, 2 BIT/SAMPLE
==========
BINARY-TREE-SEARCH

```
   1
   1   1   0   0   0   0   0   0   0   0
   1
    7.7473
   2
   1   1   1   1   0   0   0   0   0   0
   1
   14.3243
   3
   1   1   1   1   1   1   0   0   0   0
   1
   16.4817
   4
   1   1   1   1   1   1   1   1   0   0
   1
   19.2632
```

ALLE BAEUME
==========
VEKTORLAENGE 4, 1 BIT/SAMPLE

```
   4
   4   0   0   0   0   0   0   0   0   0
   1
   11.2450
   4
   2   2   0   0   0   0   0   0   0   0
   1
   10.9471
   4
   1   3   0   0   0   0   0   0   0   0
   1
   10.8929
   4
   1   1   2   0   0   0   0   0   0   0
   1
   10.8414
   4
   3   1   0   0   0   0   0   0   0   0
   1
   10.7676
   4
   2   1   1   0   0   0   0   0   0   0
   1
   10.6486
   4
   1   2   1   0   0   0   0   0   0   0
   1
   10.5002
   4
   1   1   1   1   0   0   0   0   0   0
   1
   10.4713
```

ALLE BAEUME
==========
VEKTORLAENGE 8, 0.5 BIT/SAMPLE

```
   8
   4   0   0   0   0   0   0   0   0   0
   1
    8.4593
   8
   1   3   0   0   0   0   0   0   0   0
   1
    8.2758
   8
   1   1   2   0   0   0   0   0   0   0
   1
    8.1188
   8
   3   1   0   0   0   0   0   0   0   0
   1
    8.1018
   8
   2   2   0   0   0   0   0   0   0   0
   1
    8.0214
   8
   1   2   1   0   0   0   0   0   0   0
   1
    7.8311
   8
   2   1   1   0   0   0   0   0   0   0
   1
    7.8293
   8
   1   1   1   1   0   0   0   0   0   0
   1
    7.6909
```

AUSGEWAEHLTE BAEUME
==========
VEKTORLAENGE 8, 0.625 BIT/SAMPLE

```
   8
   5   0   0   0   0   0   0   0   0   0
   1
   10.3691
   8
   2   3   0   0   0   0   0   0   0   0
   1
   10.2128
   8
   3   2   0   0   0   0   0   0   0   0
   1
   10.1644
   8
   2   1   2   0   0   0   0   0   0   0
   1
    9.9990
   8
   1   2   2   0   0   0   0   0   0   0
   1
    9.9797
   8
   2   2   1   0   0   0   0   0   0   0
   1
    9.9195
   8
   2   1   1   1   0   0   0   0   0   0
   1
    9.8796
   8
   1   1   2   1   0   0   0   0   0   0
   1
    9.7854
   8
   1   2   1   1   0   0   0   0   0   0
   1
    9.6962
   8
   1   1   1   2   0   0   0   0   0   0
   1
    9.6330
   8
   1   1   1   1   1   0   0   0   0   0
   1
    9.5595
```

AUSGEWAEHLTE BAEUME
==========
VEKTORLAENGE 8, 0.75 BIT/SAMPLE

```
   8
   6   0   0   0   0   0   0   0   0   0
   1
   12.2052
   8
   2   4   0   0   0   0   0   0   0   0
   1
   12.0328
   8
   3   3   0   0   0   0   0   0   0   0
   1
   12.0152
   8
   4   2   0   0   0   0   0   0   0   0
   1
   11.9229
   8
   1   2   3   0   0   0   0   0   0   0
   1
   11.8365
   8
   2   2   2   0   0   0   0   0   0   0
   1
   11.8148
```

```
   8
   2   1   1   2   0   0   0   0   0   0
   1
   11.7037
   8
   1   1   2   2   0   0   0   0   0   0
   1
   11.6958
   8
   1   2   1   2   0   0   0   0   0   0
   1
   11.6943
   8
   2   2   1   1   0   0   0   0   0   0
   1
   11.5682
   8
   1   2   2   1   0   0   0   0   0   0
   1
   11.5424
   8
   2   1   2   1   0   0   0   0   0   0
   1
   11.5090
   8
   1   1   1   1   1   1   0   0   0   0
   1
   11.3628
AUSGEWAEHLTE BAEUME
==================
VEKTORLAENGE 8, 0.875 BIT/SAMPLE
--------------------------------
   8
   7   0   0   0   0   0   0   0   0   0
   1
   13.9742
   8
   2   2   1   2   0   0   0   0   0   0
   1
   13.3801
   8
   1   2   2   2   0   0   0   0   0   0
   1
   13.3323
   8
   2   1   2   2   0   0   0   0   0   0
   1
   13.3224
   8
   2   2   2   1   0   0   0   0   0   0
   1
   13.2238
   8
   1   1   1   1   1   1   1   0   0   0
   1
   12.9067
AUSGEWAEHLTE BAEUME
==================
VEKTORLAENGE 8, 1 BIT/SAMPLE
----------------------------
   8
   8   0   0   0   0   0   0   0   0   0
   1
   15.5923
   8
   4   4   0   0   0   0   0   0   0   0
   1
   15.3723
   8
   2   2   2   2   0   0   0   0   0   0
   1
   14.9174
   8
   1   1   1   1   1   1   1   1   0   0
   1
   14.4216
AUSGEWAEHLTE BAEUME
==================
VEKTORLAENGE 8, 1.125 BIT/SAMPLE
--------------------------------
   8
   3   3   3   0   0   0   0   0   0   0
   1
   16.8005
   8
   1   2   2   2   2   0   0   0   0   0
   1
   16.3752
   8
   1   1   1   1   1   1   1   1   1   0
   1
   15.9203
AUSGEWAEHLTE BAEUME
==================
VEKTORLAENGE 8, 1.25 BIT/SAMPLE
-------------------------------
   8
   5   5   0   0   0   0   0   0   0   0
   1
   18.7241
   8
   2   2   2   2   2   0   0   0   0   0
   1
   18.1471
   8
   1   1   1   1   1   1   1   1   1   1
   1
   17.5215
```

```
NEUE STRATEGIE
==============
VEKTORLAENGE 4, 1 BIT/SAMPLE
----------------------------
   4
   1   1   1   1   0   0   0   0   0   0
   2
   10.5123
NEUE STRATEGIE
==============
VEKTORLAENGE 4, 1.25 BIT/SAMPLE
-------------------------------
   4
   1   1   1   1   1   0   0   0   0   0
   2
   12.7059
NEUE STRATEGIE
==============
VEKTORLAENGE 4, 1.5 BIT/SAMPLE
------------------------------
   4
   2   2   2   0   0   0   0   0   0   0
   2
   15.5167
   4
   1   1   1   1   1   1   0   0   0   0
   2
   15.0890
NEUE STRATEGIE
==============
VEKTORLAENGE 4, 1.75 BIT/SAMPLE
-------------------------------
   4
   1   1   1   1   1   1   1   0   0   0
   2
   17.2274
NEUE STRATEGIE
==============
VEKTORLAENGE 4, 2 BIT/SAMPLE
----------------------------
   4
   1   1   1   1   1   1   1   1   0   0
   2
   19.1987
   4
   2   2   2   2   0   0   0   0   0   0
   2
   18.9470
NEUE STRATEGIE
==============
VEKTORLAENGE 8, 0.5 BIT/SAMPLE
------------------------------
   8
   1   1   1   1   0   0   0   0   0   0
   2
    7.8317
NEUE STRATEGIE
==============
VEKTORLAENGE 8, 0.625 BIT/SAMPLE
--------------------------------
   8
   1   1   1   1   1   0   0   0   0   0
   2
    9.6893
NEUE STRATEGIE
==============
VEKTORLAENGE 8, 0.75 BIT/SAMPLE
-------------------------------
   8
   2   2   2   0   0   0   0   0   0   0
   2
   11.8158
   8
   1   1   1   1   1   1   0   0   0   0
   2
   11.2790
```

```
NEUE STRATEGIE
==============
VEKTORLAENGE 8, 0.875 BIT/SAMPLE
--------------------------------
   8
   1   1   1   1   1   1   1   0   0   0
   2
   12.8277
NEUE STRATEGIE
==============
VEKTORLAENGE 8, 1 BIT/SAMPLE
----------------------------
   8
   2   2   2   2   0   0   0   0   0   0
   2
   14.4196
   8
   1   1   1   1   1   1   1   1   0   0
   2
   14.3382
NEUE STRATEGIE
==============
VEKTORLAENGE 8, 1.125 BIT/SAMPLE
--------------------------------
   8
   3   3   3   0   0   0   0   0   0   0
   2
   16.6293
   8
   1   1   1   1   1   1   1   1   1   0
   2
   15.8133
NEUE STRATEGIE
==============
VEKTORLAENGE 8, 1.25 BIT/SAMPLE
-------------------------------
   8
   1   1   1   1   1   1   1   1   1   1
   2
   17.3619
   8
   2   2   2   2   2   0   0   0   0   0
   2
   16.7444
```

```
OPTIMALE BAEUME
===============
VEKTORLAENGE 4, 1 BIT/SAMPLE
----------------------------
   4
   4   0   0   0   0   0   0   0   0   0
   1
   11.2450
   4
   2   2   0   0   0   0   0   0   0   0
   1
   10.9471
   4
   1   1   1   1   0   0   0   0   0   0
   1
   10.4713
OPTIMALE BAEUME
===============
VEKTORLAENGE 4, 1.25 BIT/SAMPLE
-------------------------------
   4
   5   0   0   0   0   0   0   0   0   0
   1
   13.6389
   4
   1   2   2   0   0   0   0   0   0   0
   1
   13.1751
   4
   1   1   1   1   0   0   0   0   0   0
   1
   12.5714
OPTIMALE BAEUME
===============
VEKTORLAENGE 4, 1.5 BIT/SAMPLE
------------------------------
   4
   6   0   0   0   0   0   0   0   0   0
   1
   16.0424
   4
   3   3   0   0   0   0   0   0   0   0
   1
   15.8731
   4
   2   2   2   0   0   0   0   0   0   0
   1
   15.6494
   4
   1   1   1   1   1   0   0   0   0   0
   1
   14.9846
OPTIMALE BAEUME
===============
VEKTORLAENGE 4, 1.75 BIT/SAMPLE
-------------------------------
   4
   7   0   0   0   0   0   0   0   0   0
   1
   18.2979
   4
   1   2   2   2   0   0   0   0   0   0
   1
   17.5533
   4
   1   1   1   1   1   1   1   0   0   0
   1
   17.1707
```

```
OPTIMALE BAEUME
===============
VEKTORLAENGE 4, 2 BIT/SAMPLE
----------------------------
   4
   8   0   0   0   0   0   0   0   0   0
   1
   20.3634
   4
   4   4   0   0   0   0   0   0   0   0
   1
   20.2528
   4
   2   2   2   2   0   0   0   0   0   0
   1
   19.9020
   4
   1   1   1   1   1   1   1   1   0   0
   1
   19.2632
OPTIMALE BAEUME
===============
VEKTORLAENGE 8, 0.5 BIT/SAMPLE
------------------------------
   8
   4   0   0   0   0   0   0   0   0   0
   1
    8.4593
   8
   2   2   0   0   0   0   0   0   0   0
   1
    8.0214
   8
   1   1   1   1   0   0   0   0   0   0
   1
    7.6909
OPTIMALE BAEUME
===============
VEKTORLAENGE 8, 0.625 BIT/SAMPLE
--------------------------------
   8
   5   0   0   0   0   0   0   0   0   0
   1
   10.3691
   8
   1   2   2   0   0   0   0   0   0   0
   1
    9.9797
   8
   1   1   1   1   1   0   0   0   0   0
   1
    9.5595
OPTIMALE BAEUME
===============
VEKTORLAENGE 8, 0.75 BIT/SAMPLE
-------------------------------
   8
   6   0   0   0   0   0   0   0   0   0
   1
   12.2052
   8
   3   3   0   0   0   0   0   0   0   0
   1
   12.0152
   8
   2   2   2   0   0   0   0   0   0   0
   1
   11.8148
   8
   1   1   1   1   1   1   0   0   0   0
   1
   11.3628
```

```
OPTIMALE BAEUME
===============
VEKTORLAENGE 8, 0.875 BIT/SAMPLE
--------------------------------
   8
   7   0   0   0   0   0   0   0   0   0
   1
   13.9742
   8
   1   2   2   2   0   0   0   0   0   0
   1
   13.3323
   8
   1   1   1   1   1   1   1   0   0   0
   1
   12.9067
OPTIMALE BAEUME
===============
VEKTORLAENGE 8, 1 BIT/SAMPLE
----------------------------
   8
   8   0   0   0   0   0   0   0   0   0
   1
   15.5923
   8
   4   4   0   0   0   0   0   0   0   0
   1
   15.3723
   8
   2   2   2   2   0   0   0   0   0   0
   1
   14.9174
   8
   1   1   1   1   1   1   1   1   0   0
   1
   14.4216
OPTIMALE BAEUME
===============
VEKTORLAENGE 8, 1.125 BIT/SAMPLE
--------------------------------
   8
   3   3   3   0   0   0   0   0   0   0
   1
   16.8005
   8
   1   2   2   2   2   0   0   0   0   0
   1
   16.3752
   8
   1   1   1   1   1   1   1   1   1   0
   1
   15.9203
OPTIMALE BAEUME
===============
VEKTORLAENGE 8, 1.25 BIT/SAMPLE
-------------------------------
   8
   5   5   0   0   0   0   0   0   0   0
   1
   18.7241
   8
   2   2   2   2   2   0   0   0   0   0
   1
   18.1471
   8
   1   1   1   1   1   1   1   1   1   1
   1
   17.5215
```

```
RESIDUAL-CODEBUECHER
====================
VEKTORLAENGE 4, 1 BIT/SAMPLE
----------------------------
   4
   4   0   0   0   0   0   0   0   0   0
   1
    8.0444
   4
   2   2   0   0   0   0   0   0   0   0
   1
    7.8832
   4
   1   1   1   1   0   0   0   0   0   0
   1
    7.4772
RESIDUAL-CODEBUECHER
====================
VEKTORLAENGE 4, 1.25 BIT/SAMPLE
-------------------------------
   4
   5   0   0   0   0   0   0   0   0   0
   1
    9.8436
   4
   1   2   2   0   0   0   0   0   0   0
   1
    9.0781
   4
   1   1   1   1   0   0   0   0   0   0
   1
    8.8768
RESIDUAL-CODEBUECHER
====================
VEKTORLAENGE 4, 1.5 BIT/SAMPLE
------------------------------
   4
   6   0   0   0   0   0   0   0   0   0
   1
   11.4462
   4
   3   3   0   0   0   0   0   0   0   0
   1
   10.9975
   4
   2   2   2   0   0   0   0   0   0   0
   1
   10.9287
   4
   1   1   1   1   1   1   0   0   0   0
   1
   10.2877
RESIDUAL-CODEBUECHER
====================
VEKTORLAENGE 4, 1.75 BIT/SAMPLE
-------------------------------
   4
   7   0   0   0   0   0   0   0   0   0
   1
   12.9146
   4
   1   2   2   2   0   0   0   0   0   0
   1
   12.1764
   4
   1   1   1   1   1   1   1   0   0   0
   1
   11.8825
```

```
RESIDUAL-CODEBUECHER
====================
VEKTORLAENGE 4, 2 BIT/SAMPLE
----------------------------
   4
   8   0   0   0   0   0   0   0   0   0
   1
   14.4743
   4
   4   4   0   0   0   0   0   0   0   0
   1
   14.1975
   4
   2   2   2   2   0   0   0   0   0   0
   1
   13.8383
   4
   1   1   1   1   1   1   1   1   0   0
   1
   13.2907
RESIDUAL-CODEBUECHER
====================
VEKTORLAENGE 8, 0.5 BIT/SAMPLE
------------------------------
   8
   4   0   0   0   0   0   0   0   0   0
   1
    6.0170
   8
   2   2   0   0   0   0   0   0   0   0
   1
    5.8828
   8
   1   1   1   1   0   0   0   0   0   0
   1
    5.3589
RESIDUAL-CODEBUECHER
====================
VEKTORLAENGE 8, 0.625 BIT/SAMPLE
--------------------------------
   8
   5   0   0   0   0   0   0   0   0   0
   1
    7.4717
   8
   1   2   2   0   0   0   0   0   0   0
   1
    6.9673
   8
   1   1   1   1   1   0   0   0   0   0
   1
    6.7864
RESIDUAL-CODEBUECHER
====================
VEKTORLAENGE 8, 0.75 BIT/SAMPLE
-------------------------------
   8
   6   0   0   0   0   0   0   0   0   0
   1
    8.8156
   8
   3   3   0   0   0   0   0   0   0   0
   1
    8.5368
   8
   2   2   2   0   0   0   0   0   0   0
   1
    8.3153
   8
   1   1   1   1   1   1   0   0   0   0
   1
    7.9730
RESIDUAL-CODEBUECHER
====================
VEKTORLAENGE 8, 0.875 BIT/SAMPLE
--------------------------------
   8
   7   0   0   0   0   0   0   0   0   0
   1
   10.1478
   8
   1   2   2   2   0   0   0   0   0   0
   1
    9.3923
   8
   1   1   1   1   1   1   1   0   0   0
   1
    9.1574
RESIDUAL-CODEBUECHER
====================
VEKTORLAENGE 8, 1 BIT/SAMPLE
----------------------------
   8
   8   0   0   0   0   0   0   0   0   0
   1
   11.3920
   8
   4   4   0   0   0   0   0   0   0   0
   1
   11.0712
   8
   2   2   2   2   0   0   0   0   0   0
   1
   10.8303
   8
   1   1   1   1   1   1   1   1   0   0
   1
   10.4020
RESIDUAL-CODEBUECHER
====================
VEKTORLAENGE 8, 1.125 BIT/SAMPLE
--------------------------------
   8
   3   3   3   0   0   0   0   0   0   0
   1
   12.2411
   8
   1   2   2   2   2   0   0   0   0   0
   1
   11.9102
   8
   1   1   1   1   1   1   1   1   1   0
   1
   11.6345
RESIDUAL-CODEBUECHER
====================
VEKTORLAENGE 8, 1.25 BIT/SAMPLE
-------------------------------
   8
   5   5   0   0   0   0   0   0   0   0
   1
   13.7229
   8
   2   2   2   2   2   0   0   0   0   0
   1
   13.3252
   8
   1   1   1   1   1   1   1   1   1   1
   1
   12.8206
```

8.3 Verzeichnis der benutzten Formelzeichen

k	Vektorlänge
x_i	Abtastwert
$\bar{x}$	Gleichspannungsanteil
$\vec{X}_i$	Sprachsignalvektor
L	Anzahl der Trainingsvektoren
$\vec{Y}_j$	Codebuchvektor
n_j	Anzahl der Vektoren des Clusters j
$d(\vec{X}_i, \vec{Y}_j)$	Abstand der Vektoren $\vec{X}_i$ und $\vec{Y}_j$
$\vec{A}$	Codebuch
$\vec{A}_i$	Codebuch auf der i-ten Knotenebene
$\vec{B}$	Baumstruktur
$\vec{C}$	Baumstrukturiertes Codebuch
m	Anzahl der Knotenebenen
N	Codebuchgröße
N'_i	Zahl der Aufspaltungen eines Knotens der (i-1)-ten Ebene
N_i	Anzahl der Vektoren auf der i-ten Knotenebene

R	Informationsgehalt pro Vektor
R_i'	Zahl der binären Entscheidungen, um von der (i-1)-ten auf die i-te Knotenebene zu gelangen
R_i	Zahl der binären Entscheidungen, um auf die i-te Knotenebene zu gelangen
r	Informationsgehalt pro Sample
E_{sig}	Signalenergie
E_{dif}	Störenergie
SNR	Signal/Rauschverhältnis
ε	Abbruchkriterium
n_B	Zahl der möglichen Baumstrukturen
n_d	Anzahl der zu berechnenden Abstände
n_x	Anzahl der Multiplikationen
n_+	Anzahl der Additionen
M	Speichergröße (Vektoren)
S	Speichergröße (Vektorkomponenten)