平成 11 年度 修士論文

後方適応予測による
音声ロスレス符号化の研究
（修正版）

指導教官 森本 宏 教授

名古屋大学大学院 人間情報学研究科
社会情報学専攻 電子社会システム論講座

木村 敏幸
学籍番号 319802124

提出日
2000 年 3 月 31 日

目次

第１章 序論

1.1. はじめに

これまでの音声メディアはCD、DAT等のパッケージ型で用いるのが主流であり、音声メディアを移動するにはパッケージをそのまま輸送するしか方法がなかった。しかし、近年のインターネットの普及によって、インターネットのような通信メディアにおいて音声データを扱うことが可能になってきた。これによって移動に要する時間が短くなり、また輸送に要するコストを大幅に削減することができるため、これからの音声メディアは通信メディアを中心に発展していく可能性がある。

しかし、デジタル化した音声データの量は莫大である。例を挙げて説明すると、CDはサンプリング周波数44.1kHz、量子化ビット16bitのステレオチャンネルというフォーマットである。つまり、1曲(5分とする)当たりのデータ量を計算してみると、

$$44100(\mathrm{sec}^{-1})\times 16(\mathrm{bit})\times 2(\mathrm{channel})\times 300(\mathrm{sec}) = 423360000(\mathrm{bit}) = 52920000(\mathrm{B})$$
$$= 51679.6875(\mathrm{kB}) = 50.46844482422(\mathrm{MB}) \approx 50(\mathrm{MB})$$

なんと50MB(フロッピー１枚が約1MBだからフロッピー50枚分)もの値になってしまう。これをそのまま送信させようとすると、一般のアナログ回線(28.8kbpsとする)では、

$$50\times 10^{6}(\mathrm{B}) \div \frac{28.8\times 10^{3}}{8}(\mathrm{B/sec}) \approx 1.39\times 10^{4}(\mathrm{sec}) \approx 3.86(\mathrm{hour})$$

つまり、たかだか５分の音声データを送信しようとしても約 4 時間もの時間がかかることになる。現在の日本における通信網の大部分はアナログ回線なので、このままでは音声データを通信メディアによって普及させることは非常に困難である事がお分かり頂けるであろう。

そこで、この欠点を改善するために考え出されたのが**音声符号化**である。つまり、音声データをデジタル化した時点でデータ量をあらかじめ少なくしておくという考えである(こういった作業のことを**圧縮**という)。こうすれば通信に要する時間を(送信量が元々少ないので)少なくすることができるので、アナログ回線でも音声データの送信が実用化できるようになる。

一方、音声符号化は通信メディアのみならず、パッケージ型メディアにも応用

されている。良く知られている例としては MD が挙げられる。MD は CD と演奏時間はほぼ同じだが、大きさが CD と比べると非常に小さいという特徴がある。つまり、音声符号化の技術はパッケージの大きさを小さくするのにも一役買っているという一面も持っている。

符号化技術にはその性質上、二通りの符号化が存在する。一つは人間の聴覚に感じない程度の劣化ならそのまま無視してしまうという符号化のことで、**不可逆符号化**という。現在の音声符号化の研究は主にこちらを採用しており、様々な圧縮方法が発表されている。例えば RealAudio[1]、MP3(MPEG Audio Layer Ⅲ)[2,P177]、TwinVQ[3]などである。しかし、一旦圧縮して、再び元に戻した後(この作業のことを**解凍**という)の音声データは圧縮前のそれと比べると若干の歪みが生じている。

もう一つは音質が全く劣化しないという符号化技術である。つまり、解凍後の音声データが圧縮前の音声データと全く同じになるという符号化のことで、**可逆符号化**または**ロスレス符号化**という。この方法は不可逆符号化レベルの劣化でも許されないような場合(例えば DVD-Audio のような高品質レベルの音声データの符号化など)に用いられる。劣化がないぶん、圧縮性能は不可逆符号化に比べ劣ることになる。

また、符号化の方法としても二通り存在する。一つは**予測符号化**と呼ばれるもので、過去の入力信号列から現在の入力を予測し、その予測値と入力値の誤差を送信するという方法である。もう一つは**変換符号化**と呼ばれるもので、ブロックごとに区切った入力信号ベクトルを行列変換して、変換した値を送信するというものである。

1.2. 本研究の目的

本研究では不可逆符号化に比べてあまり研究されていない、不可逆符号化の研究においては必ず必要とされる主観評価が不要である、などの理由でロスレス符号化の研究を行うことにする。ロスレス符号化用の圧縮ツールとして思い浮かべるのは LHA や ZIP であるが、これらは元々テキストデータなどを圧縮するために開発されているので、音声データを圧縮しようとしても思うような圧縮性能を得ることができない。そこで、音声ロスレス符号化を行うには音声の性質に特化した独特の圧縮ツールを開発する必要がある。

このようなことを踏まえた上での圧縮ツールがいくつか提供されているが[9][10]、本研究では提案した手法がこれらの従来の圧縮ツールと比べるとどのくらいの性能を達成することができるかということに重点を置いている。また、符号化の方法としては、数多くの前例が出ているという理由で、予測符号化を用いることにしている。

第２章 理論的背景

2.1. A/D変換[4,P9]

音声波形は時間 t の一次元関数である。

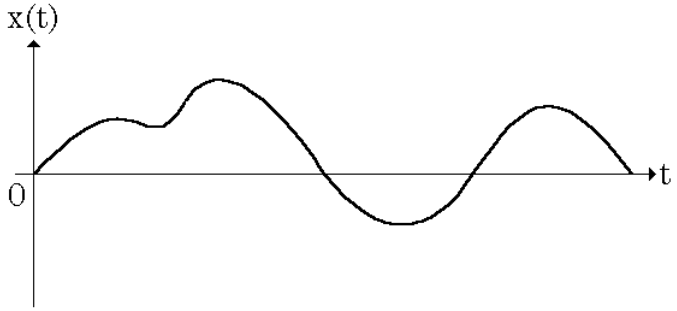


図 2-1 音声波形グラフ

しかし、通常の音声波形はアナログ信号であるので、時間 t や波形値 x(t)が共に連続値である。そのため、このままではコンピュータにデータを入力しようとしてもメモリの容量の関係で入れることができない。そのため、デジタル信号にするために連続値を離散値に変換する必要がある。この作業のことを**A/D変換**という。

A/D変換における最初の作業は**サンプリング**である。これは時間を連続値から離散値にすることである。具体的には、サンプリング周期 T ごとに波形値 x(t)を取り込む。この時のサンプリング周波数は 1/T(単位は sec^{-1}=Hz)になる。

サンプリングした後の波形値は x(nT)(n は任意の整数)となる。しかし、大抵の場合は T を省略した形で x(n)として表記するのが一般的である。以降の波形値はすべて x(n)で表すことにする。

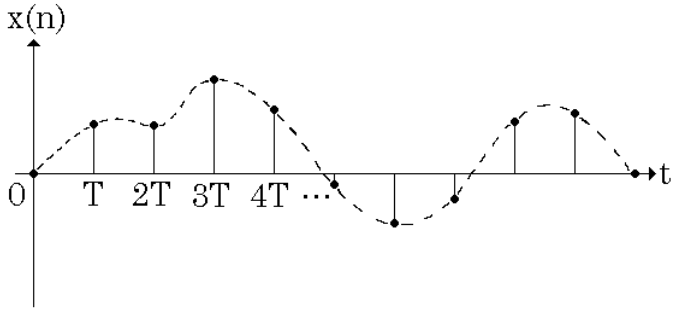


図 2-2 サンプリングした音声波形グラフ

しかし、サンプリングをしても波形値 x(n)は連続値のままである。そこで、今度は波形値を離散値に変換する必要がある。このことを**量子化**という。たいていの場合はステップサイズが常に一定である一様量子化器を用い、単位としては bit を用いる。N bit では 2^N 段階の量子化が可能になる。また、量子化後の波形値の値は整数にすることが多い。

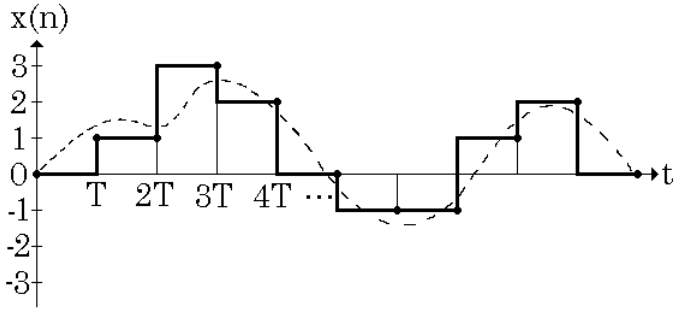


図 2-3 量子化した音声波形グラフ

以上によってアナログ信号をデジタル入力信号 x(n)に変換することができる。このようにしてデジタル化された音声信号のことを **PCM**(パルス符号変調：Pulse Code Modulation)方式という。

2.2. 予測符号化

一般に音声信号は**マルコフ情報源**である。マルコフ情報源とは過去の入力の生起確率が次の入力の生起確率に影響する情報源のことである[5,P51]。

マルコフ情報源を証明する重要な尺度として**自己相関関数**がある。自己相関関数とは次の式によって定義される関数のことである[4,P81]。

$$R(\tau) = E[x(n)x(n+\tau)] \quad (2.2.1)$$

τは 0 以上の整数である。特にτ =0 のとき、自己相関関数は入力信号列の分散σ^2に相当する。

$$R(0) = \sigma^2 = E[x(n)^2] \quad (2.2.2)$$

また、自己相関関数を R(0)で正規化したものが自己相関係数 r(τ)である。

$$r(\tau) = R(\tau)/R(0) \quad (2.2.3)$$

音声信号の自己相関係数をプロットすると、τが 0 に近い範囲では自己相関係数は 1 に近い(相関度が大きい)値になっている。

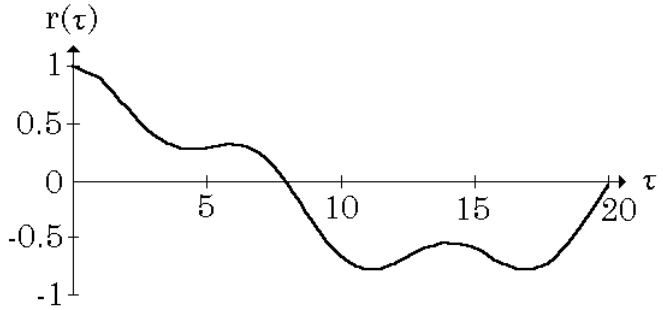


図 2-4 自己相関係数のグラフ(DING.WAV、3.4.参照)

このことから音声信号はマルコフ情報源であるということが言える。

このような音声信号の性質を利用した符号化が**予測符号化**である。代表的な予測符号化方式として挙げられるのが **DPCM**(差分 PCM : Differential PCM)である。DPCM とは過去の出力から予測した値と現在の入力の値との誤差を送信するという方法である。符号化回路としては下のようになる。

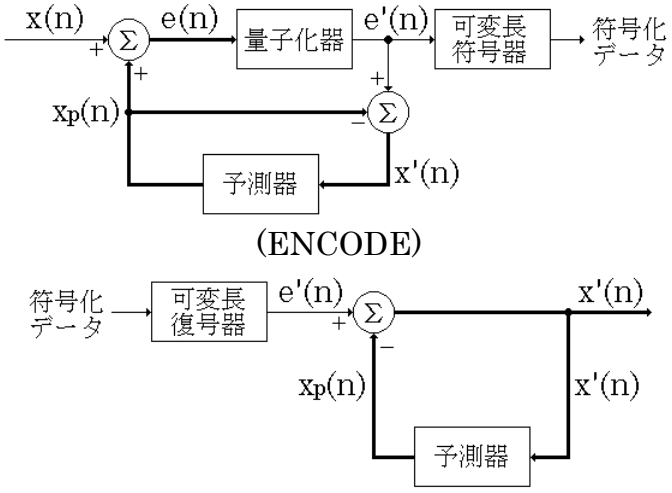


図 2-5 DPCM 回路図

x(n)は入力信号、x'(n)は出力信号、$x_p(n)$は線形予測値、e(n)は予測誤差、e'(n)は量子化予測誤差、q(n)(=e(n)-e'(n))は量子化誤差である。回路図の太線は実数値をとるという意味である。また、回路図より以下の式が成り立つ。

$$x_p(n) = \sum_{i=1}^{L} a_i x'(n-i)\ ,\ e(n) = x(n) + x_p(n)\ ,\ x'(n) = e'(n) - x_p(n)$$

a_i は予測係数である。具体的な符号化及び復号化の手順は次の通りに行う。

(ENCODE)

① 過去の出力信号列 x'(n-i)と予測係数 a_i を元に線形予測値 $x_p(n)$を計算する。
② 入力信号 x(n)を入力する。
③ 予測誤差 e(n)を x(n)+ $x_p(n)$によって計算する。
④ e(n)を量子化器にかけて、量子化予測誤差 e'(n)を生成する。
⑤ e'(n)を可変長符号器にかけて符号化データを生成する。
⑥ 符号化データを送信する。
⑦ e'(n)- $x_p(n)$より出力信号 x'(n)を算出する。
⑧ 過去の出力信号列をシフトし、x'(n)を加える。
⑨ 時間をシフトし、①から繰り返す。

(DECODE)

① x'(n-i)と a_i を元に $x_p(n)$を計算する。
② 符号化データを受け取る。
③ 符号化データを可変長復号器にかけて e'(n)を得る。
④ x'(n)を e'(n)- $x_p(n)$によって導く。
⑤ x'(n)を出力する。
⑥ 過去の出力信号列をシフトし、x'(n)を加える。
⑦ 時間をシフトし、①から繰り返す。

再生誤差(入力値と出力値の差)を計算すると、次のようになる。

$$
\begin{aligned}
\text{再生誤差} &= x(n) - x'(n) \\
&= e(n) - x_p(n) - [e'(n) - x_p(n)] \\
&= e(n) - e'(n) = q(n) \\
&= \text{量子化誤差}
\end{aligned}
$$

つまり、再生誤差は量子化誤差そのものになるので[2,P24]、いかに量子化誤差を少なくしながら効率の良い符号化方法を開発するかが予測符号化技術開発の焦点となる。

2.3. 適応予測符号化[7,P7]

先程の DPCM は一定の予測器を用いていた。つまり予測係数は常に一定であるという前提のもとにあった。ここでは予測係数は一定ではなく、入力信号が入力されるたびに適宜更新していくという予測符号化方式の説明をすることにする。このような符号化方式を **ADPCM**(適応差分 PCM : Adaptive Differential PCM)という。

これより最適な予測係数を数学的に導く方法を説明する[6]。予測誤差は以下の式によって生成される。

$$e(n) = x(n) + x_p(n) = x(n) + \sum_{i=1}^{L} a_i(n)x'(n-i) \quad (2.3.1)$$

予測係数も時間に依存する変数なので、時間に依存した形で表記する。ここでは量子化誤差はないと仮定するので、x(n)=x'(n)である。すなわち、

$$e(n) = x(n) + \sum_{i=1}^{L} a_i(n)x(n-i) \quad (2.3.2)$$

表記を見やすくするために以下のベクトルを定義することにする。

$$X = \left[x(n-1), x(n-2), \cdots, x(n-L)\right]^T$$
$$A = \left[a_1(n), a_2(n), \cdots, a_L(n)\right]^T$$

$[\ ]^T$は転置行列を表す。このベクトルを用いると、(2.3.2)式は、

$$e(n) = x(n) + X^T A = x(n) + A^T X \quad (2.3.3)$$

予測誤差のエネルギーは自乗平均誤差(=$E[\{e(n)\}^2]$)によって与えられる。(2.3.3)式より、

$$E[\{e(n)\}^2] = E[\{x(n)\}^2] + A^T E[XX^T]A + 2E[x(n)X^T]A$$
$$= E[\{x(n)\}^2] + A^T RA + 2P^T A \quad (2.3.4)$$

R は自己相関関数行列、P は相互相関ベクトルと呼ばれるものである。

$$R = \begin{pmatrix} E[x(n-1)x(n-1)] & E[x(n-1)x(n-2)] & \cdots & E[x(n-1)x(n-L)] \\ E[x(n-2)x(n-1)] & E[x(n-2)x(n-2)] & \cdots & E[x(n-2)x(n-L)] \\ \vdots & \vdots & \ddots & \vdots \\ E[x(n-L)x(n-1)] & E[x(n-L)x(n-2)] & \cdots & E[x(n-L)x(n-L)] \end{pmatrix}$$
$$= \begin{pmatrix} R(0) & R(1) & \cdots & R(L-1) \\ R(1) & R(0) & \cdots & R(L-2) \\ \vdots & \vdots & \ddots & \vdots \\ R(L-1) & R(L-2) & \cdots & R(0) \end{pmatrix}$$
$$P = \left[E[x(n)x(n-1)], E[x(n)x(n-2)], \cdots, E[x(n)x(n-L)]\right]^T$$
$$= \left[R(1), R(2), \cdots, R(L)\right]^T$$

予測誤差の最小値をとるためには、自乗平均誤差の極値をとればよい。(2.3.4)式を各予測係数で偏微分すると、

$$\nabla = \partial E[\{e(n)\}^2]/\partial A = \begin{pmatrix} \partial E[\{e(n)\}^2]/\partial a_1(n) \\ \partial E[\{e(n)\}^2]/\partial a_2(n) \\ \vdots \\ \partial E[\{e(n)\}^2]/\partial a_L(n) \end{pmatrix}$$
$$= 2RA + 2P \quad (2.3.5)$$

∇=0 として解くと、最適係数 A*が得られる。

$$RA = -P \quad (2.3.6)$$
$$\Updownarrow$$
$$A^* = -R^{-1}P \quad (2.3.7)$$

この時の自乗平均誤差は以下のようになる。

$$E[\{e(n)\}^2] = E[\{x(n)\}^2] - P^T R^{-1} P = E[\{x(n)\}^2] + P^T A^*$$
$$= \sigma^2 + P^T A^* \quad (2.3.8)$$

このように、自乗平均誤差を最小にする(つまり、圧縮効率を最大にする)最適係数は自己相関関数から求めることができる。自己相関関数を求めるには入力信号列が定常状態であることが要求される。しかし、実際の入力信号列は非定常状態であるので、長期的な範囲で統計を取っていては正しい自己相関関数を測ることができない。そこで、短期的な範囲において入力信号列を定常状態であると仮定して、その範囲内で自己相関関数を計算するという手法が取られている。この手法のことを**ブロック化**という。適応予測符号化技術においては計算に用いるブロックの長さ(ブロック長)をどれくらいにするかということも重要な要素の一つになってくる。

(2.3.6)式は L 元一次連立方程式となる。しかし、この方程式を一般のアルゴリズムで解いていたのでは、L の数が大きい時には演算量が多くなってしまうので実用的ではない。そこで、これらの連立方程式を高速に解く方法が提案されている。代表的なものとして挙げられるのが **Levinson-Durbin アルゴリズム**である。Levinson は対称テプリッツ行列による連立方程式が与えられた時に、方程式を高速に解く方法を提案した。Durbin は更に自己相関係数行列と相互相関係数ベクトルを用いる場合の連立方程式の高速法を提案した[7,P15]。これは(2.3.6)式の両辺を R(0)で正規化したものと同じになる。

$$\begin{pmatrix} r(0) & r(1) & \cdots & r(L-1) \\ r(1) & r(0) & \cdots & r(L-2) \\ \vdots & \vdots & \ddots & \vdots \\ r(L-1) & r(L-2) & \cdots & r(0) \end{pmatrix} \begin{pmatrix} a_1(n) \\ a_2(n) \\ \vdots \\ a_L(n) \end{pmatrix} = - \begin{pmatrix} r(1) \\ r(2) \\ \vdots \\ r(L) \end{pmatrix} \quad (2.3.9)$$

これらの提案法を総称して Levinson-Durbin アルゴリズムと呼んでいる。具体的な手順は次のようになる。予測係数 $a_i(n)$の括弧は省略してある。

(最初のループ、i=1)

$$w^{(1)} = r(1)$$
$$a_1^{(1)} = -w^{(1)}$$
$$u^{(1)} = 1 - \{w^{(1)}\}^2$$

(以降のループ、i=2～L)

$$w^{(i)} = \frac{r(i) + \sum_{j=1}^{i-1} a_j^{(i-1)} r(i-j)}{u^{(i-1)}}$$
$$a_j^{(i)} = a_j^{(i-1)} - w^{(i)} \times a_{(i-j)}^{(i-1)} \quad (j = 1, \cdots, i-1)$$
$$a_i^{(i)} = -w^{(i)}$$
$$u^{(i)} = (1 - \{w^{(i)}\}^2) u^{(i-1)}$$

u は自乗平均誤差である。a_i,u,w の上付きについている(i)は「i 次の」という意味である。w は **PARCOR 係数**と呼ばれるもので、この符号化システムの安定性を判別する尺度である。|w|<1 のとき、システムは安定である。

また、適応予測には二通りの方法がある。一つは現在を含めた入力信号列から自己相関係数を導いて係数を更新する方法である。この方法のことを**前方適応予測**という[7,P8]。具体的な符号化及び復号化の手順は以下の通りに行う。

(ENCODE)

① 入力信号列をシフトし、入力信号 x(n)を入力する。
② 入力信号列より自己相関係数 r(i)を求める。
③ Levinson-Durbin アルゴリズムによって予測係数 $a_i(n)$を求める。
④ 過去の出力信号列 x'(n-i)と $a_i(n)$を元に線形予測値 $x_p(n)$を計算する。
⑤ 予測誤差 e(n)を x(n)+ $x_p(n)$によって計算する。
⑥ e(n)を量子化器にかけて量子化予測誤差 e'(n)を生成する。
⑦ e'(n)及び $a_i(n)$を可変長符号器にかけて符号化データを生成する。
⑧ 符号化データを送信する。
⑨ e'(n)- $x_p(n)$より出力信号 x'(n)を算出する。
⑩ 過去の出力信号列をシフトし、x'(n)を加える。
⑪ 時間をシフトし、①から繰り返す。

(DECODE)

① 符号化データを受け取る。
② 符号化データを可変長復号器にかけて e'(n)及び $a_i(n)$を得る。

③ x'(n-i)と $a_i(n)$を元に $x_p(n)$を計算する。
④ x'(n)を e'(n)- $x_p(n)$によって導く。
⑤ x'(n)を出力する。
⑥ 過去の出力信号列をシフトし、x'(n)を加える。
⑦ 時間をシフトし、①から繰り返す。

この方法だと、現在の入力信号の統計情報が自己相関係数に含まれているため、正しい統計情報を導くことができる。また、復号化の手順が簡潔なため、デコーダの演算量が少なくて済む。しかし、予測係数を送信する必要があるため、予測係数に制約が出てくる。また、入力信号列と出力信号列を両方ともエンコーダに記憶させておく必要がある。

もう一方は過去の出力信号列から自己相関係数を導いて係数を更新する方法である。この方法のことを**後方適応予測**という[7,P8]。具体的な符号化及び復号化の手順は以下の通りに行う。

(ENCODE)
① 過去の出力信号列より自己相関係数 r(i)を求める。
② Levinson-Durbin アルゴリズムによって予測係数 $a_i(n)$を求める。
③ 過去の出力信号列 x'(n-i)と $a_i(n)$を元に線形予測値 $x_p(n)$を計算する。
④ 入力信号 x(n)を入力する。
⑤ 予測誤差 e(n)を x(n)+ $x_p(n)$によって計算する。
⑥ e(n)を量子化器にかけて量子化予測誤差 e'(n)を生成する。
⑦ e'(n)を可変長符号器にかけて符号化データを生成する。
⑧ 符号化データを送信する。
⑨ e'(n)- $x_p(n)$より出力信号 x'(n)を算出する。
⑩ 過去の出力信号列をシフトし、x'(n)を加える。
⑪ 時間をシフトし、①から繰り返す。

(DECODE)
① 過去の出力信号列より自己相関係数 r(i)を求める。
② Levinson-Durbin アルゴリズムによって予測係数 $a_i(n)$を求める。
③ 過去の出力信号列 x'(n-i)と $a_i(n)$を元に $x_p(n)$を計算する。
④ 符号化データを受け取る。
⑤ 符号化データを可変長復号器にかけて e'(n)を得る。
⑥ x'(n)を e'(n)- $x_p(n)$によって導く。
⑦ x'(n)を出力する。
⑧ 過去の出力信号列をシフトし、x'(n)を加える。
⑨ 時間をシフトし、①から繰り返す。

この方法では、予測係数は任意の値を取ることができるため、最適係数を用いることができる。また、量子化予測誤差しか送る必要が無いので、その分送信

量を少なくすることができる。しかし、復号化の時にも Levinson-Durbin アルゴリズムを使用するので、その分デコーダの演算量が増大する。また、入力信号と過去の出力信号列の統計情報が著しく異なっている場合、最適な係数を算出することができないという欠点を持っている。

2.4. 可変長符号化[5,P71]

2.3.で量子化予測誤差 e'(n)を導くことができた。しかし、実際の音声符号化では e'(n)を D 進数列の符号化データに変換して送信する必要がある。この作業を**符号化**という。また、逆の作業を**復号化**という。実際の符号化の場合、符号化データは二進数表記で表すことが多い。このため、後の符号化データはすべて二進数列(**ビット列**ともいう)に変換するものとする。

符号化は二つに大別することができる。一つは個々の e'(n)に対してすべて同じビット数の符号語を割り当てるものである。この符号化のことを**固定長符号化**(または**ブロック符号化**)という。もう一つの符号化は、個々の e'(n)に対して違ったビット数の符号語を割り当てるものである。この符号化のことを**可変長符号化**という。

音声符号化においては、大抵の場合は e'(n)は統計上の偏りを持っている。そこで、生起確率の大きい値には短いビット列を、逆に生起確率の小さい値には長いビット列を割り当てれば、全体としての平均符号長は固定長符号化を用いた時よりも小さくすることができる。このような理由で可変長符号化を用いれば更に効率の良い符号化を行うことが可能になる。

一般に、M 個のシンボル $A=\{a_1,a_2,\cdots,a_M\}$ があり、それぞれの生起確率を $\{p(a_1),p(a_2),\cdots,p(a_M)\}$、符号長を $\{l(a_1),l(a_2),\cdots,l(a_M)\}$($l(a_i)$は整数)とすると、平均符号長 L(A)は次のように定義できる。

$$L(A)=\sum_{i=1}^{M} l(a_i)p(a_i) \quad (2.4.1)$$

それでは、実用的な可変長符号化とはどのようなものなのだろうか。それには以下の二つの条件が挙げられる。まず一つは与えられたビット列がただ一通りに復号できることである。このことを**一意復号可能**という。もう一つはあるシンボルを受信したら即座に決定することができることである。このことを**瞬時復号可能**といい、そのような符号のことを**瞬時符号(語頭符号)**という。

このことを説明するために、4 つのシンボル(A,B,C,D)の符号化を考えてみよう。まず、このような符号語を考える。

A=0 , B=1 , C=10 , D=01

この符号語の時に 0101 を受信すると、

$$0-1-0-1 \rightarrow \mathrm{ABAB}$$
$$0-1-01 \rightarrow \mathrm{ABD}$$
$$0-10-1 \rightarrow \mathrm{ACB}$$
$$01-0-1 \rightarrow \mathrm{DAB}$$
$$01-01 \rightarrow \mathrm{DD}$$

という風に 5 通りもの意味に解釈できてしまう。つまり、この符号は一意復号可能ではない。これを防ぐためには区切り用の特別な符号を送信する必要があるが、それでは平均符号長が更に大きくなってしまうので効率的な符号化には適していない。

次にこのような符号語を考える。

$$\mathrm{A=0\ ,\ B=01\ ,\ C=011\ ,\ D=0111}$$

この時に 00 を受信すると、まず最初の 0 を受信した時点で A が受信される。しかし、この 0 は B,C,D の接頭語であるので、もしかしたら A ではないかもしれない。そこで次のビットを受信したところ、0 であったので、B,C,D にはなり得ないことが分かり(00 は B,C,D の接頭語ではないため)、A が決定できる。このような符号は瞬時符号ではない。この符号を用いた場合、シンボルが決定できるまでの間、過去のビットを記憶しておかなければならないので効率的ではない。

最後にこのような符号を考える。

$$\mathrm{A=0\ ,\ B=10\ ,\ C=110\ ,\ D=111}$$

この時に 0 を受信すると、A は B,C,D の接頭語ではないので、瞬時に A を決定することができる。このように実用的な可変長符号化は一意復号可能で、かつ瞬時符号であることが条件となる。

一意復号可能かつ瞬時符号であるための条件を示す数式として**クラフトの不等式**がある。クラフトの不等式は次のように定義される。

「M 個のシンボル $A=\{a_1,a_2,\cdots,a_M\}$ を D 進数列で符号化した時の符号長をそれぞれ $\{l(a_1),l(a_2),\cdots,l(a_M)\}$ とすると、符号語が一意復号可能かつ瞬時復号であるための必要十分条件は、

$$\sum_{i=1}^{M}\frac{1}{D^{l(a_i)}}\le 1 \quad (2.4.2)$$

である。」

特にビット列で符号化した時のクラフトの不等式は、

$$\sum_{i=1}^{M} \frac{1}{2^{l(a_i)}} \le 1 \quad (2.4.3)$$

となる。クラフトの不等式によって、最適な可変長符号を作ることができる。

一般に、生起確率と符号長との間には以下のような関係が成り立つ。

$$2^{-l(a_i)} \le p(a_i) < 2^{-l(a_i)+1} \quad (i = 1, \cdots, M) \quad (2.4.4)$$

底を 2 とした対数で取ると、

$$-l(a_i) \le \log_2 p(a_i) < -l(a_i) + 1 \quad (2.4.5)$$
$$l(a_i) \ge -\log_2 p(a_i) > l(a_i) - 1 \quad (2.4.6)$$

平均符号長を基準にすると、

$$-\log_2 p(a_i) \le l(a_i) < -\log_2 p(a_i) + 1 \quad (2.4.7)$$

平均値を算出すると、

$$-\sum_{i=1}^{M} p(a_i) \log_2 p(a_i) \le L(A) < -\sum_{i=1}^{M} p(a_i) \log_2 p(a_i) + 1 \quad (2.4.8)$$

ここで**エントロピー**という概念を導入する。エントロピーは下の式によって定義される。

$$H(A) = -\sum_{i=1}^{M} p(a_i) \log_2 p(a_i) \quad (2.4.9)$$

(2.4.9)式を使って(2.4.8)式を整理すると、

$$H(A) \le L(A) < H(A) + 1 \quad (2.4.10)$$

(2.4.10)式の等号が成立するのは、各生起確率が 2 の整数乗で表すことができる時である。

$$p(a_i) = 2^{-l(a_i)} \quad (i = 1, \cdots, M) \quad (2.4.11)$$

このように、可変長符号化の上限と下限を導くことができる。

第３章 音声ロスレス符号化の研究

3.1. 予測ロスレス符号化

ロスレス符号化とは入力信号と出力信号の再生誤差が全く無い符号化のことである。ここでは入力信号と出力信号にはデジタル信号を用いるという前提の元で、予測符号化のロスレス型についての説明をすることにする。

予測符号化の代表的方法である DPCM は図 2-5 のような回路図によって構成されている。この回路図の場合には再生誤差は量子化誤差そのものになるので、ロスレス符号化を達成するためには量子化誤差(量子化器によって生じる誤差)を 0 にしなければならない。つまり、DPCM 回路から量子化器を取り除いてやる必要がある。

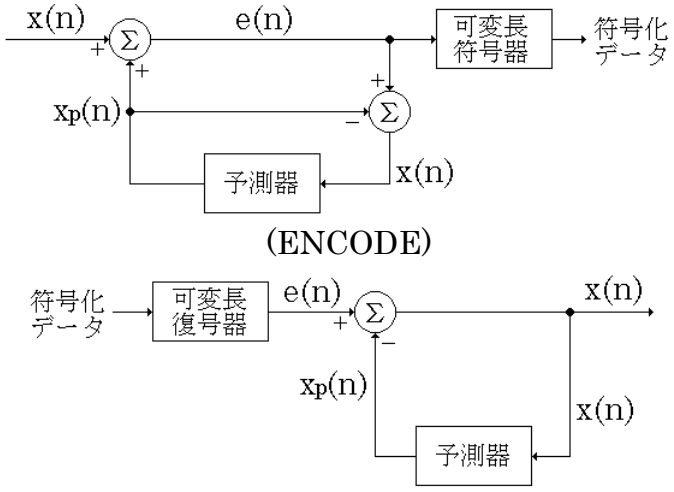


図 3-1 ロスレス版 DPCM 回路図

x(n)は入力及び出力信号、$x_p(n)$は線形予測値、e(n)は予測誤差である。回路図の関係式として次の式が成り立つ。

$$e(n) = x(n) + x_p(n) \Leftrightarrow x(n) = e(n) - x_p(n) \quad (3.1.1)$$

$$x_p(n) = \sum_{i=1}^{L} a_i x(n-i) \quad (3.1.2)$$

具体的な符号化及び復号化の手順としては、次の通りに行う。

(ENCODE)

① 過去の入力信号列 x(n-i)と予測係数 a_i を元に線形予測値 $x_p(n)$を計算する。
② 入力信号 x(n)を入力する。
③ 予測誤差 e(n)を x(n)+ $x_p(n)$によって計算する。
④ e(n)を可変長符号器にかけて符号化データを生成する。
⑤ 符号化データを送信する。
⑥ 過去の入力信号列をシフトし、x(n)を加える。
⑦ 時間をシフトし、①から繰り返す。

(DECODE)

① 過去の出力信号列 x(n-i)と a_i を元に $x_p(n)$を計算する。
② 符号化データを受け取る。
③ 符号化データを可変長復号器にかけて e(n)を得る。
④ 出力信号 x(n)を e(n)- $x_p(n)$によって導く。
⑤ x(n)を出力する。
⑥ 過去の出力信号列をシフトし、x(n)を加える。
⑦ 時間をシフトし、①から繰り返す。

この回路図によって予測ロスレス符号化を達成することができる。しかし、同時に色々な制約も出てくる。一番の制約が生じるのは予測係数である。

x(n)は整数値である。また、e(n)も整数値である。そのため、(3.1.1)式から $x_p(n)$は整数値でなくてはいけない。そのために、(3.1.2)式からも分かるように、予測係数はすべて整数値である必要がある。2.3 節において最適予測係数は数学的に算出することができると説明したが、その場合の値はほとんど実数値である。そのため、実数値を整数値に変換して用いたとしても、その分の誤差が生じてしまうため、結果として圧縮効率が悪くなってしまう。そのため、実際には予測係数に実数値を用いても予測ロスレス符号化が可能であるような回路図を考える必要がある。

ここで、図 2-5 の量子化予測誤差 e'(n)について考えよう。e'(n)は次の式によって与えられる。

$$\begin{aligned} e'(n) &= \lfloor e(n) \rfloor = \lfloor x(n) + x_p(n) \rfloor \\ &= e(n) - q(n) = \{x(n) + x_p(n)\} - q(n) \quad (3.1.3) \end{aligned}$$

q(n)は e(n)(実数値)を e'(n)(整数値)に変換したために生じた量子化誤差である。⌊e(n)⌋は、「e(n)以下の最も大きな整数」という意味である。この作業のことを**丸め込み**という。これをグラフで表すと次のようになる。

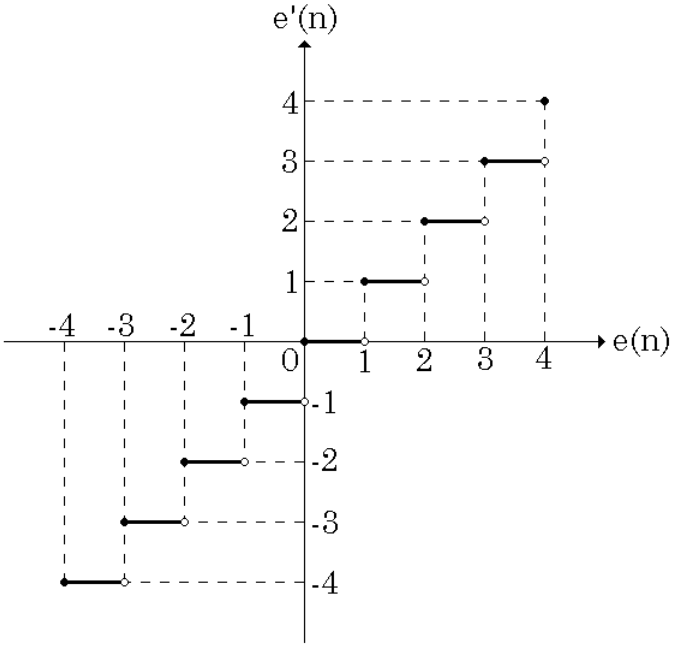


図 3-2 丸め込み関数のグラフ

つまり、q(n)は e(n)の小数部に相当し、0≤q(n)<1 である。

ここで(3.1.3)式の x(n)に注目する。x(n)は整数値であるので、小数部は 0 である。つまり、e(n)の小数部に作用する要素は $x_p(n)$だけである。ゆえに、e(n)を丸め込んでも $x_p(n)$のみを丸め込んでも e'(n)の値は変化しないということが言える。

$$e'(n) = x(n) + \lfloor x_p(n) \rfloor = x(n) + \{x_p(n) - q(n)\} \quad (3.1.3)$$

ここで $x_p(n)$-q(n)は整数値になるので、新たな変数 $x'_p(n)$を定義する。これを量子化線形予測値と呼ぶことにする

$$x'_p(n) = \lfloor x_p(n) \rfloor = x_p(n) - q(n) \quad (3.1.4)$$

$x_p(n)$ の $x'_p(n)$への丸め込みは図 3-2 となる。ゆえに、e'(n)を e(n)に書き改めると、予測誤差の算出式が導かれる。

$$e(n) = x(n) + x'_p(n) \quad (3.1.5)$$

$$x(n) = e(n) - x'_p(n) \quad (3.1.6)$$

$$x'_p(n) = \lfloor x_p(n) \rfloor = \left\lfloor \sum_{i=1}^{L} a_i x(n-i) \right\rfloor \quad (3.1.7)$$

符号化及び復号化の手順は次のように行う。

(ENCODE)

① 過去の入力信号列 x(n-i)と予測係数 a_i を元に線形予測値 $x_p(n)$を計算する。
② $x_p(n)$を丸め込んで量子化線形予測値 $x'_p(n)$を生成する。
③ 入力信号 x(n)を入力する。
④ e(n)を x(n)+ $x'_p(n)$によって計算する。
⑤ e(n)を可変長符号器にかけて符号化データを生成する。
⑥ 符号化データを送信する。
⑦ 過去の入力信号列をシフトし、x(n)を加える。
⑧ 時間をシフトし、①から繰り返す。

(DECODE)

① 過去の出力信号列 x(n-i)と a_i を元に $x_p(n)$を計算する。
② $x_p(n)$を丸め込んで $x'_p(n)$を生成する。
③ 符号化データを受け取る。
④ 符号化データを可変長復号器にかけて、e(n)を得る。
⑤ 出力信号 x(n)を e(n)- $x'_p(n)$によって計算する。
⑥ x(n)を出力する。
⑦ 過去の出力信号列をシフトし、x(n)を加える。
⑧ 時間をシフトし、①から繰り返す。

この符号化を回路によってあらわすと、図 3-3 のようになる。

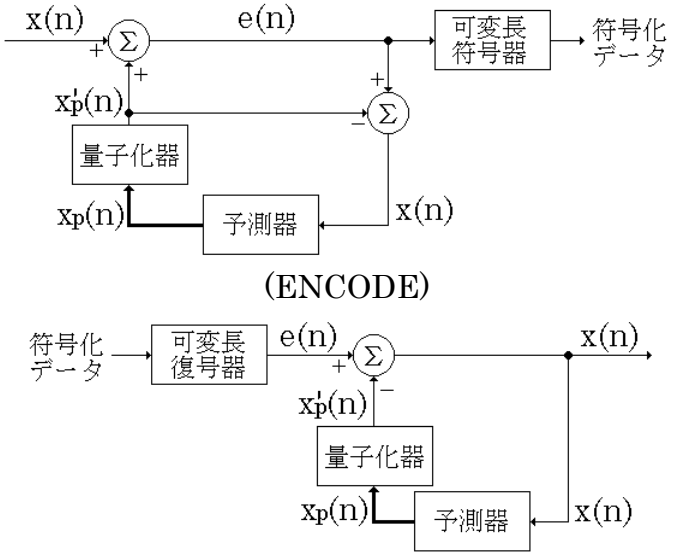


図 3-3 改良したロスレス版 DPCM 回路図

この回路ではデコーダにエンコーダのと同じ量子化器を組み込む必要があるが、量子化器は一定なので、デコーダの演算量はあまり増加しない。

この回路の長所は、$x_p(n)$を実数値のまま、つまり、予測係数 a_i を実数値のまま用いることができることである。先程のシステムでは予測係数は整数値でなければならなかったため、最適係数にほど遠い値を用いざるを得なかったが、このシステムでは最適係数をそのまま用いるため、結果として先のシステムよりも良い圧縮効率を見込むことができる。

しかし、まだ改善する点がある。それは丸め込みの部分である。一般の音声符号化の場合、予測誤差の統計分布を取ってみると、0 を中心に対称に偏っている傾向がある。丸め込みというのはこの統計分布の一定区間における確率の合計(積分でいう面積)を取ることである。

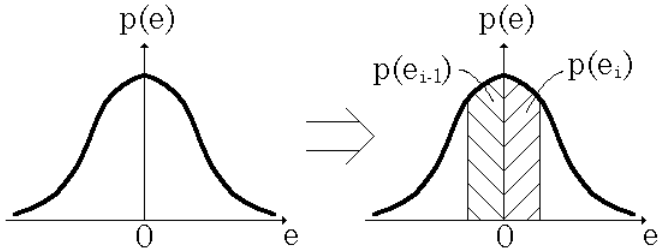


図 3-4 丸め込みの概念図

統計分布の偏りが大きくなるほどエントロピーは小さくなる。つまり、特定の区間での確率を大きくする必要がある。今までの丸め込みでは一番大きい所は $0 \le e < 1$ もしくは $-1 \le e < 0$ であったが、ステップ幅が同じである場合、$-0.5 \le e < 0.5$ と取ったほうが確率の値は大きくなる。そこで、本研究では図 3-2 の丸め込み関数ではなくて、-0.5 だけシフトした丸め込み関数を用いることにする。グラフにすると図 3-5 のようになる。

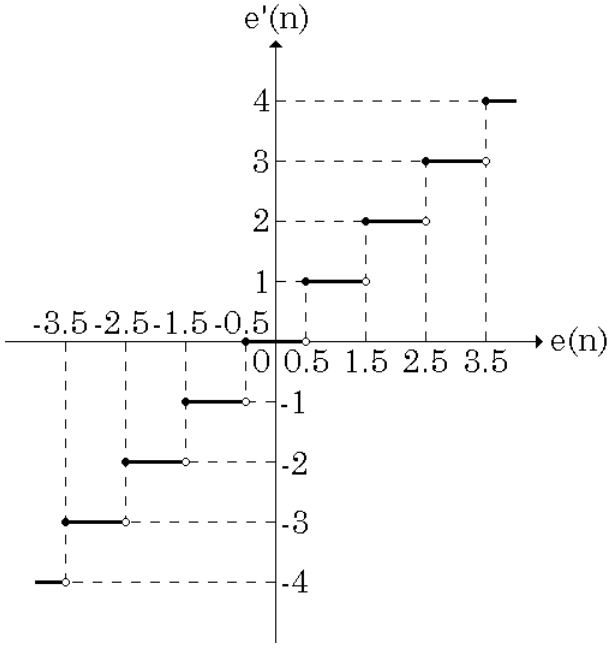


図 3-5 修正した丸め込み関数のグラフ

このときの(3.1.7)式は次のように変形される。

$$x'_p(n) = \lfloor x_p(n) + 0.5 \rfloor = \left\lfloor \sum_{i=1}^{L} a_i x(n-i) + 0.5 \right\rfloor \quad (3.1.8)$$

符号化、復号化にはそれぞれ(3.1.5)、(3.1.6)式を同じように用いる。

3.2. 適応予測符号化

2.3.と同じように、予測ロスレス符号化にも適応予測符号化を導入することによって圧縮効率を更に高めることができる。適応予測には二通りの方法(前方適応予測と後方適応予測)があることは前に説明した。ロスレス符号化においては入力信号と出力信号は同じになるので、自己相関関数を算出するには共に x(n)列を用いる。ゆえに、前方適応予測と後方適応予測の違いは現在の入力信号を自己相関関数に参照するかどうかとなる。予測ロスレス符号化でのそれぞれの符号化及び復号化手順を示すと、

(前方適応予測の場合)

(ENCODE)

① 入力信号列をシフトし、入力信号 x(n)を入力する。
② 入力信号列より自己相関係数 r(i)を求める。
③ Levinson-Durbin アルゴリズムによって予測係数 $a_i(n)$を求める。
④ 過去の入力信号列 x(n-i)と $a_i(n)$を元に線形予測値 $x_p(n)$を計算する。
⑤ $x_p(n)$を丸め込んで量子化線形予測値 $x'_p(n)$を生成する。
⑥ 予測誤差 e(n)を $x(n)+ x'_p(n)$によって計算する。
⑦ e(n)及び $a_i(n)$を可変長符号器にかけて符号化データを生成する。
⑧ 符号化データを送信する。
⑨ 時間をシフトし、①から繰り返す。

(DECODE)

① 符号化データを受け取る。
② 符号化データを可変長復号器にかけて e(n)及び $a_i(n)$を得る。
③ x(n-i)と $a_i(n)$を元に $x_p(n)$を計算する。
④ $x_p(n)$を丸め込んで量子化線形予測値 $x'_p(n)$を生成する。
⑤ 出力信号 x(n)を $e(n)- x'_p(n)$によって導く。
⑥ x(n)を出力する。
⑦ 過去の出力信号列をシフトし、x(n)を加える。
⑧ 時間をシフトし、①から繰り返す。

(後方適応予測の場合)

(ENCODE)

① 過去の入力信号列より自己相関係数 r(i)を求める。

② Levinson-Durbin アルゴリズムによって予測係数 $a_i(n)$を求める。
③ 過去の入力信号列 x(n-i)と $a_i(n)$を元に線形予測値 $x_p(n)$を計算する。
④ $x_p(n)$を丸め込んで量子化線形予測値 $x'_p(n)$を生成する。
⑤ 入力信号 x(n)を入力する。
⑥ 予測誤差 e(n)を $x(n)+ x'_p(n)$によって計算する。
⑦ e(n)を可変長符号器にかけて符号化データを生成する。
⑧ 符号化データを送信する。
⑨ 過去の入力信号列をシフトし、x(n)を加える。
⑩ 時間をシフトし、①から繰り返す。

(DECODE)
① 過去の出力信号列より自己相関係数 r(i)を求める。
② Levinson-Durbin アルゴリズムによって予測係数 $a_i(n)$を求める。
③ 過去の出力信号列 x(n-i)と $a_i(n)$を元に $x_p(n)$を計算する。
④ $x_p(n)$を丸め込んで量子化線形予測値 $x'_p(n)$を生成する。
⑤ 符号化データを受け取る。
⑥ 符号化データを可変長復号器にかけて e(n)を得る。
⑦ x(n)を $e(n)- x'_p(n)$によって導く。
⑧ x(n)を出力する。
⑨ 過去の出力信号列をシフトし、x(n)を加える。
⑩ 時間をシフトし、①から繰り返す。

今までの適応予測ロスレス符号化の研究では、前方適応予測を用いた研究報告例は見つかったが[8]、後方適応予測についての報告例はなかった。本研究では報告例が見あたらない、予測係数に任意の値を用いるのでより良い圧縮性能を見込むことができるという理由で後方適応予測による予測ロスレス符号化を採用することにする。

3.3. 可変長符号化

2.4.と同様に、今度は予測誤差 e(n)を符号化データとしてビット列に変換する必要がある。ここでは、音声符号化のケースに基づいた可変長符号化の方法を説明することにする。

一般的な可変長符号化の方法としては**ハフマン符号化**や**算術符号化**などがあるが、これらの符号化は入力系列がある一定の範囲内にのみ発生する時に使用することができるものである。つまり、今回のように発生する範囲を特定できないような場合において使用するにあたっては、予測誤差全体を一旦走査して予測誤差の発生する範囲を調べなくてはならない。しかし、その作業のおかげで符号化時間が更に長くなってしまう。そのため、これらの符号化は実用的ではない。そこで、今回のような場合は予測誤差の範囲が未定のままでも即座に符号化でき、かつ瞬時に一意復号可能である符号を使用する必要がある。

一般的な音声符号化の場合、予測誤差の統計分布はラプラス分布になる傾向がある[8]。ラプラス分布の生起確率は次のような式で表すことができる。但し、予測誤差 e は連続値である。

$$p(e) = \frac{1}{\sqrt{2\sigma^2}} \exp\left(-\sqrt{\frac{2}{\sigma^2}} |e| \right) \quad (3.3.1)$$

(3.3.1)式は以下の条件を満たしている。

$$\int_{-\infty}^{\infty} p(e)de = 1 \quad E[e] = \int_{-\infty}^{\infty} ep(e)de = 0 \quad E[e^2] = \int_{-\infty}^{\infty} e^2 p(e)de = \sigma^2$$

平均符号長の下限はエントロピーに収束することは 2.4 節で説明した。平均符号長の定義式とエントロピーの定義式を比べてみると、各符号長 l(e)は$-\log_2 p(e)$に相当することが分かる。

$$L(E)\left(= \int_{-\infty}^{\infty} p(e)l(e)de \right) \quad \geq \quad H(E)\left(= -\int_{-\infty}^{\infty} p(e)\log_2 p(e)de \right) \quad (3.3.2)$$

E は e の集合を表す。(3.3.1)式を変形すると、

$$-\log_2 p(e) = \sqrt{\frac{2}{\sigma^2}} \cdot \frac{|e|}{\ln 2} + \log_2 \sqrt{2\sigma^2} \quad (3.3.3)$$

つまり、(3.3.3)をグラフに表すと y 軸対称の直線となる。ゆえに、音声符号化のような場合には符号長が直線的に変化していく(かつどんな予測誤差の値でも符号化できる)コード表を設定する必要がある。

SHORTEN[8]では、先に述べたようなコード表が使用されている。具体的には以下の図に従って符号語が作られていく。ここでの予測誤差 e は整数値である。

	sign bit	lower bit	number of '0's	end bit
codeword (e)=	0(e>=0) 1(e<0)	|e|mod 2^n を２進表記	|e|¥2^nの数だけ 0を連ねる	1
	1	n	|e|¥2^n	1

図 3-6 SHORTEN で用いられたコード表

|e| は e の絶対値、A¥B は A÷B の商、A mod B は A÷B の剰余である。n は符号長の直線的変化の度合いを示すパラメータである(n は負でない整数)。n の値が大きいほど符号長の変化はゆるやかになる。

しかし、図 3-6 のコード表では 0 のような符号のない値でさえも sign bit を要するという欠点が生じる。そこで、本研究ではこの欠点を改良したコード表を用いることにする。具体的には sign bit を符号語の一番後ろに挿入するというものである。こうすることによって、0 の時には sign bit を付ける必要がなくなる。図に示したものが図 3-7 である。

	upper bit	number of '0's	end bit	sign bit
codeword (e)=	$\|e\| \bmod 2^n$ を２進表記	$\|e\| ¥ 2^n$の数だけ 0を連ねる	1	0(e>0) 1(e<0)
	n	$\|e\| ¥ 2^n$	1	0 or 1

図 3-7 本研究で用いたコード表

主な場合の符号木は以下の図のようになる。

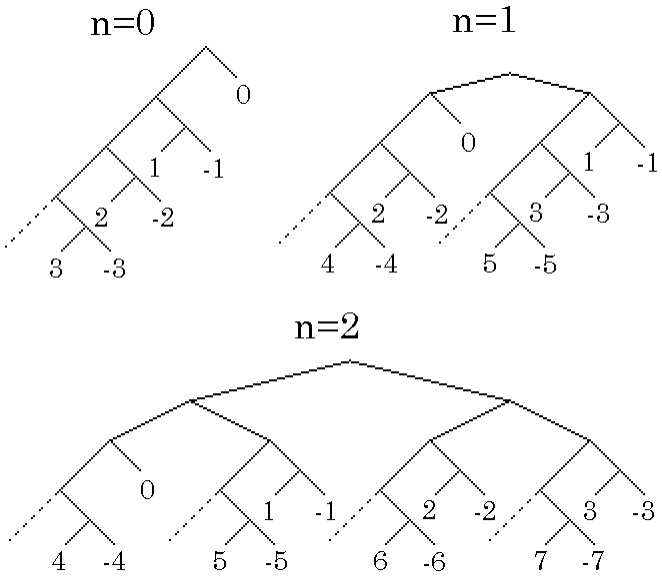


図 3-8 図 3-7 の主な場合の符号木

また、予測誤差ごとの符号長をグラフに表すと、次のようになる。

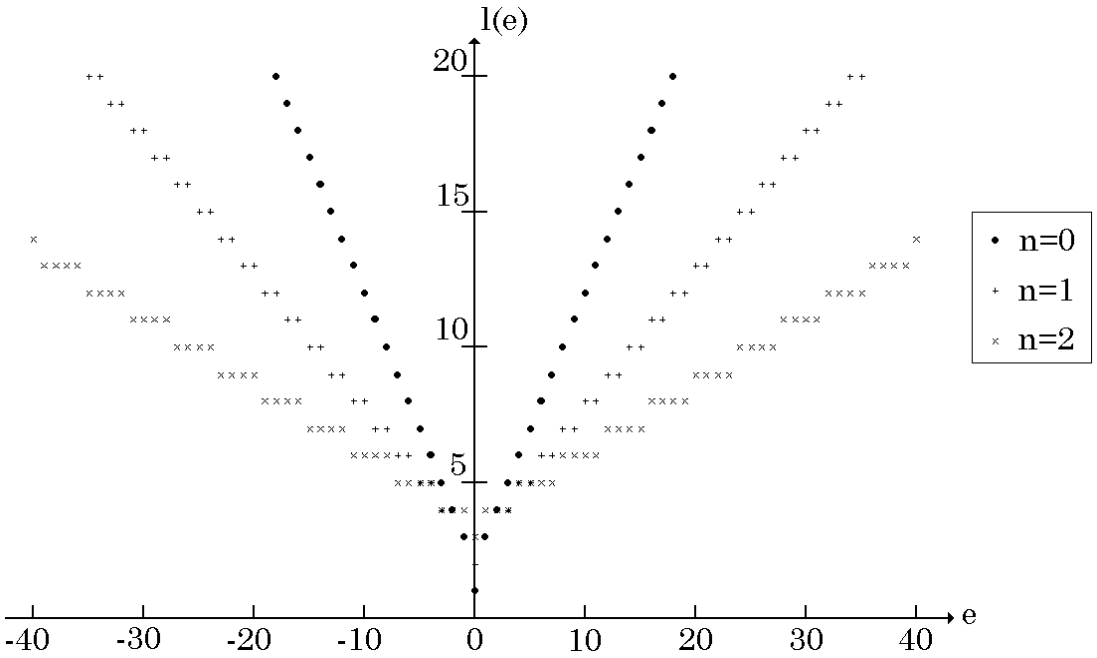


図 3-9 図 3-7 の符号長のグラフ

このように、符号長が直線的に変化しているので、ラプラス分布の可変長符号化にはこのコード表が最も適していることが分かる。

それではパラメータ n はどうやって導くのだろうか。
次のような定義を考えることにする。ある範囲(-2^n～2^n)において生起確率 p(x)の和が閾値 P(0<P<1)に到達した時に n の値が切り替わることにする[8]。これを数式にすると次のようになる。

$$\begin{aligned} P &= \int_{-2^n}^{2^n} p(x)dx \\ &= \int_{-2^n}^{2^n} \frac{1}{\sqrt{2\sigma^2}} \exp\left(-\sqrt{\frac{2}{\sigma^2}}\,|x|\right)dx \\ &= -\exp\left(-\sqrt{\frac{2}{\sigma^2}}\,2^n\right)+1 \quad (3.3.4) \end{aligned}$$

これより以下の式が導出される。

$$n = \log_2\left(-\ln(1-P)\sqrt{\frac{\sigma^2}{2}}\right) \quad (3.3.5)$$

$\sigma^2=E[\{x(n)\}^2]$は入力信号列及び出力信号列より算出できるので、n の値は随時変更することができる。

3.4. 実験

それではここからは実際に行った実験の手順と結果を順次説明していくことにする。

3.4.1. 実験方法

前節までで説明した符号化回路、後方適応予測、可変長コード表を用いて予測ロスレス符号化を行う。具体的な実験の方法としては WAV 形式(Windows OS での音声ファイル形式)のサンプルファイルを圧縮して、別の圧縮ファイルを形成するという手法を用いた。それを実行するために、コンピュータ上でソースコードをコンパイルしてプログラムを作り、プロンプト上で実行するという方式をとった。プログラム言語には C 言語(ANSI 対応)を用いた。コンパイラには Microsoft Visual C++ 6.0 を用いた。使用したコンピュータは GATEWAY 2000 G6-233 である。主なスペックは以下の通りである。

CPU：PentiumⅡ(233MHz)
Main memory：64MB
OS：Windows 95 4.00.950 B

サンプルファイルにはすべての人が同じサンプルファイルを扱うことができるという理由で、Windows OS に標準装備されている 4 種類の音声ファイル(CHIME.WAV、CHORD.WAV、DING.WAV、TADA.WAV)を用いた。このファイルの音声フォーマットはサンプリング周波数 22.05kHz、量子化ビット 8bit のモノラルファイルである。ゆえに、プログラムも 8bit、モノラル用に作成してある。

作成したソースコードは別に付属しておく。このソースコードを MS-DOS プロンプト上でコンパイルすると、以下の図のようになる。

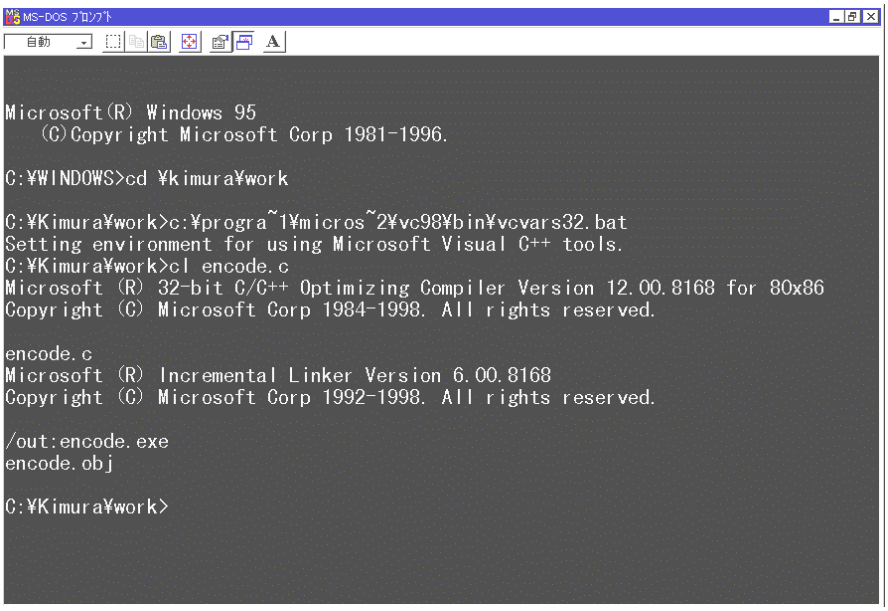


図 3-10 ソースコードをコンパイルした時の画面

作成されたプログラムはプロンプト上でこのように入力する。

＞encode (サンプルファイル名) (圧縮ファイル名)

このようにして実行すると、用いたパラメータの値、サンプルファイルの音声フォーマットとサンプル数、圧縮ファイルサイズと圧縮率、そして符号化時間が表示される。なお、圧縮ファイルはサンプルファイルのヘッダ(56Byte)とサンプルの圧縮ファイルを入れた形で作成している。実行された時の画面が図 3-11 に示してある。

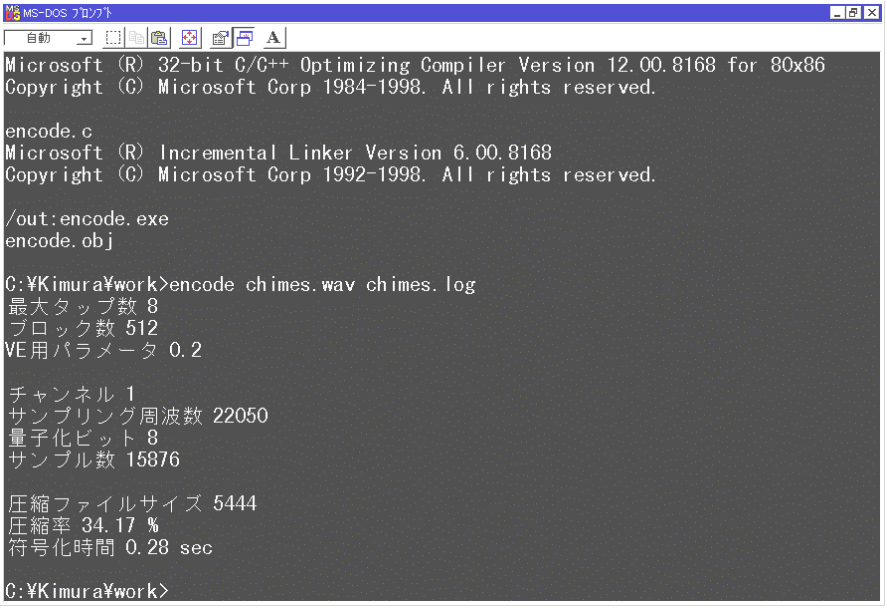


図 3-11 実行した時の画面

3.4.2. パラメータ同定のための実験

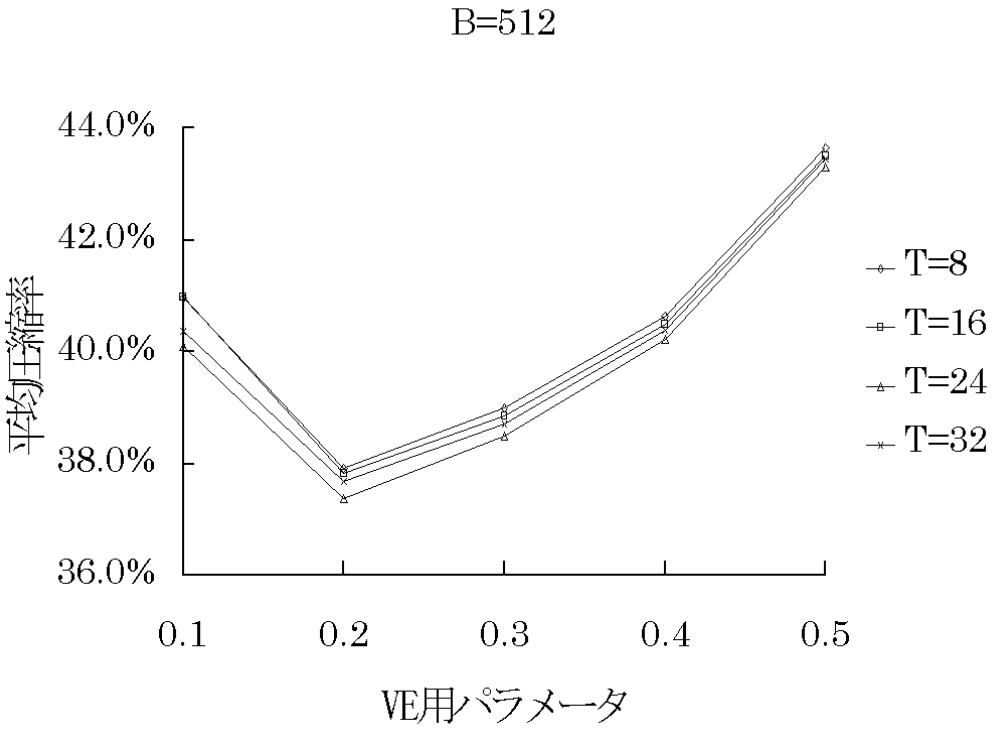
この実験で用いられるパラメータは最大タップ数（(3.1.2)の L に相当、線形予測値を算出するのに必要)、ブロック数（自己相関関数を算出するのに必要)、コード表閾値（(3.3.4)の P に相当)の三つである。実際の圧縮率はこの三つの変数に依存しているので、実験によって最適な変数値を導こうとしてもプロット数が多すぎるので対応しきれない。かといって数学的に導くこともできない。そこで、最適な変数値を導くために、次の性質を利用する。

「符号化時間は最大タップ数（以下これを T と呼ぶ）のみに比例する」

これは、予測係数を求めるには L 回ループをしなければならないが、自己相関関数は漸化式で算出するので、符号化時間はブロック数(以下これを B と呼ぶ)に依存しない。また、コード表閾値(以下これを P と呼ぶ)は(3.3.5)の n を導くのに用いているだけなので、これも符号化時間に依存することはない、ということから来ている。つまり、実用化の面では符号化にどれだけ時間がかかるかという段階に従って T を固定しても構わないという事が言える。この時点で圧縮率は T,B,P の三変数関数から B,P の二変数関数に変化する。

今度は B,P の最適値を導くわけだが、片方の変数を固定して圧縮率を一変数関数とみなした時の極小値を最適値とする。こうすることによって最適値を導くことができる。

このことを踏まえた上で T、B、P の最適値を導出してみよう。まず T を 8 の倍数(=8、16、24、32)に固定する。次に B を 512、1024 に固定して、P を 0.1、0.2、0.3、0.4、0.5 と変化させたときの平均圧縮率の変化をグラフにしてみると、図 3-12 のようになる。



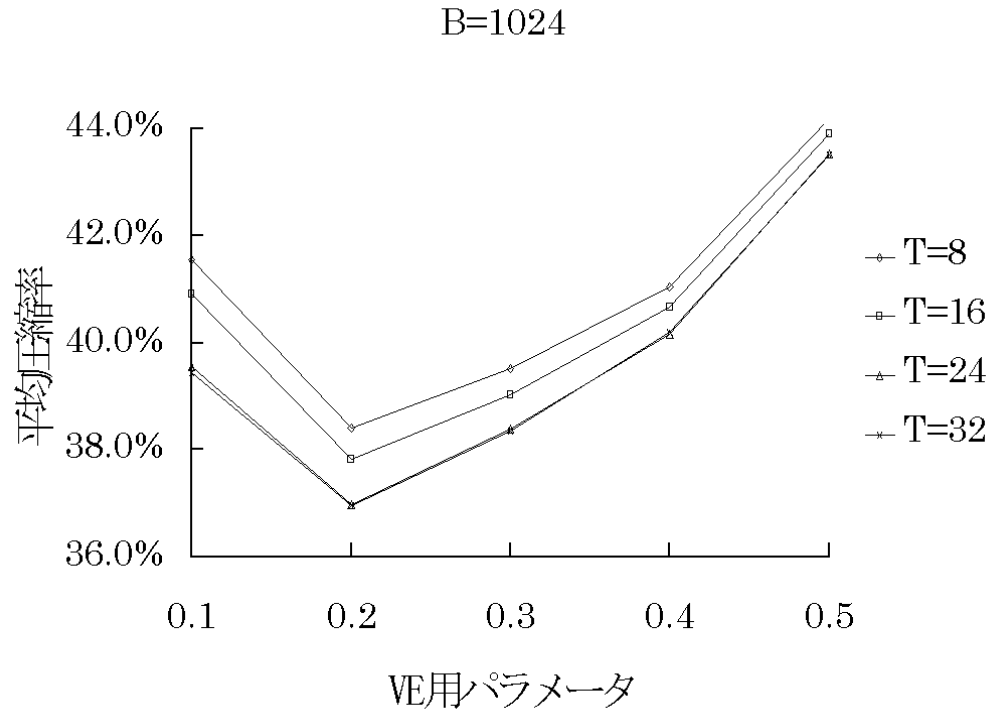


図 3-12 T、B 一定の場合の P による平均圧縮率の変化

このグラフを見てみると、どちらの場合でも P=0.2 の時に平均圧縮率が最も小さくなっている。ゆえにどのような B を設定しても P の極小値は 0.2 であるということが言える。また、T=32 での平均圧縮率は T=24 の時とほとんど変わっ

ていない(もしくは劣っている)。T を大きくしても符号化時間がいたずらに増加するだけなので、T は大きくても 24 が限界であるだということも言える。

次に、B を求めるために T、P を固定した時の平均圧縮率の変化をグラフにしてみることにする。P を 0.2 に固定し、B を 256、512、768、1024、1280 を変化させたときの平均圧縮率の変化をグラフにしてみると、図 3-13 のようになる。

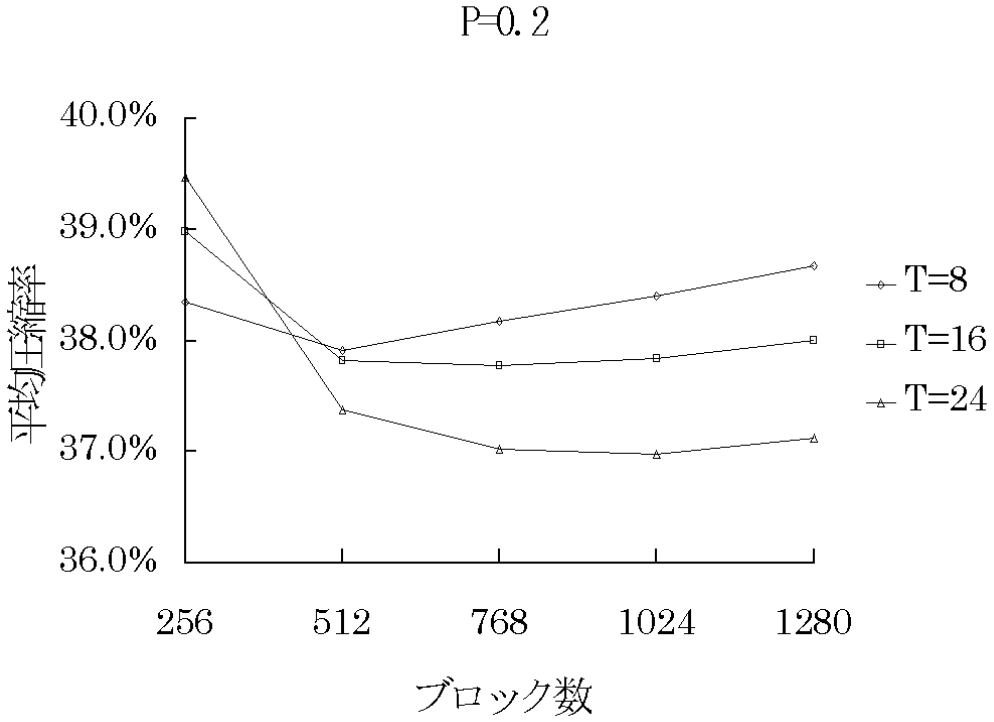


図 3-13 T、P 一定の場合の B による平均圧縮率の変化

このグラフによると、T=8 の時は B=512、T=16 の時は B=768、T=24 の時は B=1024 の時に平均圧縮率が最小になっている。ゆえに、最適変数のセットは(T,B,P)=(8,512,0.2)、(16,768,0.2)、(24,1024,0.2)の三つであることが分かる。ではこれからはこの三つのセットを用いた時の圧縮率がどのくらいの性能を満たすのかを検証することにする。

3.4.3. 圧縮率から見た実験結果

まず、どれくらいの圧縮率を達成することができたかを見てみる。比較対象としては SHORTEN(normal mode)[9]と、LTAC(block size=1024)[10]を用いた。これより以降は(T,B,P)=(8,512,0.2)は TYPE 1、(T,B,P)=(16,768,0.2)は TYPE 2、(T,B,P)=(24,1024,0.2)は TYPE 3 と表記することにする。サンプルファイルの圧縮サイズ、それに圧縮率を比較してみると、表 3-1、表 3-2 のようになる。

表 3-1 ファイルサイズの比較表(Byte 単位)

	Original	TYPE 1	TYPE 2	TYPE 3	SHORTEN	LTAC
CHIMES	15932	5398	5125	4982	6757	5139
CHORD	24994	7428	7624	7771	9049	7884
DING	11586	3256	3179	3115	4781	3181
TADA	27516	14252	14308	13721	14969	13762
Total	80028	30334	30236	29589	35556	29966

表 3-2 圧縮率の比較表(%単位)

	TYPE 1	TYPE 2	TYPE 3	SHORTEN	LTAC
CHIMES	33.88	32.17	31.27	42.41	32.26
CHORD	29.72	30.50	31.09	36.20	31.54
DING	28.10	27.44	26.89	41.27	27.46
TADA	51.80	52.00	49.87	54.40	50.01
Average	37.90	37.78	36.97	44.43	37.44

3.4.4. 符号化時間から見た実験結果

次に、符号化に要した時間がどれくらいかを調べる。今回適用した符号化方式は復号化にも同じだけ時間を要するものである。そのためにリアルタイム符号化(送信と受信とのタイムラグが生じないこと)として理想的な時間としては符号化時間がサンプルファイルの演奏時間より少ないことが条件である。実験としては符号化プログラムを実行し、得られた時間値の値を用いた。それを表にまとめたものが表 3-3 である。

表 3-3 符号化時間の比較表(sec 単位)

File name(sample)	TYPE 1	TYPE 2	TYPE 3
CHIMES(15876)	0.60	1.15	1.65
CHORD(24938)	0.77	1.53	2.36
DING(11530)	0.39	0.93	1.32
TADA(27459)	0.93	1.71	2.69
Total(79803)	2.69	5.32	8.02

全体の演奏時間は 79803／22050=3.62(sec)であるので、TYPE 1 の時のみリアルタイム符号化が可能である。しかし、これでは実用化には不充分なので、更に符号化時間を短くするために改良を行うことにする。

3.4.5. 符号化時間改善のための実験

符号化時間は T のみに依存する。それはなぜかと言うと、予測係数の更新に時間を要するからである。これまでの実験では予測係数の更新は入力信号が一つ入力されるたびに行っていた(3.2.参照)。しかし、その更新する間隔をある程度広げて、つまり、数サンプル入力されるたびに予測係数を更新するようにすれ

ば全体の符号化に要する時間を少なくすることができる。しかし、その分だけ最適な係数を得られない場合が多くなるので、圧縮率が劣ることになる。

ここでは圧縮率をなるべく劣化させずに符号化時間を改善させるためには予測係数更新の間隔をどのくらいまで広げることができるのかを検証する。TYPE1,2,3 において更新の間隔を 1(今までの実験)～20 まで変化させた場合の符号化時間の合計を示したグラフが図 3-14 である。

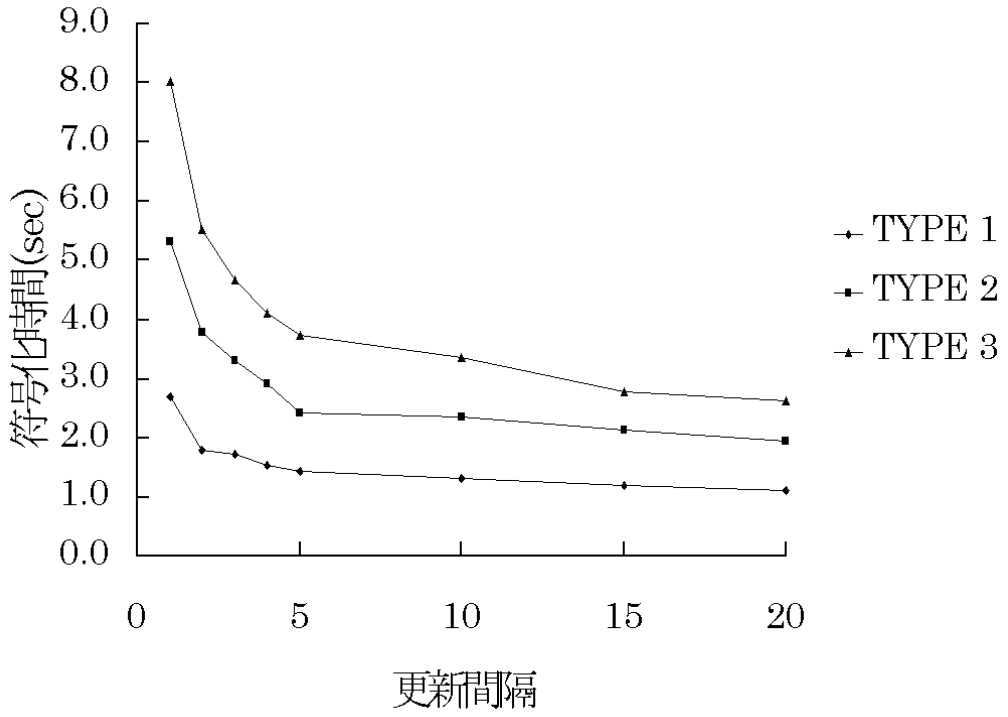


図 3-14 更新間隔による符号化時間の変化

予測通り全体的に更新間隔が大きくなるほど、符号化時間は減少している。同時に更新間隔を変化させた場合の 1 を基準とした場合の平均圧縮率誤差をグラフにしたものが図 3-15 である。

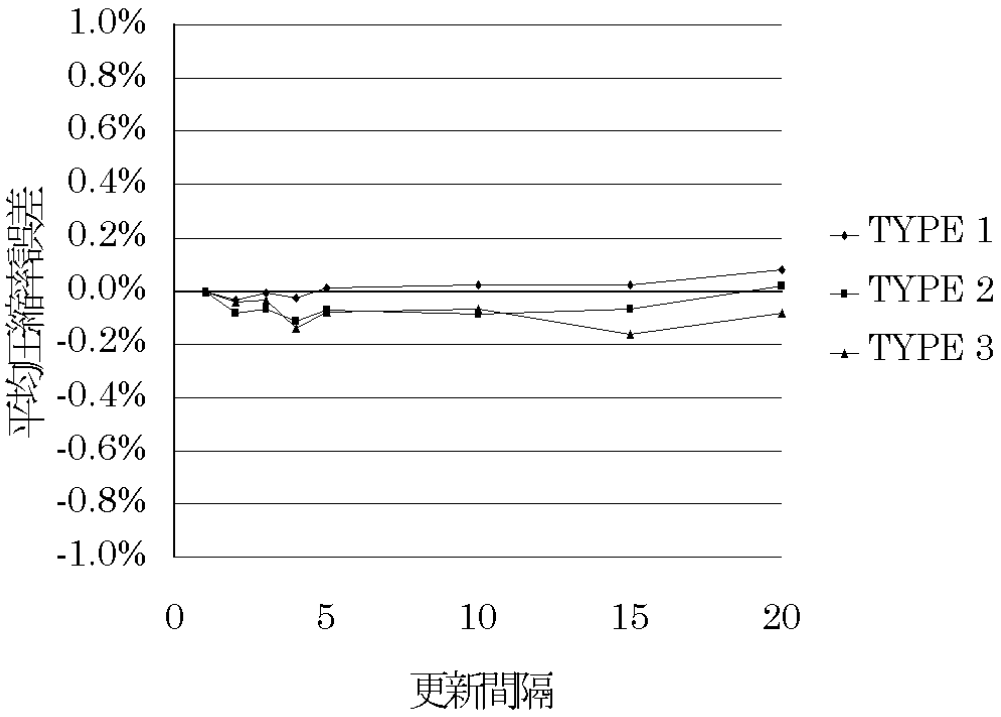


図 3-15 更新間隔による平均圧縮率誤差の変化

多少の平均圧縮率の変化はあるが、それぞれにおいての変化率は 0.2%以内であるので、ほとんど変化しないとみなしても良いだろう。更新間隔が 20 の時の圧縮率と符号化時間の表を示すと、表 3-4、表 3-5 になる。

表 3-4 符号化時間改善後の圧縮率の比較表(%単位)

File name	TYPE 1	TYPE 2	TYPE 3
CHIMES	34.17	32.41	31.42
CHORD	29.74	30.63	30.89
DING	28.08	27.18	26.79
TADA	51.84	51.91	49.76
Average	37.98	37.80	36.89

表 3-5 符号化時間改善後の符号化時間の比較表(sec 単位)

File name	TYPE 1	TYPE 2	TYPE 3
CHIMES	0.22	0.33	0.49
CHORD	0.38	0.54	0.83
DING	0.22	0.32	0.39
TADA	0.44	0.77	0.93
Total	1.26	1.96	2.64

3.4.6. 復号化実験

これまでの実験で符号化の実験は終了した。今度は圧縮したファイルを元に戻すための復号化の実験について説明することにする。

復号化も符号化の時と同じ条件で実験を行った。パラメータも符号化の時と同じ値を用いた。プログラムソースは別に付属しておく。ソースをコンパイルした後、同じように MS-DOS プロンプト上で、

＞decode (圧縮ファイル名) (復号後のファイル名)

とプログラムを実行すると圧縮ファイルの復号化が行われる。このようにして復号化の実験を行ったところ、すべてのサンプルファイルの場合において復号後のファイルが圧縮前のサンプルファイルと全く同じファイルになることが確認できた。

3.5. 考察

3.5.1. 圧縮率について

まず表 3-4 の平均圧縮率について見てみると、TYPE1,2,3 いずれの場合も SHORTEN よりも平均圧縮率は良くなっている。特に TYPE 3 は LTAC よりも平均圧縮率が良くなっている。つまり、今回提案したロスレス符号化方式は従来のものと同等、又はそれ以上の圧縮性能を達成することができるということが言える。

次に、表 3-4 での個々のサンプルファイルの圧縮率について見てみると、CHIMES と DING では 1<2<3 の順に圧縮率が良くなっている。これはタップ数が増加することに起因すると考えられる。しかし、一方では CHORD の場合は逆に悪くなっている。これは CHORD においての極小点は TYPE 1 にあるためと考えることができる。また、TADA においては 1>2<3 となっている。これは TADA においては極大点が存在するためと考えられる。しかし、平均圧縮率は 1<2<3 となっているので、たいていのファイルは 1<2<3 の順に圧縮率が良くなっていくと考えることができる。

3.5.2. 符号化時間について

TYPE 2、TYPE 3 の符号化時間は TYPE 1 に比べて約 2 倍及び約 3 倍になっている。これはタップ数が 2 倍もしくは 3 倍になるため、そのために予測係数を更新するためのループの回数が増えたためと考えられる。

また、リアルタイム符号化における実用性について考えてみると、表 3-3 の時

には TYPE 1 で可能なだけであったが、表 3-5 の時にはすべての TYPE でリアルタイム符号化が可能となることがわかる。つまり、ある程度実用性を考慮するならば、圧縮率に影響を与えない範囲で更新間隔を広げてやらなければいけないということが分かる。

3.6. まとめ及び今後の展望

本研究では後方適応予測に基づく音声予測ロスレス符号化の実験を行った。その結果、従来の音声ロスレス符号化ツールと同等の圧縮性能を達成することができた。また、復号化もできることが分かった。今後の展望としては以下のことが挙げられる。

①他の音声フォーマットでの研究
今回はサンプリング周波数 22.05kHz、量子化ビット 8bit、mono のフォーマットしか時間の都合上できなかった。音声フォーマットにはそれ以外にサンプリング周波数 44.1kHz、量子化ビット 16bit、stereo(2 チャンネル)といったモードがあるので、それらのフォーマットにおいてはどのようなパラメータの値を用いれば良いのかを探る必要がある。

②実用的な符号化ツールの作成
以上の事を成し遂げたうえで、実用的な符号化及び復号化ツールを作成する必要がある。

謝辞

この論文を執筆するにあたって、数々のご指導を頂きました森本宏教授にまず感謝致します。また、数々のご協力を頂きました森本研究室の学生の皆さまにも感謝致します。最後に、研究生活を陰で支えてくださったすべての人たちに感謝のお言葉を述べさせて頂きます。

参考文献

[1] http://www.real.com/

[2] K.R.Rao,J.J.Hwang 著,安田 浩,藤原 洋監訳,”デジタル放送・インターネットのための情報圧縮技術”,共立出版,1998

[3] http://sound.splab.ecl.ntt.co.jp/twinvq/

[4] 伊東 晋,”画像情報処理の基礎-信号・情報理論と画像符号化-”,東京理科大学出版会,1986

[5] 塩野 充,"わかりやすいディジタル情報理論",オーム社,1998

[6] 浜田晴夫,"アダプティブフィルタの基礎(その 1)",**45**,624-630,日本音響学会誌,1989

[7] 守谷健弘,"音声符号化",社団法人電子情報通信学会,1998

[8] T.Robinson,"*SHORTEN:Simple lossless and near-lossless waveform compression*", http://svr-www.eng.cam.ac.uk/~ajr/GroupPubs/Robinson94-tr156/,1994

[9] http://www.softsound.com/Shorten.html

[10] T.Liebchen,M.Purat and P.Noll,"*Improved Lossless Transform Coding of Audio Signals*",http://www-ft.ee.tu-berlin.de/~liebchen/ltac.html,1999