音楽データへの電子透かしの試み

— 離散フーリエ変換の応用 —

An Experiment of Digital Watermark for Music Data

— An Application of Digital Fourier Transform —

ネットワーク情報学部　染谷知臣　斉藤昌紀　増田圭悟　佐藤 創

School of Network and Information　Tomo-omi Someya, Masaki Saito, Keigo Masuda, Hajime Sato

Keywords : wave data, digital watermark, digital Fourier transform, synchronization, error correction

まえがき

街には音楽があふれている。街頭や喫茶店，レコード店等で流れている音楽を「あ，この曲いいな」と思っても,「あれ？ これなんて曲だったかな…」と思ったことはないだろうか。しかし，現状ではBGMとして流れているだけでは音楽のタイトルを知ることはできない。

2005年度の本学部授業科目「プロジェクト1」におけるテーマ「新しい音楽検索手法」（小林 隆助教授担当）では，街や喫茶店で流れている曲を耳にした人が携帯電話のマイクを利用した問い合わせると，その曲のタイトルやアーティストなどのデータを即時提供するサービスを行うビジネスモデルが企画された。本稿は，このプロジェクトの一環として「電子透かしの実現」に取り組んだグループの活動報告である.

現代では音楽は通常，デジタル方式でCDやMDなどの媒体に記録されている。ほんの20年ほど前までは，レコード盤や磁気テープなどアナログ方式の記録しか考えられなかった。レコードは盤面に刻まれた溝を針がなぞり，溝の凹凸がもたらす微妙な振動を電気的に増幅して音楽演奏を再現するものであった。

記録方式のデジタル化によって音楽をコンピュータで処理できるようになり，音楽情報を付加することも容易になった。著作権情報などを気づかれないように付加する，いわゆる「電子透かし」も考案された。電子透かしは記録媒体上での検出を目的としたものが多いが，我々が製作を試みた電子透かしは音楽として一度アナログ化された後でも透かしを検出できる特徴をもっている。上記のプロジェクトを実現するに当たり，重要な役割を果たすのが電子透かし技術である。

フーリエ変換は電子透かし技術を支える数理的な方法である。この技術の開発には数理的原理に関する知識よりも，アルゴリズム的な理解とプログラミング能力，実験結果から現象を予測する論理的思考力が求められる。その実践を通じてフーリエ変換についての理解が深まる。大切なことは自分たちで考えることである。（染谷・佐藤）

1　システムの概要

このプロジェクトにおいて開発される音楽検索手法を3EMSと呼ぶ。これは3E Music Searchの略で，3Eとは

・Everytime（いつでも）
　… 検索した情報はいつでも試聴，ダウンロード可能
・Everywhere（どこでも）
　… 気になった曲情報を，その場で検索
・Easy（簡単に）
　… 携帯電話をかざすだけで，すぐに回答

を表す。3EMSは音楽へのＩＤ情報の埋め込み・抽出技術を開発し，その音楽を配信する画期的なサービスを目指す。3EMSにおいて提唱する電子透かし方式に要求された仕様は,「スピーカーから流れる透かしの入った音声から曲を特定するIDを検出すること」である。

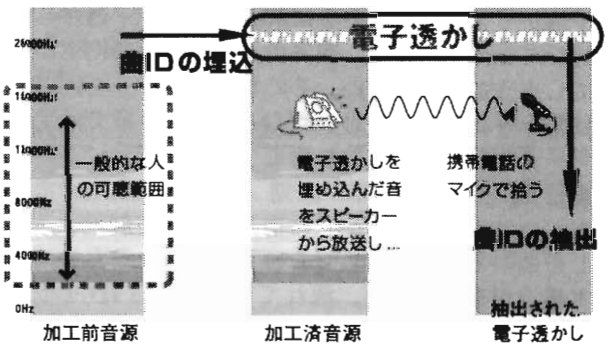


図 1.1　電子透かし概念図

一般に電子透かしとは,「画像や動画，音声などのマルティメディアデータに，画質や音質にはほとんど影響を与えずに特定の情報を埋め込む技術，およびその特定の情報」のことである。多くの方法が世界中で発明・発表されている。著作権情報の埋め込みに利用され，改ざんや不正利用等からの保護のために暗号化されて埋め込まれるなどセキュリティを重視したものが多いが，今回開

発した電子透かしは単に曲を特定するID（35ビットのコード）を検出することだけが目的であるため，そのようなセキュリティの必要はない。

この電子透かしは人の耳では聞こえない高周波領域にID情報を付加するものである。透かしの検出には一般的なPCマイクを使い，いったんwaveファイルに録音し，検出プログラムにかけてIDの検出を行う。加工対象とする音声ファイルの形式には，wave形式，PCM，48 kHzが想定されている。これはこの透かしの挿入方法でより多くの情報を伝達するためである。録音側のファイルも同じスペックの録音形式を要求する。

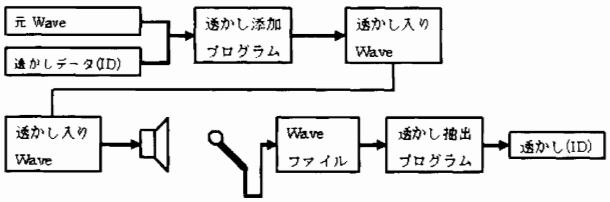


図 1.2　システム全体図

本システムの開発上で必要となった基礎知識と技術について以下に解説する。

まず，2章では我々の電子透かしを実現する上で必要なファイル構造について述べる。加工するデータの構造が分からなければ電子透かしも埋め込むことはできない。電子透かしは高周波帯域を加工して埋め込むため，高周波成分を音楽データから取り出す必要がある。音楽データの周波数成分を分析するには「フーリエ変換」がよく使われる。さらに周波数成分からもとの音楽データに戻すには「フーリエ逆変換」が必要になる。ここが電子透かしの技術の中でも一番重要な部分である。3章ではこのフーリエ変換とその高速化版であるFFTついて，電子透かしを入れ方を中心に解説する。電子透かしを埋め込むことはできたが，空気中を伝わる間に内容が変化してしまうかもしれない。そのためにエラーを回避できる仕組みを組み込まなければならない。4章では受信したデータからエラーを取り除く「誤り訂正」について解説を行う。そして，5章「同期　データを取出すための工夫」では主に電子透かしの構造について解説する。透かしを曲のどの位置からでも確実にとらえるためにはデータ構造が重要である。流れていく曲からIDを取り出す工夫を紹介する。これらの技術は我々の電子透かしシステムには最低限必要な技術である。最後に，発展・反省・課題では　時間切れにより開発を中止した方式についても解説する。

各技術がどのような役割を果たしているのかを頭に置きながら読んでもらいたい。（染谷）

2　waveファイル構造

実際に電子透かしを入れるにはどのようにしたらよいのか。圧縮など加工が施されていないwave形式が手始めによいと考えた。透かしを入れるにはまず音源のファイルから音楽データを抜き出さなければならない。そのためには音源に使われている「waveファイルの構造」が分からなければ始まらない。

waveファイルは次の表に示すように先頭からフォーマットされている。その主要部分を説明する。

各フィールドの内容	サイズ
RIFFヘッダ	4バイト
ファイルサイズ	4バイト
waveヘッダ	4バイト
fmtチャンク	4バイト
fmtチャンクのバイト数	4バイト
PCMの種類	2バイト
チャンネル数	2バイト
サンプリングレート	4バイト
データ速度	4バイト
ブロックサイズ	2バイト
サンプルあたりのビット数	2バイト
拡張部分のサイズ	2バイト
拡張部分のデータ	2^{16} バイト未満
dataチャンク	4バイト
データ部分のバイト数	4バイト
データ部分	2^{32} バイト未満

1) RIFFヘッダ

waveファイルはRIFF（Resourse Interchange File Format）という形式のファイルの一種で，「データの説明」と「データ」が順に記録されている。

2) その他のヘッダとチャンク

waveとfmtとdataのヘッダ・チャンクの部分はファイルの形式を表すので必須の項目となる。このデータは正確に記録されていないとファイルの認識が出来ないので，ファイルの読み込み・再生などが出来ない。ファイルサイズについても正確に記録されている必要がある。この部分でデータのサイズが正確に記録されていない場合，ファイルが再生できなかったり，途中で再生不可のエラーが表示され，ファイルが中断されることがある。

3) PCM（Pulse Code Modulation）

音声をデジタルデータに変換する方式の一つで，音の波形を一定の時間間隔で数値化したものが記録される。数値は表現ビット数に応じて量子化される。記録されたデータ量は，(1秒間の数値化回数)×(データの表現ビット

数) × (記録時間) である。一般的に音楽 CD は PCM 方式を利用している。

4) PCM の種類に関する注意

基本的な PCM の種類はリニア PCM で，バイナリ表示でファイルを見ると 01 00 と 16 進表示される。この場合は拡張部分がない。通常音楽 CD に使われている PCM は非圧縮の形式であるが，他の PCM の中にはデータを圧縮してファイルのサイズを小さくしているものもある。この圧縮されているファイルを元の状態に戻すための拡張情報を格納しているのが拡張部分となる。PCM の種類によっては拡張部分を持つものもあるのでフォーマットを操作する場合には注意が必要である。

5) サンプリングレート

1 秒間にどれだけの回数データをサンプリングするかを表す。1 秒間にサンプリングするデータ数が多いほど音質が良くなる。例えばサンプリングレートが 44.1 kHz なら 1 秒間に 44100 回データをサンプリングする。48kHz なら 1 秒間に 48000 のデータとなる。この 2 つのレートの差は 1 秒間に 3900 回となり，その分 48kHz のレートの方が実際の音により近い表現ができるということになる。

6) データ速度

音声ファイルを 1 秒間再生するために必要なバイト数を表す。したがってデータ速度は，サンプリングレート×チャンネル数× 1 サンプルあたりの表現バイト数となる。

7) データ部分

データ部分における 1 サンプルあたりの表現ビット数には 8 ビットと 16 ビットの 2 通りがある。8 ビット表記の場合は 0 から 255 までの正の整数値で表され，16 ビット表記の場合は 1 ビットを＋/－の符号として使用するために，−32768 から +32767 までの整数型の値で表される。また無音地帯の値は 8 ビット表記では 127，16 ビット表記では 0 になっている。(斉藤)

3 フーリエ変換と FFT

このシステム開発における中心技術はフーリエ変換であり，そのプログラムが重要な役割を果たす ([1] 参照)。自作のプログラムのほか，公開されていた FFT のプログラムを採用した。

3.1 フーリエ変換

まず，フーリエ変換が何をするかを一言で表現するなら『音の強弱を表す数値の列で構成されたファイルを一定間隔で区切り，各々を周波数成分の強弱を表す数値の列のファイルに変換する』と言える。自作のフーリエ変換プログラムを用いた一例をグラフで示すと図 3.1 のようになる。上のグラフは縦軸が音の強さ，横軸が時間である。下のグラフの縦軸は各周波数成分の強さであるが横軸は周波数であり，上のグラフの波形にどの周波数がどの程度含まれているのかを知ることができる。

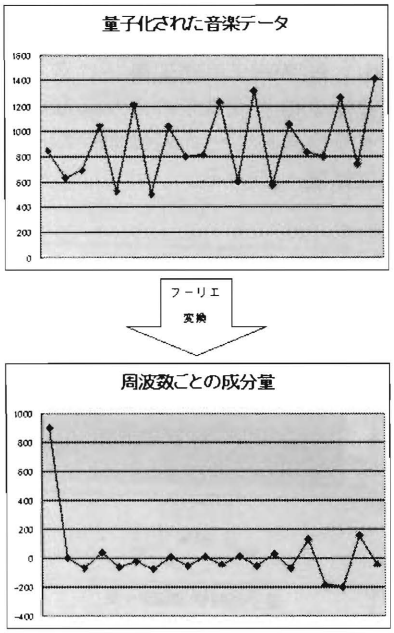


図 3.1　フーリエ変換概念図

図 3.1 に対応する数値データを図 3.2 に示す。左側が変換前の波形を表す数値ファイル（整数列），右側が変換後の各周波数の強弱をあらわす数値ファイル（実数列）である。どちらも数値の列であるが，明らかに数値が異なっていることからも変換が行われたことがわかる。

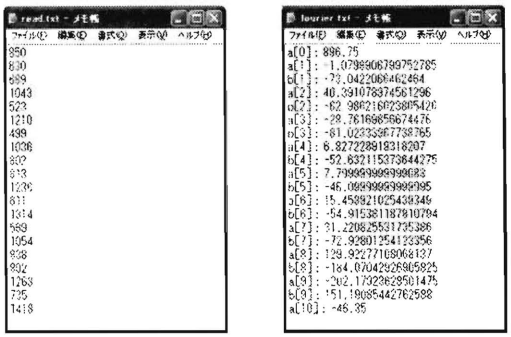


図 3.2　フーリエ変換実行結果

我々の作成した電子透かし技術は，フーリエ変換後の値が数値であることを利用し，この段階で電子透かしを付加している。次に説明するフーリエ逆変換も我々の作成した電子透かしに欠かせない技術である。

3.2 フーリエ逆変換

フーリエ変換と同様にフーリエ逆変換を一言で表現するなら，まさにフーリエ変換の逆で，『一定間隔で区切った各周波数の強弱をあらわす数値ファイルから波形を表す数値ファイルを生成する』となり，フーリエ変換と対をなす。自作したフーリエ逆変換プログラムを用いて上記の例を逆変換すると図 3.3 のようになる。周波数成分の量を表すグラフからもとの時間軸に沿って波形を表す最初のグラフに戻っている。

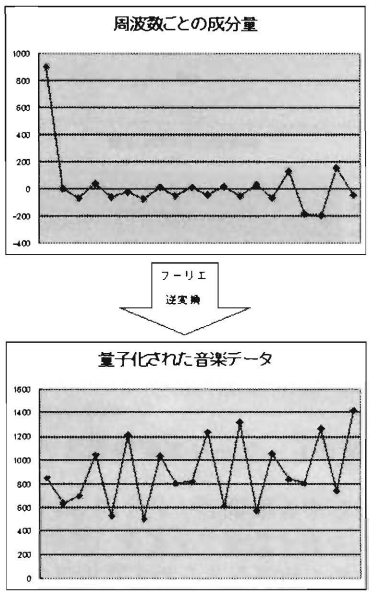


図 3.3 フーリエ逆変換概念図

図 3.3 に対応する数値データを図 3.4 に示す。左側が逆変換前（変換後）の各周波数の強弱を表す数値ファイル，右側が逆変換後（変換前）の数数値ファイルである。

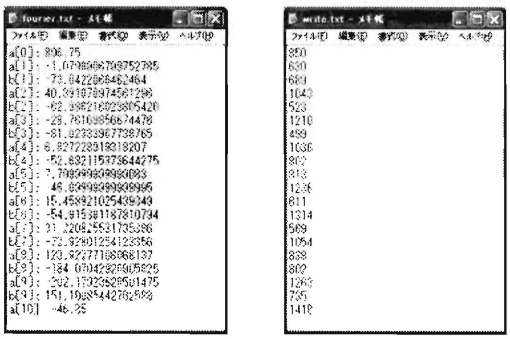


図 3.4 フーリエ逆変換実行結果

フーリエ逆変換の最大のポイントは，実行結果をくらべてみるとわかるように，フーリエ変換後のファイルが完全にフーリエ変換前のファイルに戻ることである。この性質を利用することで，電子透かしを付加したファイルを生成することが可能になる。

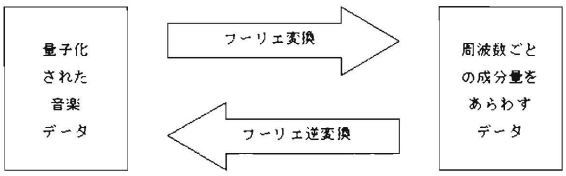


図 3.5 フーリエ変換とフーリエ逆変換の関係

3.3 電子透かしの付加

前項で我々の電子透かしは，フーリエ変換とフーリエ逆変換を用いていると述べた。具体的に言えば，wave ファイルのデータ部分をフーリエ変換し，そこで得られた各周波数の強弱を表す数値ファイルの一部を加工して情報を埋め込み，そのファイルをフーリエ逆変換して透かしの埋め込まれた wave ファイルを生成するのである。

それを詳しく解説するために，まずファイルの内容について説明する。フーリエ変換の対象となる wave ファイルのデータ部分は数値の列で構成されていて，電子透かしを付加した wave ファイルのデータ部分も同様に数値の列で構成される。フーリエ変換プログラムは wave ファイルを読み，そのデータ部分を配列（これを配列 1 とする）に格納し，配列 1 のフーリエ変換を行い，その出力結果を新たな配列（これを配列 2 とする）に代入し，配列 2 の値をファイルに書き込んでいる。

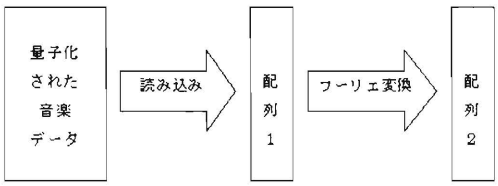


図 3.6 フーリエ変換プロセス

次に，フーリエ変換は複素数を使った計算がもととなっているため，入力が実数の数列であるのに，フーリエ変換を行うと実数値と虚数値とが出力される。したがって，配列 2 は実数部の配列 a と虚数部の配列 b が合わさった形になっている。自作したフーリエ変換プログラムでは配列 2 にあたる配列は次のようになっている。

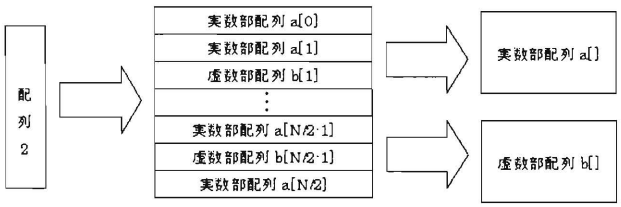


図 3.7 フーリエ変換内配列 2 の構造

図 3.7 には変数 N が使われているが，この N は wave ファイルのデータ部分から読み込むサンプルデータの 1 セット分の個数を意味する。我々の作成した電子透かしでは，全サンプルデータを $N = 2048$ 個ずつに区切り，2048 個のサンプルデータを 1 セットとしてフーリエ変換を行う。その結果として 1 セットごとに 2048 個の値が得られる。N が偶数であるから配列 2 の内容は図のようになる。すなわち，$a[0]$ は波形の直流成分の大きさで，$a[\frac{N}{2}]$ は最も高い周波数成分の大きさであり，$b[0]$ と $b[\frac{N}{2}]$ は常に 0 である。1 セット分のサンプルデータの数列を $x[0], x[1], \cdots, x[N-1]$ とすれば，

$$x[m] = a[0] + \sum_{k=1}^{\frac{N}{2}-1} \left(a[k] \cos \frac{2\pi mk}{N} + b[k] \sin \frac{2\pi mk}{N} \right) + a[\tfrac{N}{2}] \cos \pi m \qquad (m = 0, 1, \cdots, N-1)$$

の関係が成り立つ。なお，N が奇数の場合は配列 b の最後の要素 $b[\frac{N-1}{2}]$ にも計算値が代入される形になり，$b[0] = 0$ のままとする。複素数で考えるフーリエ変換では k 番目と $N-k$ 番目の値が互いに共役となる。この性質を理解することが電子透かしを付加する上で重要である。

3.4 透かしを入れる位置

フーリエ変換におけるファイルの内容が説明されたところで再び電子透かしの付加について解説する。

実際に電子透かしを入れるのに当たってどの位置に透かしのデータを入れるのが適当であるのかを決めなければならない。一般的に人の可聴域の上限は 20kHz 付近と言われている。そこで電子透かしを 20kHz 以上の音域に入れることとした。しかし，いくらでも高い位置に入れればよいというものでもない。一般的にマイクの周波数特性は人間の可聴域に合わせて作られており，20kHz を境に急激に悪くなっていくものが多い。マイクの周波数特性によってすかしのデータが読み取れなくなることを防ぐため，双方のぎりぎりのラインを狙い，20kHz を目安に電子透かしの埋め込みを行った。以降この電子透かしを「データ用透かし」と呼ぶ。

我々の電子透かしは，フーリエ変換後の配列 2 の一部を加工して情報を埋め込む。具体的には，透かしを付加する目標の周波数を設定し，次の計算式によって対応する配列要素の位置を計算し，そこに特定の値を代入する。こうして加工した配列 2 をフーリエ逆変換することで電子透かしが付加される。まず，記号を次のように設定する。

W：目標の周波数（20 kHz や 21 kHz）
S：サンプリングレート（44.1 kHz や 48 kHz）
N：サンプルデータ数（2048）
J：実数部配列 a の要素番号（0 から始まる整数）
K：配列 2 の要素番号（1 から始まる整数）

◎ 電子透かしを入れる配列 a の中の位置を求める計算式

$$J = W \div (S \div N) \tag{1}$$

我々の電子透かしは，サンプリングレート $S = 48$ kHz，1 セットのサンプルデータ数 $N = 2048$ で行っているので，$W =$ 約 20 kHz の周波数に電子透かしを付加したい場合は以下のようになる。

$$J \fallingdotseq 20 \div (48 \div 2048) = 853.333\cdots \quad \Rightarrow \quad J = 853$$

この計算より，実数部配列の 853 番目 $a[853]$ に透かし情報を入れればよいことがわかる（透かし情報の正確な周波数は $W = J \times S/N \fallingdotseq 19.992$ kHz となる）。

実際には虚数部も含む配列 2 における先頭からの位置 K を求める必要がある。実数部配列の最初 $a[0]$ と虚数部配列の最初 $b[0]$ が 1 番目と 2 番目に入り，その後も実数部と虚数部が交互に入っているので，$a[J]$ の前には J 個の a の要素と同数の b の要素がある。したがって，

$$K = 2J + 1$$

となる。我々の電子透かしで $J = 853$ の場合には，

$$K = 2J + 1 = 2 \times 853 + 1 = 1707$$

となり，配列 2 において 1707 番目に透かし情報を入れればよいことがわかる。

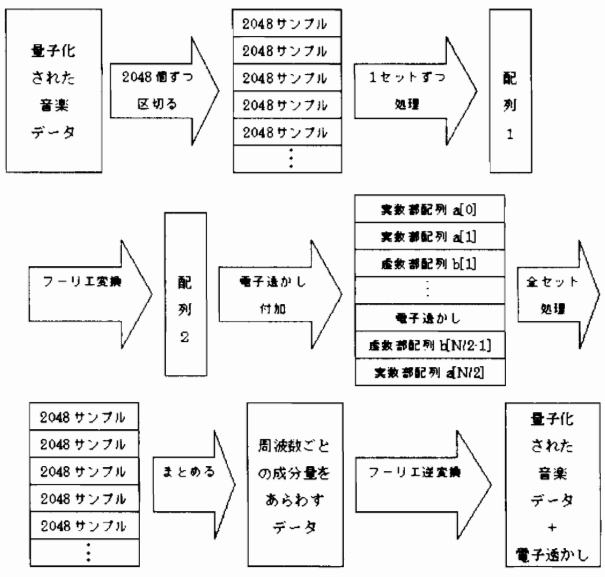


図 3.8 電子透かし付加の仕組み

先に説明したように我々の電子透かしでは，2048 個のサンプルデータを 1 セットとし，この 1 セットに対して 1 ビットの付加情報を考える。具体的には，1 セット分のサンプルデータをフーリエ変換して得られる配列 2 の $K = 1707$ 番目の値を強制的に定数 $C(> 0)$ または 0 に置き換える。このように加工された配列 2 のデータがフー

リエ逆変換されたとき，波形として周波数 W のコサイン波を含む，含まない，の違いで1ビットが表現される。

付加したいIDなどの情報は文字コードなどの2進数列に置き換え，1，0に対応する1セット分にはそれぞれ電子透かしを付加する，付加しないという処理を行う。これらを数セット集めると意図する付加情報が表現されているという仕組みである。夏休みの段階では，フーリエ変換した後の配列2の K 番目に付加したい情報である曲IDの数を直接代入していたが，電子透かしを抽出する際にはその情報のほとんどが失われ，実際に使えるものではなかった。そこで，現在使っている2048サンプルを1セットとして処理する方法が考え出された。この方法では，電子透かしを抽出したときに付加したものと完全に同じである必要はなく，あるの許容範囲（閾値の上下）で一致していればよいので，情報が失われにくいという利点がある。

3.5 FFT（高速フーリエ変換）

1ビットの付加情報を抽出するのに1セット2048サンプル（$S = 48$ kHz のときサンプリング時間は $N/S =$ 約0.047秒）を処理していく方法は，時間がかかるという欠点があった。数秒で曲IDを認識するためにはプログラムの高速化を行う必要があり，そこで新しく取り入れた技術がFFT（高速フーリエ変換）である。FFTといってもフーリエ変換にほかならず，プログラムの機能は今まで説明したことと同じである。違いは，アルゴリズムの改良により処理スピードが格段に上がっている点と変換するときの配列の使い方にある。FFTで用いる配列を図にすると図3.10のようになる。

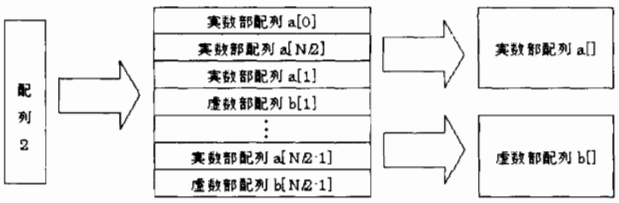


図 3.9 FFT プログラムの配列2の構造

前項の図とくらべてわかるように，配列の使い方が変わっている。これは，同じ計算をできるだけ反復しないように計算アルゴリズムを改良して計算量を減らし，処理を高速化しているためである。この他にも高速化のテクニックがいろいろ使われているが，電子透かしを付加する上でとくに注意する点はない。

なお，FFTのプログラムはインターネットのサイト(`http://www.kurims.kyoto-u.ac.jp/~ooura/index-j.html`)に公開されている大浦拓哉教授（京都大）によるものを使用した。（増田）

4 誤り訂正

電子透かしを解析することによりビット列を得るが，そのビット列が正しいビット列とは限らない。ビット列が間違っていた場合，違う曲IDを取得してしまう危険がある。これを避けるためには，ビット列が間違っていることを検出し，ビット列をもう一度取得する必要がある。しかし，ユーザビリティの面を考えると，たった1ビットでも誤りが存在しただけで正しいビット列が得られるまで録音を繰り返し行うのは実用上好ましいとは言えない。理想的な方法は録音を繰り返すことなく，誤りを検出し，誤りを自己訂正する方法である。この問題を解決する情報の表現法が誤り訂正符号である。誤り訂正符号には様々なものが考案されている。

4.1 通信路のモデル

はじめに，通信中にどのような現象が生ずるかをモデル化した図を示す。このモデルはShannonの通信系のモデルとして知られている。

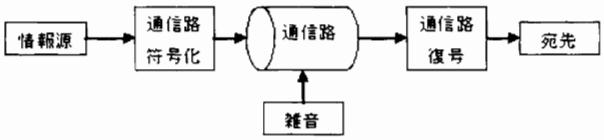


図 4.1 Shannon の通信系モデル

このモデルを使って情報が送・受信されるときの状況を説明する。まず送信元の情報がある。情報を送信するときは，通信に適した形にデータを変換する。この変換を広義での符号化という。通信を行う通路の概念をこのモデルでは通信路という。通信路には外的な要因で様々な雑音が混入する。雑音は通信中の情報を狂わせることがある。相手先に届いた情報は符号化を解き，送信元の情報と同じ形に戻す。この変換を広義での復号という。この復号の中で一般に雑音の混入によって失われた情報の回復を図る。こうして宛先に情報は届けられる。

通信中に情報が失われないために，送信時に予め組織的な冗長性を付加し，雑音による情報の変形を回復することが可能な構造をもつ誤り訂正符号が考案された。

4.2 ハミング符号

図4.2に示された通信路は入出力ともに0，1の2値である。通信中に $0 \to 1$，$1 \to 0$ のエラーがどちらも同程度発生する場合に，2進対称通信路と呼ばれる。ハミング符号はこのような通信路を想定して考案された，古典的な誤り訂正符号である。仕組みは簡単であるが，現代の高度な誤り訂正符号の基礎となった基本的な符号方式である。

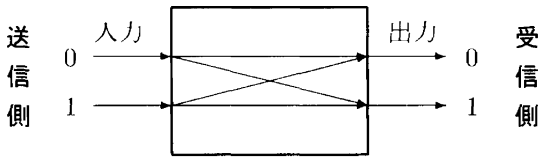


図 4.2　2 進対称通信路

ハミング符号は本来の情報ビットに検査ビットと呼ばれる冗長なビットを数ビット加えることにより,「情報ビット列＋検査ビット列」の中で「1 ビット誤り訂正」または「1 ビット誤り訂正と 2 ビット誤り検出」が可能となるように設計されたコード体系である。

誤り訂正能力はそれほど高くないため，コンピュータ内部の通信など信頼性の高い通信路に用いるのが一般的であるが，我々は開発期間が短かったため，雑音の多い通信路でありながら符号の仕組みが単純である理由で採用した。ここでは 4 ビットの情報に検査ビットを加える符号化とその復号の方法について解説する。実際のプログラムでも 4 ビットの情報に対してこの符号化を利用している。

4.3　ハミング符号による符号化

4 ビットの情報を $a_1\,a_2\,a_3\,a_4$ で表す（a_k は 0 または 1）。これに 3 個の検査ビット p_1, p_2, p_3 を付け加える変換

$$a_1\,a_2\,a_3 a_4 \;\longmapsto\; a_1\,a_2\,a_3\,a_4\,p_1\,p_2\,p_3$$

を符号化という。p_1, p_2, p_3 は次の生成式で求める。

$$\begin{cases} p_1 = a_1 + a_2 + a_3 \\ p_2 = a_1 + a_2 \qquad + a_4 \\ p_3 = a_1 \qquad + a_3 + a_4 \end{cases}$$

この演算 + は排他的論理和（$1+1=0$）である。どの桁 a_k も 2 回以上式に現われている。この生成式により 0000〜1111 の 16 種類の情報を $a_1\,a_2\,a_3\,a_4\,p_1\,p_2\,p_3$ で表すと以下のようになる。これらの 16 個のビット列の集まりを (7,4) ハミング符号と呼ぶ（[2] 参照）。

0000	0 0 0 0 0 0 0	1000	1 0 0 0 1 1 1
0001	0 0 0 1 0 1 1	1001	1 0 0 1 1 0 0
0010	0 0 1 0 1 0 1	1010	1 0 1 0 0 1 0
0011	0 0 1 1 1 1 0	1011	1 0 1 1 0 0 1
0100	0 1 0 0 1 1 0	1100	1 1 0 1 0 0 1
0101	0 1 0 1 1 0 1	1101	1 1 0 1 0 1 0
0110	0 1 1 0 0 1 1	1110	1 1 1 0 1 0 0
0111	0 1 1 1 0 0 0	1111	1 1 1 1 1 1 1

この中から一次独立なものを選び出し，並び替えると

$$G = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix}$$

となる。G を生成行列と呼ぶ。符号化は線形変換

$$[a_1, a_2, a_3, a_4, p_1, p_2, p_3] = [a_1, a_2, a_3, a_4]\,G$$

で表される。また，生成式を変形すると

$$\begin{cases} a_1 + a_2 + a_3 \qquad + p_1 \qquad\quad = 0, \\ a_1 + a_2 \qquad + a_4 \qquad + p_2 \qquad = 0, \\ a_1 \qquad + a_3 + a_4 \qquad\quad + p_3 = 0 \end{cases}$$

なる。これから得られる行列

$$H = \begin{bmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 1 \end{bmatrix}$$

を検査行列と呼び，次の性質が復号に利用される。

任意のハミング符号 $a_1\,a_2\,a_3\,a_4\,p_1\,p_2\,p_3$ に対して

$$[a_1, a_2, a_3, a_4, p_1, p_2, p_3]\,H^T = [0, 0, 0]$$

である（H^T は H の転置行列）。

4.4　ハミング符号による復号

4 ビットの情報 $a_1\,a_2\,a_3\,a_4$ を符号化した送信ビット列 $\boldsymbol{x} = a_1\,a_2\,a_3\,a_4\,p_1\,p_2\,p_3$ が通信路を通して送信されたとき，受信ビット列を $\boldsymbol{y} = b_1\,b_2\,b_3\,b_4\,b_5\,b_6\,b_7$ とする。この $\boldsymbol{y}$ を元の 4 ビットの情報 $a'_1\,a'_2\,a'_3\,a'_4$ に変換すること

$$b_1\,b_2\,b_3\,b_4\,b_5\,b_6\,b_7 \;\longmapsto\; a'_1\,a'_2\,a'_3\,a'_4$$

を復号という。しかし，受信ビット列 $\boldsymbol{y}$ は送信ビット列 $\boldsymbol{x}$ と異なることがあるから，正しい復元

$$a'_1\,a'_2\,a'_3\,a'_4 = a_1\,a_2\,a_3\,a_4$$

がなされるとは限らない。これを通信中に雑音ビット列 $\boldsymbol{e} = e_1\,e_2\,e_3\,e_4\,e_5\,e_6\,e_7$（雑音を 1 とする）が加わって，$\boldsymbol{y} = \boldsymbol{x} + \boldsymbol{e}$ となったと考える。

	a_1	a_2	a_3	a_4	p_1	p_2	p_3	$\cdots$	$\boldsymbol{x}$ (送信列)
+	e_1	e_2	e_3	e_4	e_5	e_6	e_7	$\cdots$	$\boldsymbol{e}$ (雑音列)
	b_1	b_2	b_3	b_4	b_5	b_6	b_7	$\cdots$	$\boldsymbol{y}$ (受信列)

このとき，関係式

$$\boldsymbol{y} + \boldsymbol{e} = (\boldsymbol{x} + \boldsymbol{e}) + \boldsymbol{e} = \boldsymbol{x}$$

が成り立つから，もし雑音ビット列 $\boldsymbol{e}$ が分かれば送信ビット列 $\boldsymbol{x}$ の復元（誤りの訂正）ができて，情報 $a_1\,a_2\,a_3\,a_4$ が正しく伝送されることになる。

ハミング符号は，雑音ビット列 $\boldsymbol{e}$ の中の雑音がただ1箇所に限るとき，誤り訂正が可能であるという性質をもつ。復号の手順を以下に示す。

受信ビット列 $\boldsymbol{y}$ を検査行列 H で変換した3ビット

$$[s_1, s_2, s_3] = \boldsymbol{y}\,H^T$$

をシンドロームという。$\boldsymbol{x}\,H^T = [0,0,0]$ であったから

$$\boldsymbol{y}\,H^T = (\boldsymbol{x}+\boldsymbol{e})\,H^T = \boldsymbol{x}\,H^T + \boldsymbol{e}\,H^T = \boldsymbol{e}\,H^T$$

であり，シンドロームは雑音ビット列 $\boldsymbol{e}$ だけから決定する。$\boldsymbol{e}$ の中の雑音がただ1箇所に限るとき，シンドローム $\boldsymbol{s} = s_1\,s_2\,s_3$ は $\boldsymbol{e}$ と次のように1対1に対応する。

誤り位置	雑音ビット列 $\boldsymbol{e}$	シンドローム $\boldsymbol{s}$
1	1 0 0 0 0 0 0	1 1 1
2	0 1 0 0 0 0 0	1 1 0
3	0 0 1 0 0 0 0	1 0 1
4	0 0 0 1 0 0 0	0 1 1
5	0 0 0 0 1 0 0	1 0 0
6	0 0 0 0 0 1 0	0 1 0
7	0 0 0 0 0 0 1	0 0 1
なし	0 0 0 0 0 0 0	0 0 0

したがって復号手順は次のようになる。

受信ビット列 $\boldsymbol{y}$ からシンドロームを計算.
対応表から誤り位置を定め，送信ビット列 $\boldsymbol{x}$ を復元.
ハミング符号表から情報ビット列 $a_1\,a_2\,a_3\,a_4$ を決定.

例えば受信ビット列 $\boldsymbol{y}$ が 0 1 0 1 1 1 1 のとき，以下のような復号過程となる。

シンドローム $[s_1, s_2, s_3] = \boldsymbol{y}\,H^T = [0,1,0]$ の計算

0101111	0101111	0101111
1110100	1101010	1011001
0100100	0101010	0001001
$s_1 = 0$	$s_2 = 1$	$s_3 = 0$

($\boldsymbol{y}$と行列 H の各行と，要素ごとの積の排他的論理和)

対応表より誤り位置6を知り，送信ビット列 $\boldsymbol{x}$ を復元（受信ビット列 $\boldsymbol{y}$ の第6ビットを反転）

$$\begin{array}{rcl} \boldsymbol{y} & = & 0\,1\,0\,1\,1\,1\,1 \\ +\boldsymbol{e} & = & 0\,0\,0\,0\,0\,1\,0 \\ \hline \boldsymbol{x} & = & 0\,1\,0\,1\,1\,0\,1 \end{array}$$

ハミング符号表から情報ビット列 $a_1\,a_2\,a_3\,a_4$ を復号

$\boldsymbol{x} = 0101101 \longrightarrow a_1a_2a_3a_4 = \underline{0101}$ (完了)

少し工夫して次のようにすれば，シンドロームと誤り位置の2進表現とを一致させることができる。

生成行列 $G = \begin{bmatrix} 1101001 \\ 0101010 \\ 1001000 \\ 1110000 \end{bmatrix}$，検査行列 $H = \begin{bmatrix} 0001111 \\ 0110011 \\ 1010101 \end{bmatrix}$

4.5 拡大ハミング符号

ハミング符号では1ビットの誤りは訂正できても2ビットの誤りは訂正できない。もし2ビットの誤りが発生すると，上記の誤り訂正の結果，復号に誤りが生ずことがある。2ビットの誤りを訂正できないまでもその検出が可能になるように改良したものが拡大ハミング符号である。生成行列 G の各行にパリティビットを追加して4行8列とし，検査行列 H を

$$H = \begin{bmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

とすればよい。シンドロームの第4ビットは，1ビット誤りのとき常に1であるのに対して，2ビット誤りのときは0となるので，その検出が可能となる。(染谷)

5 同期 – データを取出すための工夫

ビット列として高速に流れているに wave 形式の音楽データに埋め込まれた透かし情報を読み取る際に，情報の開始位置が正確に特定される必要がある。そのために我々は2種類の同期を実現している。

5.1 ビット同期

5.1.1 ビット同期の目的

1ビットを表示するために2048サンプルのデータをひとまとめにしてフーリエ変換を施し，ある特定の周波数帯が閾値より大きいか小さいかで0か1かを表しデータとする。しかし，この2048サンプルの正確な先頭を見つけないと正しいデータ（0か1か）は得られない。そのために電子透かしとしてデータを埋め込む周波数帯とは別の周波数帯に2048サンプルの先頭を見つけるための透かしを埋め込み，それを抽出・識別することでビット同期（2048の先頭）とする。

ビット同期を埋め込む周波数帯は，マイクの周波数特性による信号の劣化を防ぐため，20kHzになるべく近く，かつ，互いの周波数帯に影響しあわないことを考え，21kHz帯にビット同期を埋め込むことを決めた。

5.1.2 ビット同期の入れ方

ビット同期はファイルの先頭から 2048 サンプルごとに 1（設定値を入力する），0（入力しない），1，0，…という順番で透かしを入れていく。基本的に透かしを埋め込むのは高周波になるので任意で値を入力しなければ 0 に近い非常に小さな値になると考えられる。

5.1.3 ビット同期の抽出

電子透かし抽出を行う対象ファイルのデータ部分を先頭から 2048 サンプル取り出し配列に入れ，その配列に対してフーリエ変換を行う。そしてビット同期情報を埋め込んだ周波数帯 (図 3.7 に示した配列 a の 896 番目) に現れた数値と 2048 サンプルの先頭位置 (何回目のフーリエ変換かを表す) を取得，保存しておく。この配列 a の中の位置 $J = 896$ は，ビット同期信号の周波数（$W = 21$ kHz）から計算式 (1)

$$J = W \div (S \div N) = 21 \div (48 \div 2048) = 896$$

によって得られる。次にサンプルの取得位置を 1 つだけ後ろにずらして配列に入れ，再びフーリエ変換し $a[J]$ の数値を取得する。そして先に保存しておいた値より大きければ値を更新し，さらにサンプルの先頭位置も更新する。これを 4096 回繰り返す（3 ビット分行うことで必ずビット同期情報の開始点が現れる）。その結果，保存されるサンプルの先頭位置は $a[J]$ の値が最大となる位置，すなわちビット同期情報の開始点と一致したものとなる。この時の先頭がデータ部分の電子透かしにおいても先頭となる。

図 5.1 は，2048 サンプルの束を 1 サンプルずつずらして 6144 回フーリエ変換して得られるフーリエ展開の係数 $a[J]$ の値の変化を，グラフで表したものである。このグラフは縦軸が $a[J]$ の値の大きさ，横軸がサンプルを取り出す際の先頭位置である。

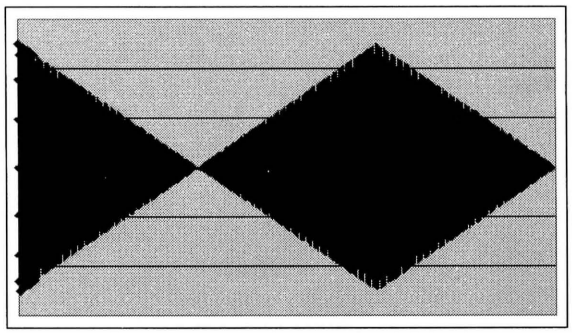


図 5.1 展開係数 $a[J]$ の値のグラフ

$a[J]$ の値は正にも負にもなり，図 5.1 はそのグラフがひし形のようになることを示す。$a[J]$ が最大値をとる位置を探しているので負の値は役に立たない。また値が上下に激しく変動するので値の頂点をつなぎ合わせた線（包絡線）の傾きが不安定になる。

これを改善するために絶対値処理を行うことを思いついた。その結果，最も大きい値をとる位置が正確に得られると考えられる（図 5.2 参照）。今後の改善点としたい。なお，絶対値には単純な $|a[J]|$ と複素係数 $a[J] + \mathrm{i}\, b[J]$ の絶対値 $\sqrt{a[J]^2 + b[J]^2}$ とがあるが，ビット同期情報を埋め込んだ周波数帯の信号の強さを意味している後者の方がよいであろう（付録 1(2) 参照）。

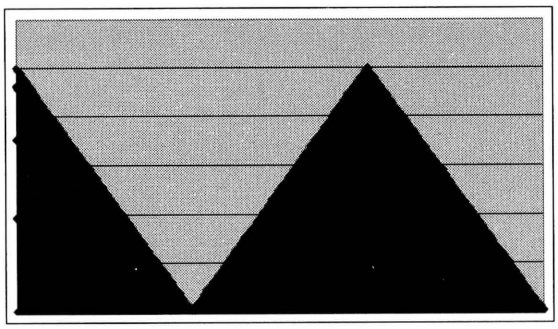


図 5.2 絶対値 $|a[J]|$ のグラフ

5.1.4 ビット同期とデータ用電子透かし

ここではデータ用電子透かしと同期用帯域の関係について 2048 サンプルを 1 ビットとするビットレベルで解説する。解説のために電子透かしを埋め込んだ音楽ファイルをスペクトラム・アナライザ（既製ソフト）で解析した図を使う（図 5.3）。スペクトラム・アナライザは，横に時間軸，縦に周波数をとり，音が強く出力されている周波数を赤に近い色で，音が弱いところを緑色で表示する。

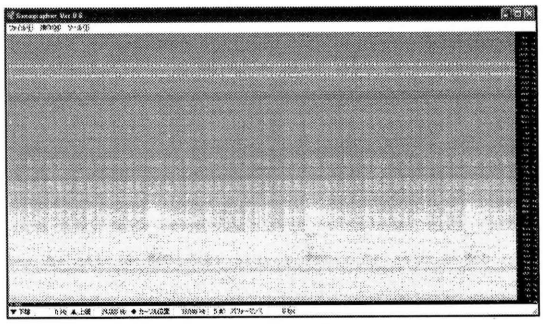


図 5.3 スペクトル分布

この図を見ると高周波帯域に 1 組の平行線ができているのがわかる。これが電子透かしであり，上側の線が同期を取るための電子透かし，下側の線が ID のデータが入っている電子透かしである。上側の線は 21kHz 帯，下側の線は 20kHz 帯にはいっている。これらの電子透かしは，音が出ているか出ていないかで 1 と 0 を表現し，そ

れを1ビットとしていることは既に説明したとおりである。わかりやすいように図にすると以下のような関係にある。

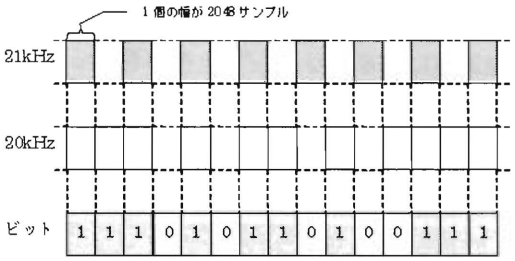


図 5.4　同期用電子透かしとデータ用電子透かし

下側のデータの電子透かしは，データ構造で説明したユニットのビット列を連続的に入れているため電子透かしの線が上側の同期を取るための電子透かしにくらべると不規則になっている。それにくらべ，上側の同期を取るための電子透かしは，音が出力されている1の状態と無音の0の状態が交互に規則的に並んでいる。0と1を交互に入れることで，同期を取るための電子透かしが付加されている21kHzの周波数に電子透かしが入っている2048サンプルをすばやく見つけ出し，データの電子透かしを正確に検出することができるようになっている。(斉藤)

5.2　ユニット同期

5.2.1　データ用透かしの構造

我々の電子透かしでは10進数4桁の音楽ID情報を付加することを想定している。この情報を桁ごとに2進数で表現すると各々4ビット必要になり，それをさらにハミング符号で表現するので各桁ごとに7ビットの長さになる。これをブロックと呼ぶことにする。図5.5は，1ブロックがデータ部の4ビットとパリティ部の3ビットであることを示している。

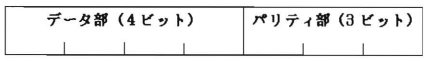


図 5.5　ブロック内構造

10進数4桁のIDに対応して4ブロックが必要になり，さらに先頭にID検出用のビットパターン（同期ビット列）をつける。同期ビット列も7ビット（1ブロック）からなるので合わせて5ブロックとなり，この5ブロック（35ビット）の塊をユニットと呼ぶ。同期ビット列はユニットの始まりを宣言し，次のブロックからIDのビット列が開始されることを意味している。同期ビット列があることで以降のブロックが何桁目の数字に対応するかがわかる。このユニットを曲中に繰り返し連続して挿入することにより，IDを曲のどの位置からでも読み出せるようになっている（図5.6)。

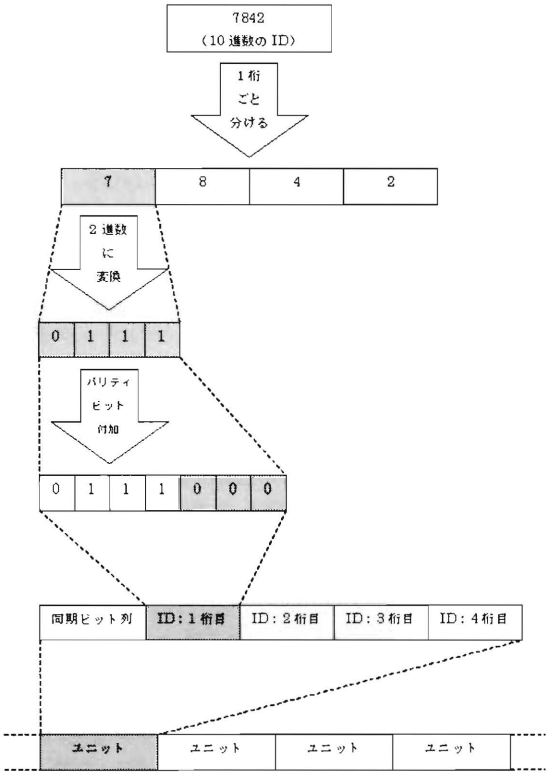


図 5.6　電子透かしデータ構造

ID情報を検出するときは同期ビット列と同じビットパターンを見つけ，そこから4ブロック分のビット列を連続して読み取り，ブロックごとに10進数に変換してID情報を抽出する。しかし，ビット誤り率が高いため，さらに次のユニットのビット列も読み取り，2ユニットを比較することでID抽出の精度を上げている。それでもデータが正しくない場合は，ハミング符号によって誤りを訂正し，正しいユニットのビット列を取得する。

（反省すれば，ここで場合分けをしないで無条件に誤り訂正復号すれば，改良版が得られるものと思われる。）

すでに述べたように我々の電子透かしは，2048サンプルごとに1ビットの情報を埋め込む。したがって，1ユニットのサンプル数は

$$2048 \times 7 \times 5 = 71680$$

である。これはサンプリングレートが48 kHzのとき，約1.5秒の時間間隔に相当する。

5.2.2　ユニット同期

同期ビット列の役割がユニットの先頭の宣言であることはすでに述べた。その必要性について説明をしておく。ビットレベルでは1と0が並んでいるだけなので，どこがIDの先頭（ユニットの先頭）か見つけることが難し

い。そのためにユニットの先頭に同期ビット列（7 ビット）を置く。同期ビット列を用いる同期を「ビット同期」と区別して,「ユニット同期」と呼ぶ。

図 5.7 は 7842 という数列を並べたものであるが，連続的に繰り返すので同期ビット列がないと読み始める場所がわからない。特定のビットパターンを同期ビット列とすれば，ID の先頭を知ることができる。

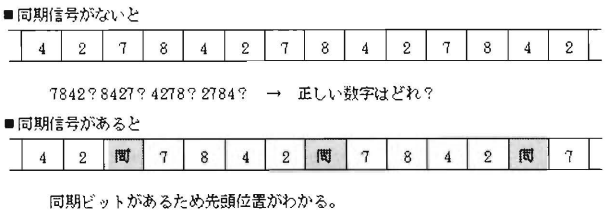


図 5.7　ユニット同期の効果

また，電子透かしは同期ビット列を起点としてその後に続くブロックの読み取りを行うため，同期ビット列は確実に正しい位置で検出しなければならない。同期ビット列には ID の表現に使われる 10 種類の数字の 2 進コード（+検査ビット）以外のパターンを使用する必要がある。しかし条件はこれだけではいけない。図 5.8 のように偶然隣り合ったブロックを跨ぐようにして同期ビット列と同じビットパターンが出現する可能性がある。この点も考慮して決めなければならない。

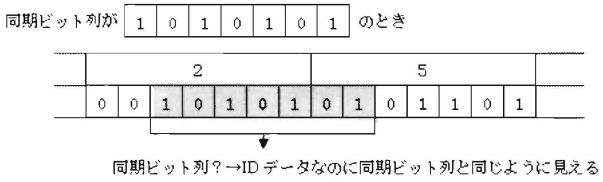


図 5.8　同期ビット列の誤検出

このような点を考慮した結果，我々の電子透かしには「0111111」というビットパターンを使用した。このほかにも同期ビット列になりうるビットパターンは存在する。（斉藤・増田）

6　発展・反省・課題

6.1　15 ビット版 – より多くの情報を

このプログラムはこれまでのプログラムの反省点をふまえ，より多くの情報を載せることを目標にしていたが，開発にかける時間がなくなってしまったため，実験段階のものとなっている。

これまでのプログラムはビットごとに同期を取り，ビットパターン照合を行って，ユニットの先頭を検出していたが，この方法ではビットパターンが偶然ビット反転により本来の位置以外の部分にも出現した場合に，誤作動を起こす可能性がある。また，同期ビット列自体がビット反転により損傷していた場合，同期ビット列にはエラー訂正符号化処理されていないため，修復もできず，ID を抽出できなくなってしまう。

そこで，同期ビットをブロックの先頭 1 ビットのみにつけるようにし，ブロックごとに同期をとることにした。また，拡大ハミング符号を用いてブロック長を 8 ビットに拡張し，2 ビットの誤りも検出できるようにした。8 ビットのうち前 4 ビットを情報ビット，後ろ 4 ビットを検査ビットとする。4 ビットの情報ビットのうち，先頭の 1 ビットは先頭フラグとする。これはユニットの先頭であることを示し，ユニットの先頭になるブロックにのみ付加する。残りの 3 ビットが情報の本体になる。このようなブロックを 5 つで 1 ユニットを組むと，3 ビット×5 ブロックで合計 15 ビットのデータを送信できる。これにより 2 の 15 乗，つまり 0～32767 までの ID の表現が可能になる。従来の表現範囲が 0～9999 までだったのに比べ，埋め込む情報は格段に増えたといえる。

以上のことをまとめると以下の図のようになる。

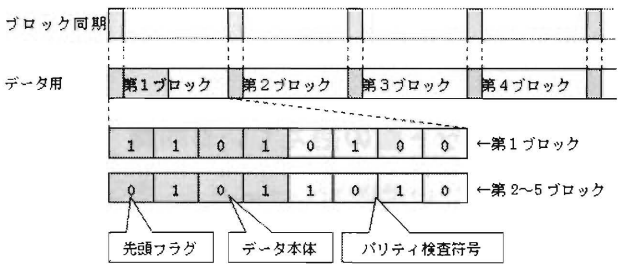


図 6.1　ブロック同期信号とデータ帯域の関係

データを読み取るときはまず，ビットの同期を取るときと同様に 2048 サンプルずつ FFT を行い，ブロック同期用の周波数帯の値が一番大きくなる部分を探し，そこをブロックの先頭位置であることを確定する。次に，その位置から無条件に 80 ビット分（2 ユニット分）を解析し，従来のプログラムと同様にビットを比較する方法により訂正する。

さらに拡大ハミング符号を使ってエラーをチェックし，訂正を行う。ここで訂正不能なエラーが発見された場合は処理を中止する。ここまでの処理でブロックごとに正しいビット列を得られたことになるが，このままではブロックの順序が狂っている可能性がある。もし，2 番目や 3 番目のブロックが先頭になっていた場合，ID が変化してしまい信頼性が失われてしまう。

そこで全ブロックの先頭フラグをチェックし，フラグの立っているブロックを探す。5 つのブロックはシーケンシャルに並んでいるため，先頭のブロックが見つかれば，正しい順序に並べ替えることが可能である。先頭フラグは並べ替えのキーとなるため，ビット反転は許されない。そのためパリティビットの生成時に先頭フラグとデータ本体をあわせて一つのデータとして処理し，エラー訂正により元のビット列を得られるようにしているのである。

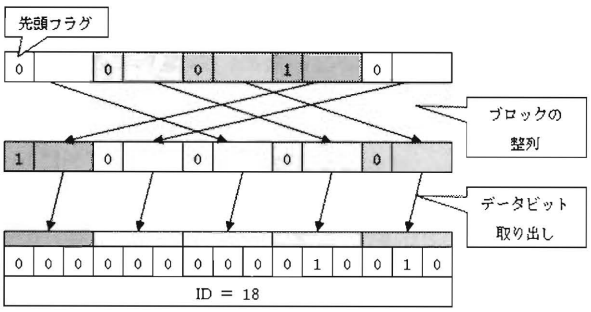


図 6.2　データの並べ替え

並べ替えたビット列の中からデータ本体の部分の 3 ビットを取り出して並べていく。これを結合したものが最終的に透かしのデータとなる。上図の場合 2 進数で 18 を表しているため，この透かしが入っていた曲の ID は 18 であることがわかる。

以上の改良版を実装したが，正確にブロックの同期信号の位置を読み取ることができなかった。今後この方式を開発するにはその問題を解決するための実験が必要である。

6.2　15 ビット版の抱えている問題

今回開発した 10 進版は将来的に付加できる情報量が足りなくなる問題が考えられるため，10 進版よりも多くの情報を付加できる 15 ビット版の開発を試みた。しかし，15 ビット版は 10 進版と同様の過程で透かしを抽出しようとしてもうまく抽出することができなかった。

具体的には，電子透かしを付加することは 10 進版と同様に可能だが，同期信号を埋め込んだ周波数領域に出てくる数値のグラフの頂点が不安定になり，10 進版のようにデータの先頭を正確に導き出すことができなかった。直接の解決策にはならないと思うが，現在使っている FFT のプログラムを sin 関数で行っているものを cos 関数で行うと多少精度が上がるかもしれない。これは，10 進版でも言えることなので，まず 10 進版で試してみるとよい。簡単に説明するとデータの先頭を導き出す際に sin 関数の場合頂点が 2 つ存在してしまうのに対し，cos 関数は頂点が 1 つだけしか存在せず，データの先頭が導き出しやすいためである。

6.3　高周波の限界

物理的な法則から高周波は減衰しやすい。また，我々の作成した電子透かしは高周波領域に情報を付加していることは既に説明してある。つまり，高周波領域に電子透かしを埋め込んだ wave ファイルを端末で録音して電子透かしを抽出するには，なるべく近くで録音しなければならないという『距離』の問題がでてくる。我々が実験した段階では 1.5 メートルまではほぼ 100 %の確率で取れることが確認できている。1.5 メートルでは距離が短すぎるため更なる改良が必要となる。

またプロジェクトの発表会の時に企業の方に指摘していただいた課題として高周波では使用している技術が簡単にわかってしまう事が挙げられる。音楽ファイルを不正に複製されないようにするために電子透かしを使う場合は，電子透かしを人の耳にも聞こえる低い周波数に付加している。我々の電子透かしは，人に使ってもらうための電子透かしで隠す必要がないとはいえ，高周波領域に付加されている様子はスペクトラムアナライザーを用いれば一目瞭然である。この様子から使っている技術がわかり，技術の盗用，複製も容易になってしまう。

6.4　スピーカーによる違い

電子透かしを開発しているときはいつも同じスピーカーを使っていたため気づかなかったが，スピーカーによってマイクの距離の間隔に差があることがプロジェクトの発表会の日にわかり，調整に苦労した。このスピーカーによる違いはまったく同じ型のスピーカーでも起きる可能性があり，どのスピーカーでも同様のサービスが提供できるようにするためには，プログラムの修正を行う必要があるかもしれない。（染谷）

あとがき

本学情報科学研究所の企画「イブニングセミナー」で，フーリエ変換の入門的な話をしたことがきっかけとなって，このプロジェクトの「電子透かし開発グループ」の学生諸君との交流が生まれた。

高い周波数の振動を断続的に付加することによって，同期信号や透かしを表現するアイディアに親近感をもった。これは典型的な離散フーリエ変換の応用の一つである。さらに誤り訂正符号を取り入れ，実際にその効果を発揮している状況を知ることができて，喜ばしい限りである。本文はこのグループ 3 名の学生の執筆によるものであるが，かなり綿密に議論をして分かりにくい表現などに手直しをお願いした。また，この開発で主役をつとめた離散フーリエ変換の定義式と計算結果，および同期信号の処理方式の提案を付録として載せることにした。

このグループの並々ならぬ努力と実力を高く評価したいと考えて，この活動報告を共著で本論集に投稿したが，それを快く歓迎してくださった編集委員に深く感謝したい。（佐藤）

付録1　フーリエ変換の定義と計算

この論文に記されているフーリエ変換の定義，および，ビット同期におけるフーリエ変換の値の計算結果を記す。

(1) フーリエ変換の定義

数学でフーリエ変換というと積分変換

$$f(x) \longmapsto \widehat{f}(\xi) = \int_{-\infty}^{\infty} f(x)\, \mathrm{e}^{\mathrm{i}\, 2\pi x \xi}\, \mathrm{d}x$$

を意味することが多いが，ここで応用されているものはその離散版である。離散フーリエ変換の対象は有限の長さの数列で，それを同じ長さの数列に変換する方法である。以下ではその長さを N で表し，それが偶数のときに限って記す（奇数 N の場合，わずかに変更すればよい。[1] 参照）。

離散フーリエ変換の定義には表現方法や定数の掛け方の違いでいくつかのヴァリエーションがあるが，基本的には，実数列 $\boldsymbol{x} = (x_n; n = 0, 1, \cdots, N-1)$ に対して

$$x_n = a_0 + \sum_{k=1}^{N/2} a_k \cos k\, n\, \theta + \sum_{k=1}^{N/2-1} b_k \sin k\, n\, \theta \qquad (2)$$
$$(\text{ここに，} \theta = 2\pi/N)$$

という有限三角級数（離散フーリエ展開）を求めることが目的である。展開係数 a_k, b_k は数列におけるそれぞれ振動数 k のコサイン波とサイン波の振幅（音の強さ）を表し，コサイン波とサイン波を合成した振動数 k の波の振幅は $\sqrt{a_k^2 + b_k^2}$ となる。a_0 は数列の平均値で，波形としては直流成分を意味する。

離散フーリエ変換は式 (2) を満たす N 個の係数 $a_k (k = 0, 1, \cdots, N/2)$, $b_k (k = 1, 2, \cdots, N/2-1)$ を求めることであるが，その計算には次の複素型の変換式が利用されることが多い。

$$y_n = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} x_k\, w^{k\, n} \quad (n = 0, 1, \cdots, N-1). \qquad (3)$$

ここに定数 w は，i を虚数単位として

$$w = \mathrm{e}^{\mathrm{i}\, \theta} = \cos\theta + \mathrm{i} \sin\theta$$

で定義される。w は単位円周の N 等分点であり，$w^N = 1$ が成り立つ。とくに y_0, $y_{N/2}$ は実数で，展開係数との間に

$$\begin{cases} a_0 = \dfrac{y_0}{\sqrt{N}}, \quad a_k = \dfrac{2\,\Re(y_k)}{\sqrt{N}}, \quad b_k = \dfrac{2\,\Im(y_k)}{\sqrt{N}}, \\ a_{N/2} = \dfrac{y_{N/2}}{\sqrt{N}}, \quad \sqrt{a_k^2 + b_k^2} = \dfrac{2\,|y_k|}{\sqrt{N}} \end{cases} \qquad (4)$$
$$(k = 1, 2, \cdots, N/2-1)$$

の関係がある（$\Re(z), \Im(z)$ はそれぞれ z の実数部と虚数部を表す）。$y_{N-k} = \overline{y_k}$ の性質（$\overline{z}$ は z の共役複素数）があるので，N 個の複素数は N 個の実数で表される。

(3) 式のフーリエ変換に対応する逆変換は

$$x_n = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} y_k\, w^{-k\, n} \quad (n = 0, 1, \cdots, N-1) \qquad (5)$$

である。これは (3) 式の w をその逆数 w^{-1} で置き換えた形となっている。

離散フーリエ変換，およびその逆変換の定義として

$$y_n = \sum_{k=0}^{N-1} x_k\, w^{k\, n} \quad (n = 0, 1, \cdots, N-1), \qquad (6)$$

$$x_n = \frac{1}{N} \sum_{k=0}^{N-1} y_k\, w^{-k\, n} \quad (n = 0, 1, \cdots, N-1) \qquad (7)$$

が用いられることある。この定義によれば，関係式 (4) は

$$\begin{cases} a_0 = \dfrac{y_0}{N}, \quad a_k = \dfrac{2\,\Re(y_k)}{N}, \quad b_k = \dfrac{2\,\Im(y_k)}{N}, \\ a_{N/2} = \dfrac{y_{N/2}}{N}, \quad \sqrt{a_k^2 + b_k^2} = \dfrac{2\,|y_k|}{N} \end{cases} \qquad (8)$$

となる。関係式 (4) に比べて分数の分母の値が大きくなることに注意されたい。

(2) 図 5.1 のグラフの概形と微細構造

本文 3.3 に記されているように，元データの 1 セット分 $\boldsymbol{x} = (x_n; n = 0, 1, \cdots, N-1)$ への電子透かしは次のようになされる。特定の整数 $J\,(0 < J \leq N/2)$ と定数 C を設定し，$\boldsymbol{x}$ のフーリエ変換 $\boldsymbol{y} = (y_n; n = 0, 1, \cdots, N-1)$ において $y_J = C$, $y_{N-J} = \overline{C}$ の置き換えを行い，それを逆変換したものが透かし入りデータ $\boldsymbol{x}'$ である。

例えば，サンプルレート $S = 48$ kHz，長さ $N = 2048$ のデータに周波数 $W = 21$ kHz の高周波を埋め込むならば，計算式 (1) により $J = W \div (S \div N) = 896$ となる。実際には，(6) 式を利用した FFT において $C = 2 \times 10^6$ としている。これは関係式 (8) により，$a_J \fallingdotseq 1953$, $b_J = 0$ に対応し，

$$a_J \cos J\, n\, \theta \qquad (n = 0, 1, \cdots, N-1) \qquad (9)$$

という振動を付加することにあたる。当然，$a_J = b_J = 0$ とおけば全くこの振動はなくなる（元々この周波数帯の振動が小さいならば，a_J, b_J を 0 とおく代わりに元の値のままにしても同じ効果がある）。周波数 W の振動の有無が 1 ビットの情報となっている。

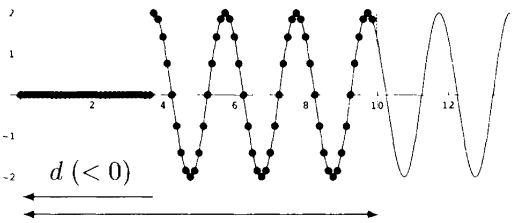
本文 5.1.3 によれば「ビット同期」の信号は，周波数 W の振動 (9) の埋め込みの有無を長さ N で交互に繰り返えすことで表現する。この振動の開始点が埋め込み情報の開始点と同期するので，その開始点の検出が重要

である。開始点の検出は，透かし入りデータから長さ N の枠を次々とスライドしながら1セット分をフーリエ変換し，$|y_J|$ の値に注目する。明らかに，枠全体に振動があれば $y_J = C$，枠内に振動が全くなければ $y_J = 0$ となる。その以外では $0 < |y_J| < |C|$ であると考えられるから（C が実数でも $|y_J|$ は複素数），$|y_J|$ が最大値をとる位置がわかればよい。

値 $|y_J|$ の変化が近似的に図5.1のような3角波になることを計算で示す。枠の位置が振動開始点より左に d サンプル分ずれているとき，枠内のデータの周波数 W の成分は，$d \le 0$ として

$$x_n^{(d)} = \begin{cases} 0, & 0 \le n \le -d-1, \\ A\cos J(n+d)\theta, & -d \le n \le N-1 \end{cases}$$

となる。簡単な例として，$A = 2, N = 80, J = 5, d = -30$ のときの数列 $x_n^{(d)}$ $(n = 0, 1, \cdots, N-1)$ のグラフを示す。



1セット分のデータ $x_n^{(d)}$ $(n = 0, 1, \cdots, N-1)$ のフーリエ変換を $y_n^{(d)}$ $(n = 0, 1, \cdots, N-1)$ とおく。$A = 2$ として，$\cos k\theta = (w^k + w^{-k})/2$ と等比級数の和の公式により，(6) 式の定義で計算すると

$$\begin{aligned} y_n^{(d)} &= \sum_{k=0}^{N-1} x_k^{(d)} w^{kn} \\ &= \sum_{k=-d}^{N-1} \left(w^{J(k+d)} + w^{-J(k+d)} \right) w^{kn} \\ &= w^{-nd} \sum_{k=0}^{N+d-1} \left(w^{(n+J)k} + w^{(n-J)k} \right) \end{aligned}$$

が得られる。とくに $n = J$ のとき，

$$\begin{aligned} y_J^{(d)} &= w^{-Jd} \sum_{k=0}^{N+d-1} \left(w^{2Jk} + 1 \right) \\ &= w^{-Jd} \left(\frac{1 - w^{2Jd}}{1 - w^{2J}} + N + d \right) \\ &= (N+d)\, w^{-Jd} + \frac{\sin J\theta\, d}{\sin J\theta}\, w^{-J} \end{aligned}$$

である（$y_J^{(0)} = N$, $y_J^{(N)} = 0$）。その実数部と虚数部は

$$\begin{aligned} \mathrm{Re}\, y_J^{(d)} &= (N+d)\cos J\theta\, d + (\cot J\theta)\sin J\theta\, d, \\ \mathrm{Im}\, y_J^{(d)} &= -(N+d+1)\sin J\theta\, d \end{aligned}$$

となる。これらは d の増加とともに直線的に振幅が増加する三角関数である。図5.1は実数部 $\mathrm{Re}\, y_J^{(d)}$ を縦軸，d を横軸にとったグラフであり，図5.2は絶対値 $|\mathrm{Re}\, y_J^{(d)}|$ を縦軸，d を横軸にとったグラフである（ただし，右半分のみ）。

その絶対値について

$$\begin{aligned} \left|y_J^{(d)}\right|^2 &= (N+d)^2 + \frac{1 - \cos 2J\theta\, d}{1 - \cos 2J\theta} \\ &\quad + (N+d)\left((\cot J\theta)\sin 2J\theta\, d - \cos 2J\theta\, d + 1 \right) \end{aligned}$$

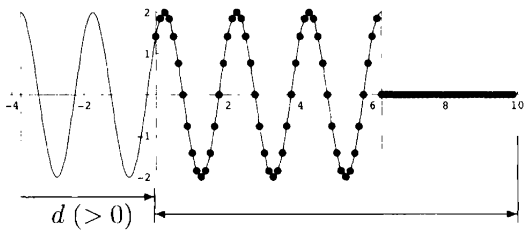
であり，テーラー展開による近似式

$$\left|y_J^{(d)}\right| \fallingdotseq N + d + \frac{1}{2}\left((\cot J\theta)\sin 2J\theta\, d - \cos 2J\theta\, d + 1 \right)$$

は，直線と三角関数の和の形で表されている。

同様にして，枠の位置が振動開始点より右に d サンプル分ずれているとき，$d > 0$ と考えて

$$x_n^{(d)} = \begin{cases} A\cos J(n+d)\theta, & 0 \le n \le N-d-1, \\ 0, & N-d \le n \le N-1 \end{cases}$$



とすれば，

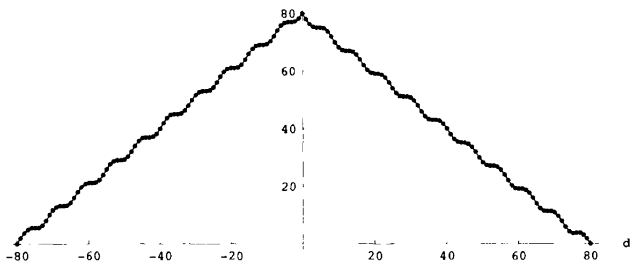
$$y_J^{(d)} = (N-d)\, w^{-Jd} - \frac{\sin J\theta\, d}{\sin J\theta}\, w^{-J},$$

$$\begin{aligned} \left|y_J^{(d)}\right|^2 &= (N-d)^2 + \frac{1 - \cos 2J\theta\, d}{1 - \cos 2J\theta} \\ &\quad - (N-d)\left((\cot J\theta)\sin 2J\theta\, d - \cos 2J\theta\, d + 1 \right), \end{aligned}$$

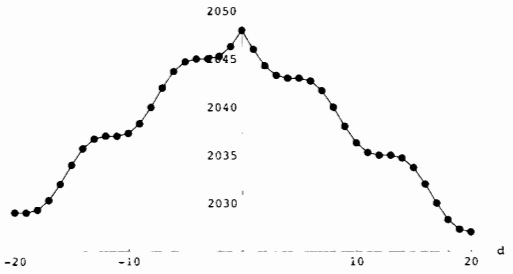
$$\left|y_J^{(d)}\right| \fallingdotseq N - d - \frac{1}{2}\left((\cot J\theta)\sin 2J\theta\, d - \cos 2J\theta\, d + 1 \right)$$

が得られる。

$-N \le d \le N$ の区間で絶対値 $\left|y_J^{(d)}\right|$ のグラフは近似的に三角形（$N - |d|$）になる（これを1周期として三角波を形成する）。



上の図は振動が誇張されているが，$N = 2048, J = 896$ で実装したときの絶対値 $|y_J^{(d)}|$ のグラフは，図 5.2（の輪郭）と同様に三角形とほとんど見分けがつかない。しかし，その頂上付近を拡大してみるとやはり次のように波打っていて，実際のデータではさらに雑音の影響があるからピークの検出はそれほど容易ではなかろう。



付録2　ピーク位置の推定法

本システムでは同期をとるために，付録1で述べた“枠”を1個ずつ右にずらしながら（d の値は不明），$2N$ 回実数部 $\mathrm{Re}\, y_J^{(d)}$ の値を計算し，それが最大値（図 5.1 の三角形のピーク）をとる位置を求めている（本文 5.1.3）。FFT の計算が速いので支障をきたさないものの，できるならフーリエ変換の計算回数を減らしたいところである。ここでは絶対値 $|y_J^{(d)}|$ のピーク位置を求める簡易計算法を提案する。

周期 $2N$ は定数として固定する。最初の観測点からピークの位置までのサンプルデータの個数を X とする。

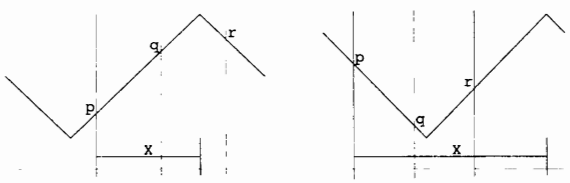
仮に信号が正確に三角波（上昇下降の傾斜と長さが等しい）場合には，2回または3回の計算で十分である。

(1) その最大値 A と最小値 B が既知ならば $p = |y_J^{(d)}|$ とその右隣りの値 $q = |y_J^{(d+1)}|$ から X がわかる（これは実用的ではない）。

$$\begin{cases} X = \dfrac{p-B}{A-B}N, & p < q \text{ のとき} \\ X = \dfrac{A+p-2B}{A-B}N, & p > q \text{ のとき} \end{cases}$$

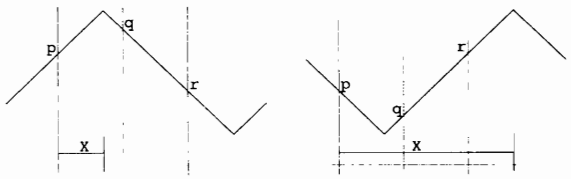
(2) 最大値と最小値が不明でも $N/2$ ずつ離れた3点の値 $p = |y_J^{(d)}|, q = |y_J^{(d+N/2)}|, r = |y_J^{(d+N)}|$ から次のようにして X の値がわかる。

- $|p-q| \le |q-r|$ のとき



$$\begin{cases} X = \dfrac{-p+2q-r}{4(q-r)}N, & q > r \text{ のとき} \\ X = \dfrac{-p+2q-r}{4(q-r)}N + N, & q < r \text{ のとき} \end{cases}$$

- $|p-q| < |q-r|$ のとき



$$\begin{cases} X = \dfrac{3p-2q-r}{4(q-r)}N, & p < q \text{ のとき} \\ X = \dfrac{3p-2q-r}{4(q-r)}N + N, & p > q \text{ のとき} \end{cases}$$

実際には信号の減衰や雑音の混入により三角波に微小の振動が加わるので，この X の値は正確ではない。しかし，X の“理論値”付近の点を数個計算すれば実際の X の値を求めることができるであろう。

また，次のようなアイディアもある。

(3) (2) の3点に加えて，第4点 $s = |y_J^{(d+3N/2)}|$ を計算し，(2) と同じ方法で q, r, s から X' を求め，X との平均

$$X'' = \frac{1}{2}\big(X + \mathrm{mod}\,(X', 2N)\big)$$

の付近を探索する。

(4) 等間隔にスキップしながら $|y_J|$ を計算し，値が減少したら例えば半分の間隔で戻ることを繰り返す。

(5) ランダムに例えば数10個の点を計算し，統計処理を施す．例えば，最小二乗法で三角形の頂点の位置を推定する。

実際に数値計算によって試してみるとよいが，興味深い結果を得ることが期待される。（佐藤）

参考文献

[1] 佐藤 創，「離散 Fourier 変換とその性質」，講義資料，2000.

[2] イエルン ユステセン，トム ホーホルト（坂田省二郎ほか訳），「誤り訂正符号入門」，森北出版，2005.

[3] 松井甲子雄，「電子透かしの基礎」，森北出版，1998.