ウェーブレット変換して遊ぶ

目　　次

1. 概要 1

2. 試行環境 2

3. ファイル構成 2

4. スクリプトの実行 2

5. ウェーブレットの多重解像度分解 4

6. スケーリング関数とウェーブレット母関数(ハール関数) 5

6-1. スケーリング関数による近似 6

6-2. ウェーブレット母関数による誤差 7

7. ウェーブレット変換 8

7-1. ハール関数におけるウェーブレット変換 8

7-2. ウェーブレット変換の一般化 9

7-3. ウェーブレット変換の実装 10

8. あとがき 11

9. 参考文献 11

1. 概要

ウェーブレットは、フランス人 Morlet によって、石油探査で人口地震波の解析に用いられたと言われているらしい。JPEG2000 では、DCT の代わりに採用されいるので、10 年ほど前に少しだけ勉強した覚えがある。昔参考にした書籍を読み返しているうちに、なんとなくまとめておきたくなった。ちなみに、ウェーブレット(wavelet)とは、辞書によると「小さな波」や「さざなみ」を意味する。

ウェーブレット解析の多重解像度分解は直感的に分かりやすい。関数を大まかに近似して、それと残りの部分とに分ける。その近似を更に近似と残りの部分とに分ける…といったことの繰り返し。そこには、難しい理論は存在しない。こうした簡単な概念の歴史は意外と浅いことに驚かされる。この感覚は、書籍「これなら分かる応用数学教室」で体感できる。

その手順は、スケーリング関数を平行移動しながら次々と近似し、信号を線形結合する。そして、次第にスケーリング関数の幅を広げて荒っぽく近似していく。その時の誤差成分を管理するのがウェーブレット関数である。ウェーブレット関数は、スケーリング関数と共に平行移動して、次第に幅を広げていく。

ここで、重要なのは、スケーリング関数とウェーブレット関数が直交関係にあることである。しかも、ウェーブレット関数は、その幅が広がっていく各段階(レベル)においても、直交基底を作っていることである。こうした互いの性質を持つ関数の組み合わせは、ハール関数のような実に簡単な特性で実現できるところにウェーブレット解析の凄さがあるように思える。

信号解析でよく用いられるフーリエ解析が、周波数領域で時間的な情報が失われるのに対して、ウェーブレット解析では時間的周波数解析が可能である。ウェーブレット解析とは、どの場所(空間情報)でどれだけのスケール(周波数情報)の振動があるかを同時に指定するものである。

2. 試行環境

Win2K + Cygwin 1.5.25-11 + Octave 2.1.73 + gnuplot 4.2.3
Fedora8 + Octave 3.0.0 + gnuplot 4.2.2
尚、gnuplot は、X 環境で試行。

[注意]
Fedora8 で wxWidgets をインストールしていると、gnuplot の Terminal type が default で'wxt'になる。
ちなみに、Cygwin のデフォルトは、’x11’。
とりあえず一緒にするために、“set terminal x11”(graph_*.bash 参照)を指定している。

3. ファイル構成

dwt1d.m	: 一次元離散ウェーブレット変換の関数定義	: octave スクリプト
idwt1d.m	: 一次元逆離散ウェーブレット変換の関数定義	: octave スクリプト
dwt2d.m	: 二次元離散ウェーブレット変換の関数定義	: octave スクリプト
idwt2d.m	: 二次元逆離散ウェーブレット変換の関数定義	: octave スクリプト
test_dwt1d.m	: 一次元離散ウェーブレット変換(テスト用)	: octave スクリプト
test_dwt2d.m	: 二次元離散ウェーブレット変換	: octave スクリプト
dwt2d.dat	: 二次元離散ウェーブレット変換後の生成データ	: テキスト
idwt2d.dat	: 二次元逆離散ウェーブレット変換後の生成データ	: テキスト
graph_rgb.bash	: RGB データをグラフ表示する	: bash スクリプト
sample_x_64x64.dat	: 画像サンプル	: テキスト
sample64x64.dat	: 画像サンプル	: テキスト
sample128x128.dat	: 画像サンプル	: テキスト
sample320x320.dat	: 画像サンプル	: テキスト

4. スクリプトの実行

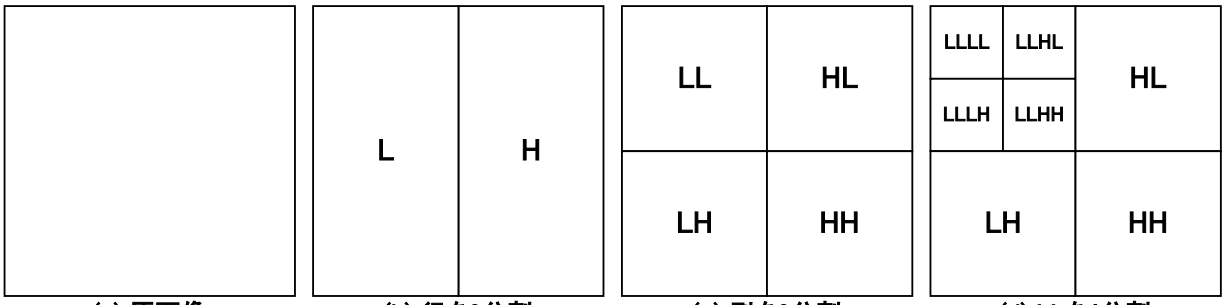
```
(コマンド 1) test_dwt2d.m [画像サイズ] [データファイル]
(コマンド 2) graph_rgb.bash [RGB ファイル]
    [使用例]
  $ ./test_dwt2d.m -64 sample64x64.dat      #  二次元ウェーブレット変換の実行
  $ ./graph_rgb.bash sample64x64.dat        #  サンプル画像の表示
  $ ./graph_rgb.bash dwt2d.dat              #  ウェーブレット変換後の表示
  $ ./graph_rgb.bash idwt2d.dat             #  逆ウェーブレット変換後の表示
```

！注意: 画像サイズは、正方形を前提にしている。よって、-64 を指定すると 64×64 となる。

test_dwt2d.m の実行によって生成するデータは以下のとおり。

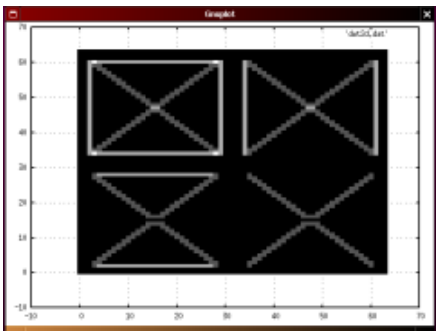
dw2d.dat	: 二次元離散ウェーブレット変換後のデータ
idw2d.dat	: 二次元逆離散ウェーブレット変換後のデータ

二次元ウェーブレット変換は、まず行方向に対して一次元ウェーブレット変換を行い、低域成分と高域成分に分割する(下図(b)参照)。次に、列方向に低域成分と高域成分に分割する(下図(c)参照)。これで一組の二次元処理が完了する。更に深く二次元処理をする時は、LL の部分を分割する(下図(d)参照)。尚、本プログラムは 2 分割まで対応している。

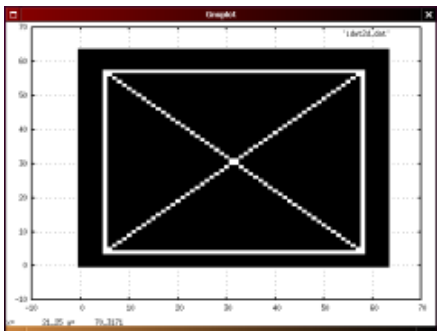


(a) 原画像　(b) 行を2分割　(c) 列を2分割　(d) LLを4分割

図 4.1 二次元ウェーブレットの処理イメージ

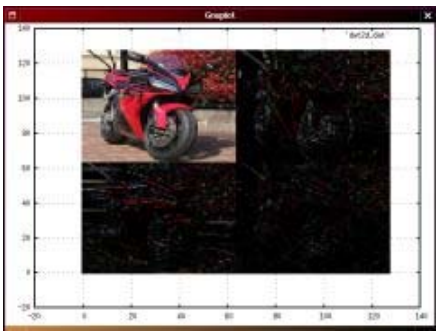


(a) 二次元 DWT



(b) 二次元 IDWT

図 4.2 sample_x_64x64.dat による DWT/IDWT



(a) 二次元 DWT

(b) 二次元 IDWT

図 4.3 sample128x128.dat による DWT/IDWT

[実行時間の測定]

以下、ドベシィ数列の N=1 で実行した結果を示す。

Win2K + Cygwin 1.5.25-11 + Octave 2.1.73 + gnuplot 4.2.3
CPU : Intel Pentium4 2.8GHz
Memory : 1GB

---

width x height = 64 x 64 : data = sample64x64.dat process time = 00:04:22.452
width x height = 128 x 128 : data = sample128x128.dat process time = 00:17:38.858
width x height = 320 x 320 : data = sample320x320.dat process time = 01:49:31.217

Fedora8 + Octave 3.0.0 + gnuplot 4.2.2
CPU : AMD Athlon ThunderBird 1.2GHz
Memory : 512MB

---

width x height = 64 x 64 : data = sample64x64.dat process time = 00:00:27.754
width x height = 128 x 128 : data = sample128x128.dat process time = 00:01:50.693
width x height = 320 x 320 : data = sample320x320.dat process time = 00:11:30.389

！コメント: Linux 環境の方がマシンが非力であるにも関わらず、処理時間が思いっきり速い。
特に Cygwin 環境で、320 x 320 画像を動かすと 2 時間近くもかかるので、やる気がしない。
尚、処理時間の見方は、hh:mm:ss.xxx(時:分:秒.xxx)である。
<u>処理時間からしてプログラムの書き方にセンスの無さを感じる。</u>

5. ウェーブレットの多重解像度分解

任意の信号 $f_0(x)$に対して、近似値 $f_1(x)$と、誤差値 $g_1(x)$に分解する。更に近似値 $f_1(x)$を分解し、それを繰り返す。
その多重分解を、次式で表す。

$$
\begin{aligned}
f_0(x) &= g_1(x) + f_1(x) \\
&= g_1(x) + g_2(x) + f_2(x) \\
&= g_1(x) + g_2(x) + g_3(x) + f_3(x) \\
&\quad \cdots \\
&= g_1(x) + g_2(x) + g_3(x) + g_4(x) + \cdots + g_n(x) + f_n(x)
\end{aligned}
\qquad \text{……式 5.1}
$$

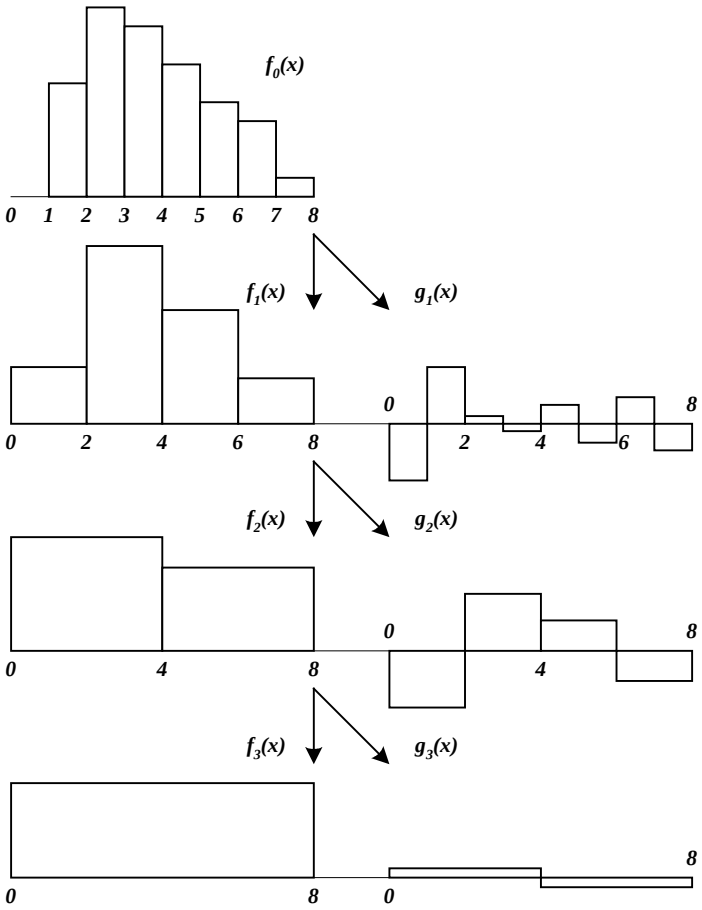


図 5.1 多重分解のイメージ (N=8)

このように、ウェーブレット解析の概念は単純である。任意の信号を異なるスケール成分へ分解するところは、フーリエ解析に似ている。

6. スケーリング関数とウェーブレット母関数(ハール関数)

スケーリング関数 $\Phi(x)$とウェーブレット母関数 $\Psi(t)$は、互いに直交し、以下の関係を満たす必要がある。

・ スケーリング関数を横に 2 倍広げたものは、スケーリング関数を平行移動したものの線形結合で表せる。

・ ウェーブレット母関数を横に 2 倍広げたものは、スケーリング関数を平行移動したものの線形結合で表せる。

上記の条件を満たす様々な関数が知られているが、ここでは簡単なハール関数を取り上げる。

$$\Phi(x) = \begin{cases} 1 & (0 \leqq x < 1) \\ 0 & (otherwise) \end{cases} \quad \text{……式 6.1}$$

$$\Psi(x) = \begin{cases} 1 & (0 \leqq x < 1/2) \\ -1 & (1/2 \leqq x < 1) \\ 0 & (otherwise) \end{cases} \quad \text{……式 6.2}$$

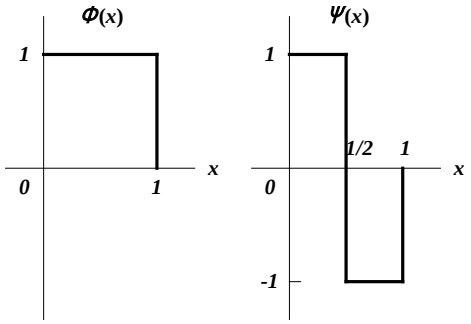


図 6.1 スケーリング関数 $\Phi(x)$とウェーブレット母関数 $\Psi(t)$

ここで、分解する段数をレベル j、平行移動を k とし、スケール関数 $\Phi_{j,k}(x)$ とウェーブレット母関数 $\Psi_{j,k}(t)$を多重分解に適応したイメージが下図である。

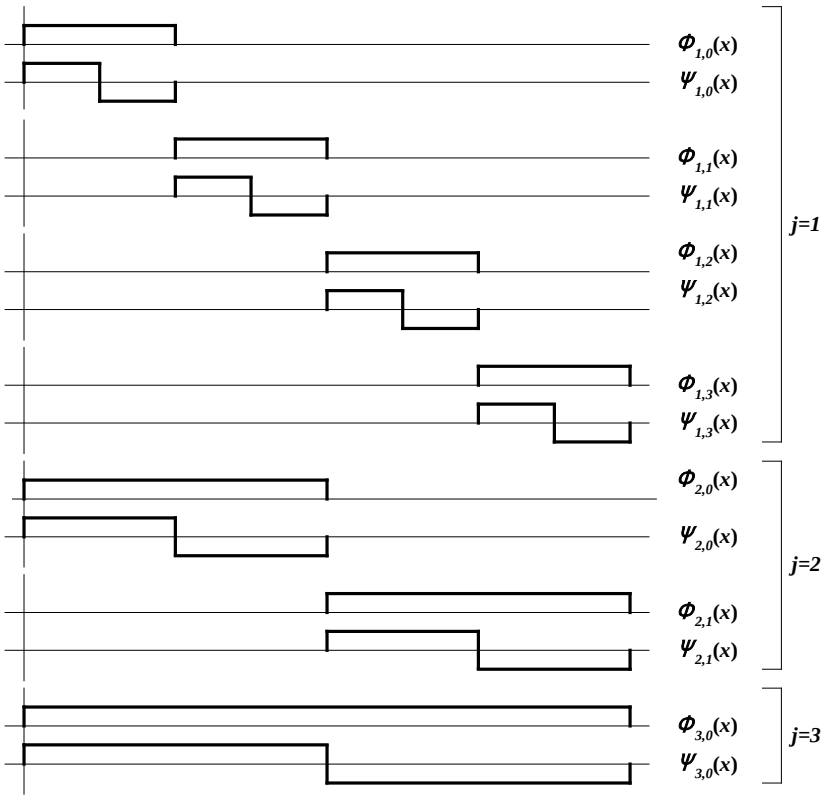


図 6.2 スケーリング関数とウェーブレット母関数を多重分解に適応したイメージ

6-1. スケーリング関数による近似

連続信号 $f_0(x)$から一定間隔で N 個の値 $c_0, c_1, \ldots, c_{n-1}$を…サンプルしたとする。
これのスケール成分をスケーリング関数 $\Phi(x)$で表す。

$$f_0(x) = \sum_{k=0}^{N-1} c_{0,k}\, \Phi(x - k)\,, \qquad c_{0,k} = c_k \qquad \text{……..式 6.1.1}$$

$\Phi(x - k)$は、スケーリング関数を k だけ平行移動したものである。

$$f_1(x) = \sum_{k=0}^{N/2-1} c_{1,k}\, \Phi(x/2 - k)\,, \qquad c_{1,k} = (c_{0,2k} + c_{0,2k+1})\,/\,2 \qquad \text{……..式 6.1.2}$$

$\Phi(x/2 - k) = \Phi((x - 2k)/2)$は、スケーリング関数を 2 倍の幅にして 2k だけ平行移動したものである。

$$f_2(x) = \sum_{k=0}^{N/4-1} c_{2,k}\, \Phi(x/4 - k)\,, \qquad c_{2,k} = (c_{1,2k} + c_{1,2k+1})\,/\,2 \qquad \text{……..式 6.1.3}$$

$\Phi(x/4 - k) = \Phi((x - 4k)/4)$は、スケーリング関数を 4 倍の幅にして 4k だけ平行移動したものである。
これを一般化すると、次のようになる。

$$f_j(x) = \sum_{k=0}^{N/2^j-1} c_{j,k}\, \Phi(x/2^j - k)\,, \qquad c_{j,k} = (c_{j-1,2k} + c_{j-1,2k+1})\,/\,2 \qquad \text{……..式 6.1.4}$$

ここで、次のようにおくと、

$$\Phi_{j,k}(x) = \Phi(x/2^j - k) \qquad \text{……..式 6.1.5}$$

注目すべきは、次の関係が成り立つことである（当り前なのだが…）。
つまり、各レベル j において内積が 0 ということは、平行移動しながら直交基底を作っている。

$$\int_0^N \Phi_{j,k}(x)\, \Phi_{j,k'}(x)dx = 0, \quad k \neq k' \qquad \text{……..式 6.1.6}$$

6-2. ウェーブレット母関数による誤差

次に、誤差成分をウェーブレット母関数 $\Psi(t)$で、表す。

$$
\begin{aligned}
g_1(x) = f_0(x) - f_1(x) \\
&= \begin{cases} c_{0,0} - (c_{0,0} + c_{0,1}) / 2 & 0 \leqq x < 1 \\ c_{0,1} - (c_{0,0} + c_{0,1}) / 2 & 1 \leqq x < 2 \\ c_{0,2} - (c_{0,2} + c_{0,3}) / 2 & 2 \leqq x < 3 \\ c_{0,3} - (c_{0,2} + c_{0,3}) / 2 & 3 \leqq x < 4 \\ \cdots\cdots & \cdots\cdots \end{cases} \\
&= \begin{cases} (c_{0,0} - c_{0,1}) / 2 & 0 \leqq x < 1 \\ -(c_{0,0} - c_{0,1}) / 2 & 1 \leqq x < 2 \\ (c_{0,2} - c_{0,3}) / 2 & 2 \leqq x < 3 \\ -(c_{0,2} - c_{0,3}) / 2 & 3 \leqq x < 4 \\ \cdots\cdots & \cdots\cdots \end{cases} \\
&= \sum_{k=0}^{N/2-1} d_{1,k}\, \Psi(x/2 - k), \qquad d_{1,k} = (c_{0,2k} - c_{0,2k+1}) / 2
\end{aligned}
$$

……..式 6.2.1

$\Psi(x/2 - k) = \Psi((x - 2k)/2)$は、ウェーブレット母関数を 2 倍の幅にして 2k だけ平行移動したものである。

$$
\begin{aligned}
g_2(x) = f_1(x) - f_2(x) \\
&= \begin{cases} c_{1,0} - (c_{1,0} + c_{1,1}) / 2 & 0 \leqq x < 2 \\ c_{1,1} - (c_{1,0} + c_{1,1}) / 2 & 2 \leqq x < 4 \\ c_{1,2} - (c_{1,2} + c_{1,3}) / 2 & 4 \leqq x < 6 \\ c_{1,3} - (c_{1,2} + c_{1,3}) / 2 & 6 \leqq x < 8 \\ \cdots\cdots & \cdots\cdots \end{cases} \\
&= \begin{cases} (c_{1,0} - c_{1,1}) / 2 & 0 \leqq x < 2 \\ -(c_{1,0} - c_{1,1}) / 2 & 2 \leqq x < 4 \\ (c_{1,2} - c_{1,3}) / 2 & 4 \leqq x < 6 \\ -(c_{1,2} - c_{1,3}) / 2 & 6 \leqq x < 8 \\ \cdots\cdots & \cdots\cdots \end{cases} \\
&= \sum_{k=0}^{N/4-1} d_{2,k}\, \Psi(x/4 - k), \qquad d_{2,k} = (c_{1,2k} - c_{1,2k+1}) / 2
\end{aligned}
$$

……..式 6.2.2

$\Psi(x/4 - k) = \Psi((x - 4k)/4)$は、ウェーブレット関数を 4 倍の幅にして 4k だけ平行移動したものである。
これを一般化すると、次のようになる。

$$
g_j(x) = \sum_{k=0}^{N/2^j-1} d_{j,k}\, \Psi(x/2^j - k), \qquad d_{j,k} = (c_{j-1,2k} - c_{j-1,2k+1}) / 2
$$

……..式 6.2.3

ここで、次のようにおくと、

$$
\Psi_{j,k}(x) = \Psi(x/2^j - k)
$$

……..式 6.2.4

注目すべきは、次の関係が成り立つことである（これも当り前なのだが…）。
つまり、各レベル j において内積が 0 ということは、平行移動しながら直交基底を作っている。

$$
\int_0^N \Psi_{j,k}(x)\, \Psi_{j,k'}(x)dx = 0, \quad k \neq k'
$$

……..式 6.2.5

また、k≠k では、互いに重ならないから、$\Phi_{j,k}(x)$と $\Psi(t)_{j,k}$ の積は 0 となる。
k=k においても、前半は+1 倍で後半が-1 倍だから、積分すると 0 となる。
よって、任意の k において、下記の式が成り立つ（これまた当り前なのだが…）。
つまり、スケーリング関数 $\Phi(x)$とウェーブレット母関数 $\Psi(t)$の内積が 0 であり、直交関係にある。

$$
\int_0^N \Psi_{j,k}(x)\, \Phi_{j,k'}(x)dx = 0
$$

……..式 6.2.6

7. ウェーブレット変換

7-1. ハール関数におけるウェーブレット変換

ここで、任意の関数 $f_0(x)$は、上記の結果を代入すると、以下のようになる。

$$f_0(x) = g_1(x) + g_2(x) + \ .... + g_n(x) + f_n(x)$$

$$= \sum_{j=1}^{n} g_j(x) + f_j(x)$$

$$= \sum_{j=1}^{n} \sum_{k=0}^{N/2^j-1} d_{j,k}\, \Psi_{j,k}(x) + \sum_{k=0}^{N/2^i-1} c_{j,k}\, \Phi_{j,k}(x) \qquad \text{……式 7.1.1}$$

ここで、後ろの項は、最後のレベル j = n で、平行移動はしないので k = 0 となる(図 6.2 参照)。
また、$\Phi_{n,0}(x)$ =1 なので、上式は以下のようになる。

$$f_0(x) = \sum_{j=1}^{n} \sum_{k=0}^{N/2j-1} d_{j,k}\, \Psi_{j,k}(x) + c_{n,0} \qquad \text{……式 7.1.2}$$

ここで、ウェーブレット $\Psi_{j,k}(x)$と定数関数 $\Phi_{n,0}(x)$のノルムを求める。
$\Psi_{j,k}(x)$は、2j 上で±1 で、$\Phi_{n,0}(x)$ =1 なので、二乗積分は以下のようになる。

$$\| \Psi_{j,k} \|^2 = \int_0^N \Psi_{j,k}(x)^2\, \mathrm{dx} = \int_0^{2^j} dx = 2^j \qquad \text{……式 7.1.3}$$

$$\| \Phi_{n,0} \|^2 = \int_0^N \Phi_{n,0}(x)^2\, \mathrm{dx} = \int_0^N dx = N \qquad \text{……式 7.1.4}$$

よって、それぞれのノルムは以下のようになる。

$$\| \Psi_{j,k} \| = 2^{j-1}, \qquad \| \Phi_{n,0} \| = \sqrt{N} \qquad \text{……式 7.1.5}$$

ここで、重要なのが、以下の2つが、それぞれ正規直交基底となっていることである。

$$\{ \Psi_{j,k} / 2^{j-1} \}, \qquad \Phi_{n,0} / \sqrt{N} \qquad \text{……式 7.1.6}$$

そして、展開係数 $d_{j,k}$ と $c_{n,0}$ は、以下の式で表される。

$$d_{j,k} = 1/2^j \int_0^N f(x)\, \Psi_{j,k}(x)dx, \qquad c_{n,0} = 1/N \int_0^N f(x)dx \qquad \text{……式 7.1.7}$$

つまり、ウェーブレット変換とは、与えられた関数から展開係数を計算することと、その展開係数から元の関数を計算することである。

7-2. ウェーブレット変換の一般化

スケーリング関数 $\Phi(x)$は、多重分解する時に各段階(レベル)で直交関係が成り立っているなどのウェーブレットの性質を満足できれば良いので、ハール関数にこだわる必要はない。ただし、2 倍に広げると、もとのスケーリング関数の平行移動の線形結合で次のような形で表現されなければならない。

$$\Phi(x) = \sum_k p_k \Phi(2x - k) \qquad \text{……式 7.2.1}$$

これを 2 スケール関係式と呼ぶ。
$\{p_k\}$は、係数数列であり、ハール関数では、$p_0 = 1$, $p_1 = 1$ ということになる。

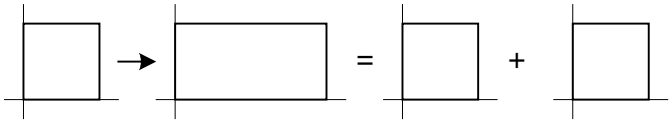


図 7.1 ハールのスケーリング関数の 2 スケール関係のイメージ

また、ウェーブレット母関数 $\Psi(x)$は、直交関係にあるスケーリング関数 $\Phi(x)$によって定まる。よって、2 倍に広げると、スケーリング関数の平行移動の線形結合で次のような形で表現されなければならない。

$$\Psi(x) = \sum_k q_k \Phi(2x - k) \qquad \text{……式 7.2.2}$$

$\{q_k\}$は、係数数列であり、ハール関数では、$q_0 = 1$, $q_1 = -1$ ということになる。

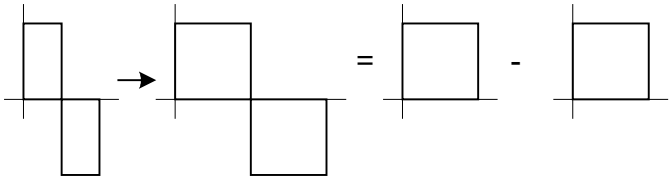


図 7.2 ハールのウェーブレット母関数と 2 スケール関係のイメージ

$\{p_k\}$, $\{q_k\}$は、生成数列と呼ばれる。

ここで、式 6.1.4 と式 6.2.3 を使って、一般化した分解アルゴリズムを示す。

$$c_{j+1,n} = 1/2 \sum_k g_k c_{j,2n+k}\ , \qquad d_{j+1,n} = 1/2 \sum_k h_k c_{j,2n+k} \qquad \text{……式 7.2.3}$$

$\{g_k\}$, $\{h_k\}$は、分解数列と呼ばれ、生成数列$\{p_k\}$, $\{q_k\}$により定まる。
ハール関数では、$g_0 = 1$, $g_1 = 1$, $h_0 = 1$, $h_1 = -1$ ということになる。
そして、合成アルゴリズムは以下のようになる。

$$c_{j,n} = \sum_k (p_k c_{j+1,n-k} + q_k d_{j+1,n-k}) \qquad \text{……式 7.2.4}$$

7-3. ウェーブレット変換の実装

ここでは、連続ウェーブレット変換に立ち返ろう。
横方向にスケール a 倍、横方向に b だけ平行移動したウェーブレット関数は次式で表せる。

$$\Psi_{a,b}(x) = 1/\sqrt{a}\ \Psi(\ (x - b)/a\ ) \qquad \text{……式 7.3.1}$$

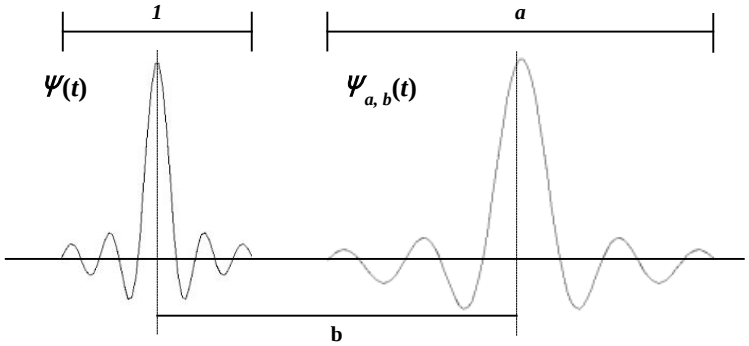


図 7.3 ウェーブレット母関数の変形イメージ

ここで、$1/\sqrt{a}$ は、正規化のための係数である。パラメータ a, b は、2 進分割によって、次のように離散化する。

$$a = 2^j,\ b = 2^j k \qquad \text{……式 7.3.2}$$

すると、ウェーブレット関数は、以下のように変形できる。

$$\Psi_{j,k}(x) = 1/\sqrt{2^j}\ \Psi(\ (x - 2^j k)\ /2^j)$$

$$= \sqrt{2^{-j}}\ \Psi(2^{-j} x - k) \qquad \text{……式 7.3.3}$$

つまり、スケーリング関数とウェーブレット関数は次のように表現できる。

$$\Phi_{j,k}(x) = \sqrt{2^{-j}} \sum_k p_k\ \Phi(2^{-j} x - k)\ , \qquad \Psi_{j,k}(x) = \sqrt{2^{-j}} \sum_k q_k\ \Phi(2^{-j} x - k) \qquad \text{……式 7.3.4}$$

よって、j=1, j=2, j=3 の時、次のように、$1/\sqrt{2}$ ずつ正規化されることになる。

$$\Phi_{1,k}(x) = 1/\sqrt{2} \sum_k p_k\ \Phi(2x - k)\ , \qquad \Psi_{1,k}(x) = 1/\sqrt{2} \sum_k q_k\ \Phi(2x - k)$$

$$\Phi_{2,k}(x) = 1/\sqrt{4} \sum_k p_k\ \Phi(4x - k)\ , \qquad \Psi_{2,k}(x) = 1/\sqrt{4} \sum_k q_k\ \Phi(4x - k) \qquad \text{……式 7.3.5}$$

$$\Phi_{3,k}(x) = 1/\sqrt{8} \sum_k p_k\ \Phi(4x - k)\ , \qquad \Psi_{2,k}(x) = 1/\sqrt{8} \sum_k q_k\ \Phi(4x - k)$$

よって、ハール関数も正規化して、$\{p_k\} = \{\ 1/\sqrt{2},\ 1/\sqrt{2}\ \}$を使うのが良いのだろう。
<u>＊ド素人だから自信がないが、後で述べるドベシィ数列で一般化されていて、N=1 の時がまさしく、ハール関数に相当している。</u>
ちなみに、上式を図 6.2 に対応したハールの基底ベクトルは下式ように表せる。

$$\begin{bmatrix} \Psi_{1,0}(x) \\ \Psi_{1,1}(x) \\ \Psi_{1,2}(x) \\ \Psi_{1,3}(x) \\ \Psi_{2,0}(x) \\ \Psi_{2,1}(x) \\ \Psi_{3,0}(x) \\ \Phi_{3,0}(x) \end{bmatrix} = 1/\sqrt{8} \begin{bmatrix} 2 & -2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & -2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & -2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 2 & -2 \\ \sqrt{2} & \sqrt{2} & -\sqrt{2} & -\sqrt{2} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \sqrt{2} & \sqrt{2} & -\sqrt{2} & -\sqrt{2} \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \qquad \text{……式 7.3.6}$$

ハール関数と同様に離散ウェーブレット変換に用いられるものに、ドベシィ(Daubechies)の数列がある。
数列の要素数を 2N 個とすると、ドベシィの数列$\{p_k\}$は、以下の条件を満たす。

$$\begin{cases} \sum_{k=0}^{2N-1} p_k = \sqrt{2} \\ \sum_{k=0}^{2N-1} p_k^2 = 1 \\ \sum_{k=0}^{2N-1} p_k p_{k+2n} = 0, \ (n \neq 0) \end{cases}$$

……..……..式 7.3.6

$$q_k = (-1)^k p_{2N-k-1}$$
$$g_k = p_k$$
$$h_k = q_k$$

……..……..式 7.3.7

上式の$\{p_k\}$がスケーリング関数として使われる。
ウェーブレットを表す数列$\{q_k\}$は、$\{p_k\}$を時間軸方向で反転し、係数の符号を交互に反転することにより得られる。
N = 1 の時を計算すると、次のようになる。これがハール関数の正規化したものに相当する。
$\{p_k\} = \{ 1/\sqrt{2}, 1\sqrt{2} \} = \{ 0.707106781, 0.707106781\}$

N = 2 以上は、参考文献に記載されている。プログラムでは、N = 2 を試してみた。
$\{p_k\} = \{ 0.482962913145 , 0.836516303738, 0.224143868041, -0.129409522551\}$

8. あとがき

数学のド素人が、昔を思い出しながら、なんとなく理解したところをまとめてみた。ハール関数までは簡単なしくみである。しかし、ドベシィ数列については、真面目に定理の勉強をする必要がありそうだ。おいらには、この数列がウェーブレットの性質を満足していると信じて使うしかない。

9. 参考文献

1.「これなら分かる応用数学教室」 金谷健一著 共立出版 2003 年 6 月

2.「ディジタル画像処理入門」 酒井幸市著 コロナ社 1997 年 12 月
この本は古いので、もう売ってないかもしれない。
ちなみに、CQ 出版社から OPEN DESIGN BOOKS 版があるようだ。
それには、サンプルプログラム(VC++)が CD で添付されているのを本屋で見かけた。

[改版履歴]

V1.0	新版	2008.05.01