1次元ウェーブレット変換

1．信号の階層的近似

連続信号から一定間隔でN個の値$c_0, c_1, \cdots, c_{N-1}$をサンプルしたとする．
ただしNは２のべき乗とする．すなわち　$N=2^n$
サンプル間隔を１とする長さの単位をとり，
信号を次の段階関数で近似する．

レベル0

$$f^{(0)} = \begin{cases} c_0 & 0 \le x < 1 \\ c_1 & 1 \le x < 2 \\ \vdots & \\ c_{N-1} & N-1 \le x < N \end{cases}$$

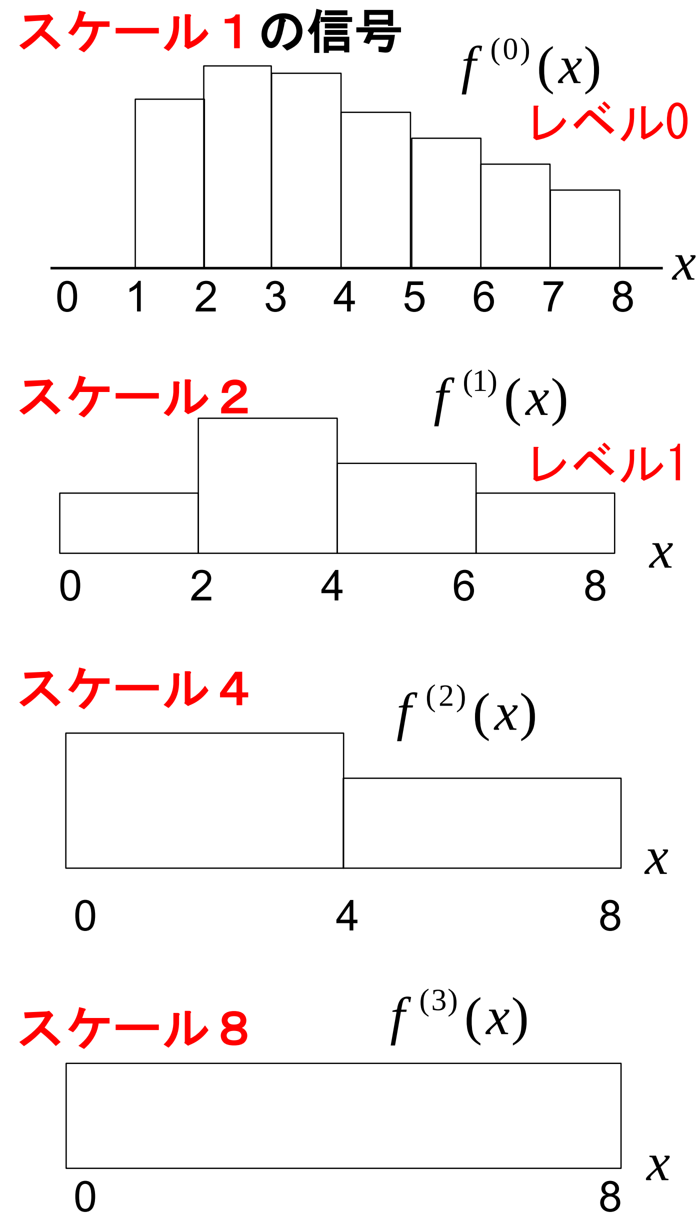
次に，連続する2画素ごとに，値をその平均値で置き換えたものをf ⁽¹⁾(x)とする．

レベル1

$$f^{(1)} = \begin{cases} c_0^{(1)} \ (=(c_0 + c_1)/2) & 0 \le x < 2 \\ c_1^{(1)} (=(c_2 + c_3)/2) & 2 \le x < 4 \\ \vdots & \\ c_{N/2-1}^{(1)} \ (=(c_{N-2} + c_{N-1})/2) & N-2 \le x < N \end{cases}$$

値が一定の区間の幅を**スケール**と呼ぶ．

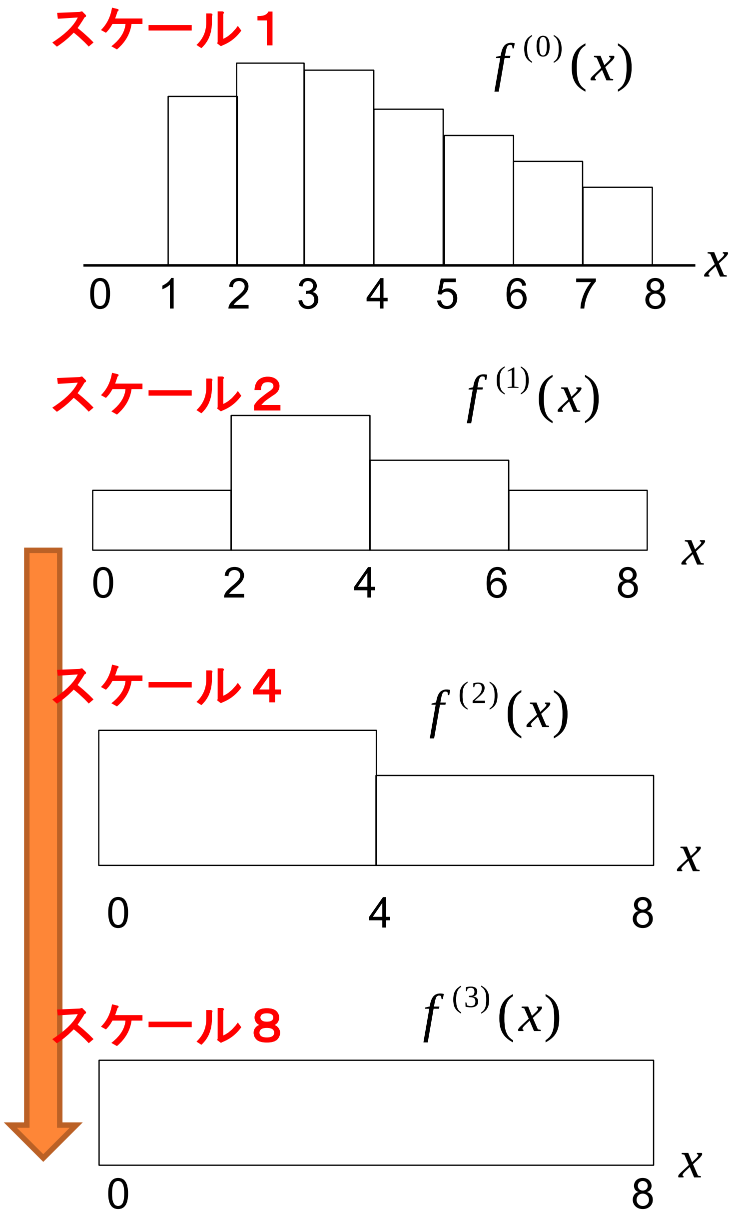
$N=8=2^3$の例



１．信号の階層的近似（つづき）

さらに，連続する４画素ごとに同様な平均操作を行ったスケール４の信号をf$^{(2)}$(x)とする．

$$
f^{(2)} = \begin{cases} c_0^{(2)} \; (= (c_0^{(1)} + c_1^{(1)})/2) & 0 \le x < 4 \\ c_1^{(2)} (= (c_2^{(1)} + c_3^{(1)})/2) & 4 \le x < 8 \\ \vdots & \\ c_{N/4-1}^{(2)} \; (= (c_{N/2-2}^{(1)} + c_{N/2-1}^{(1)})/2) & N-4 \le x < N \end{cases}
$$



以下同様にすると，最終的にf$^{(n)}$(x)がスケールNの定数関数となる．

$$
f^{(n)}(x) = c_1^{(n)} (= (c_0^{(n-1)} + c_1^{(n-1)})/2) \qquad 0 \le x < N
$$

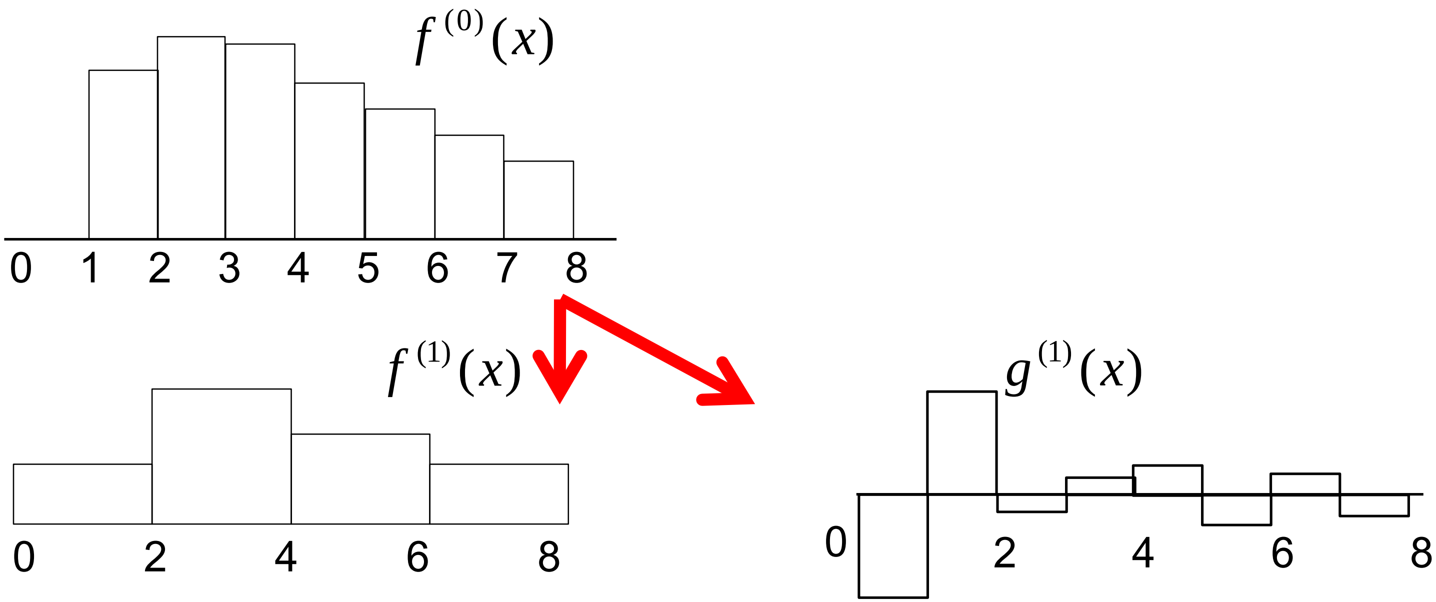
スケールの逆数を**解像度**と呼ぶ．

階層的近似：逐次的にさまざまな解像度の信号を作り出すこと．

２． 多重解像度分解

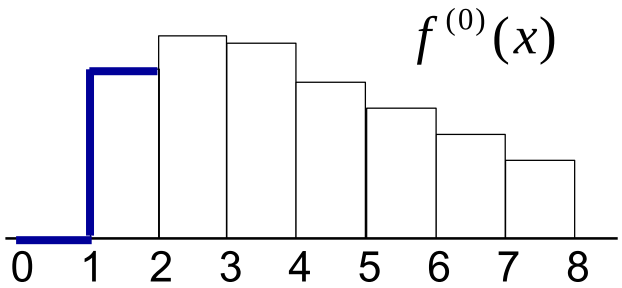
スケール１のもっとも詳細な信号
$f^{(0)}(x)$ とその次のスケール２の近似
$f^{(1)}(x)$ との差を $g^{(1)}(x)$ とする.

$$\boxed{f^{(0)}(x) = g^{(1)}(x) + f^{(1)}(x)}$$

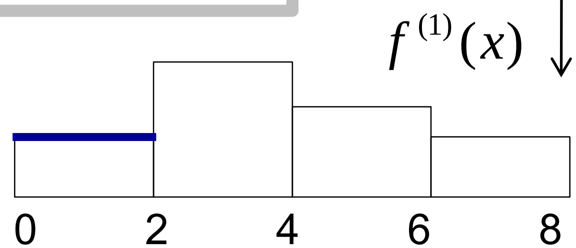


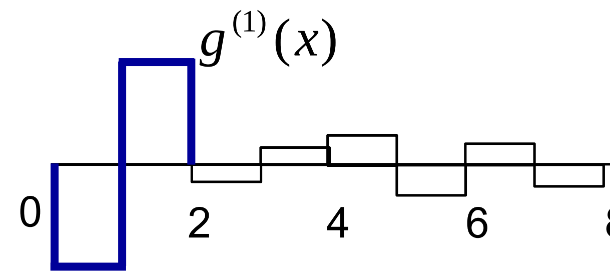
平均　　　　　　　　差分

２． 多重解像度分解



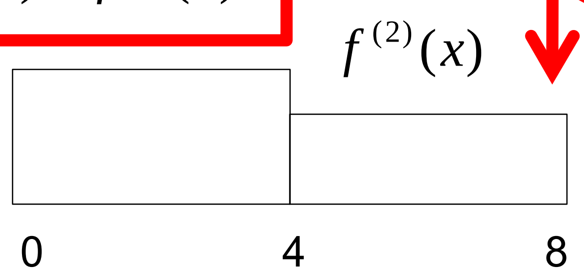
$$\boxed{f^{(0)}(x) = g^{(1)}(x) + f^{(1)}(x)}$$

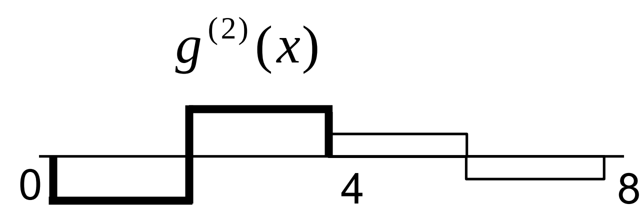




次に，スケール２の近似$f^{(1)}(x)$とスケール４の近似$f^{(2)}(x)$ との差を$g^{(2)}(x)$とする．

$$\boxed{f^{(1)}(x) = g^{(2)}(x) + f^{(2)}(x)}$$





２．多重解像度分解

スケール１のもっとも詳細な信号 $f^{(0)}(x)$ とその次のスケール２の近似 $f^{(1)}(x)$ との差を $g^{(1)}(x)$ とする.

$$f^{(0)}(x)=g^{(1)}(x)+f^{(1)}(x)$$

次に，スケール２の近似 $f^{(1)}(x)$ とスケール４の近似 $f^{(2)}(x)$ との差を $g^{(2)}(x)$ とする.

$$f^{(1)}(x)=g^{(2)}(x)+f^{(2)}(x)$$

同様に

$$f^{(2)}(x)=g^{(3)}(x)+f^{(3)}(x)$$

最終的に

$$f^{(n-1)}(x)=g^{(n)}(x)+f^{(n)}(x)$$

まとめると

$$\begin{aligned}
f^{(0)}(x)&=g^{(1)}(x)+f^{(1)}(x)\\
&=g^{(1)}(x)+g^{(2)}(x)+f^{(2)}(x)\\
&=g^{(1)}(x)+g^{(2)}(x)+g^{(3)}(x)+f^{(3)}(x)\\
&\ \ \vdots\\
&=g^{(1)}(x)+g^{(2)}(x)+\cdots+g^{(n)}(x)+f^{(n)}(x)
\end{aligned}$$

信号を異なるスケールの成分へ分解することを**多重解像度分解**と呼ぶ.

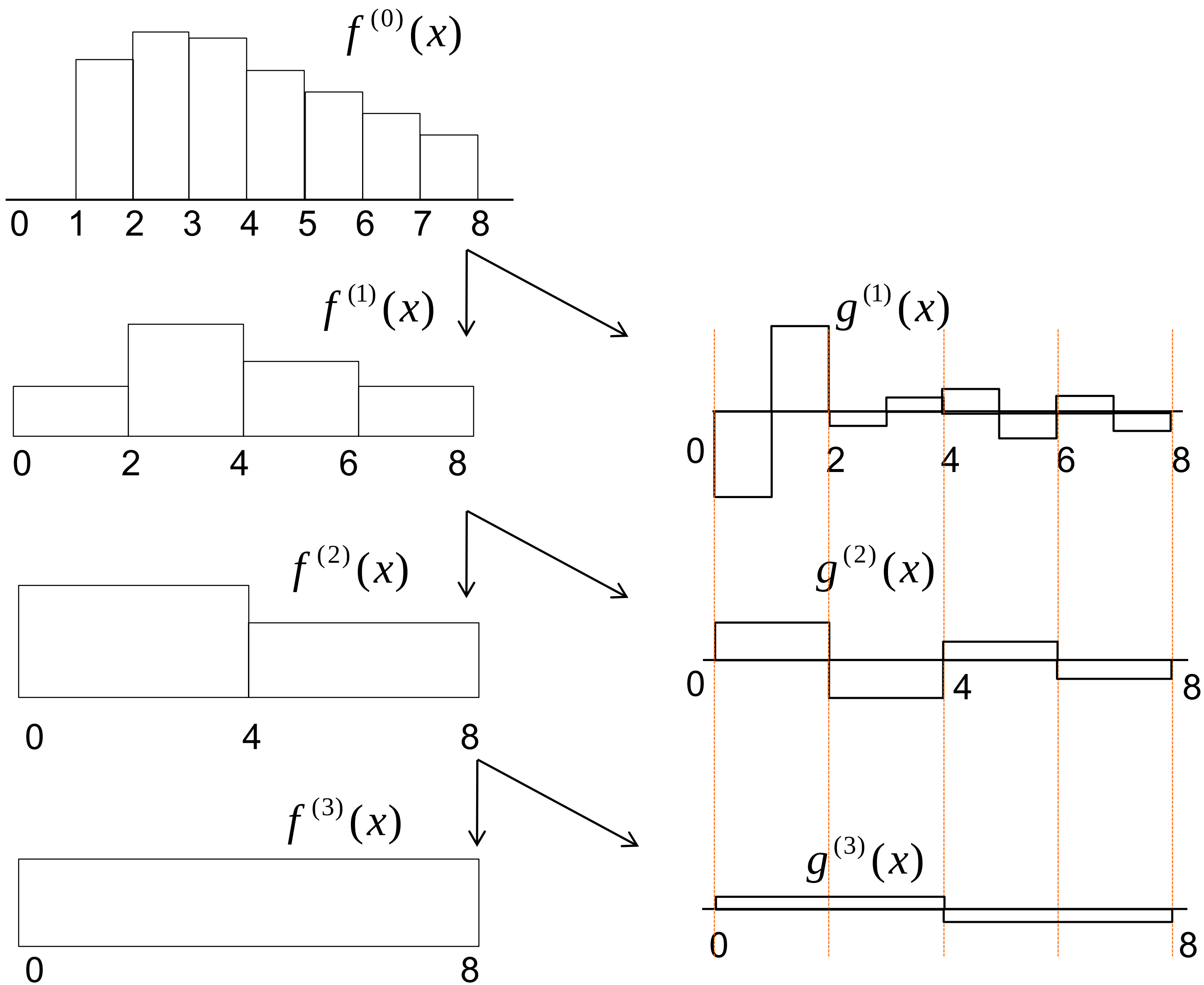
フーリエ変換とウェーブレット変換の違い

信号を異なるスケールの成分へ分解する
ことを**多重解像度分解**と呼ぶ.

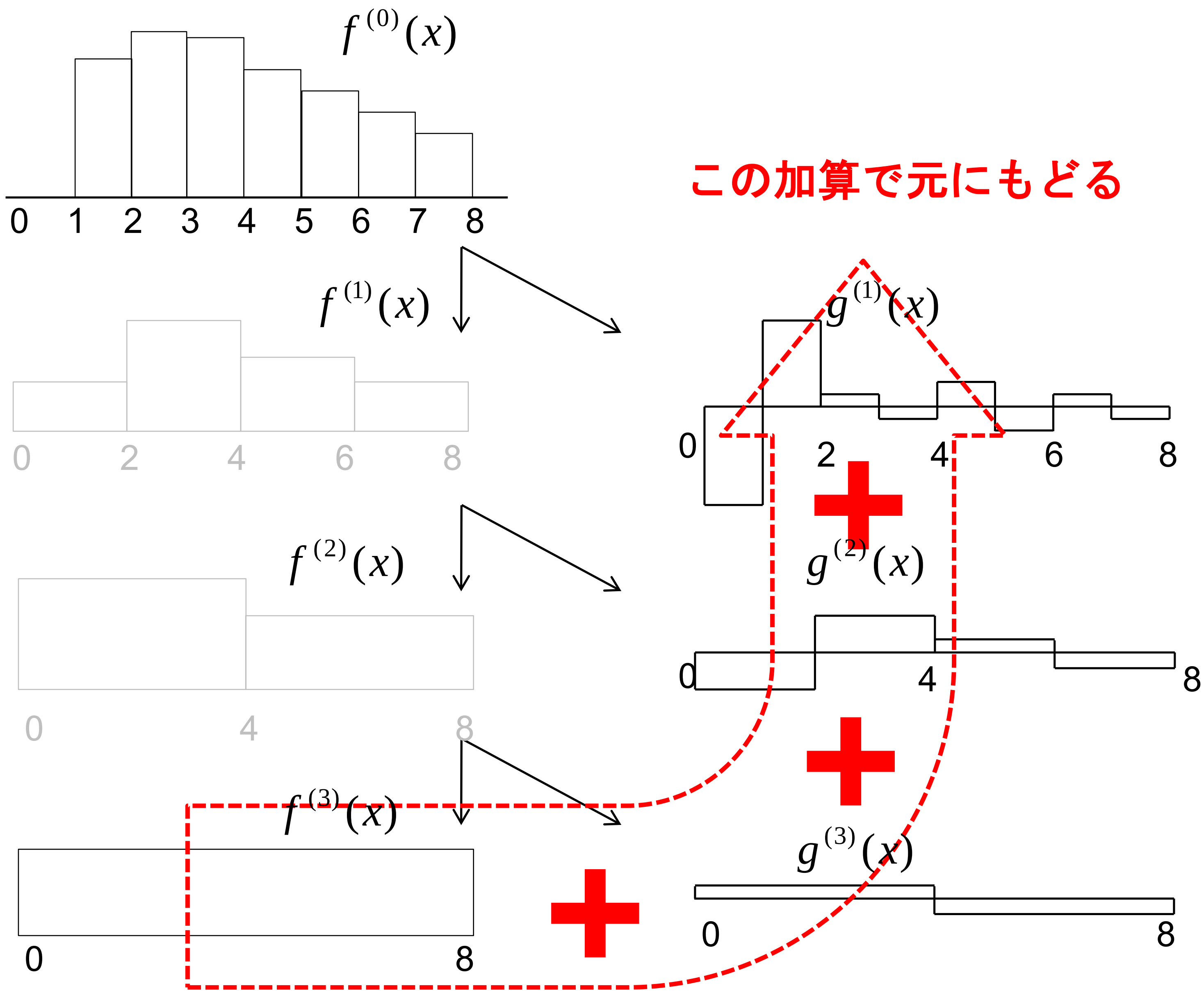
フーリエ解析との違い
フーリエ解析：各成分は正弦波であり,
振幅が場所によらず一定

多重解像度分解：各成分の振幅が場所ご
とに変化

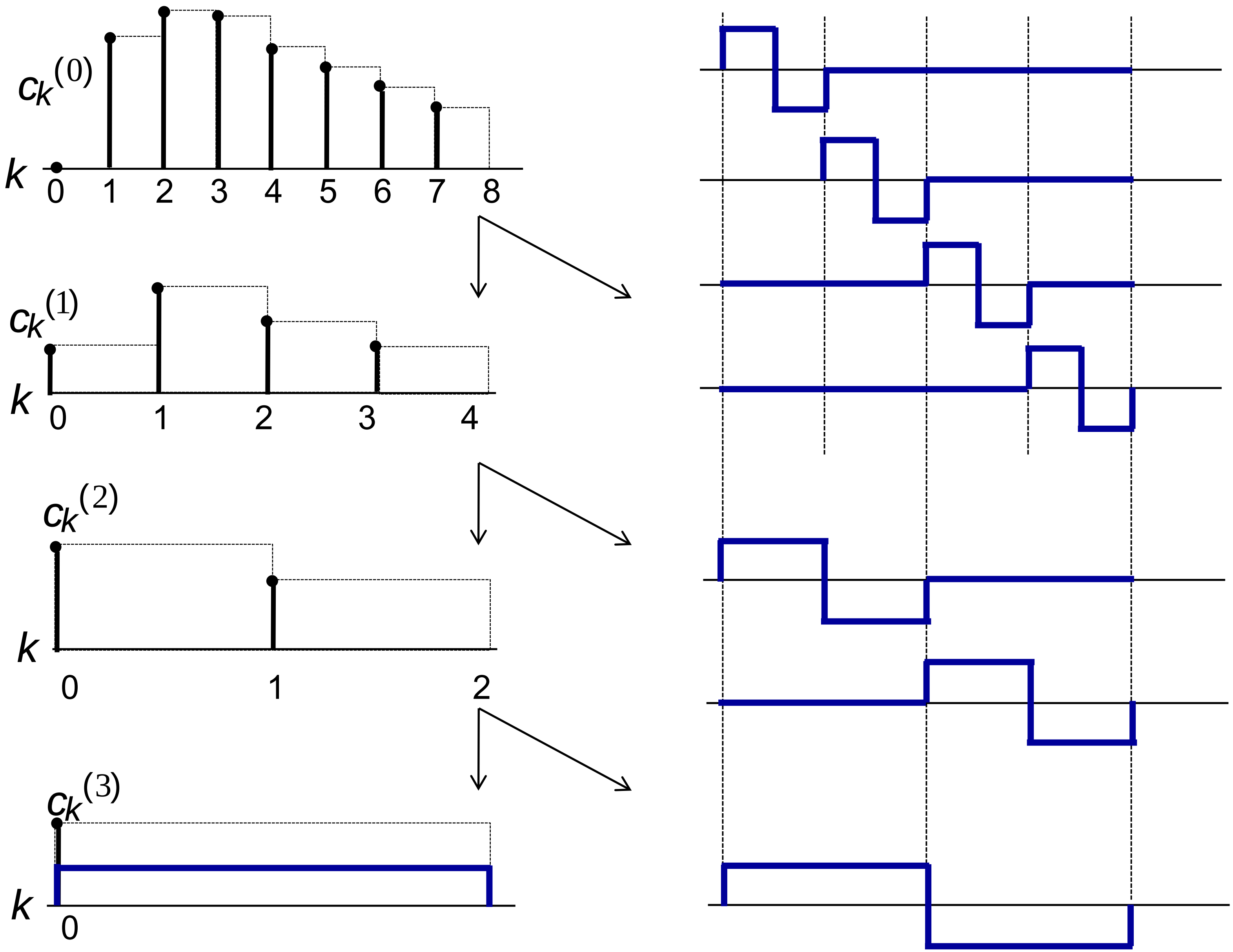
関数f$^{(0)}$(x)の多重解像度分解



関数f$^{(0)}$(x)の多重解像度分解

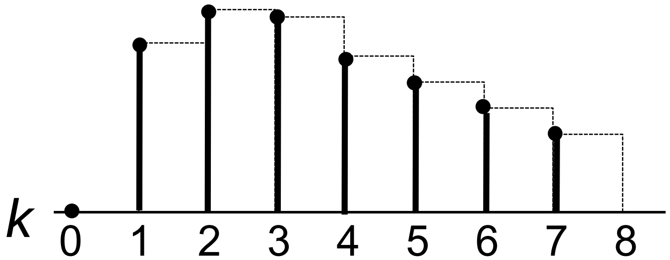


データ $c_k^{(0)}$, k=1, ···, 7の多重解像度分解

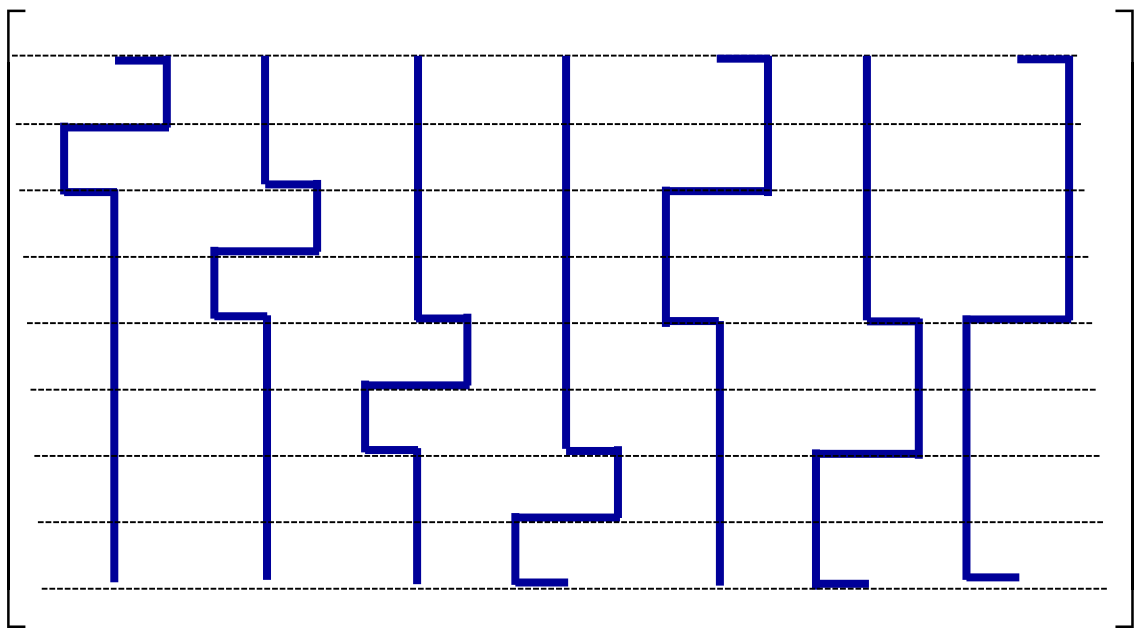


これらを基底関数として，重みをつけて合成したものと考えることができる.

ウェーブレット変換（展開）の行列表現



$$\begin{bmatrix} f^{(0)}(0) \\ f^{(0)}(1) \\ f^{(0)}(2) \\ f^{(0)}(3) \\ f^{(0)}(4) \\ f^{(0)}(5) \\ f^{(0)}(6) \\ f^{(0)}(7) \end{bmatrix} =$$



$$\begin{bmatrix} d_0^{(1)} \\ d_1^{(1)} \\ d_2^{(1)} \\ d_3^{(1)} \\ d_0^{(2)} \\ d_1^{(2)} \\ d_0^{(3)} \\ c_0^{(3)} \end{bmatrix}$$

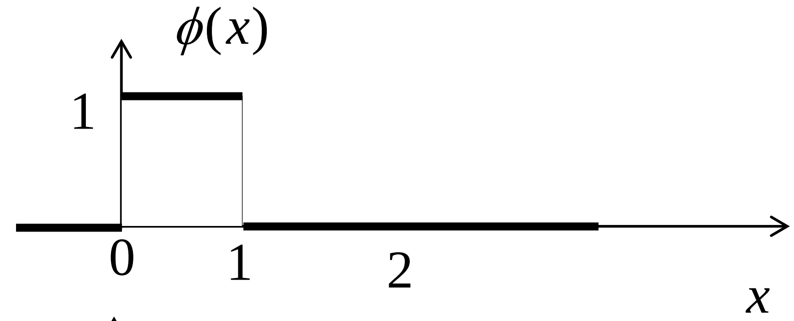
各列はウェーブレットの基底関数（基底ベクトル）
であり、互いに直交している。
その波形には局所性があることがわかる。

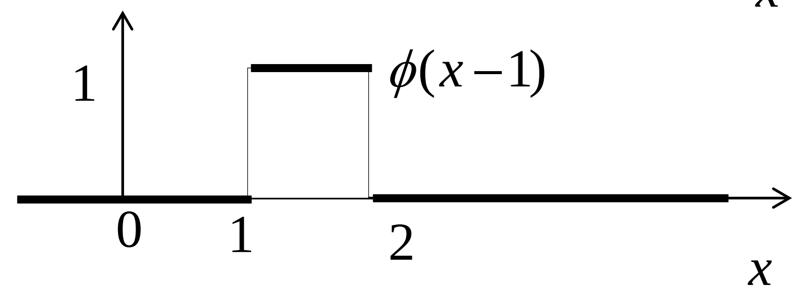
$f^{(n)}(x)$ のスケーリング関数による表現

スケーリング関数の定義

$$\phi(x)=\begin{cases}1 & 0\le x<1\\ 0 & \text{その他}\end{cases}$$

台：値が０でない区間（support）



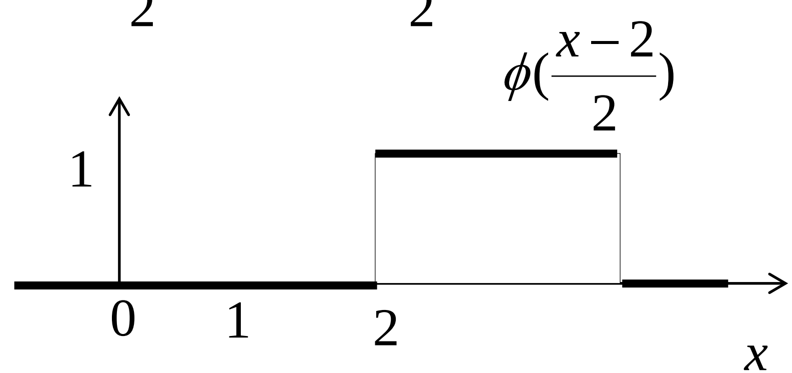


$$f^{(0)}(x)=\sum_{k=0}^{N-1}c_k^{(0)}\phi(x-k),\quad c_k^{(0)}=c_k$$

$$f^{(1)}(x)=\sum_{k=0}^{N/2-1}c_k^{(1)}\phi(\frac{x}{2}-k),\quad c_k^{(1)}=\frac{c_{2k}^{(0)}+c_{2k+1}^{(0)}}{2}$$

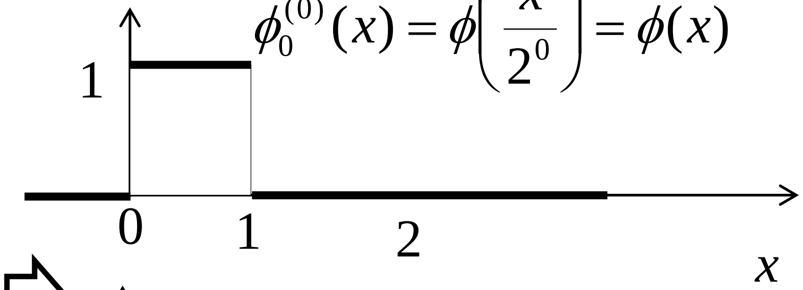
$$\Leftarrow\quad \phi(\frac{x}{2}-k)=\phi(\frac{x-2k}{2})$$

$$f^{(2)}(x)=\sum_{k=0}^{N/4-1}c_k^{(2)}\phi(\frac{x}{4}-k),\quad c_k^{(2)}=\frac{c_{2k}^{(1)}+c_{2k+1}^{(1)}}{2}$$



スケーリング関数の直交性の確認

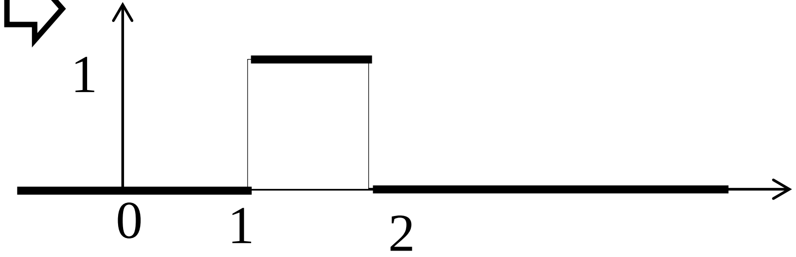
$$\phi_0^{(0)}(x)=\phi\left(\frac{x}{2^0}\right)=\phi(x)$$



区間[0, N)上の関数p(x), q(x)の内積を

$$(p,q)=\int_0^N p(x)q(x)dx$$

内積は0



$$\phi_1^{(0)}(x)=\phi\left(\frac{x}{2^0}-1\right)=\phi(x-1)$$

と定義し,

$$\phi_k^{(j)}(x)=\phi\left(\frac{x}{2^j}-k\right)$$

とおくと

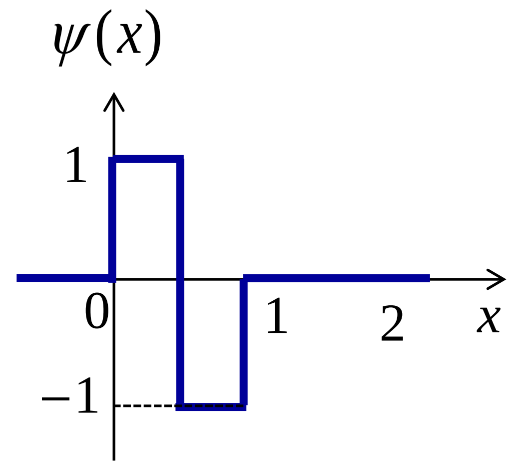
$$(\phi_k^{(j)},\phi_{k'}^{(j)}),\int_0^N \phi_k^{(j)}(x)\phi_{k'}^{(j)}(x)dx=0, \qquad k\neq k'$$

が成り立つ。すなわちϕは直交系となる。

差分データg(x)のウェーブレット母関数による表現

ウェーブレット母関数の定義

$$\psi(x)=\begin{cases}1 & 0\le x<1/2\\ -1 & 1/2\le x<1\\ 0 & \text{その他}\end{cases}$$



$$g^{(1)}(x)=f^{(0)}(x)-f^{(1)}(x)$$

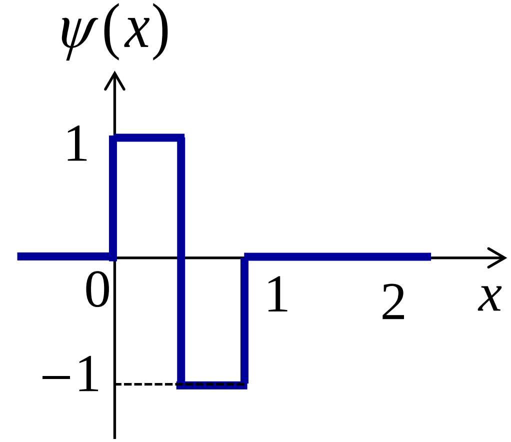
$$=\begin{cases}c_0^{(0)}-(c_0^{(0)}+c_1^{(0)})/2 & 0\le x<1\\ c_1^{(0)}-(c_0^{(0)}+c_1^{(0)})/2 & 1\le x<2\\ c_2^{(0)}-(c_2^{(0)}+c_3^{(0)})/2 & 2\le x<3\\ c_4^{(0)}-(c_2^{(0)}+c_3^{(0)})/2 & 3\le x<4\\ \vdots & \vdots\\ c_{N-2}^{(0)}-(c_{N-2}^{(0)}+c_{N-1}^{(0)})/2 & N-2\le x<N-1\\ c_{N-2}^{(0)}-(c_{N-2}^{(0)}+c_{N-1}^{(0)})/2 & N-1\le x<N\end{cases}$$

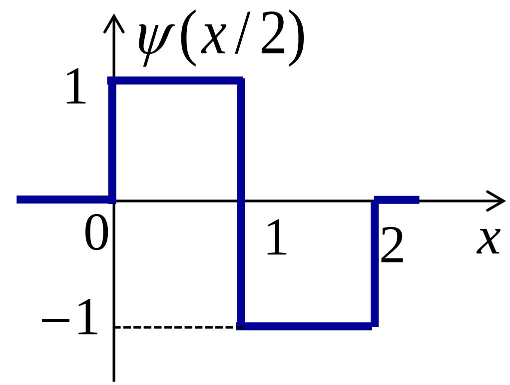
$$
g^{(1)}(x) = f^{(0)}(x) - f^{(1)}(x)
$$

$$
= \begin{cases}
c_0^{(0)} - (c_0^{(0)} + c_1^{(0)})/2 & 0 \le x < 1 \\
c_1^{(0)} - (c_0^{(0)} + c_1^{(0)})/2 & 1 \le x < 2 \\
c_2^{(0)} - (c_2^{(0)} + c_3^{(0)})/2 & 2 \le x < 3 \\
c_4^{(0)} - (c_2^{(0)} + c_3^{(0)})/2 & 3 \le x < 4 \\
\vdots & \vdots \\
c_{N-2}^{(0)} - (c_{N-2}^{(0)} + c_{N-1}^{(0)})/2 & N-2 \le x < N-1 \\
c_{N-2}^{(0)} - (c_{N-2}^{(0)} + c_{N-1}^{(0)})/2 & N-1 \le x < N
\end{cases}
$$

$$
= \begin{cases}
(c_0^{(0)} - c_1^{(0)})/2 & 0 \le x < 1 \\
-(c_0^{(0)} - c_1^{(0)})/2 & 1 \le x < 2 \\
(c_2^{(0)} - c_3^{(0)})/2 & 2 \le x < 3 \\
-(c_2^{(0)} - c_3^{(0)})/2 & 3 \le x < 4 \\
\vdots & \vdots \\
(c_{N-2}^{(0)} - c_{N-1}^{(0)})/2 & N-2 \le x < N-1 \\
-(c_{N-2}^{(0)} - c_{N-1}^{(0)})/2 & N-1 \le x < N
\end{cases}
$$

$$
= \sum_{k=0}^{N/2-1} d_k^{(1)} \psi(x/2 - k)
$$





左において

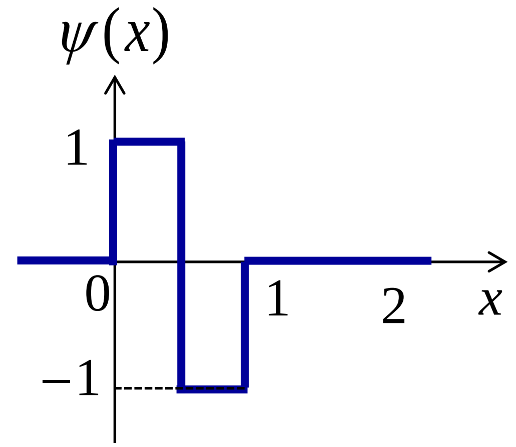
$$
d_k^{(1)} = \frac{c_{2k}^{(0)} - c_{2k+1}^{(0)}}{2}
$$

と置いた。

$$
\psi(x/2 - k) = \psi((x-2k)/2)
$$

はΨの台を区間[0, 2)に広げて
2kだけ平行移動したもの。

$$
\begin{aligned}
g^{(2)}(x) &= f^{(1)}(x) - f^{(2)}(x) \\
&= \begin{cases}
c_0^{(1)} - (c_0^{(1)} + c_1^{(1)})/2 & 0 \le x < 2 \\
c_1^{(1)} - (c_0^{(1)} + c_1^{(1)})/2 & 2 \le x < 4 \\
\vdots & \vdots \\
c_{N-2}^{(1)} - (c_{N-2}^{(1)} + c_{N-1}^{(1)})/2 & N-4 \le x < N-2 \\
c_{N-1}^{(1)} - (c_{N-2}^{(1)} + c_{N-1}^{(1)})/2 & N-2 \le x < N
\end{cases} \\
&= \sum_{k=0}^{N/4-1} d_k^{(2)} \psi(x/4 - k)
\end{aligned}
$$



ただし

$$
d_k^{(2)} = \frac{c_{2k}^{(1)} - c_{2k+1}^{(1)}}{2}
$$

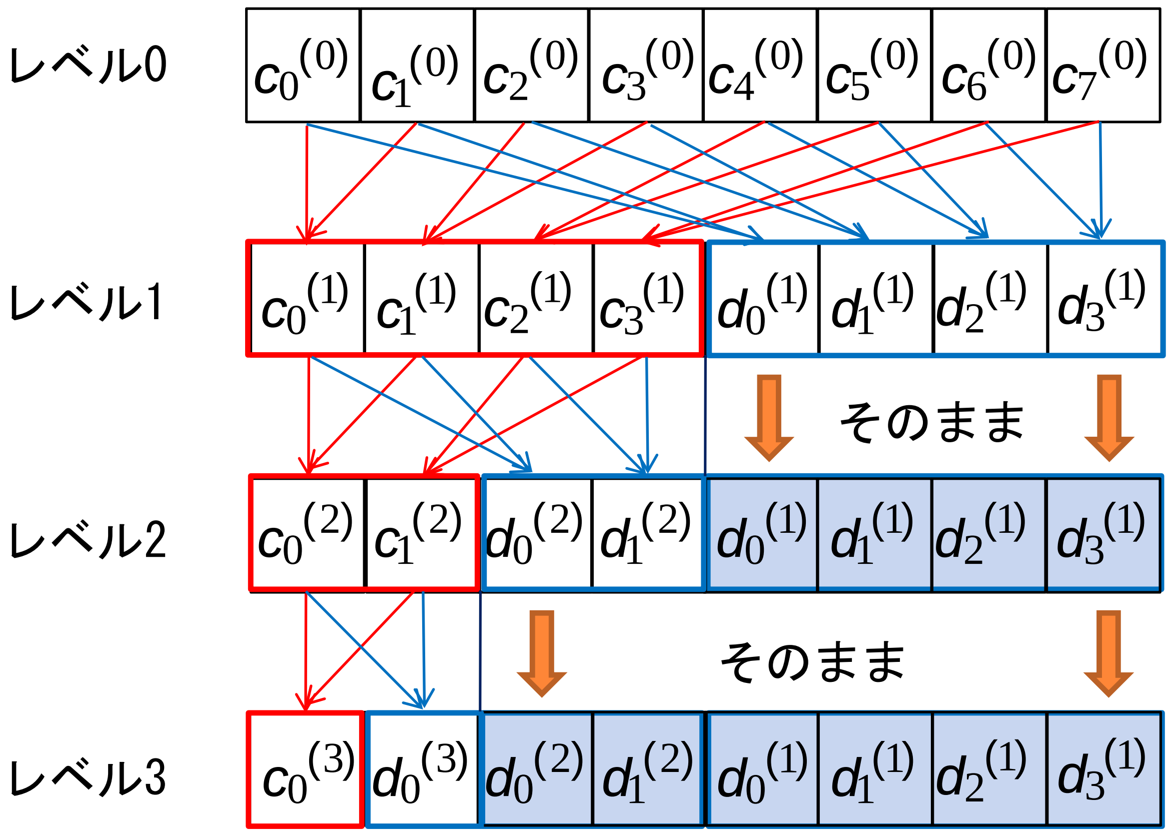
一般にスケール２^{j}の関数$g^{(j)}(x)$は次のように書ける。

$$
d_k^{(j)} = \frac{c_{2k}^{(j-1)} - c_{2k+1}^{(j-1)}}{2}
$$

レベルｊ、場所ｋのウェーブレットを次のように定義する。

$$
\psi_k^{(j)}(x) = \psi\left(\frac{x}{2^j} - k\right)
$$

係数の計算過程と配列への保存　１次元



ウェーブレット変換

スケーリング関数とウェーブレットの間には直交性がある。

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

すべてレベルと場所のウェーブレットの間には直交性がある。

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

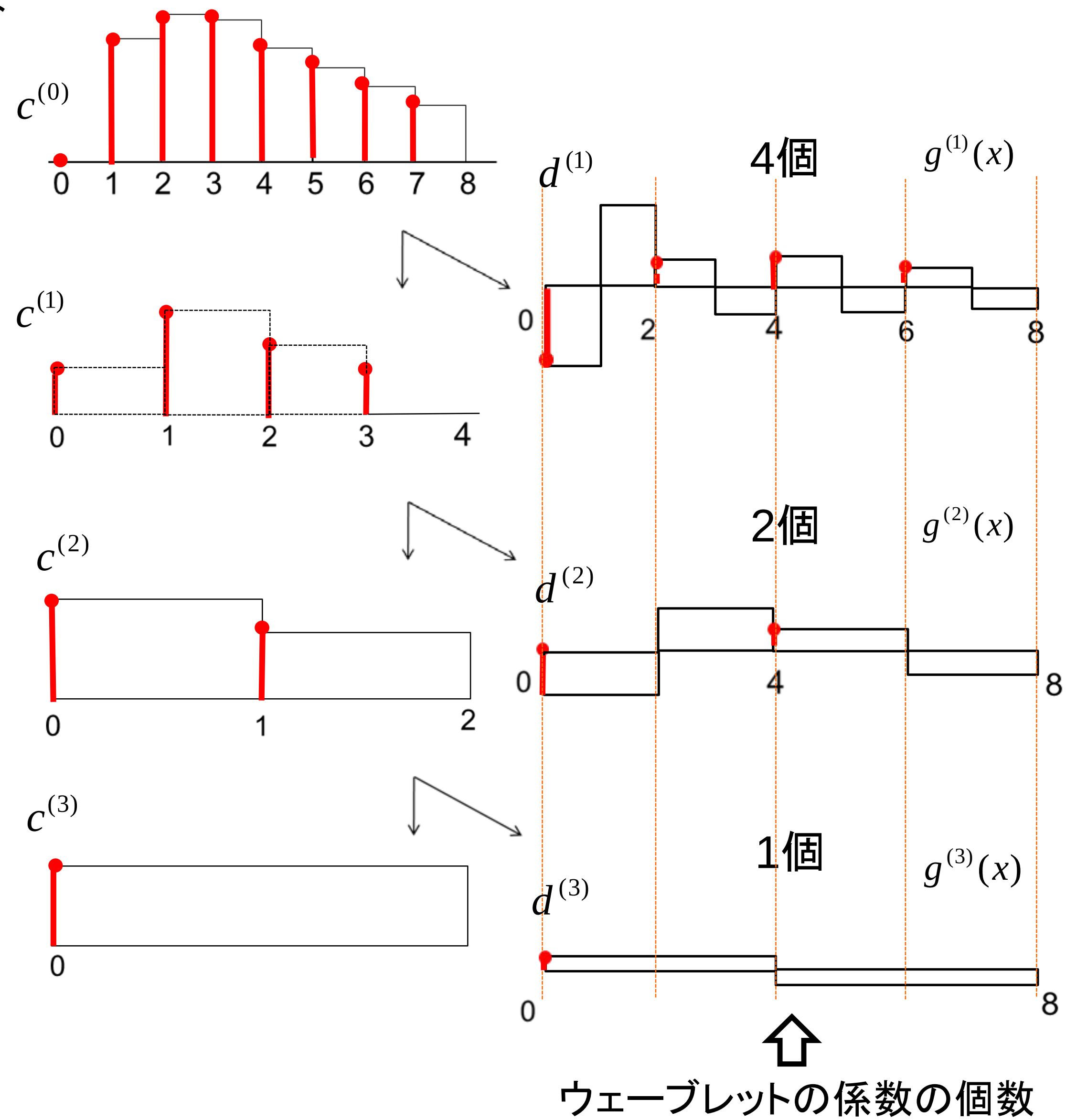
元のデータをすべてのウェーブレットと1つのスケーリング関数で表す場合、ウェーブレットの係数の個数は以下のようになる。

$$\sum_{j=1}^{n} \frac{N}{2^j} = \frac{N}{2} + \frac{N}{4} + \cdots + \frac{N}{2^n}$$

$$= 2^{n-1} + 2^{n-2} + \cdots + 1$$

$$= 2^n - 1$$

$$= N - 1$$



ウェーブレット変換　つづき

ウェーブレット変換

$$f(x)=\sum_{j=1}^{n}\sum_{k=0}^{N/2^j-1} d_k^{(j)}\psi_k^{(j)}(x)+c_0^{(n)}$$

すべてのウェーブレットの展開係数であらわされること。

ただし

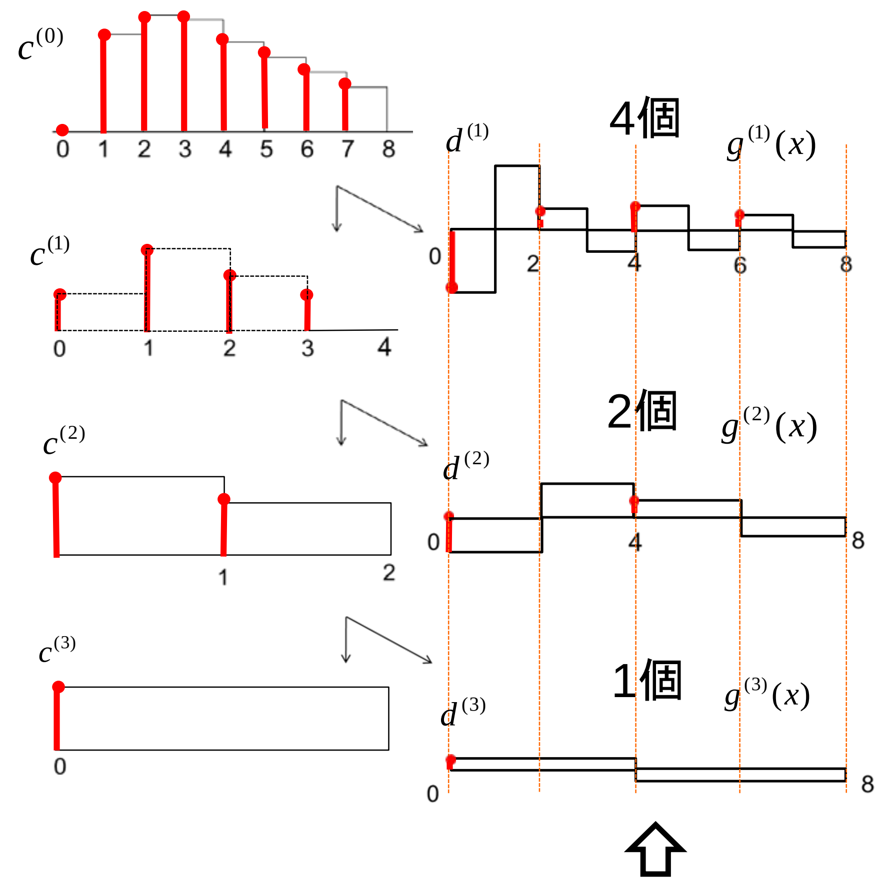
$$\left\|\psi_k^{(j)}\right\|^2=\int_0^N \psi_k^{(j)}(x)^2 dx=\int_0^{2^j} dx=2^j$$

$$\left\|\phi_0^{(n)}\right\|^2=\int_0^N \phi_0^{(n)}(x)^2 dx=\int_0^{N} dx=N$$

逆に展開係数は以下のように計算される。

$$d_k^{(j)}=\frac{1}{2^j}\int_0^N f(x)\psi_k^{(j)}(x)dx$$

$$c_0^{(n)}=\frac{1}{N}\int_0^N f(x)dx$$



ウェーブレットの係数の個数

下降サンプリングと上昇サンプリング

下降サンプリング

$$c_k^{(1)} = \frac{c_{2k}^{(0)} + c_{2k+1}^{(0)}}{2}, d_k^{(1)} = \frac{c_{2k}^{(0)} - c_{2k+1}^{(0)}}{2}, k = 0, \cdots, \frac{N}{2} - 1$$

⇩

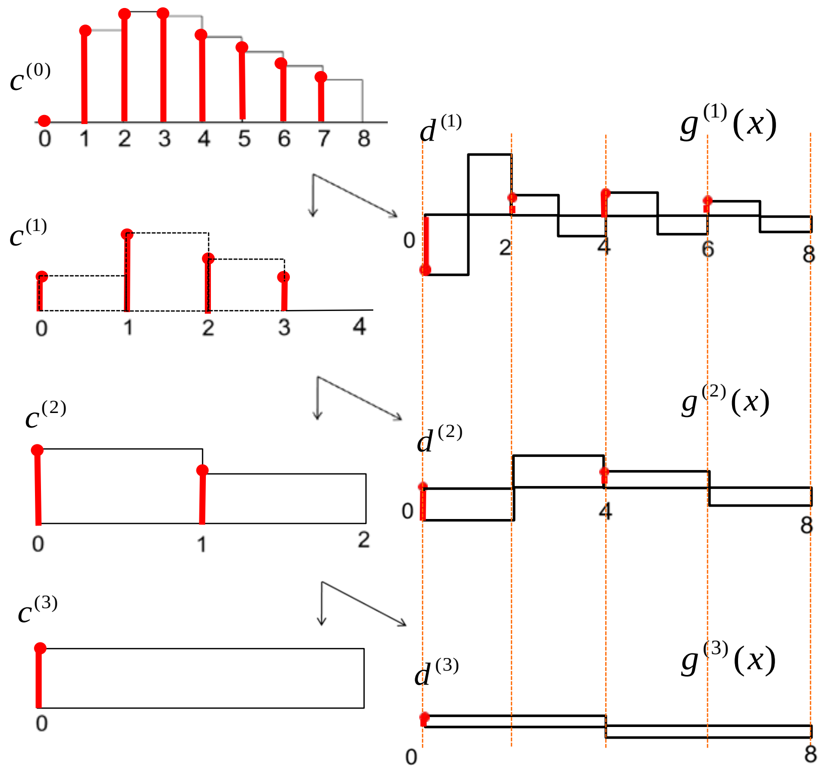
$$c_k^{(2)} = \frac{c_{2k}^{(1)} + c_{2k+1}^{(1)}}{2}, d_k^{(2)} = \frac{c_{2k}^{(1)} - c_{2k+1}^{(1)}}{2}, k = 0, \cdots, \frac{N}{4} - 1$$

⇩

$$c_k^{(3)} = \frac{c_{2k}^{(2)} + c_{2k+1}^{(2)}}{2}, d_k^{(3)} = \frac{c_{2k}^{(2)} - c_{2k+1}^{(2)}}{2}, k = 0, \cdots, \frac{N}{8} - 1$$

⇩

$$c_k^{(n)} = \frac{c_{2k}^{(n-1)} + c_{2k+1}^{(n-1)}}{2}, d_k^{(n)} = \frac{c_{2k}^{(n-1)} - c_{2k+1}^{(n-1)}}{2}, k = 0$$



演算回数

$$c_k^{(j)}\text{の数}: \sum_{j=1}^{n} \frac{N}{2^j} = \frac{N}{2} + \frac{N}{4} + \cdots + \frac{N}{2^n} = 2^{n-1} + 2^{n-2} + \cdots + 1 = 2^n - 1 = N - 1$$

$$d_k^{(j)}\text{の数}: N - 1$$

1つの値の計算に加算数が1回、除算が1回要るので、演算は合計4(N-1)回

下降サンプリングと上昇サンプリング

上昇サンプリング

$$c_{2k}^{(0)}=c_k^{(1)}+d_k^{(1)},\ c_{2k+1}^{(0)}=c_k^{(1)}-d_k^{(1)},\ k=0,1,2,\ldots\frac{N}{2}-1$$

⇧

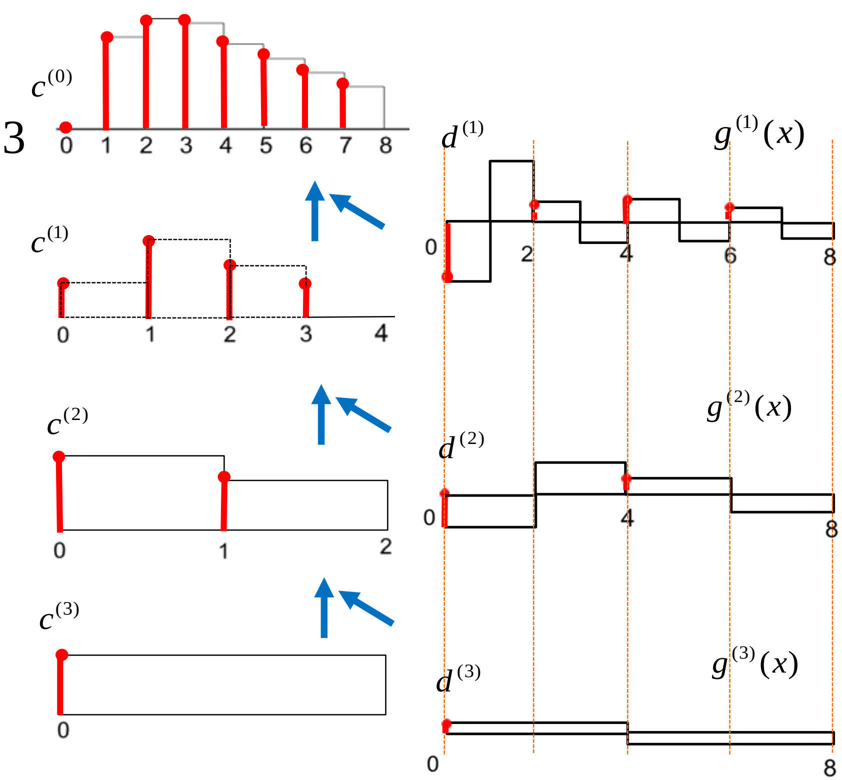
$$c_{2k}^{(n-3)}=c_k^{(n-2)}+d_k^{(n-2)},\ c_{2k+1}^{(n-3)}=c_k^{(n-2)}-d_k^{(n-2)},\ k=0,1,2,3$$

⇧

$$c_{2k}^{(n-2)}=c_k^{(n-1)}+d_k^{(n-1)},\ c_{2k+1}^{(n-2)}=c_k^{(n-1)}-d_k^{(n-1)},\ k=0,1$$

⇧

$$c_{2k}^{(n-1)}=c_k^{(n)}+d_k^{(n)},\ c_{2k+1}^{(n-1)}=c_k^{(n)}-d_k^{(n)},\ k=0$$



演算回数

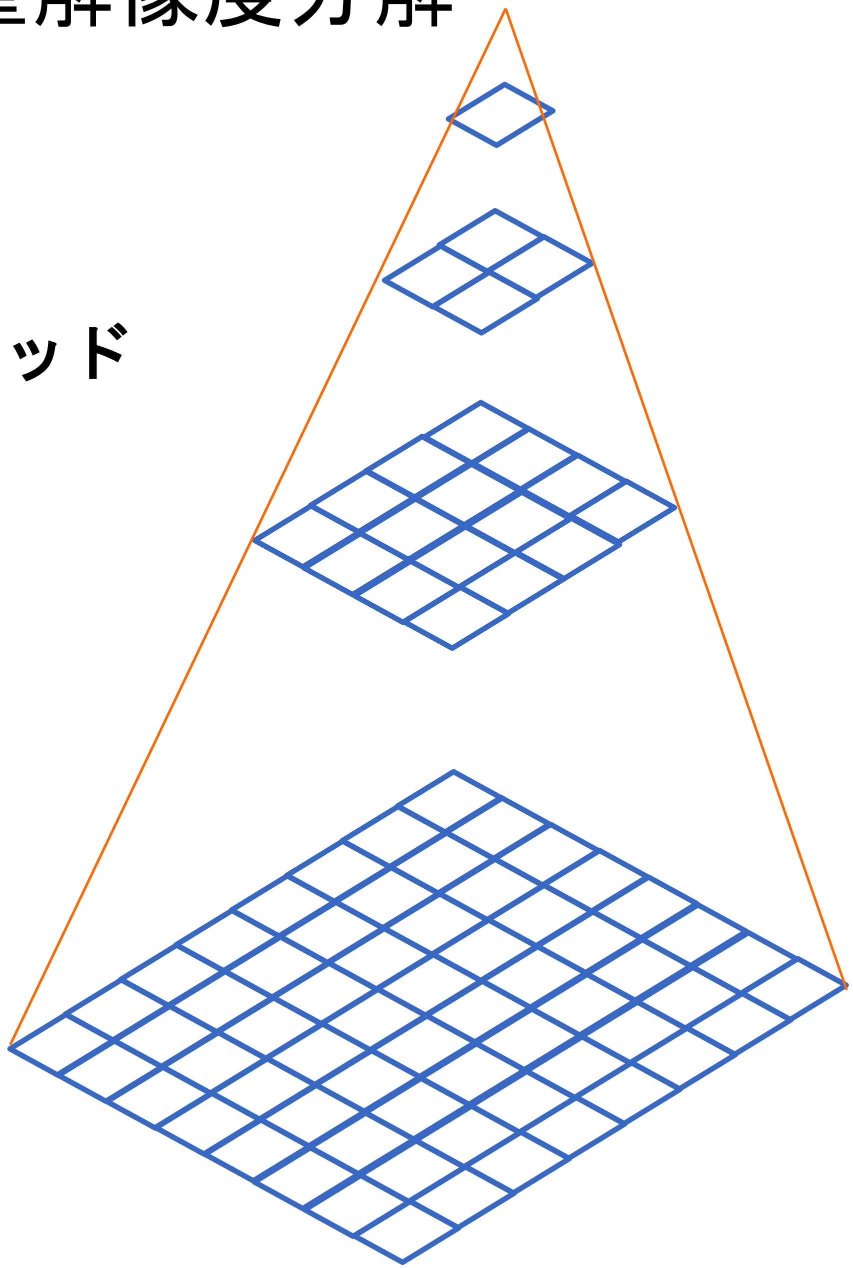
$c_k^{(j)}$の数$:2(N-1)$

1つの値の計算に加算数が1回要るので、演算は合計2(N-1)回

２次元画像の多重解像度分解

階層的近似：逐次的にさまざまな解像度の信号を作り出すこと.

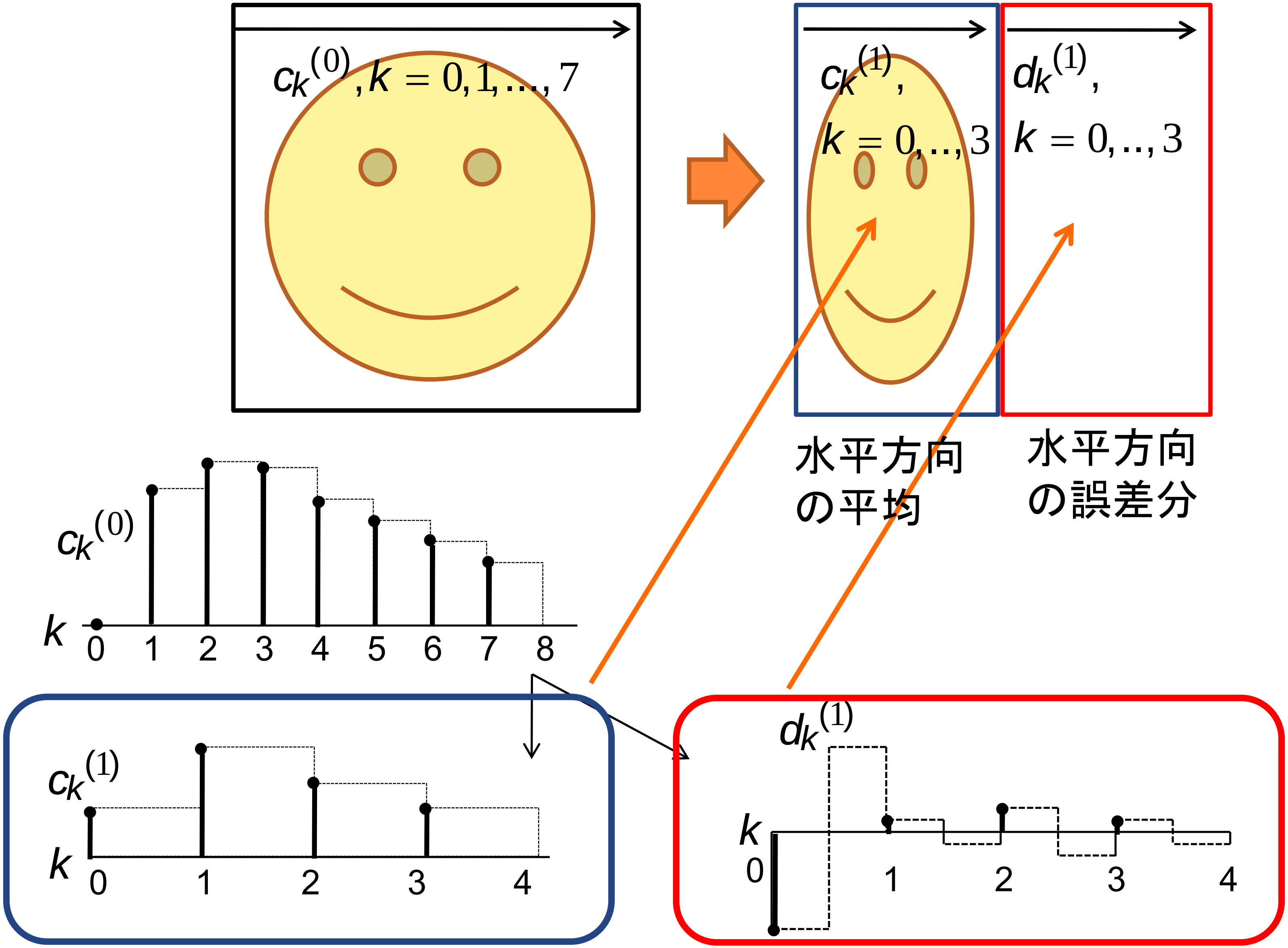
画像の場合⇒**ピラミッド**



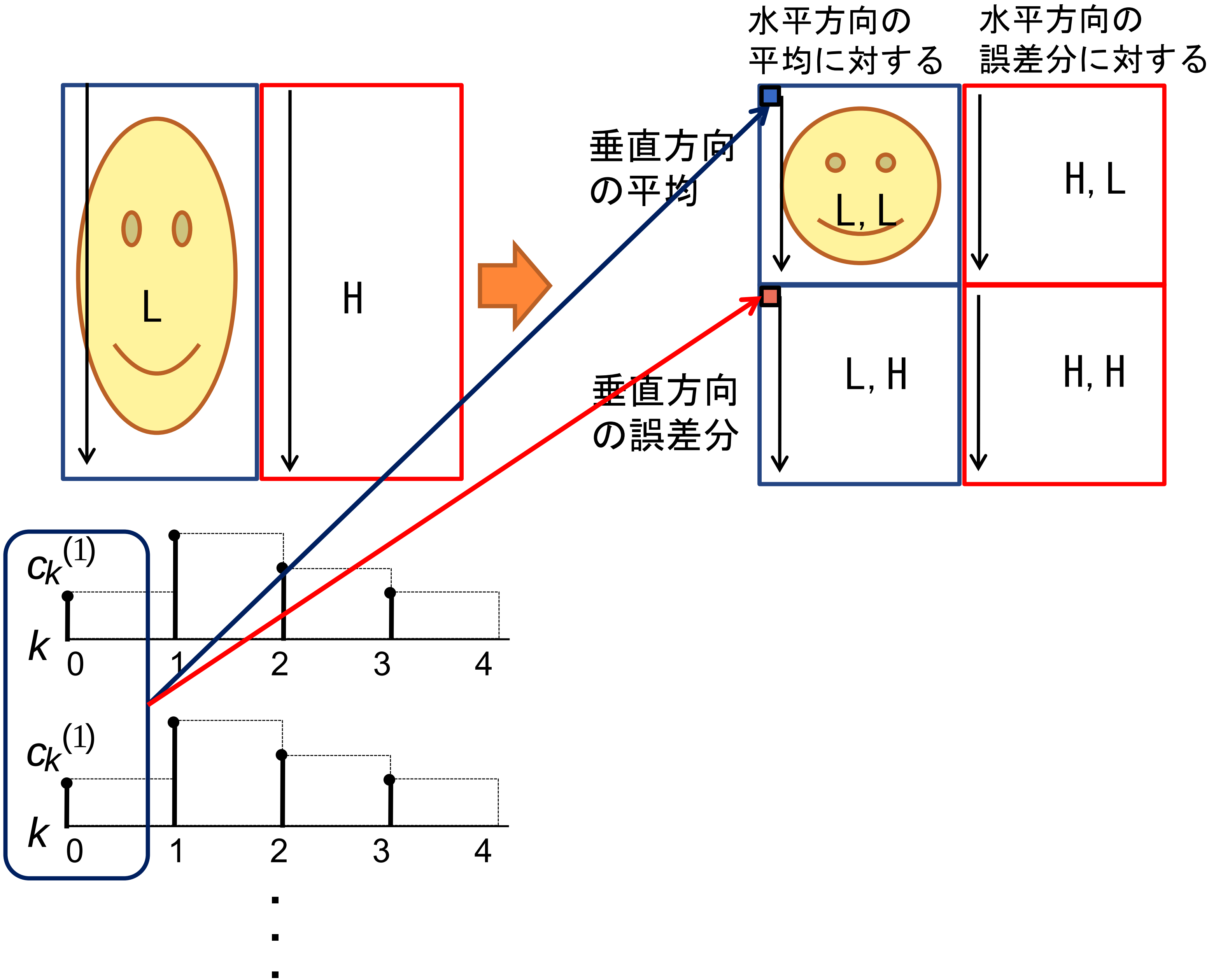
２次元ウェーブレット変換

２次元画像の多重解像度分解

画像の横のラインごとに，１段階の多重解像度分解を実施

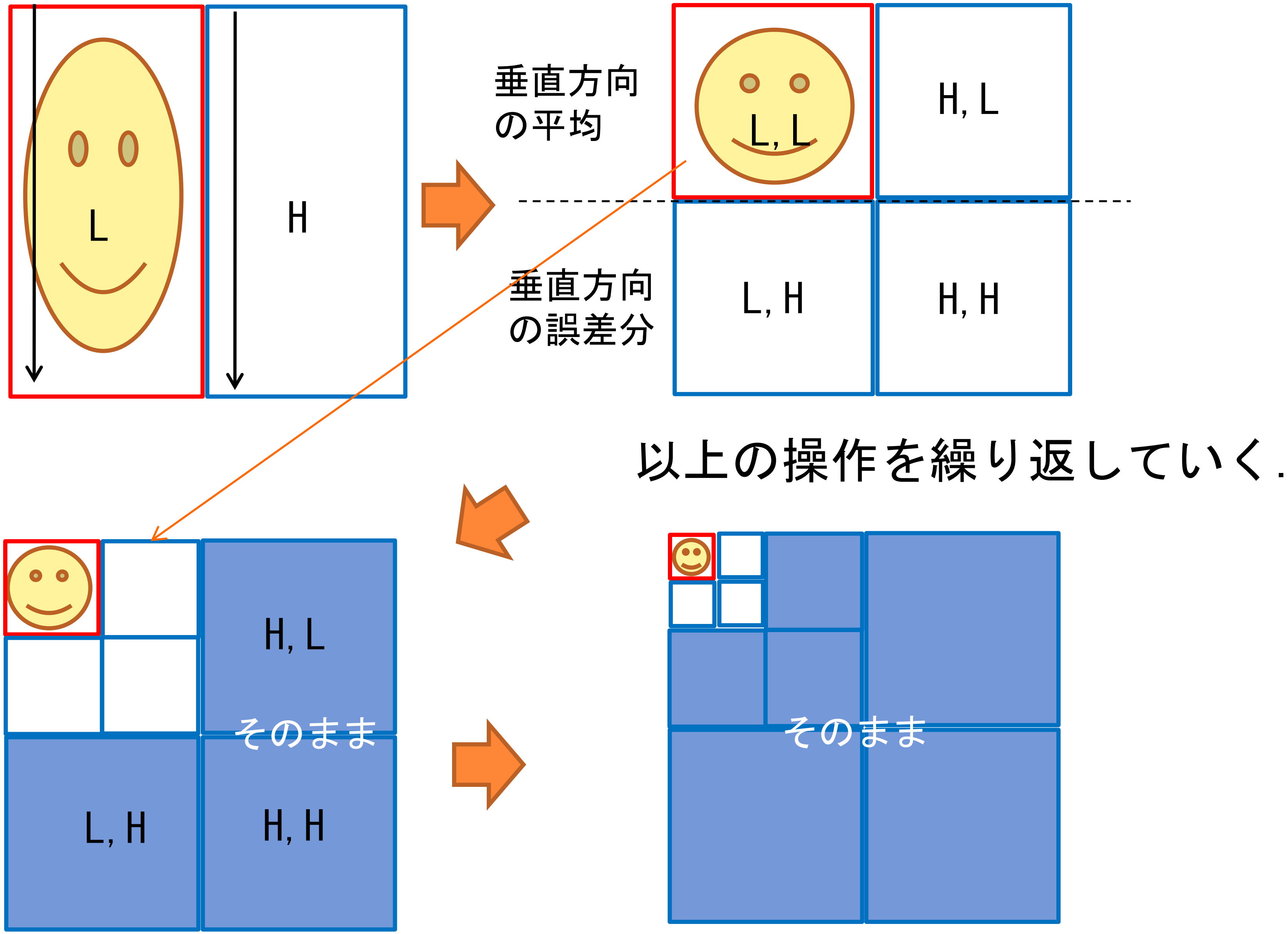


２次元画像の多重解像度分解
画像の縦のラインごとに，１段階の多重解像度分解を実施



２次元画像の多重解像度分解（つづき）

画像の縦のラインごとに，１段階の多重解像度分解を実施



ウェーブレット変換の応用

ウェーブレット縮退によるノイズの低減

一般的な特徴

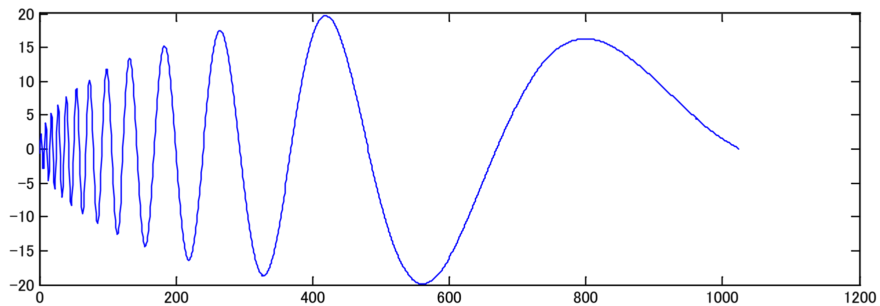
１．信号の特徴は，少数のウェーブレット展開係数で表現可能である.

２．白色ノイズは，すべての展開係数に影響を及ぼす.

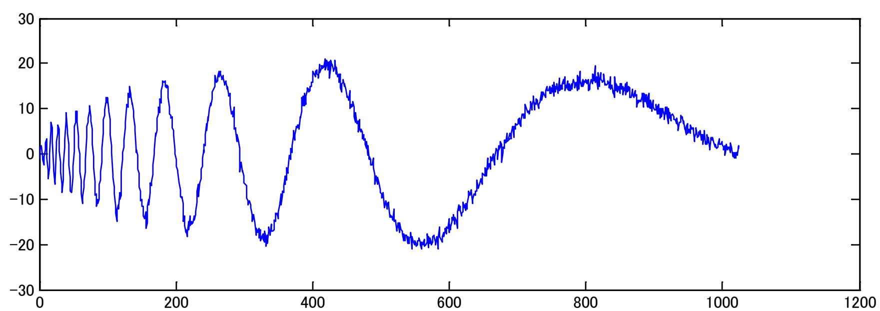
処理法

　絶対値の小さい展開係数をノイズとみなし，絶対値があるしきい値より小さい展開係数を０にして信号の再構成を行うと，信号の特徴を保持したまま，ノイズ成分が減少する.

原信号

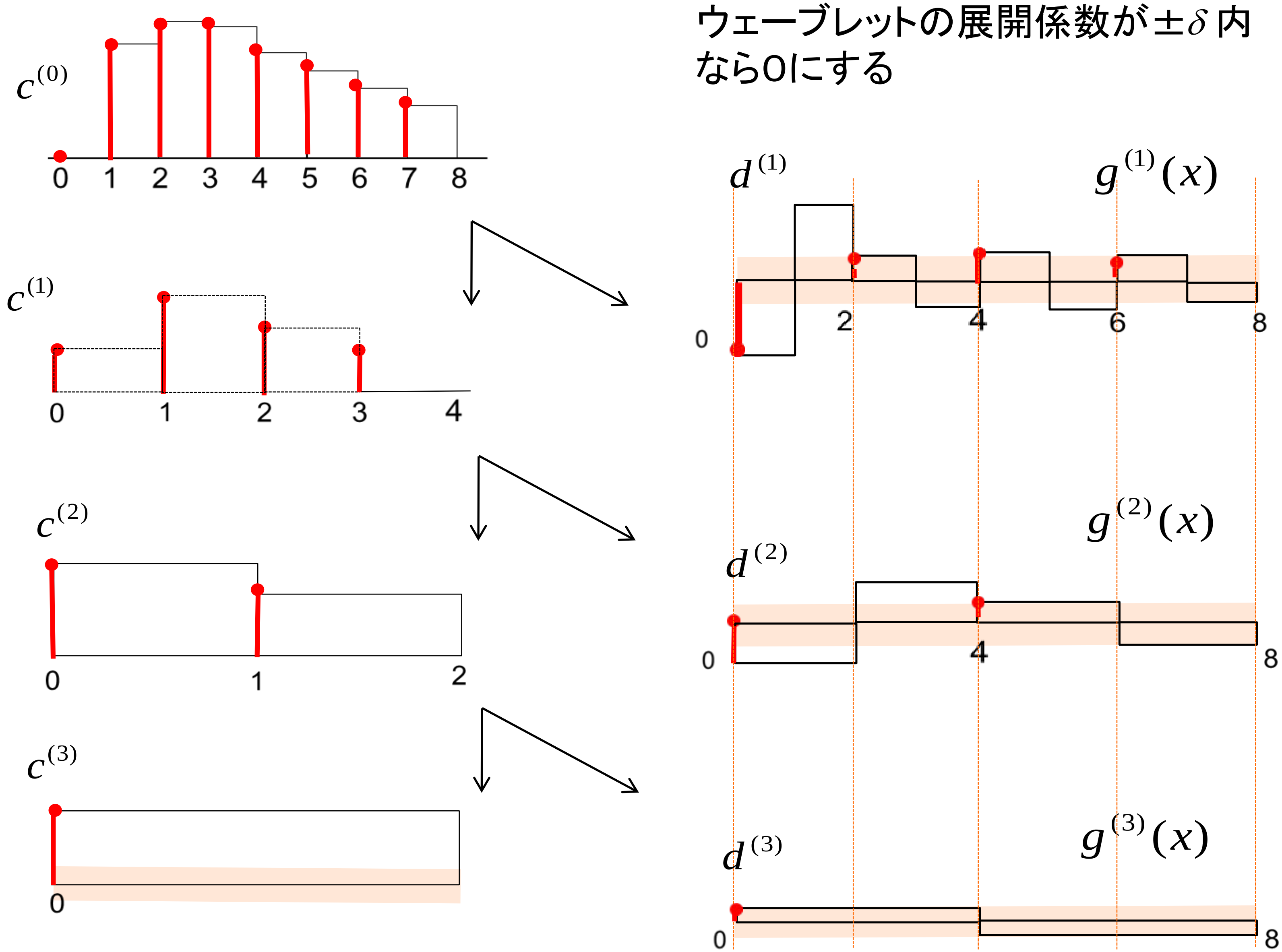


ノイズ加算



ウェーブレット縮退によるノイズの低減

ウェーブレットの展開係数が±δ内ならOにする



実験：フーリエ変換との比較

フーリエ変換

1次元信号をフーリエ変換し、絶対値で閾値以下のフーリエ係数を強制的に0に置き換える。その後、フーリエ逆変換をする。

シミュレーションでは、3段階の閾値でノイズ除去を実施。

ウェーブレット変換

1次元信号をウェーブレット変換し、絶対値で閾値以下のウェーブレット係数を強制的に0に置き換える。その後、ウェーブ逆変換をする。
閾値の設定方法として以下が提案されている。

シミュレーションでは、3段階の閾値でノイズ除去を実施。

$T = \alpha \cdot \sigma\sqrt{2\ln n}$　nはデータ数、σはノイズの標準偏差 （Donoho, 1995)

α = 0.5,　1,　1.5

デモソフト Denoise_Fourier.m

```
n = 1024;    % 信号のサンプル数 n = 1024
L = 10;      % 信号の最大展開レベル  n = 2^L
J = 9;       % J レベルまで展開  (J =< L)
threshold =0.02;

t = (1:n)./n;
sig = 40.0*sqrt(t.*(1-t)).*sin((2.2*pi)./(t+.08));
                    % ドップラー信号の生成
noise = randn(1,n);  % 白色ノイズの生成
sdv = std(noise)     % ノイズの標準偏差
s0 = sig+noise;      % 信号にノイズを加える

S0 = fft(s0);
S0a = abs(S0)/max(abs(S0));
S0 = S0 .* (abs(S0a) >= threshold);
s1 = real(ifft(S0));
subplot(2,1,1);
plot(s0);
subplot(2,1,2);
```

左から続く

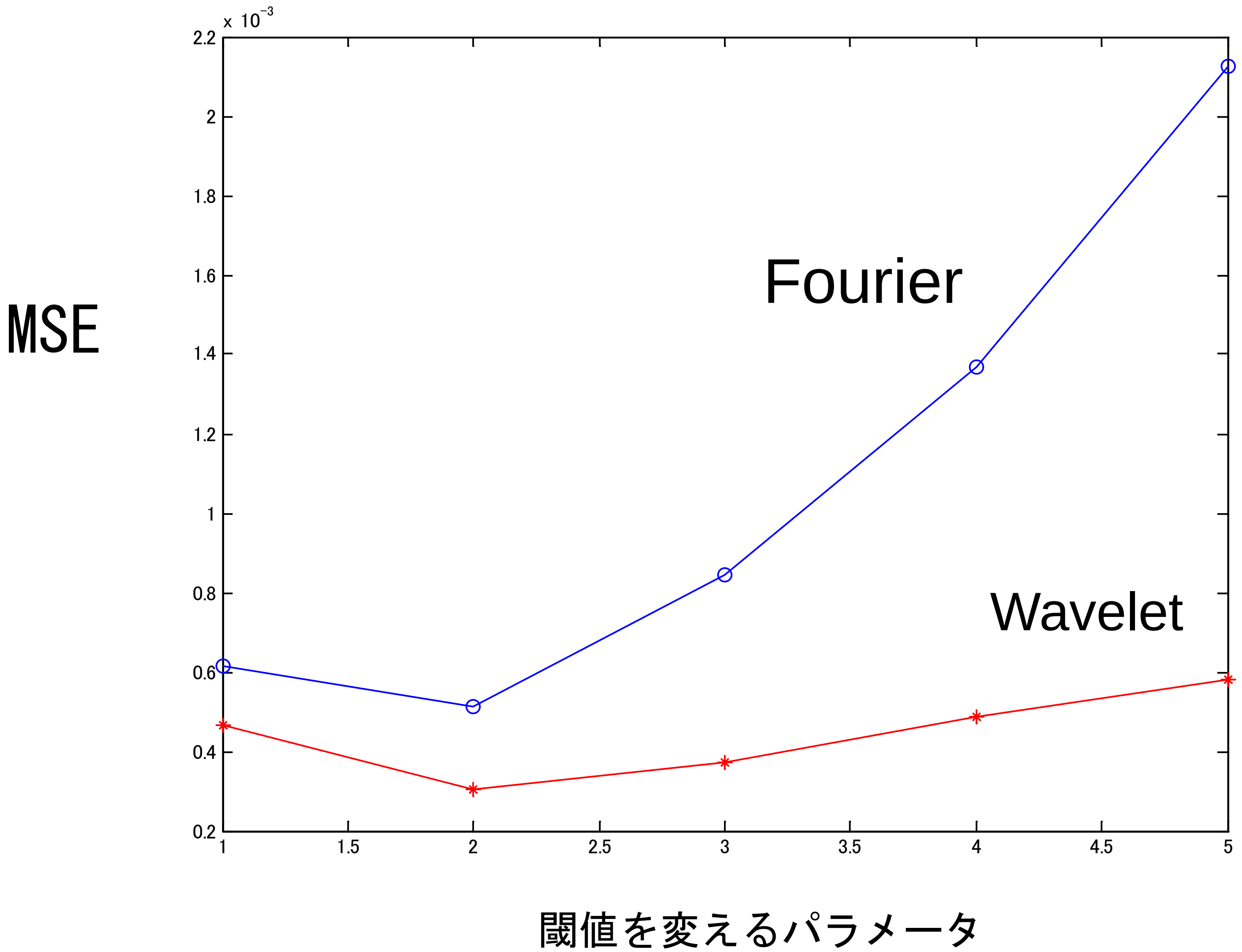
```
figure
plot(abs(S0)); % スペクトルを表示

format long
mse = sum((s1(:)-sig(:)).^2)/(n*n)
format short
```

デモソフト Denoise.m

```
n = 1024;    % 信号のサンプル数 n = 1024
L = 10;      % 信号の最大展開レベル  n = 2^L
J = 9;       % J レベルまで展開  (J =< L)

t = (1:n)./n;
sig = 40.0*sqrt(t.*(1-t)).*sin((2.2*pi)./(t+.08));
                    % ドップラー信号の生成
noise = randn(1,n);  % 白色ノイズの生成
sdv = std(noise)     % ノイズの標準偏差
s0 = sig+noise;      % 信号にノイズを加える

p = [0.482962913145 0.836516303738 ...
    0.224143868042 -0.129409522551];
                    % ドベシィの数列 p_k (N=2)

sup = length(p);     % 数列の長さ
q = -( (-1).^(1:sup) ).*p(sup:-1:1) ;
                    % ドベシィの数列 q_k の生成
coef = zeros(1,n);   % 展開係数の保持エリア
sk = s0;             % スケーリング係数の初期値設定
```

左から続く

```
% レベル 1 から レベル J までの高速ウェーブレット変換
for j = 1:J
   [sk,wk] = fwt1d(sk,p,q);
     % スケーリング係数の 1次元ウェーブレット変換
   coef((2^(L-j)+1):(2^(L-j+1))) = wk;
     % ウェーブレット展開係数の保持
   coef(1:(2^(L-j))) = sk;   % スケーリング係数の保持
end

threshold = sdv*sqrt(2*log(n))
           % ウェーブレット縮退のためのしきい値
coef = coef .* (abs(coef) >= threshold);
           % ウェーブレット縮退
           % 絶対値がしきい値未満の展開係数を 0 にする

% 再構成
sk = coef(1:(2^(L-J)));  % レベル J  のスケーリング係数
for j = J:-1:1
      wk = coef((2^(L-j)+1):(2^(L-j+1)));
           % レベル j のウェーブレット展開係数
      sk = ifwt1d(sk,wk,p,q);
           % 1次元ウェーブレット逆変換により、
           % レベル j のスケーリング係数を求める
end
```



MSE
x 10^-3
2.2
2
1.8
1.6
1.4
1.2
1
0.8
0.6
0.4
0.2
1
1.5
2
2.5
3
3.5
4
4.5
5
Fourier
Wavelet
閾値を変えるパラメータ

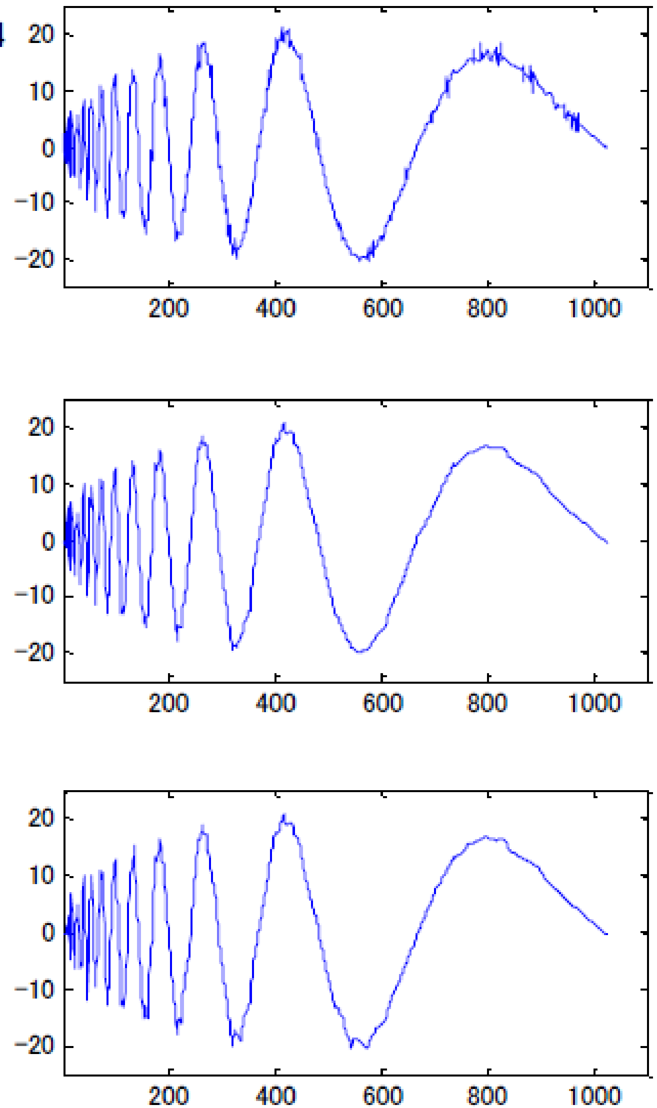
ウェーブレット縮退によるノイズ低減

x 10^{-4}

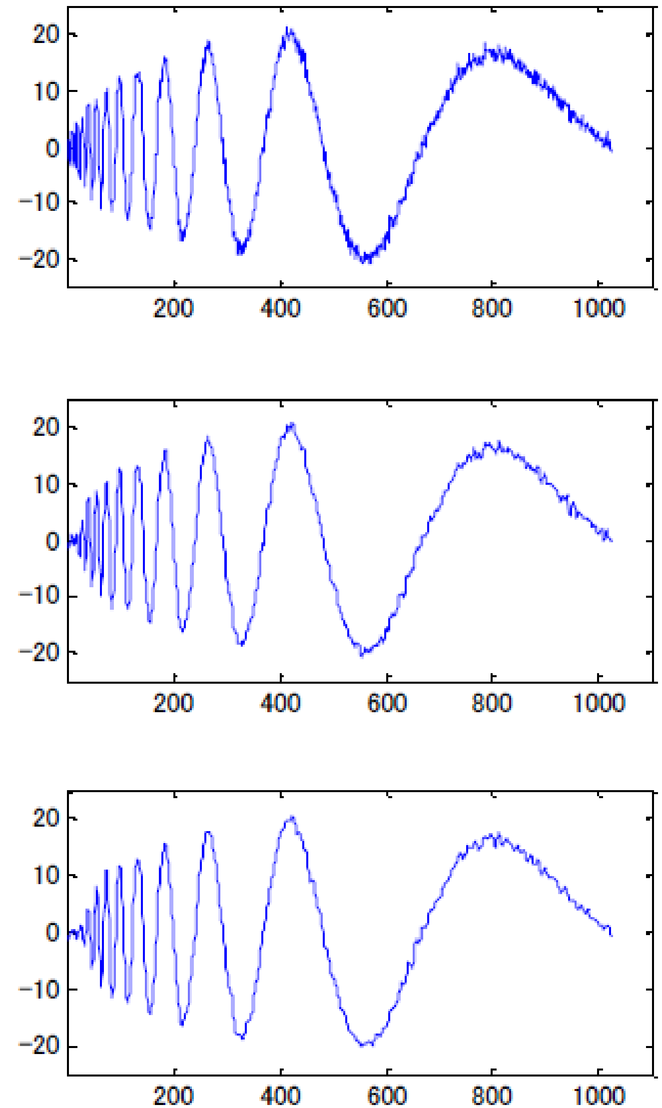
4.8

4.1

6.6



ウェーブレット空間でのノイズ除去



x 10^{-4}

6.2

5.3

8.3

フーリエ空間でのノイズ除去

画像圧縮の例（MATLABソースコード）


```
%「ウェーブレットによる信号処理と画像処理」
%  中野ら著、共立出版  から引用

n = 512;        % 2次元信号の正方配列の一辺の長さ
L = 9;          % 2^L = n
J = L-2;        % J レベルまで展開
rate = 10;      % 上位 (rate)% の展開係数で再構成
quality = 50; % quality set in JPEG
s0 = zeros(n,n); % s0 の初期設定

A = imread('yuki_s.tif','tif');
B = A(1:512, 101:612,2);
s0 = double(B(:,:));

imwrite(B, 'yuki_s.jpg','jpeg', 'Quality', quality);
C  = double(imread('yuki_s.jpg'));

 p = [ 0.482962913145 0.836516303738 ...
       0.224143868042 -0.129409522551];
  % ドベシィのスケーリング関数の数列 p_k (N=2)
sup = length(p);  % 数列の長さ
q = -( (-1).^(1:sup) ).*p(sup:-1:1);
             % ドベシィのウェーブレットの数列 q_k の
生成
coef = zeros(n,n); % 展開係数の保持エリア
sk = s0;        % スケーリング係数の初期値設定
```

左から続く

```
% レベル 1 からレベル J までの展開
for j = 1:J
 [sk,wvk,whk,wdk] = fwt2d(sk,p,q);
   % 2次元高速ウェーブレット変換（1レベル分）
 coef(1:2^(L-j),1:2^(L-j)) = sk;  % スケーリング係数の保持
 coef(1:2^(L-j),2^(L-j)+1:2^(L-j+1)) = wvk;
 coef(2^(L-j)+1:2^(L-j+1),1:2^(L-j)) = whk;
 coef(2^(L-j)+1:2^(L-j+1),2^(L-j)+1:2^(L-j+1)) = wdk;
   % ウェーブレット展開係数の保持
end

ordered = reverse(sort(abs(coef(:))));

index = round(length(s0(:))* rate/100)
   % 配列要素の上位 (rate)% に相当する配列のインデックス
threshold = ordered(index)
coef = coef .* (abs(coef) >= threshold);

% 再構成 （sk に結果が入る）
sk = coef(1:2^(L-j),1:2^(L-j));  % レベルJ のスケーリング係数
for j = J:-1:1
 wvk = coef(1:2^(L-j),2^(L-j)+1:2^(L-j+1));
   % レベル j のウェーブレット展開係数
 whk = coef(2^(L-j)+1:2^(L-j+1),1:2^(L-j));
 wdk = coef(2^(L-j)+1:2^(L-j+1),2^(L-j)+1:2^(L-j+1));
 sk = ifwt2d (sk, wvk,whk,wdk, p, q);
   % 2次元高速ウェーブレット逆変換（1レベル分）
end
```

画像圧縮への応用

original

原画像

JPEG

JPEG圧縮画像(quality=50)

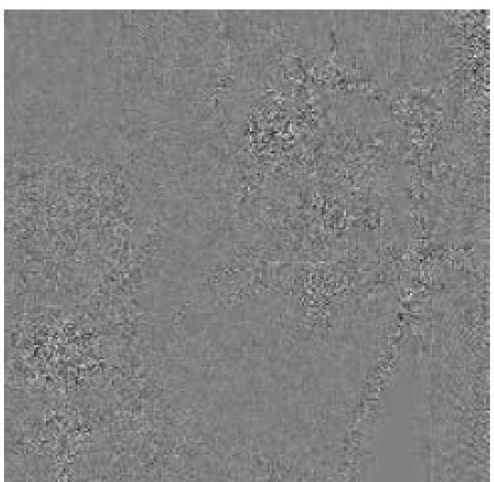
Wavelet

Wavelet圧縮画像(上位10%のみ残す)

画像圧縮への応用

original

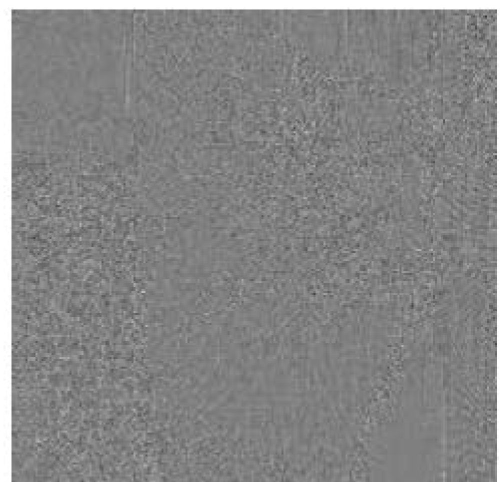
原画像



JPEG

JPEG MSE = 1.5e+001

原画像との差分



Wavelet

Wavelet MSE = 1.1e+001

２次元のウェーブレット展開係数