Доля П.Г.
Харьковский Национальный Университет
механико – математический факультет
2014 г.

Математические методы обработки изображений.

В настоящем пособии излагаются методы, которые используются при обработке изображений. Перед вами часть, которая знакомит с понятием вейвлета и вейвлет преобразования. Описываются основные идеи и формулы, используя наиболее простые из вейвлетов – вейвлеты Хаара. Приводится код Matlab функций и сценариев, поясняющих их работу. Описываются способы использования вейвлетов при хранении и обработке изображений. Дается описание основных функций Wavelet Toolbox Matlab для работы с сигналами и изображениями.

Введение в теорию вейвлетов. Вейвлеты Хаара и их применение.

Оглавление

5.1 Ведение

Рассмотрим изображение, состоящее из двух точек с яркостями $\{x_1,x_2\}$. Эти значения могут быть заменены средним значением a и полуразностью d:

$$a=(x_1+x_2)/2,\ d=(x_1-x_2)/2$$

«Вейвлет - преобразованием» исходной последовательности $\{x_1,x_2\}$ является последовательность $\{a,\ d\}$, зная которую, можно восстановить исходные значения

$$x_1=a+d,\ x_2=a-d.$$

В этом представлении информация не добавляется и не теряется. Но польза от такой замены может быть, если значения x_1 и x_2 близки. В этом случае разность d мала и для ее хранения можно использовать меньший объем памяти или

можно вообще отбросить d и «изображение» $\{x_1,x_2\}$ заменить его приближением $\{a\}$. Мы получаем сжатие изображения. Восстановленным изображением будет $\{a, a\}$.

Рассмотрим большее «изображение» $\{x_1, x_2, x_3, x_4\}$. Вычислим средние значения и разности

$$\begin{aligned} a_{1,0} &= (x_1 + x_2)/2\,,\ a_{1,1} = (x_3 + x_4)/2 \\ d_{1,0} &= (x_1 - x_2)/2\,,\ d_{1,1} = (x_3 - x_4)/2 \end{aligned} \tag{1}$$

(первый из индексов показывает номер многошагового процесса). Мы получили новое представление $\{a_{1,0}, a_{1,1}, d_{1,0}, d_{1,1}\}$ «изображения», которое содержит столько же значений, сколько исходное. Если удалить числа $d_{1,0}, d_{1,1}$, то мы получим сжатое изображение $\{a_{1,0}, a_{1,1}\}$. Применяя туже процедуру к оставшемуся изображению можно записать

$$a_{0,0} = (a_{1,0} + a_{1,1})/2\,,\ d_{0,0} = (a_{1,0} - a_{1,1})/2 \tag{2}$$

или

$$a_{0,0} = \frac{a_{1,0} + a_{1,1}}{2} = \left(\frac{x_1 + x_2}{2} + \frac{x_3 + x_4}{2} \right) \Big/ 2 = \frac{x_1 + x_2 + x_3 + x_4}{4} \tag{3}$$

$$d_{0,0} = \frac{a_{1,0} - a_{1,1}}{2} = \left(\frac{x_1 + x_2}{2} - \frac{x_3 + x_4}{2} \right) \Big/ 2 = \frac{x_1 + x_2 - x_3 - x_4}{4} \tag{4}$$

Пренебрегая $d_{0,0}$, можно заменить все изображение $\{x_1, x_2, x_3, x_4\}$ изображением, состоящим из одного числа $\{a_{0,0}\}$ - средним значением яркостей всех пикселей. Значение $a_{0,0}$ представляет самый приблизительный уровень информации об изображении, т.е. информацию при самом низком разрешении. Значения $\{a_{1,0}, a_{1,1}\}$, вместе взятые, представляют информацию на следующем, более высоком уровне разрешения. Они выражаются через $\{a_{0,0}, d_{0,0}\}$ по формулам

$$a_{1,0} = a_{0,0} + d_{0,0},\ a_{1,1} = a_{0,0} - d_{0,0} \tag{5}$$

Исходные значения/яркости пикселей $\{x_1, x_2, x_3, x_4\}$ представляют самое высокое разрешение изображения. Эти значения могут быть восстановлены обращением (1)

$$x_1 = a_{1,0} + d_{1,0}\ ;\ x_2 = a_{1,0} - d_{1,0}\ ; x_3 = a_{1,1} + d_{1,1}; x_4 = a_{1,1} - d_{1,1} \tag{6}$$

Учитывая (5), мы можем восстановить яркости пикселей из общего среднего $a_{0,0}$ и разностей $d_{0,0}, d_{1,0}, d_{1,1}$

$$\begin{aligned} x_1 &= a_{1,0} + d_{1,0} = a_{0,0} + d_{0,0} + d_{1,0} \\ x_2 &= a_{1,0} - d_{1,0} = a_{0,0} + d_{0,0} - d_{1,0} \\ x_3 &= a_{1,1} + d_{1,1} = a_{0,0} - d_{0,0} + d_{1,1} \\ x_4 &= a_{1,1} - d_{1,1} = a_{0,0} - d_{0,0} - d_{1,1} \end{aligned} \tag{7}$$

Последовательность

$$\{a_{0,0}, d_{0,0}, d_{1,0}, d_{1,1}\} \tag{8}$$

является альтернативным представлением исходного «изображения» и состоит из общего среднего и значений разностей для различных уровней детализации. Последовательность (8) называется вейвлет – преобразованием исходной последовательности $\{x_1, x_2, x_3, x_4\}$. При этом мы имеем несколько вариантов для сжатия, оставляя больше или меньше членов в последовательности (8).

Учитывая (1), (3) и (4), «вейвлет – преобразование» исходного изображения определяется по формулам

$$a_{0,0} = \frac{x_1 + x_2 + x_3 + x_4}{4}, \qquad d_{0,0} = \frac{x_1 + x_2 - x_3 - x_4}{4} \tag{9}$$

$$d_{1,0} = \frac{x_1 - x_2}{2}, \qquad d_{1,1} = \frac{x_3 - x_4}{2}$$

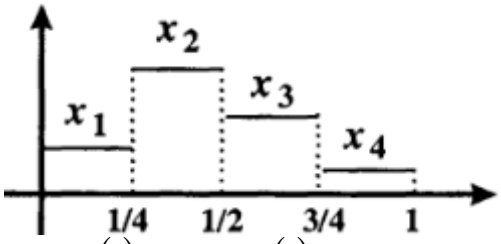
Теперь предположим, что мы рассматриваем наше изображение $\{x_1, x_2, x_3, x_4\}$ как кусочно – постоянную функцию на единичном интервале

$$f(t) = x_1 H_{[0,1/4)}(t) + x_2 H_{[1/4,1/2)}(t) + x_3 H_{[1/2.3/4)}(t) + x_4 H_{[3/4,1)}(t) \tag{10}$$

где $H_{[a,b)}(t)$ характеристическая функция интервала [*a*, *b*), т.е.

$$H_{[a,b)}(t) = \begin{cases} 1, a \le t < b \\ 0, \textit{иначе} \end{cases}$$

На следующем рисунке показано, как может выглядеть график такой кусочно – постоянной функции



Функции $H_{[1/4,1/2)}(t)$, $H_{[1/2,3/4)}(t)$, $H_{[3/4,1)}(t)$ получаются из функции $H_{[0,1/4)}(t)$ сдвигом, например, $H_{[1/4,1/2)}(t) = H_{[0,1/4)}(t - 1/4)$. Кроме того, $H_{[0,1/4)}(t)$ - это масштабирование характеристической функции единичного интервала, т.е. $H_{[0,1/4)}(t) = H_{[0,1)}(2^2 t)$. Т.о. все характеристические функции в выражении (10) могут быть записаны как масштабированные и сдвинутые версии одной единственной функции $H_{[0,1)}(t)$ для которой мы введем специальное обозначение

$$\phi(t) = H_{[0,1)}(t)$$

Эту функцию ϕ называют масштабирующей функцией. Определим функции

$$\phi_{n,j}(t) = \phi(2^n t - j),\ j = 0,1,\ldots,2^n - 1$$

Тогда $\phi_{0,0}(t) = \phi(t)$ и

$$\phi_{1,0}(t) = \phi(2t) = \{1, t \in [0, 1/2); 0, t \notin [0, 1/2)\}$$
$$\phi_{1,1}(t) = \phi(2t - 1) = \{1, t \in [1/2, 1); 0, t \notin [1/2, 1)\}$$

Длина носителя функций $\phi_{k,j}$ уменьшается в два раза при увеличении k на единицу. Это значит, что длина носителя $\phi_{k+1,j}$ составляет половину длины носителя $\phi_{k,j}$.

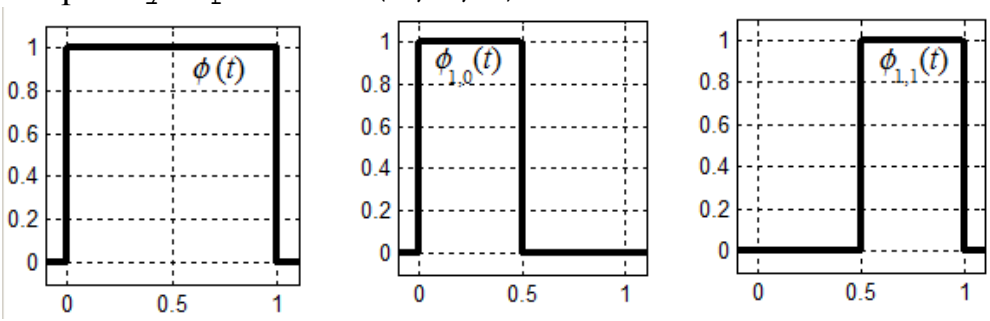
Пример. Построим графики функций $\phi(t), \phi_{1,0}(t), \phi_{1,1}(t)$. Вначале создадим функцию `psiHaar(x,n,j)` (\ch05\psiHaar.m)

```
% ненормированные масштабирующие функции Хаара
% x - вектор строка аргументов
function y=psiHaar(x,n,j)
y=psiHaarBase(2.^n*x-j);
% базисная масштабирующая функция Хаара
function y=psiHaarBase(x)  % подфункция
y=(x>=0 & x <1).*1.0;
```

Теперь напишем сценарий построения графика масштабирующей функции (\ch05\ch05ex03.m)

```
% Графики масштабирующих функций
close all; clear all;
x=-0.1:0.0025:1.1;      % диапазон для графика
y1=psiHaar(x,1,1);
plot(x,y1,'k-','LineWidth',3);  % график;
axis([-0.1 1.1 -0.1 1.1]); grid on;
```

Для построения графиков функций $\phi(t), \phi_{1,0}(t), \phi_{1,1}(t)$ мы меняем параметры n и j в строке `y1=psiHaar(x,1,1)`.



□

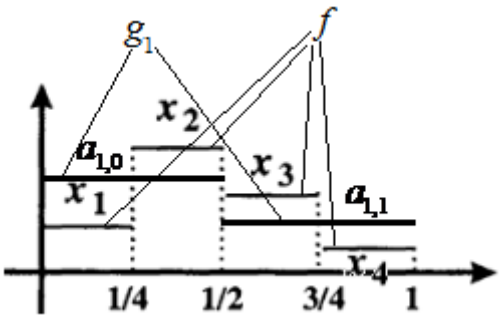
Используя введенные обозначения, функция $f(t)$ может быть выражена с помощью этих функций

$$f(t) = x_1 H_{[0,1/4)}(t) + x_2 H_{[1/4,1/2)}(t) + x_3 H_{[1/2.3/4)}(t) + x_4 H_{[3/4,1)}(t) =$$
$$= x_1 \varphi_{2,0}(t) + x_2 \varphi_{2,1}(t) + x_3 \varphi_{2,2}(t) + x_4 \varphi_{2,3}(t)$$

Попытаемся теперь представить функцию $f(t)$ не через значения $\{x_1, x_2, x_3, x_4\}$, а через значения среднего и разностей $\{a_{0,0}, d_{0,0}, d_{1,0}, d_{1,1}\}$, которые мы получили выше (см. формулы (9)). Для этого вначале построим функцию

$$g_1(t) = a_{1,0}\phi_{1,0}(t) + a_{1,1}\phi_{1,1}(t) \qquad (11)$$

где $a_{1,0} = (x_1 + x_2)/2$, $a_{1,1} = (x_3 + x_4)/2$. Очевидно, что она не совпадает с f. Графики функций $f(t)$ и $g_1(t)$ приведены на следующем рисунке



Для выделения деталей нам потребуется еще одна «базисная» функция, которая должна позволять выражать разность. Определим ее следующим образом

$$\psi(t)=H_{\left[0,\frac{1}{2}\right)}(t)-H_{\left[\frac{1}{2},1\right)}(t)=\begin{cases}1,\ t\in[0,1/2)\\-1,\ t\in[1/2,1)\\0,\ t\notin[0,1)\end{cases}$$

Она называется базисной вейвлет - функцией.

По-аналогии с предыдущим, введем масштабированные и сдвинутые варианты этой функции следующим образом

$$\psi_{n,j}(t)=\psi\left(2^n t-j\right),\ j=0,1,\ldots,2^n-1$$

Тогда, например, $\psi_{0,0}(t)=\psi(t)$ и

$$\psi_{1,0}(t)=\psi(2t)=\begin{cases}1,\ t\in[0,1/4)\\-1,\ t\in[1/4,1/2)\\0,\ t\notin[0,1/2)\end{cases},\qquad \psi_{1,1}(t)=\psi(2t-1)=\begin{cases}1,\ t\in[1/2,3/4)\\-1,\ t\in[3/4,1)\\0,\ t\notin[1/2,1)\end{cases}$$

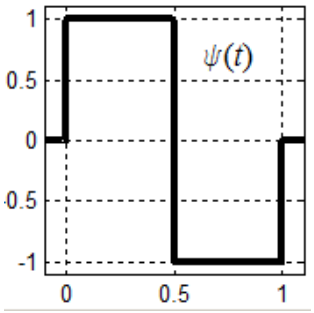
Пример. Построим графики функций $\psi(t),\psi_{1,0}(t),\psi_{1,1}(t)$. Вначале создадим функцию `ksiHaar(x,n,j)` (\ch05\ksiHaar.m)

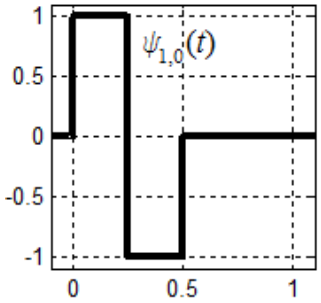
```
% ненормированные вейвлеты Хаара
% x - вектор строка аргументов
function y=ksiHaar(x,n,j)
y=ksiHaarBase(2.^n*x-j);
% базисный вейвлет Хаара
function y=ksiHaarBase(x)
y=(x>=0 & x <0.5).*1.0-(x>=0.5 & x <1).*1.0;
```

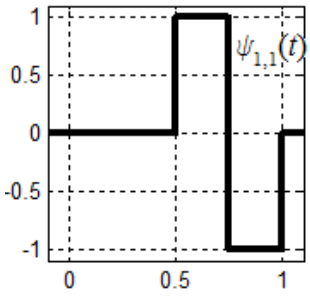
Теперь напишем сценарий построения графика вейвлет функции (\ch05\ch05ex03.m)

```
% Графики вейвлетов Хаара
close all; clear all;
x=-0.1:0.0025:1.1;       % диапазон для графика
y2=ksiHaar(x,1,0);
figure,plot(x,y2,'k-','LineWidth',3);    % график;
axis([-0.1 1.1 -1.1 1.1]); grid on;
```

На следующем рисунке показаны графики функций $\psi(t)$, $\psi_{1,0}(t)$ и $\psi_{1,1}(t)$, построенные с помощью этого сценария (мы меняли второй и третий аргументы при вызове функции `ksiHaar(x,1,0)` в третьей строке кода).



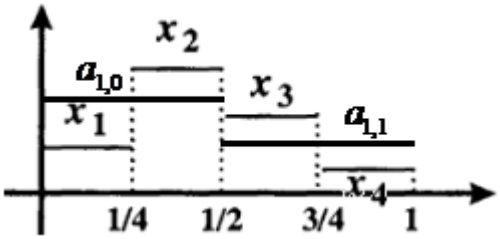




□

Вернемся к задаче выражения кусочно – постоянной функции (10) через средние значения и разности или, пользуясь новой терминологией, через функции масштабирования и вейвлеты $\psi_{n,j}(t)$.

Напомним, что первое приближение (11) $g_1(t)=a_{1,0}\phi_{1,0}(t)+a_{1,1}\phi_{1,1}(t)$ не совпадает с исходной функцией. Для наглядности последующих вычислений напомним график функции $g_1(t)$ и формулы (1)



$$a_{1,0}=(x_1+x_2)/2,\ a_{1,1}=(x_3+x_4)/2$$
$$d_{1,0}=(x_1-x_2)/2,\ d_{1,1}=(x_3-x_4)/2$$

Рассмотрим отличие $g_1(t)$ от $f(t)$ на всех интервалах длиной 1/4. На [0, 1/4) разность равна

$$f(t)-g_1(t)=x_1-a_{1,0}=x_1-\frac{x_1+x_2}{2}=\frac{x_1-x_2}{2}=d_{1,0}$$

На [1/4, 1/2) разность равна

$$f(t)-g_1(t)=x_2-a_{1,0}=x_2-\frac{x_1+x_2}{2}=\frac{x_2-x_1}{2}=-d_{1,0}$$

Здесь мы использовали формулы (1). Вспоминая какой вид имеет функция $\psi_{1,0}(t)$, для интервала [0,1/2) можем записать (второе слагаемое $a_{1,1}\psi_{1,1}(t)$ на этом участке равно нулю)

$$f(t)=g_1(t)+d_{1,0}\psi_{1,0}(t)=a_{1,0}\phi_{1,0}(t)+d_{1,0}\psi_{1,0}(t)$$

Аналогично, на интервале [1/2,3/4) имеем

$$f(t)-g_1(t)=x_3-a_{1,1}=x_3-\frac{x_3+x_4}{2}=\frac{x_3-x_4}{2}=d_{1,1}$$

На [3/4, 1] разность равна

$$f(t)-g_1(t)=x_4-a_{1,1}=x_4-\frac{x_3+x_4}{2}=\frac{x_4-x_3}{2}=-d_{1,1}$$

Тогда для t принадлежащего интервалу [1/2, 1] имеем

$$f(t)=g_1(t)+d_{1,1}\psi_{1,1}(t)=a_{1,1}\phi_{1,1}(t)+d_{1,1}\psi_{1,1}(t)$$

Объединяя эти формулы, мы получим новое представление для f на всем интервале [0, 1)

$$f(t)=a_{1,0}\,\varphi_{1.0}(t)+a_{1,1}\,\varphi_{1.1}(t)+d_{1,0}\,\psi_{1.0}(t)+d_{1,1}\,\psi_{1.1}(t)$$

Теперь заменим в правой части последнего выражения первые два слагаемых на $a_{00}\,\phi_{0,0}(t)$, а последние два оставим без изменений

$$g_0(t) = a_{0,0}\,\varphi_{0.0}(t) + d_{1,0}\,\psi_{1.0}(t) + d_{1,1}\,\psi_{1.1}(t)$$

Учитывая (9) на [0,1/4] разность между $f(t)$ и $g_0(t)$ будет равна ($\psi_{1,1}$ на этом интервале равна 0, а $\psi_{1,0}$ равна 1)

$$f(t) - g_0(t) = x_1 - a_{0,0} - d_{1,0} = x_1 - \frac{x_1 + x_2 + x_3 + x_4}{4} - \frac{x_1 - x_2}{2} =$$

$$= \frac{4x_1 - (x_1 + x_2 + x_3 + x_4) - 2(x_1 - x_2)}{4} = \frac{x_1 + x_2 - x_3 - x_4}{4} = d_{0,0}$$

На [1/4,1/2) мы получим ($\psi_{1,1}$ на этом интервале все еще 0, а $\psi_{1,0}$ равна –1)

$$f(t) - g_0(t) = x_2 - a_{0,0} + d_{1,0} = x_2 - \frac{x_1 + x_2 + x_3 + x_4}{4} + \frac{x_1 - x_2}{2} =$$

$$= \frac{4x_2 - (x_1 + x_2 + x_3 + x_4) + 2(x_1 - x_2)}{4} = \frac{x_1 + x_2 - x_3 - x_4}{4} = d_{0,0}$$

Аналогично, на [1/2,3/4) имеем

$$f(t) - g_0(t) = x_3 - a_{0,0} - d_{1,1} = x_3 - \frac{x_1 + x_2 + x_3 + x_4}{4} - \frac{x_3 - x_4}{2} =$$

$$= \frac{4x_3 - (x_1 + x_2 + x_3 + x_4) - 2(x_3 - x_4)}{4} = \frac{-x_1 - x_2 + x_3 + x_4}{4} = -d_{0,0}$$

и на [3/4,1) имеем

$$f(t) - g_0(t) = x_4 - a_{0,0} + d_{1,1} = x_4 - \frac{x_1 + x_2 + x_3 + x_4}{4} + \frac{x_3 - x_4}{2} =$$

$$= \frac{4x_4 - (x_1 + x_2 + x_3 + x_4) + 2(x_3 - x_4)}{4} = \frac{-x_1 - x_2 + x_3 + x_4}{4} = -d_{0,0}$$

Следовательно, на всем интервале [0, 1) можно записать $f(t) = g_0(t) + d_{0,0}\psi_{0,0}(t)$ или окончательно

$$f(t) = a_{0,0}\,\varphi_{0.0}(t) + d_{0,0}\,\psi_{0.0}(t) + d_{1,0}\,\psi_{1.0}(t) + d_{1,1}\,\psi_{1.1}(t) \qquad (12)$$

Это выражение является аналогом последовательности (8) $\{a_{0,0}, d_{0,0}, d_{1,0}, d_{1,1}\}$ для функции $f(t)$.

5.2 Вейвлеты Хаара

Рассмотрим пространство V^0 всех функций, постоянных на интервале [0, 1). V^0 является линейным векторным пространством. Масштабирующая функция

$$\phi(t) = \phi_{0,0}(t) = \begin{cases} 1, 0 \le t < 1 \\ 0, \textit{иначе} \end{cases} \qquad (1)$$

принадлежит V^0 и является тривиальным базисом в этом пространстве.

Совокупность кусочно-постоянных функций, являющихся константами на интервалах [0,1/2) и [1/2,1) также образует линейное векторное пространство V^1. Масштабирующие функции

$$\phi_{1,0}(t)=\phi(2t)=\begin{cases}1, 0\le t<1/2\\ 0, иначе\end{cases} \quad \text{и} \quad \phi_{1,1}(t)=\phi(2t-1)=\begin{cases}1, 1/2\le t<1\\ 0, иначе\end{cases}$$

принадлежат этому пространству и образуют в нем базис. При этом функции постоянные на интервале [0,1) являются постоянными на каждом из интервалов [0,1/2) и [1/2,0). Поэтому каждый элемент V^0 является элементом V^1, т.е. $V^0 \subset V^1$.

Аналогично определим V^2 как пространство кусочно-постоянных функций на интервалах [0,1/4), [1/4,1/2), [1/2,3/4), [3/4,1) и, далее, определим V^n как пространство кусочно-постоянных функций на равноотстоящих интервалах длиной $1/2^n$. Ясно, что V^n является линейным векторным пространством. Масштабирующие функции $\phi_{n,j}(t)=\phi(2^n t-j),\ j=0,1,\ldots,2^n-1$ имеют носитель

$$0\le 2^n t-j<1,\ \frac{j}{2^n}\le t<\frac{j+1}{2^n}, \operatorname{supp}(\phi_{n,j}(t))=\left[\frac{j}{2^n},\frac{j+1}{2^n}\right),$$

и определяются выражением

$$\phi_{n,j}(t)=\begin{cases}1,\ j/2^n\le t<(j+1)/2^n\\ 0, иначе\end{cases},\ j=0,1,\ldots,2^n-1. \qquad (2)$$

Они образуют базис в пространстве V^n. Кроме того, пространства V^n последовательно вложены друг в друга

$$V^0\subset V^1\subset\ldots\subset V^n\subset\ldots$$

Векторное пространство с введенным в нем скалярным произведением называется евклидовым пространством. В нашем случае скалярное произведение определяется как

$$(f,g)=\int_0^1 f(t)g(t)dt.$$

Поэтому все пространства V^n евклидовы. Две функции называются ортогональными, если $(f,g)=0$. Тогда

$$(\phi_{1,0},\phi_{1,1})=\int_0^1 \phi_{1,0}(x)\phi_{1,1}(x)dx=0,$$

поскольку носители функций $\phi_{1,0}$ и $\phi_{1,1}$ не пересекаются. Поэтому $\phi_{1,0}$ и $\phi_{1,1}$ образуют ортогональный базис пространства V^1. Аналогично при $i\ne j$

$$(\phi_{n,i},\phi_{n,j})=\int_0^1 \phi_{n,i}(x)\phi_{n,j}(x)dx=0,$$

Это означает, что $\phi_{n,j}(t),\ j=0,1,\ldots,2^n-1$ образует множество взаимно ортогональных базисных векторов в V^n.

Для двух евклидовых пространств $E_1\subset E_2$ можно рассмотреть подмножество векторов из E_2, которые ортогональны всем векторам из E_1. Оно называется ортогональным дополнением пространства E_1 в пространство E_2 и, как легко видеть, само является векторным (и евклидовым) пространством.

Для пространств $V^n \subset V^{n+1}$ рассмотрим ортогональное дополнение V^n в V^{n+1}, которое обозначим через W^n:

$$W^n = \left\{ v \in V^{n+1} : (v, f) = 0 \ \ \forall \ f \in V^n \right\}$$

Построим базис в пространстве W^n. Рассмотрим функцию

$$\psi(t) = \phi(2t) - \phi(2t-1) = \psi_{0,0}(t) = \begin{cases} 1, \ t \in [0,1/2) \\ -1, t \in [1/2,1) \\ 0, \ t \notin [0,1) \end{cases} \qquad (3)$$

и построим функции $\psi_{n,j}(t) = \psi(2^n t - j), \ j = 0,1,..2^n - 1$. Легко видеть, что носитель этих функций такой же как и у функций $\phi_{n,j}(t)$

$$0 \le 2^n t - j < 1, \ \frac{j}{2^n} \le t < \frac{j+1}{2^n}, \operatorname{supp}(\psi_{n,j}(t)) = \left[\frac{j}{2^n}, \frac{j+1}{2^n} \right)$$

Число n характеризует уровень разрешения. Чем больше n, тем более узкие носители имеют функции $\psi_{n,j}(t)$ и $\phi_{n,j}(t)$.

Функции $\psi_{n,j}(t)$ равны плюс единице на участке

$$0 \le 2^n t - j < 1/2, \ \frac{j}{2^n} \le t < \frac{j+1/2}{2^n},$$

равны минус единице на участке

$$1/2 \le 2^n t - j < 1, \ \frac{j+1/2}{2^n} \le t < \frac{j+1}{2^n},$$

и равны нулю на оставшейся части вещественной оси. Т.е.

$$\psi_{n,j}(t) = \begin{cases} 1, \frac{j}{2^n} \le t < \frac{j}{2^n} + \frac{1}{2^{n+1}} \\ -1, \frac{j}{2^n} + \frac{1}{2^{n+1}} \le t < \frac{j}{2^n} + \frac{1}{2^n} \\ 0, \ иначе \end{cases} \qquad (4)$$

Длина интервалов на которых $\psi_{n,j}(t)$ постоянны, составляют половину длины интервалов на которых элементы пространства V^n являются константами. Другими словами, $\psi_{n,j}(t) \in V^{n+1}$ при каждом j. Кроме того, ясно, что $(\psi_{n,j}, f) = 0$ для каждого $f \in V^n$. Действительно,

$$(\psi_{n,j}, f) = \int_0^1 \psi_{n,j}(t) \left(\sum_{k=0}^{2^n-1} x_k \phi_{n,k}(t) \right) dt = \sum_{k=0}^{2^k-1} x_k \int_0^1 \psi_{n,j}(t) \phi_{n,k}(t) \, dt$$

Но при $k \ne j$ носители функций $\psi_{n,j}(t)$ и $\phi_{n,k}(t)$ не пересекаются и $\int_0^1 \psi_{n,j}(t) \phi_{n,k}(t) \, dt = 0$, а при $k = j$ носители функций $\psi_{n,j}(t)$ и $\phi_{n,k}(t)$ совпадают и, поскольку $\phi_{n,k}(t)$ постоянна на участке своего носителя, а $\psi_{n,k}$ на половине

этого участка равна +1, а на другой половине равна –1 , то также имеем $\int_0^1 \psi_{n,k}(t)\,\phi_{n,k}(t)\,dt = 0$. Таким образом, $\psi_{n,j}(t) \in W^n$ при каждом j.

Поскольку $W^n \subset V^{n+1}$, то размерность W^n не больше 2^{n+1} размерности V^{n+1}. С другой стороны система функций $\psi_{n,j}(t)$ содержит 2^n взаимно ортогональных линейно независимых векторов в W^n. Т.е. размерность W^n равна, по крайней мере, 2^n. С другой стороны базис $V^n \subset V^{n+1}$ содержит тоже 2^n элементов – функций $\phi_{n,j}(t)$, каждая из которых ортогональна элементам подпространства W^n. Это значит, что совокупность функций

$$\left\{\phi_{n,j}(t), \psi_{n,j}(t)\right\},\ j = 0,1,..2^n - 1. \qquad (5)$$

образует новый базис в пространстве V^{n+1}. Напомним, что старым базисом в этом пространстве V^{n+1} была система функций $\phi_{n+1,j}(t)$, $j = 0,1,..2^{n+1} - 1$. Это также означает, что $V^{n+1} = V^n \oplus W^n$, где значок $\oplus$ обозначает прямую сумму векторных пространств.

Предположим, что W^n содержит вектор, который не может быть выражен через систему векторов $\psi_{n,j}(t)$. Тогда он будет содержать в своем разложении элементы $\phi_{n,j}(t)$, но тогда он не будет принадлежать ортогональному дополнению к V^n. Таким образом, размерность пространства W^n равна 2^n и функции $\psi_{n,j}(t)$, $j = 0,1,..2^n - 1$ образуют в нем базис. Функции $\psi_{n,j}(t)$ называют вейвлетами.

Удобно работать с нормированными масштабирующими функциями и вейвлетами. Норма вектора в евклидовом пространстве определяется через скалярное произведение как $\| f \| = \sqrt{(f, f)}$. Вектор $\boldsymbol{u}$ называется нормированным, если $\| \boldsymbol{u} \| = 1$. Нормировка любого вектора выполняется путем деления вектора на его норму, например, $\boldsymbol{u} = \boldsymbol{f} / \| \boldsymbol{f} \|$. Чтобы нормировать $\psi_{n,j}(t)$ и $\phi_{n,j}(t)$ нужно определить их нормы

$$\| \phi_{n,j} \|^2 = \int_0^1 \phi_{n,j}^2(t)\,dt = \int_{j/2^n}^{(j+1)/2^n} 1 dt = \frac{1}{2^n}$$

Тогда $\| \phi_{n,j} \| = \frac{1}{\sqrt{2^n}}$. Аналогично $\| \psi_{n,j} \| = \frac{1}{\sqrt{2^n}}$.

Переопределим функции $\psi_{n,j}(t)$ и $\phi_{n,j}(t)$ так, чтобы они были нормированными

$$\phi_{n,j}(t) = \sqrt{2^n}\ \phi\left(2^n t - j\right),\ j = 0,1,\ldots,2^n - 1, \qquad (6)$$

$$\psi_{n,j}(t) = \sqrt{2^n}\ \psi\left(2^n t - j\right),\ j = 0,1,\ldots,2^n - 1, \qquad (7)$$

где функции $\phi(t)$ и $\psi(t)$ заданы выражениями (1) и (3).

Замечание. Если пределы изменения j не ограничивать диапазоном $j = 0,1,\ldots,2^n - 1$, а разрешить j принимать любые целые значения $j \in \boldsymbol{Z}$, то

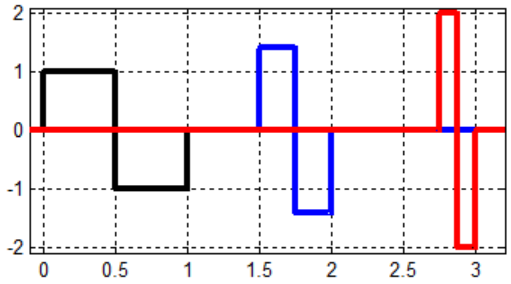
системами функций $\psi_{n,j}(t)$ и $\phi_{n,j}(t)$ можно приближать любые кусочно-непрерывные функции, заданные на вещественной оси.

Пример. Построим графики нормированных вейвлетов Хаара. Вначале создадим функцию `ksiNormHaar(x,n,j)` (\ch05\ksiNormHaar.m)

```
% нормированные вейвлеты Хаара
% x - вектор строка аргументов
function y=ksiNormHaar(x,n,j)
y=ksiHaarBase(2.^n*x-j).*(2.^(n/2));
% базисный вейвлет Хаара
function y=ksiHaarBase(x)
y=(x>=0 & x <0.5).*1.0-(x>=0.5 & x <1).*1.0;
```

Теперь напишем сценарий построения графика вейвлет функции (\ch05\ch05ex04.m)

```
% Графики вейвлетов Хаара
close all; clear all;
x=-0.1:0.0025:3.2;       % диапазон для графика
y2=ksiNormHaar(x,0,0);
plot(x,y2,'k-','LineWidth',3);    % график;
grid on; hold on;
y3=ksiNormHaar(x,1,3);
plot(x,y3,'b-','LineWidth',3);  % график;
y4=ksiNormHaar(x,2,11);
plot(x,y4,'r-','LineWidth',3);  % график;
axis([min(x) max(x) min(y4)-0.1 max(y4)+0.1]); grid on;
```



Здесь мы построили графики следующих нормированных вейвлетов $\psi(t), \psi_{1,3}(t), \psi_{2,11}(t)$.

□

Функции $\psi_{n,j}(t)$, заданные выражениями (7), называют базисными вейвлетами Хаара, поскольку они образуют базис пространства W^n. Как будет показано позже, вейвлеты можно определить как любые базисные функции для этого ортогонального дополнения. Часто вейвлетами называют любые элементы пространства W^n. Функция $\phi(t)$ называется масштабирующей функцией Хаара, а функция $\psi(t)$ называется материнским вейвлетом Хаара.

Замечание. При построении вейвлетов можно разрешить использование отрицательных значений *n*.

Замечание. При n=1,2,... справедливы соотношения

$$\varphi(x)=\frac{1}{\sqrt{2}}\left(\varphi_{1,0}(x)+\varphi_{1,1}(x)\right)=\frac{1}{2}\left(\varphi_{2,0}(x)+\varphi_{2,1}(x)+\varphi_{2,2}(x)+\varphi_{2,3}(x)\right)=\ldots=$$

$$=\frac{1}{\sqrt{2^n}}\sum_{j=0}^{2^n-1}\varphi_{n,j}(x)$$

или

$$\varphi(x)=\frac{1}{\sqrt{2^n}}\sum_{j=0}^{2^n-1}\varphi\left(2^n x-j\right) \tag{8}$$

Последнее соотношение часто называют основным уравнением кратномасштабного анализа.

5.3 Вейвлет - преобразование

Пусть дана последовательность 2^n чисел $\left\{x_1,x_2,\ldots,x_{2^n}\right\}$. Ее можно отождествить с кусочно – постоянной функцией из V^n вида

$$f(t)=\frac{x_1}{\sqrt{2^n}}\phi_{n,0}(t)+\ldots+\frac{x_{2^n}}{\sqrt{2^n}}\phi_{n,2^n-1}(t)=\tilde{x}_1\phi_{n,0}(t)+\ldots+\tilde{x}_{2^n}\phi_{n,2^n-1}(t), \tag{1}$$

где $\tilde{x}_j=x_j/\sqrt{2^n}$. Делитель $\sqrt{2^n}$ здесь появляется вследствие того, что функции $\phi_{n,j}(t)$ теперь нормированные (см. выражения (2.6)).

Первым шагом вейвлет – преобразования последовательности $\left\{x_1,x_2,\ldots,x_{2^n}\right\}$ является разложение $f(t)$ по альтернативному базису (2.5) пространства V^n, половину которого составляют вейвлеты.

$$\begin{aligned} f(t)=a_{n-1,0}\,\phi_{n-1,0}(t)+\ldots+a_{n-1,2^{n-1}-1}\,\phi_{n-1,2^{n-1}-1}(t)+ \\ +d_{n-1,0}\,\psi_{n-1,0}(t)+\ldots+d_{n-1,2^{n-1}-1}\psi_{n-1,2^{n-1}-1}(t) \end{aligned} \tag{2}$$

Первая строка представляет грубое приближение функции $f(t)$, которое принадлежит пространству V^{n-1}, а вторая – отражает детали, которыми $f(t)$ отличается от «грубого» приближения.

Коэффициенты $d_{n-1,0},\ldots,d_{n-1,2^{n-1}-1}$ при базисных вейвлет функциях оставляем, а первую часть суммы, которую мы обозначим как $g_{n-1}\in V^{n-1}$,

$$g_{n-1}(t)=a_{n-1,0}\,\phi_{n-1,0}(t)+\ldots+a_{n-1,2^{n-1}-1}\,\phi_{n-1,2^{n-1}-1}(t) \tag{3}$$

подвергнем дальнейшему преобразованию. Поскольку $g_{n-1}\in V^{n-1}$, то функцию $g_{n-1}(t)$, можно разложить по альтернативному базису пространства V^{n-1}, состоящему из масштабирующих функций $\phi_{n-2,j}$ и вейвлетов $\psi_{n-2,j}$. Затем этот процесс переразложения можно продолжить.

Определим коэффициенты переразложения (2). Умножим обе части выражения (2) скалярно на $\phi_{n-1,j}$. Учитывая, что функция $\phi_{n-1,j}$ ортогональна

всем функциям $\psi_{n-1,k}$, ортогональна всем функциям $\phi_{n-1,k}$ при $k \neq j$ и нормирована, получаем

$$\left(f, \phi_{n-1,j}\right) = a_{n-1,j} \quad \text{или} \quad \int_0^1 f(t)\phi_{n-1,j}(t)dt = a_{n-1,j} \tag{4}$$

Теперь подставим в (4) правую часть равенства (1) и учтем, что $\operatorname{supp}\left(\phi_{n-1,j}(t)\right) = \left[\frac{j}{2^{n-1}}, \frac{j+1}{2^{n-1}}\right)$. Имеем

$$a_{n-1,j} = \int_{j/2^{n-1}}^{(j+1)/2^{n-1}} \left(\tilde{x}_1\phi_{n,0}(t) + \ldots + \tilde{x}_{2^n}\phi_{n,2^n-1}(t)\right)\cdot\sqrt{2^{n-1}}\,dt$$

Поскольку

$$\operatorname{supp}\left(\phi_{n-1,j}(t)\right) = \left[\frac{j}{2^{n-1}}, \frac{j+1}{2^{n-1}}\right) = \left[\frac{2j}{2^n}, \frac{2(j+1)}{2^n}\right) = \left[\frac{2j}{2^n}, \frac{2j+1}{2^n}\right) \cup \left[\frac{2j+1}{2^n}, \frac{2(j+1)}{2^n}\right),$$

то получаем далее

$$a_{n-1,j} = \sqrt{2^{n-1}} \int_{j/2^{n-1}}^{(j+1)/2^{n-1}} \left(\tilde{x}_{2j+1}\phi_{n,2j}(t) + \tilde{x}_{2j+2}\phi_{n,2j+1}(t)\right)dt =$$

$$= \sqrt{2^{n-1}} \int_{2j/2^n}^{(2j+1)/2^n} \tilde{x}_{2j+1}\phi_{n,2j}(t)dt \quad + \sqrt{2^{n-1}} \int_{(2j+1)/2^n}^{(2j+2)/2^n} \tilde{x}_{2j+2}\phi_{n,2j+1}(t)dt =$$

$$= \sqrt{2^{n-1}}\tilde{x}_{2j+1}\frac{1}{2^n}\sqrt{2^n} + \sqrt{2^{n-1}}\tilde{x}_{2j+2}\frac{1}{2^n}\sqrt{2^n} = \frac{\tilde{x}_{2j+1} + \tilde{x}_{2j+2}}{\sqrt{2}}$$

Т.о.

$$a_{n-1,j} = \frac{\tilde{x}_{2j+1} + \tilde{x}_{2j+2}}{\sqrt{2}}, \quad j = 0,1,\ldots,2^{n-1} \tag{5}$$

Аналогично, используя свойство ортогональности и нормированности функций $\psi_{n,j}$, вычислим коэффициенты $d_{n-1,j}$. Умножим обе части выражения (2) скалярно на $\psi_{n-1,j}$. Учитывая, что функция $\psi_{n-1,j}$ ортогональна всем функциям $\phi_{n-1,k}$, ортогональна всем функциям $\psi_{n-1,k}$ при $k \neq j$ и нормирована, получаем

$$\left(f, \psi_{n-1,j}\right) = d_{n-1,j} \quad \text{или} \quad \int_0^1 f(t)\,\psi_{n-1,j}(t)dt = d_{n-1,j}$$

Подставим в (4) правую часть равенства (1) и учтем, что $\operatorname{supp}\left(\psi_{n-1,j}(t)\right) = \left[\frac{j}{2^{n-1}}, \frac{j+1}{2^{n-1}}\right)$. Имеем

$$d_{n-1,j} = \int_{j/2^{n-1}}^{(j+1)/2^{n-1}} \left(\tilde{x}_1\,\phi_{n,0}(t) + \ldots + \tilde{x}_{2^n}\,\phi_{n,2^n-1}(t)\right)\psi_{n-1,j}(t)dt$$

Поскольку

$$\operatorname{supp}\left(\phi_{n-1,j}(t)\right) = \left[\frac{j}{2^{n-1}}, \frac{j+1}{2^{n-1}}\right) = \left[\frac{2j}{2^n}, \frac{2(j+1)}{2^n}\right) = \left[\frac{2j}{2^n}, \frac{2j+1}{2^n}\right) \cup \left[\frac{2j+1}{2^n}, \frac{2(j+1)}{2^n}\right),$$

то получаем далее

$$d_{n-1,j} = \int_{j/2^{n-1}}^{(j+1)/2^{n-1}} \left(\tilde{x}_{2j+1}\phi_{n,2j}(t) + \tilde{x}_{2j+2}\,\phi_{n,2j+1}(t) \right) \psi_{n-1,j}(t)\,dt =$$

$$= \sqrt{2^{n-1}} \int_{2j/2^n}^{(2j+1)/2^n} \tilde{x}_{2j+1}\phi_{n,2j}(t)(+1)dt \;\; + \sqrt{2^{n-1}} \int_{(2j+1)/2^n}^{(2j+2)/2^n} \tilde{x}_{2j+2}\phi_{n,2j+1}(t)(-1)dt =$$

$$= \sqrt{2^{n-1}}\tilde{x}_{2j+1}\frac{1}{2^n}\sqrt{2^n} - \sqrt{2^{n-1}}\tilde{x}_{2j+2}\frac{1}{2^n}\sqrt{2^n} = \frac{\tilde{x}_{2j+1} - \tilde{x}_{2j+2}}{\sqrt{2}}$$

Т.о.

$$d_{n-1,j} = \frac{\tilde{x}_{2j+1} - \tilde{x}_{2j+2}}{\sqrt{2}}, \;\; j = 0,1,\ldots,2^{n-1} \qquad (6)$$

Обратим внимание на квадратный корень в знаменателях формул (5), (6), который появляется в результате нормирования функций $\phi_{n,j}$ и $\psi_{n,j}$. Без этого мы имели бы полусумму и полуразность, как в п. 5.1.

Вот код Matlab функции, выполняющей один шаг вейвлет – преобразования (ch05\w1step.m)

```
% выполнение одного шага вейвлет преобразования
% принимает вектор f; length(f) - должно быть степенью двойки
% возвращает вектор коэф. a и вектор коэф. d
function [a,d]=w1step(f)
L=length(f);
m=L/2;
a=zeros(1,m);
d=zeros(1,m);
dv=sqrt(2);
for j=0:m-1
    a(j+1)=(f(2*j+1)+f(2*j+2))/dv;
    d(j+1)=(f(2*j+1)-f(2*j+2))/dv;
end
```

Пример 1. Тестирование функции w1step (ch05\ch05ex05.m)

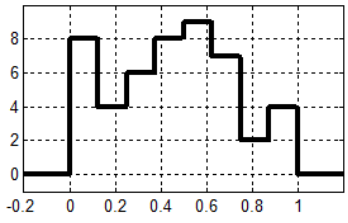
```
% тестирование функции w1step
close all; clear all;
f=[8 4 6 8 9 7 2 4];% length(f) - должно быть степенью двойки
n=length(f);
f=f./sqrt(n);  % преобразуем f к нормированным psiNormHaar
x=-0.2:0.0025:1.2;
%  график исходной кусочно - постоянной функции f
y=0;
for i=1:n
    y=y+f(i).*psiNormHaar(x,3,i-1);
end
plot(x,y,'k-','LineWidth',3);   % график
axis([min(x) max(x) min(y)-1 max(y)+1]); grid on;
[a,d]=w1step(f);   % первый шаг вейвлет - преобразования
% график первой усредняющей кусочно - постоянной функции
g=0;
```

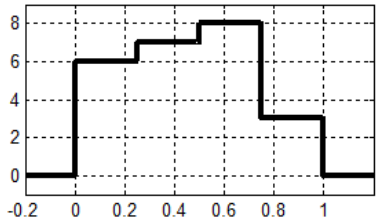
```
for i=1:n/2
    g=g+a(i).*psiNormHaar(x,2,i-1);
end
figure,plot(x,g,'k-','LineWidth',3);     % график
axis([min(x) max(x) min(g)-1 max(g)+1]); grid on;
%  график первой детализирующей кусочно - постоянной функции
h=0;
for i=1:n/2
    h=h+d(i).*ksiNormHaar(x,2,i-1);
end
figure,plot(x,h,'k-','LineWidth',3);     % график
axis([min(x) max(x) min(h)-1 max(h)+1]); grid on;
set(gcf,'Color',[1 1 1]);
% проверка - складываем усредняющую и уточняющую функции
ys=g+h;
figure,plot(x,ys,'k-','LineWidth',3);    % график
axis([min(x) max(x) min(ys)-1 max(ys)+1]); grid on;
```

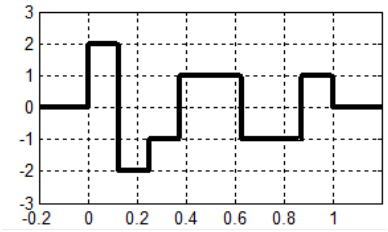
Получаем следующие значения коэффициентов *a* и *d*:

```
a =    3.0000    3.5000    4.0000    1.5000
d =    1.0000   -0.5000    0.5000   -0.5000
```

Ниже на первом графике показана исходная кусочно – постоянная функция; на втором – график усредняющей функции; на третьем – график детализирующей функции.







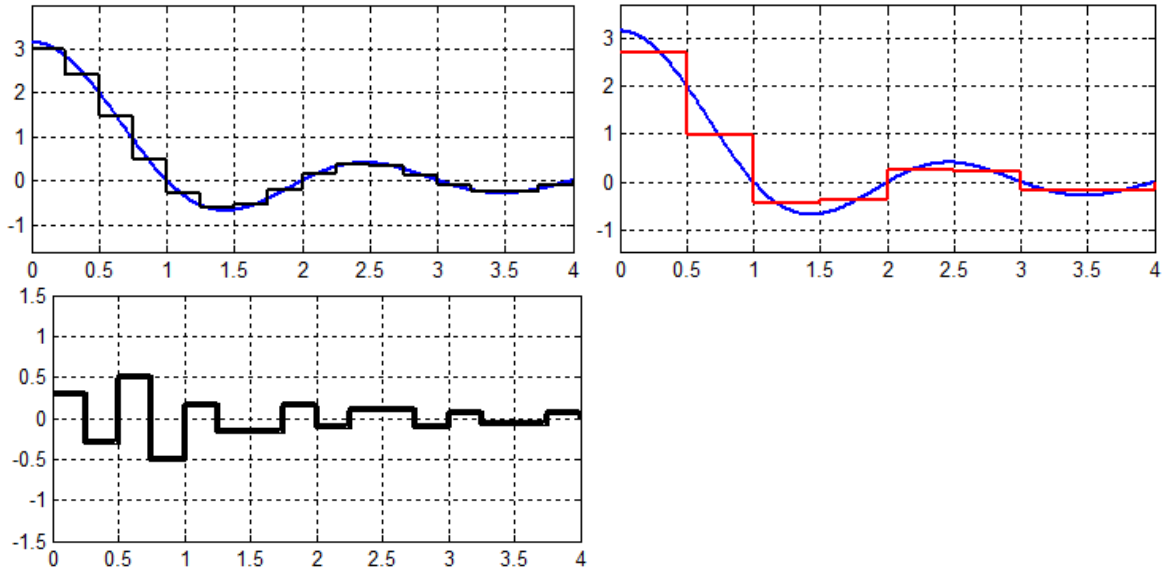
В последних трех строчках кода выполняется проверка – строится график суммы усредняющей и детализирующей функций. Он полностью совпал с первым графиком. □

Пример 2. Рассмотрим на отрезке [0,4] кусочно – постоянную функцию f, имеющую 16 участков и приближающую функцию $\frac{\sin \pi x}{x}$. Построим для f усредняющую и детализирующую функции (ch05\ch05ex06.m)

```
% один шаг вейвлет преобразования функции sin(pi*x)/x
close all; clear all;
xx=0:0.01:4; yy=sin(pi.*xx)./xx;  % значения для графика кривой

% вектор значений приближающей кусочно-постоянной функции
xp=0:0.25:4;
yp=sin(pi.*xp)./xp;
yp(1)=pi;         % предел в нуле = pi вместо NaN
yL=yp(1:16);      % левые значения
yR=yp(2:17);      % правые значения
f=(yL+yR)/2;      % значения в середине интервалов
n=length(f);      % length(f) - должно быть степенью двойки
f=f./2;           % преобразуем f к нормированным psiNormHaar
```

```
%  график приближающей кусочно - постоянной функции f
x=0:0.0025:4;
y=0;
for i=1:n
    y=y+f(i).*psiNormHaar(x,2,i-1);
end
plot(xx,yy,'b-',x,y,'k-','LineWidth',2);
axis([min(x) max(x) min(y)-1 max(y)+1]); grid on;

[a,d]=w1step(f);          % первый шаг вейвлет - преобразования

% график усредняющей кусочно - постоянной функции
g=0;
for i=1:n/2
    g=g+a(i).*psiNormHaar(x,1,i-1);
end
figure,plot(xx,yy,'b-',x,g,'r-','LineWidth',2);
axis([min(x) max(x) min(g)-1 max(g)+1]); grid on;
%  график детализирующей кусочно - постоянной функции
h=0;
for i=1:n/2
    h=h+d(i).*ksiNormHaar(x,1,i-1);
end
figure,plot(x,h,'k-','LineWidth',3);
axis([min(x) max(x) min(h)-1 max(h)+1]); grid on;
% проверка - складываем усредняющую и детализирующую функции
ys=g+h;
figure,plot(xx,yy,'b-',x,ys,'k-','LineWidth',2);
axis([min(x) max(x) min(ys)-1 max(ys)+1]); grid on;
```

Ниже на первом рисунке показана исходная функция и ее кусочно – постоянное приближение; на втором – график усредняющей функции; на третьем – график детализирующей функции.



В последних трех строчках кода выполняется проверка – строится график суммы усредняющей и детализирующей функций. Он полностью совпал с первым графиком.

Заметим, что при разложении приближения кусочно – постоянной функции *f* мы использовали базисные масштабирующие функции $\phi_{2,j}$, поскольку их носители имеют длину 0.25. Эта длина нужна, чтобы на отрезке [0,4] было 16 точек вектора *f*. По этой же причине мы масштабировали вектор значений в строке `f=f./2`, которая является следствием формулы $\tilde{x}_j = x_j / \sqrt{2^n}$ при *n*=2.

□

На первом шаге ВП мы используем массив значений $\{\tilde{x}_i\}, \tilde{x}_j = x_j / \sqrt{2^n}$ кусочно постоянной функции (1). В результате одного шага вейвлет преобразования мы получаем два массива коэффициентов $\{a_{n-1,j}\}$ и $\{d_{n-1,j}\}$, входящих в разложение (2) этой же функции. Коэффициенты $\{d_{n-1,j}\}$ сохраняем, а функцию g_{n-1} (см. (3), она содержит только новые коэффициенты $\{a_{n-1,j}\}$ и является грубым приближением исходной функции) рассматриваем как исходную и подвергаем ее тому же преобразования. После этого мы получаем новые массивы коэффициентов $\{a_{n-2,j}\}$ и $\{d_{n-2,j}\}$. Опять коэффициенты $\{d_{n-2,j}\}$ сохраняем, а последовательность $\{a_{n-2,j}\}$ рассматриваем как коэффициенты разложения по функциям $\phi_{n-2,j}(t)$ новой функции g_{n-2}(t). Она является еще более грубым приближением исходной функции и которую мы подвергаем снова тому же преобразованию. Этот процесс продолжаем до тех пор, пока массивы $\{a_{0,j}\}$ и $\{d_{0,j}\}$ не станут одноэлементными (т.е. они будут состоять из элементов $a_{0,0}$ и $d_{0,0}$ соответственно).

Коэффициенты разложения при переходе от *k*-го уровня к $k-1$ изменяются следующим образом

$$a_{k-1,j} = \frac{a_{k,2j} + a_{k,2j+1}}{\sqrt{2}}, \quad j = 0,1,\ldots,2^{k-1} \tag{7}$$

$$d_{k-1,j} = \frac{a_{k,2j} - a_{k,2j+1}}{\sqrt{2}}, \quad j = 0,1,\ldots,2^{n-1} \tag{8}$$

Они полностью совпадают с формулами (5), (6), если начальную последовательность коэффициентов $\{\tilde{x}_j\}_{j=1}^{2^m}$ обозначить $\{a_{n,j}\}_{j=0}^{2^n-1}$, т.е. $a_{n,j} = \tilde{x}_{j+1}$, $j = 0,1,\ldots 2^{n-1}-1$ (здесь индекс j у $a_{n,j}$ на единицу меньше, чем у соответствующего ему $\tilde{x}_j$).

Расставляя коэффициенты перед соответствующими базисными функциями и складывая, получаем следующее представление исходной кусочно – постоянной функции

$$\begin{aligned} f(t) = a_{0,0}\phi_{0,0}(t) + d_{0,0}\psi_{0,0}(t) + d_{1,0}\psi_{1,0}(t) + d_{1,1}\psi_{1,1}(t) + \\ + d_{2,0}\psi_{2,0}(t) + \ldots + d_{2,3}\psi_{2,3}(t) + \ldots + \\ + d_{n-1,0}\,\psi_{n-1,0}(t) + \ldots + d_{n-1,2^{n-1}-1}\psi_{n-1,2^{n-1}-1}(t) \end{aligned} \tag{9}$$

Это есть разложение функции $f(t)$ по вейвлетам Хаара. При n=2 мы получаем уже записанное нами ранее представление (1.12).

Пример 3. Выполним вейвлет преобразование функции $\sin(2\pi x)$ на отрезке [0,1]. Для этого выполним многократно одношаговое преобразование. На первом шаге в качестве аргумента функции `[a,d]=w1step(f)` используем массива векторов $\{\tilde{x}_i\}$. Значения коэффициентов `d` сохраняем, а коэффициенты `a` передаем функции `[a,d]=w1step(a)` в качестве аргументов для выполнения следующего шага вейвлет преобразования. Эту процедуру выполняем до тех пор, пока вектор `a` коэффициентов не станет одноэлементным (ch05\ch05ex07.m).

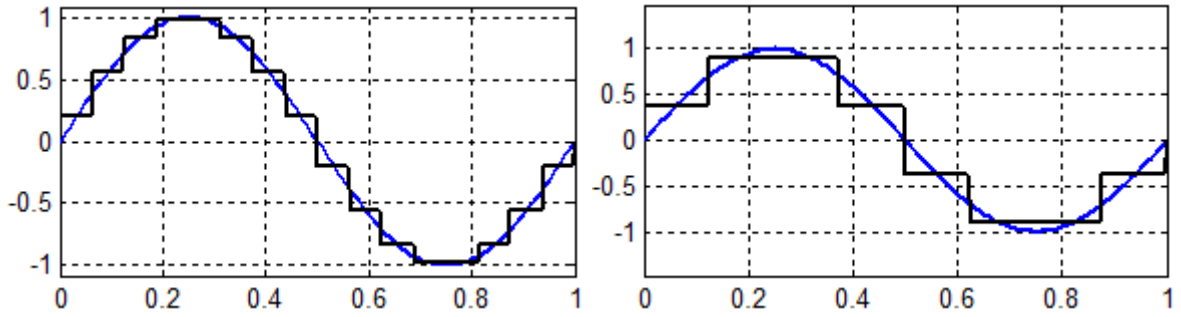
```
% разложение функции sin(2*pi*x) по вейвлетам Хаара
close all; clear all;
xx=0:0.01:1; yy=sin(2*pi.*xx);  % значения для графика кривой
% построение исходной кусочно-постоянной функции
n=4;        % назначаем
N=2^n;      % количество интервалов (степень
step=1/N;
xf=step/2:step:1-step/2; % точки в серединах интервалов
yf=sin(2*pi.*xf);
f=yf./sqrt(N);     % для разложения по нормированным psi
x=0:0.005:1;
y=0;
for i=1:N
    y=y+f(i).*psiNormHaar(x,n,i-1);
end
plot(xx,yy,'b-',x,y,'k-','LineWidth',2);
axis([min(x) max(x) min(y)-0.1 max(y)+0.1]); grid on;

% определение коэффициентов вейвлет разложения
wd=zeros(1,N);
[a,d]=w1step(f);   % первый шаг вейвлет - преобразования
for i=n-1:-1:0
    wd(2^i+1:2^(i+1))=d;
    if length(a)==1 , break, end  % выход при одноэлементном a
    [a,d]=w1step(a);
end
wd(1)=a;
disp(wd);    % вывод коэффициентов вейвлет разложения

% вычисление функции во всех точках вектора x с использованием
% коэффициентов вейвлет разложения и построение графика
w=wd(1);
k=2;
for i=0:n-1
    for j=0:2^i-1
        w=w+wd(k).*ksiNormHaar(x,i,j);
        k=k+1;
    end
end
```

```
figure,plot(xx,yy,'b-',x,w,'k-','LineWidth',2);  % график слева
axis([min(x) max(x) min(y)-0.1 max(y)+0.1]); grid on;
% отбрасываем самые мелкие детализирующие коэф.(при i=n-1)
k=2;
w=wd(1);
for i=0:n-2    % до n-2, а не как ранее до n-1
    for j=0:2^i-1
        w=w+wd(k).*ksiNormHaar(x,i,j);
        k=k+1;
    end
end
figure,plot(xx,yy,'b-',x,w,'k-','LineWidth',2); % график справа
axis([min(x) max(x) min(y)-0.5 max(y)+0.5]); grid on;
```

Вот значения коэффициентов вейвлет разложения при n=4.

```
0         0.6407    -0.0000    0.0000    -0.1327    0.1327
0.1327   -0.1327    -0.0637   -0.0264    0.0264     0.0637
0.0637    0.0264    -0.0264   -0.0637
```

Ниже слева показан график ступенчатой функции, полученной разложением по всем вейвлет – функциям, справа – график ступенчатой функции, полученной отбрасыванием части разложения, содержащей первую детализирующую функцию (самые мелкие детали).



□

Проясним вейвлет преобразование, используя символику прямой суммы векторных пространств.

На первом шаге мы имеем функцию $f(t)\in V^n = V^{n-1}\oplus W^{n-1}$. Мы разлагаем функцию $f(t)$ в сумму двух функций $g_{n-1}(t)\in V^{n-1}$ и $w_{n-1}(t)\in W^{n-1}$ так, что $f(t)=g_{n-1}(t)+w_{n-1}(t)$. На втором шаге функцию $g_{n-1}(t)\in V^{n-1}=V^{n-2}\oplus W^{n-2}$ мы разлагаем в сумму двух функций $g_{n-2}(t)\in V^{n-2}$ и $w_{n-2}(t)\in W^{n-2}$ таких, что $g_{n-1}(t)=g_{n-2}(t)+w_{n-2}(t)$. Фактически для исходной функции $f(t)\in V^n = V^{n-2}\oplus W^{n-2}\oplus W^{n-1}$ на втором шаге мы получаем представление в виде $f(t)=g_{n-2}(t)+w_{n-2}(t)+w_{n-1}(t)$. Продолжая этот процесс далее, можем сказать, что для функции $f(t)\in V^n = V^0\oplus W^0\oplus W^1\oplus\ldots\oplus W^{n-1}$ мы получаем разложение

$$f(t)=g_0(t)+w_0(t)+\ldots+w_{n-2}(t)+w_{n-1}(t) \qquad (10)$$

Для каждой функции последней суммы мы имеем

$$g_0(t)=a_{0,0}\phi_{0,0}(t)$$
$$w_0(t)=d_{0,0}\psi_{0,0}(t)$$
$$w_1(t)=d_{1,0}\psi_{1,0}(t)+d_{1,1}\psi_{1,1}(t)$$
$$w_2(t)=d_{2,0}\psi_{2,0}(t)+d_{2,1}\psi_{2,1}(t)+d_{2,2}\psi_{2,2}(t)+d_{2,3}\psi_{2,3}(t)$$
$$\ldots$$
$$w_{n-1}(t)=d_{n-1,0}\,\psi_{n-1,0}(t)+\ldots+d_{n-1,2^{n-1}-1}\psi_{n-1,2^{n-1}-1}(t)$$

Подставляя эти выражения в (10), мы приходим к формуле (9).

Для дискретных последовательностей вейвлет преобразованием называется переход от последовательности коэффициентов $\{\tilde{x}_i\}, \tilde{x}_j = x_j/\sqrt{2^n}$ к последовательности

$$\{a_{0,0}, d_{0,0}, d_{1,0}, d_{1,1}, d_{2,0},\ldots,d_{n-1,0},\ldots,d_{n-1,2^{n-1}-1}\} \qquad (11)$$

Количество элементов этой последовательности равно $1+1+2+2^2+\ldots+2^{n-1}=2^n$, столько же, сколько элементов в исходной последовательности $\{x_i\}$.

Часто вейвлет преобразование останавливают на некотором шаге, не доводя его до последнего коэффициента $a_{0,0}$. Например, вместо последовательности (11) получают последовательность

$$\{a_{k,0}, a_{k,1},\ldots,a_{k,2^k-1}, d_{k,0},\ldots,d_{k,2^k-1},\ldots,d_{n-1,0},\ldots,d_{n-1,2^{n-1}-1}\} \qquad (12)$$

которая также содержит

$$2^k+2^k+2^{k+1}+\ldots+2^{n-1}=2^k+2^k\left(1+2+2^2+\ldots+2^{n-k-1}\right)=2^k+2^k\frac{1-2^{n-k}}{1-2}=2^n$$

элементов. Последовательность (12) используется тогда, когда заранее известно, что самое грубое приближение, которое может быть использовано, представляется коэффициентами $\{a_{k,0}, a_{k,1},\ldots,a_{k,2^k-1}\}$.

Процесс разложения/декомпозиции дискретной последовательности в средние $a_{k,j}$ и детализирующие $d_{k,j}$ значения при различных масштабах детализации *k* называется кратномасштабным анализом.

Напишем функцию `wtransform`, которая выполняет многошаговый процесс вейвлет преобразования. Ее аргументами являются вектор – строка исходных коэффициентов $\{a_{n,j}=\tilde{x}_{j+1}\}$ и необязательный номер уровня *k* детализации, на котором следует остановиться. Если *k* опущен, то он равен нулю (ch05\wtransform.m).

```
% Многошаговый процесс вейвлет преобразования
% Аргументы:
% вектор строка исходных коэффициентов f с length(f)=2^n
% необязательный номер уровня k детализации
% Возвращает последовательности a и d
function [a,d]=wtransform(f,varargin)
N=length(f); % количество значений
n=log2(N);
```

```
if mod(n,1)~=0, error('Длина вектора f должна быть степенью
двойки'), end
nargs=length(varargin);
if nargs==0
    k=0;
elseif nargs==1
    k=varargin{1};
else
    error('Слишком много аргументов');
end;
if mod(k,1)~=0, error('2-й аргумент должен быть целым числом');
end
if (k<0 || k>n) , error('Неверно задан 2-й аргумент'); end
if k==n    % ничего не делаем
    a=f;
    d=[];
    return;
end
% определение коэффициентов вейвлет разложения
wd=zeros(1,N);
[aa,dd]=w1step(f);   % первый шаг вейвлет - преобразования
for i=n-1:-1:k
    wd(2^i+1:2^(i+1))=dd;
    if length(aa)==2^k , break, end  % не вызывать w1step с
одноэлементным a
    [aa,dd]=w1step(aa);
end
wd(1:2^k)=aa;
a=wd(1:2^k);
d=wd(2^k+1:N);
```

Здесь мы привели несложный код, реализующий по полученным ранее формулам (5) и (6), многократное применение одношагового вейвлет преобразования. Имеются и более быстрые алгоритмы дискретного ВП, в частности, существует алгоритм быстрого вейвлет преобразования. Однако это не является предметом нашего обсуждения в данной главе.

Прежде, чем продемонстрировать работу функции `wtransform`, напишем функцию `wcalc`, которая вычисляет вектор значений кусочно – постоянной функции, заданной последовательностями `a` и `d` в точках некоторого вектора `x` (ch05\wcalc.m).

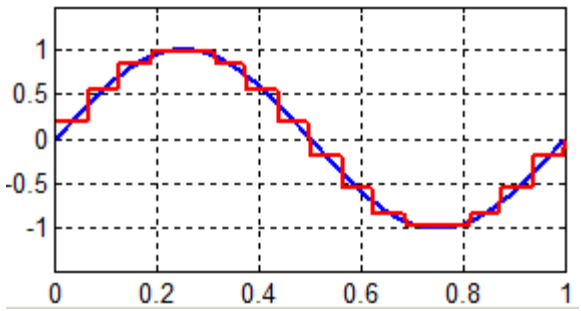
```
% вычисления значений вейвлет ряда [a,d] в массиве точек x
function y=wcalc(x,a,d)
na=log2(length(a));
N=length(a)+length(d); % количество элементов
n=log2(N);
% display(n);
if mod(na,1)~=0, error('Длина вектора a должна быть степенью
двойки'), end
if mod(n,1)~=0, error('Длина вектора d Задана неверно'), end

% значения от усредняющей функции
ya=a(1).*psiNormHaar(x,na,0);
```

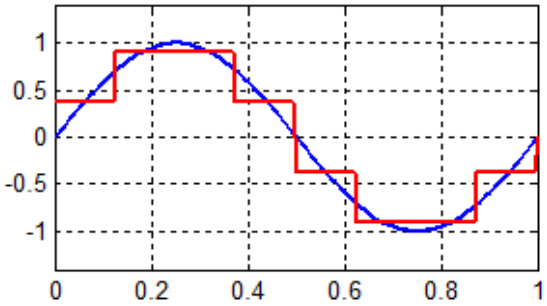
```
for i=2:2^na
    ya=ya+a(i).*psiNormHaar(x,na,i-1);
end
% значения от детализирующих функций
yd=0;
k=1;
for i=na:n-1
    for j=0:2^i-1
        yd=yd+d(k).*ksiNormHaar(x,i,j);
        k=k+1;
    end
end
y=ya+yd;     % результирующие значения
```

Пример 4. Продемонстрируем использование наших функций на примере кусочно – постоянной функции $f(x)$, приближающей функцию $\sin(2\pi x)$ на отрезке [0,1]. Зададим функцию $f(x)$ на 16 участках, как в примере 3 этого параграфа (ch05\ch05ex08.m).

```
close all; clear all;
xx=0:0.01:1; yy=sin(2*pi.*xx);  % значения для графика кривой
% построение исходной кусочно-постоянной функции
n=5;         % назначаем
N=2^n;       % количество интервалов (степень
step=1/N;
xf=step/2:step:1-step/2; % точки в серединах интервалов
yf=sin(2*pi.*xf);
f=yf./sqrt(N);      % для разложения по нормированным psi
[a,d]= wtransform(f,0);  % полное вейвлет преобразование
x=0:0.005:1;
y=wcalc(x,a,d);   % вычисление значений функции по вейвлет
коэффициентам
figure,plot(xx,yy,'b-',x,y,'r-','LineWidth',2);
axis([min(x) max(x) min(y)-0.5 max(y)+0.5]); grid on;
set(gcf,'Color',[1 1 1]);
```

Следующий график слева построен этим сценарием.





Если мы хотим отбросить самые мелкие детали, то мы можем выполнить следующие команды (продолжение сценария ch05\ch05ex08.m).

```
[a,d]= wtransform(f,3)  % Один шаг вейвлет преобразования
y=wcalc(x,a,[]);
figure,plot(xx,yy,'b-',x,y,'r-','LineWidth',2); % рис. справа
axis([min(x) max(x) min(y)-0.5 max(y)+0.5]); grid on;
```

Предыдущий рисунок справа.

□

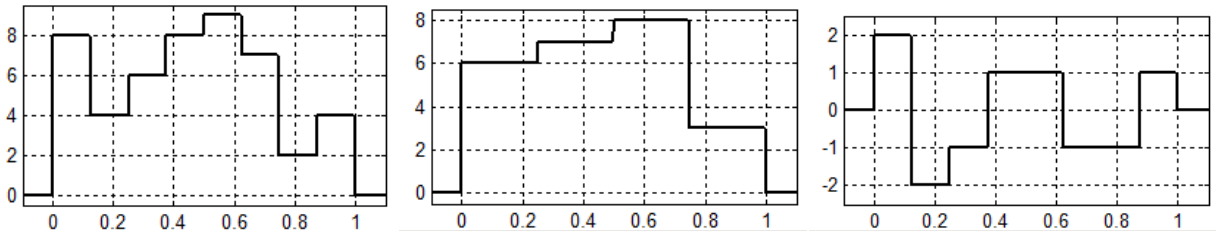
Пример 5. Продемонстрируем использование наших функций на примере кусочно – постоянной функции $f(x)$, заданной вектором в примере 1 (ch05\ch05ex09.m).

```
close all; clear all;
f=[8 4 6 8 9 7 2 4];% length(f) - должно быть степенью двойки
N=length(f);
f=f./sqrt(N);  % преобразуем f к нормированным psi
k=2;  % уровень детализации
[a,d]= wtransform(f,k)  % один шаг вейвлет преобразования
x=-0.1:0.005:1.1;
y=wcalc(x,a,d); % вычисл.знач. функции по вейвлет коэффициентам
figure,plot(x,y,'k-','LineWidth',2);
axis([min(x) max(x) min(y)-0.5 max(y)+0.5]); grid on;
y=wcalc(x,a,[]);  % усредняющая функция
figure,plot(x,y,'k-','LineWidth',2);
axis([min(x) max(x) min(y)-0.5 max(y)+0.5]); grid on;
a=zeros(1,2^k);  % обнуляем коэффициенты усредняющей функции
y=wcalc(x,a,d);  % детализирующая функция
figure,plot(x,y,'k-','LineWidth',2);
axis([min(x) max(x) min(y)-0.5 max(y)+0.5]); grid on;
```

Получаем следующие значения коэффициентов a и d:

```
a =    3.0000    3.5000    4.0000    1.5000
d =    1.0000   -0.5000    0.5000   -0.5000
```

Ниже на первом графике показана исходная кусочно – постоянная функция `f`; на втором – график усредняющей функции; на третьем – график детализирующей функции. Все полностью совпадает с результатами примера 1.



□

5.4 Обратное вейвлет - преобразование

Обращение вейвлет преобразования состоит в определении исходной последовательности $\{x_1, x_2, \ldots, x_{2^n}\}$ из последовательности коэффициентов (3.11) $\{a_{0,0}, d_{0,0}, d_{1,0}, d_{1,1}, d_{2,0}, \ldots, d_{n-1,0}, \ldots, d_{n-1,2^{n-1}-1}\}$ или (3.12).

Как мы видели, один шаг вейвлет преобразования, заключающийся в переходе от k-го уровня к $k-1$, выглядит следующим образом (см. формулы (3.7) и (3.8))

$$a_{k-1,j} = \frac{a_{k,2j} + a_{k,2j+1}}{\sqrt{2}}, \quad j = 0,1,\ldots,2^{k-1}$$

$$d_{k-1,j} = \frac{a_{k,2j} - a_{k,2j+1}}{\sqrt{2}}, \quad j = 0,1,\ldots,2^{n-1}$$

Тогда

$$a_{k,2j} = \frac{a_{k-1,j} + d_{k-1,j}}{\sqrt{2}}$$

$$a_{k,2j+1} = \frac{a_{k-1,j} - d_{k-1,j}}{\sqrt{2}}$$

Т.о., если есть последовательность (3.12) на $(k-1)$ – ом уровне детализации

$$\{a_{k-1,0},...,a_{k-1,2^{k-1}-1}, d_{k-1,0},...,d_{k-1,2^{k-1}-1}, d_{k,0},...,d_{n-1,0},...,d_{n-1,2^{n-1}-1}\}$$

то один шаг назад будет состоять в выборе первых 2^k элементов

$$\{a_{k-1,0}, a_{k-1,1},...,a_{k-1,2^{k-1}-1}, d_{k-1,0},...,d_{k-1,2^{k-1}-1}\}$$

и замене их последовательностью

$$\left\{\frac{a_{k-1,0} + d_{k-1,0}}{\sqrt{2}}, \frac{a_{k-1,0} - d_{k-1,0}}{\sqrt{2}},..., \frac{a_{k-1,2^{k-1}-1} - d_{k-1,2^{k-1}-1}}{\sqrt{2}}\right\},$$

которую мы обозначаем $\{a_{k,0}, a_{k,1},...,a_{k,2^k-1}\}$. В результате мы переходим к последовательности

$$\{a_{k,0},...,a_{k,2^k-1}, d_{k,0},...,d_{k,2^k-1},...,d_{n-1,0},...,d_{n-1,2^{n-1}-1}\},$$

которая представляет k – й уровень детализации.

Напишем функцию `iwtransform`, выполняющую обратное вейвлет преобразование (ОВП). Она использует 3 параметра. Первые два – это последовательности `a` и `d`. Третий задает приращения уровня детализации относительно существующего уровня (количество обратных шагов вейвлет преобразования). Если этот аргумент опущен, то принимается равным 1. Функция возвращает новые значения последовательностей `a` и `d`. (ch05\ iwtransform.m)

```
% Многошаговый процесс обратного вейвлет преобразования
% Аргументы: последовательности a и d с length(a)=2^n
% необязательный индекс приращения уровня детализации k
% k= varargin; если k=0 оставляем все без изменений,
% если k=1 выполняем один шаг ОВП, и т.д.
% Возвращает последовательности a и d
function [ar,dr]=iwtransform(a,d,varargin)
if isempty(d)
    ar=a;
    dr=d;
    return;
end
La=length(a); na=log2(La);
if mod(na,1)~=0
     error('Длина вектора a должна быть степенью двойки')
end
Ld=length(d);
n=log2(Ld+La);
if mod(n,1)~=0, error('Длина вектора d задана неверно'), end
if n<na, error('Длина вектора d задана неверно'), end
```

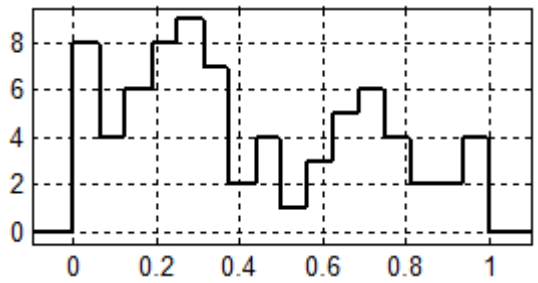
```
% случай n=na соответствует isempty(d)=1 и уже рассмотрен
nargs=length(varargin);
if nargs==0
    k=1;
elseif nargs==1
    k=varargin{1};
else
    error('Слишком много аргументов');
end;
if mod(k,1)~=0
    error('3-й аргумент должен быть целым числом');
end

if (k<0) , error('Неверно задан 3-й аргумент'); end

if ( k>n-na)
    k=n-na;
    warning('MATLAB:msg1','3-й аргумент выбран макс. %d',k);
end

for p=1:k
    [a,d]=iw1step(a,d);
end
ar=a;
dr=d;

% подфункция предполагает, что длины векторов a и d заданы
корректно
% один шаг назад вейвлет преобразования
function [ar,dr]=iw1step(a,d)
if isempty(d)
    ar=a;
    dr=d;
    return;
end
La=length(a);
aa=zeros(1,2*La);
j=0;
for i=1:La
    aa(2*j+1)=(a(j+1)+d(j+1))/sqrt(2);
    aa(2*j+2)=(a(j+1)-d(j+1))/sqrt(2);
    j=j+1;
end
ar=aa;
dr=d(La+1:length(d));
```

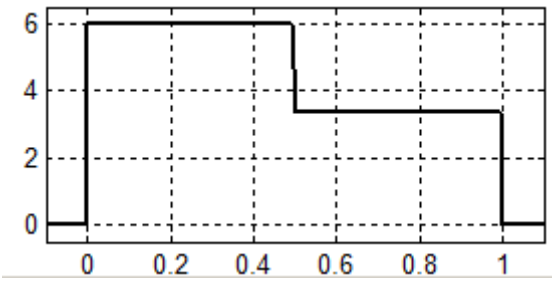
Пример. Выполним обратное вейвлет преобразование (ОВП) для числовой последовательности, используя созданную функцию `iwtransform` (ch05\ch05ex11.m)

```
% тестирование функции iwtransform
close all; clear all;
```

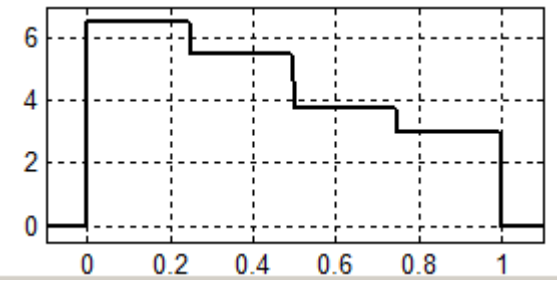
```
f=[8 4 6 8 9 7 2 4 1 3 5 6 4 2 2 4];% length(f) - должно быть
степенью двойки
N=length(f);
f=f./sqrt(N);  % преобразуем f к нормированным psiNormHaar
x=-0.1:0.005:1.1;

kw=1;
[a,d]= wtransform(f,kw);
y=wcalc(x,a,d); % вычисл. знач. функции по вейвлет коэф.
figure,plot(x,y,'k-','LineWidth',2);    % 1
axis([min(x) max(x) min(y)-0.5 max(y)+0.5]); grid on;
set(gcf,'Color',[1 1 1]);

ym=wcalc(x,a,[]);  % усредняющая функция при выбранном kw
figure,plot(x,ym,'k-','LineWidth',2);   % 2
axis([min(x) max(x) min(ym)-0.5 max(ym)+0.5]); grid on;
set(gcf,'Color',[1 1 1]);

kiw=1;
[a,d]=iwtransform(a,d,kiw)  % обратное вейвлет преобразование

ym=wcalc(x,a,[]);  % усредняющая функция после обращения
figure,plot(x,ym,'k-','LineWidth',2);   % 3
axis([min(x) max(x) min(ym)-0.5 max(ym)+0.5]); grid on;
set(gcf,'Color',[1 1 1]);

a=zeros(1,2^(kiw+kw));
yd=wcalc(x,a,d);     % детализирующая функция после обращения
figure,plot(x,yd,'k-','LineWidth',2);    % 4
axis([min(x) max(x) min(yd)-0.5 max(yd)+0.5]); grid on;
set(gcf,'Color',[1 1 1])

y=ym+yd;
figure,plot(x,y,'k-','LineWidth',2);    % 5
axis([min(x) max(x) min(y)-0.5 max(y)+0.5]); grid on;
set(gcf,'Color',[1 1 1])
```

На следующем рисунке показана исходная функция, построенная функцией `plot` с комментарием %1. На втором графике показана усредняющая функция, построенная по полученной последовательности `a`.





На следующем графике показана усредняющая функция, построенная по последовательности `a`, полученной после одного шага ОВП. Она создается функцией `plot`, помеченной комментарием %3.



□

5.5 Вейвлеты при обработке изображений

Пусть дана матрица монохромного изображения

$$\left[x_{i,j} \right], i = 1,\ldots,2^n;\ j = 1,\ldots,2^n \tag{1}$$

размера $2^n \times 2^n$. Ее можно представить как кусочно постоянную функцию $f(s,t)$ двух переменных, заданную на единичном квадрате $[0,1]\times[0,1]$ плоскости (s,t),

$$f(s,t) = \sum_{i=1}^{2^n} \sum_{j=1}^{2^n} x_{i,j} H_{I_i \times I_j}(s,t) \tag{2}$$

Здесь

$$I_i \times I_j = \left[\frac{i-1}{2^n}, \frac{i}{2^n} \right] \times \left[\frac{j-1}{2^n}, \frac{j}{2^n} \right] = \left\{ (s,t) : s \in \left[\frac{i-1}{2^n}, \frac{i}{2^n} \right),\ t \in \left[\frac{j-1}{2^n}, \frac{j}{2^n} \right) \right\}$$

и

$$H_{I_i \times I_j}(s,t) = \begin{cases} 1, (s,t) \in I_i \times I_j \\ 0, (s,t) \notin I_i \times I_j \end{cases} = H_{I_i}(s) H_{I_j}(t) = \frac{\phi_{n,i-1}(s)}{\sqrt{2^n}} \frac{\phi_{n,j-1}(t)}{\sqrt{2^n}} \tag{3}$$

Введенные здесь обозначения показывают, что параметр s откладывается по вертикали, также как и индекс i, который нумерует строки матрицы $x_{i,j}$. Подставляя (3) в (2), получаем

$$f(s,t) = \frac{1}{2^n} \sum_{i=1}^{2^n} \sum_{j=1}^{2^n} x_{i,j} \phi_{n,i-1}(s) \phi_{n,j-1}(t) = \frac{1}{2^n} \sum_{i=1}^{2^n} \left\{ \sum_{j=1}^{2^n} x_{i,j} \phi_{n,j-1}(t) \right\} \phi_{n,i-1}(s) =$$

$$= \frac{1}{2^n} \sum_{i=1}^{2^n} z_i(t) \phi_{n,i-1}(s) \tag{4}$$

где

$$z_i(t) = \sum_{j=1}^{2^n} x_{i,j} \phi_{n,j-1}(t) \tag{5}$$

Для каждого i уравнение (5) выглядит также как (3.1) и можно выполнить один шаг одномерного ВП. Получаем новый набор уравнений для $z_i(t), i = 1,\ldots,2^n$ вида (см. формулу (3.2))

$$z_i(t) = \sum_{j=0}^{2^{n-1}-1} a^i_{n-1,j} \phi_{n-1,j}(t) + \sum_{j=0}^{2^{n-1}-1} d^i_{n-1,j} \psi_{n-1,j}(t) \tag{6}$$

Для каждого i это соответствует применению одного шага одномерного ВП к i – ой строке исходного изображения (1). Теперь подставим (6) обратно в (4) и переупорядочим члены

$$f(s,t)=\frac{1}{2^n}\sum_{i=1}^{2^n} z_i(t)\phi_{n,i-1}(s)=\frac{1}{2^n}\sum_{i=1}^{2^n}\left(\sum_{j=0}^{2^{n-1}-1} a^i_{n-1,j}\phi_{n-1,j}(t)+\sum_{j=0}^{2^{n-1}-1} d^i_{n-1,j}\psi_{n-1,j}(t)\right)\phi_{n,i-1}(s)=$$

$$=\frac{1}{2^n}\left(\sum_{j=0}^{2^{n-1}-1}\left\{\sum_{i=1}^{2^n} a^i_{n-1,j}\phi_{n,i-1}(s)\right\}\phi_{n-1,j}(t)+\sum_{j=0}^{2^{n-1}-1}\left\{\sum_{i=1}^{2^n} d^{\,i}_{n-1,j}\phi_{n,i-1}(s)\right\}\psi_{n-1,j}(t)\right) \qquad (7)$$

$$=\frac{1}{2^n}\left(\sum_{j=0}^{2^{n-1}-1}\alpha_j(s)\phi_{n-1,j}(t)+\sum_{j=0}^{2^{n-1}-1}\beta_j(s)\psi_{n-1,j}(t)\right),$$

где $\alpha_j(s)=\sum_{i=1}^{2^n} a^i_{n-1,j}\phi_{n,i-1}(s)$ и $\beta_j(s)=\sum_{i=1}^{2^n} d^{\,i}_{n-1,j}\phi_{n,i-1}(s)$. Выражения для $\alpha_j(s)$ и $\beta_j(s)$ при фиксированном j опять похожи на (3.1) и к ним тоже может быть применено одномерное ВП. Это эквивалентно применению одномерного ВП к каждому столбцу преобразованного изображения. Выполняя это, получаем

$$\alpha_j(s)=\sum_{i=1}^{2^n} a^i_{n-1,j}\phi_{n,i-1}(s)=\sum_{i=0}^{2^{n-1}-1}\tilde{a}^{\,j}_{n-1,i}\phi_{n-1,i}(s)+\sum_{i=0}^{2^{n-1}-1}\tilde{d}^{\,j}_{n-1,i}\psi_{n-1,i}(s)$$

$$\beta_j(s)=\sum_{i=1}^{2^n} d^{\,i}_{n-1,j}\phi_{n,i-1}(s)=\sum_{i=0}^{2^{n-1}-1}\tilde{\tilde{a}}^{\,j}_{n-1,i}\,\phi_{n-1,i}(s)\;+\sum_{i=0}^{2^{n-1}-1}\tilde{\tilde{d}}^{\,j}_{n-1,i}\psi_{n-1,i}(s)$$

Тогда

$$f(s,t)=\frac{1}{2^n}\left(\sum_{j=0}^{2^{n-1}-1}\alpha_j(s)\phi_{n-1,j}(t)+\sum_{j=0}^{2^{n-1}-1}\beta_j(s)\psi_{n-1,j}(t)\right)=$$

$$=\frac{1}{2^n}\left(\sum_{j=0}^{2^{n-1}-1}\left\{\sum_{i=0}^{2^{n-1}-1}\tilde{a}^{\,j}_{n-1,i}\phi_{n-1,i}(s)+\sum_{i=0}^{2^{n-1}-1}\tilde{d}^{\,j}_{n-1,i}\psi_{n-1,i}(s)\right\}\phi_{n-1,j}(t)+\right.$$

$$\left.+\sum_{j=0}^{2^{n-1}-1}\left\{\sum_{i=0}^{2^{n-1}-1}\tilde{\tilde{a}}^{\,j}_{n-1,i}\,\phi_{n-1,i}(s)\;+\sum_{i=0}^{2^{n-1}-1}\tilde{\tilde{d}}^{\,j}_{n-1,i}\psi_{n-1,i}(s)\right\}\psi_{n-1,j}(t)\right)$$

После перестановки слагаемых, получаем

$$f(s,t)=\sum_{i=0}^{2^{n-1}-1}\sum_{j=0}^{2^{n-1}-1} a^{n-1}_{i,j}\,\phi_{n-1,j}(t)\phi_{n-1,i}(s)+\sum_{i=0}^{2^{n-1}-1}\sum_{j=0}^{2^{n-1}-1} h^{n-1}_{i,j}\,\phi_{n-1,j}(t)\psi_{n-1,i}(s)+$$

$$+\sum_{i=0}^{2^{n-1}-1}\sum_{j=0}^{2^{n-1}-1} v^{n-1}_{i,j}\,\psi_{n-1,j}(t)\phi_{n-1,i}(s)+\sum_{i=0}^{2^{n-1}-1}\sum_{j=0}^{2^{n-1}-1} d^{n-1}_{i,j}\,\psi_{n-1,j}(t)\psi_{n-1,i}(s), \qquad (8)$$

где $a^{n-1}_{i,j}=\frac{1}{2^n}\tilde{a}^{\,j}_{n-1,i}$, $h^{n-1}_{i,j}=\frac{1}{2^n}\tilde{d}^{\,j}_{n-1,i}$, $v^{n-1}_{i,j}=\frac{1}{2^n}\tilde{\tilde{a}}^{\,j}_{n-1,i}$, $d^{n-1}_{i,j}=\frac{1}{2^n}\;\tilde{\tilde{d}}^{\,j}_{n-1,i}$. В результате функция $f(x,y)$ раскладывается по функциям $\phi_{n-1,j}(t)\phi_{n-1,i}(s)$, $\phi_{n-1,j}(t)\psi_{n-1,i}(s)$, $\psi_{n-1,j}(t)\phi_{n-1,i}(s)$ и $\psi_{n-1,j}(t)\psi_{n-1,i}(s)$.

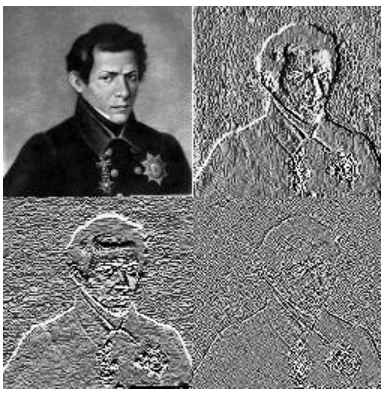
Таким образом, один шаг двумерного ВП заключается в применении одного шага одномерного ВП к каждой строке изображения (1), а затем – в

применении одного шага одномерного ВП к каждому столбцу матрицы с преобразованными строками.

Пример. Применим один шаг двумерного ВП к изображению (ch05\ch05ex13.m)

```
% один шаг двумерного ВП
close all; clear all;
f=im2double(imread('Lobachevsky01.tif'));   % 256 x 256
% f=im2double(imread('lenaTest1.jpg'));
imshow(f,[]);
[M,N]=size(f);
n=log2(M);
p=1;  % количество шагов ВП
k=n-p;
% не выполняем проверки корректности M,N,n,k
a=zeros(M,N/2^p);
d=zeros(M,2^n-N/2^p);
for i=1:M
    [a(i,:),d(i,:)]=wtransform(f(i,:),k);
end

ff=cat(2,a,d)';  % матрица коэффициентов a,d
fa=zeros(M,N/2^p);
fd=zeros(M,2^n-N/2^p);
for i=1:M
    [fa(i,:),fd(i,:)]=wtransform(ff(i,:),k);
end
fff=cat(2,fa,fd)'/M;
figure,imshow(fff,[]);

% для наглядности выполняем гистограммную эквализацию
% 3-х четвертей матрицы коэффициентов
fff(1:M/2,1:N/2)=mat2gray(fff(1:M/2,1:N/2));
fff(1:M/2,N/2+1:end)=histeq(mat2gray(fff(1:M/2,N/2+1:end)),256);
fff(M/2+1:end,1:N/2)=histeq(mat2gray(fff(M/2+1:end,1:N/2)),256);
fff(M/2+1:end,N/2+1:end)=...
          histeq(mat2gray(fff(M/2+1:end,N/2+1:end)),256);
figure,imshow(fff,[]);
```

На следующем рисунке слева показан результат – представлена матрица коэффициентов как изображение после одного шага двумерного ВП. Черные пиксели представляют маленькие значения коэффициентов *d*. Для наглядности мы приводим справа второе изображение этой матрицы, выполнив гистограммную эквализацию каждой из четвертей полученной матрицы коэффициентов по отдельности.



□

Следует обратить внимание на то, что для выполнения второго (и последующих) шага двумерного ВП нужно использовать только левый верхний квадрант обработанной на предыдущем шаге матрицы.

Прежде, чем рассматривать дальнейшие примеры, напишем код функции, выполняющей несколько шагов двумерного ВП (ch05\w2transform.m).

```
% Многошаговый процесс двумерного вейвлет преобразования
% Аргументы: квадратная матр. изображения f с size(f)=2^n x 2^n
% Необязательный аргумент: p - количество шагов двумерного ВП
% при p=0 ничего не делаем, при p=1 - один шаг ВП, и т.д.
% при p=-1 или аргумент p опущен делаем все шаги ВП до конца
% Возвращает: матрицу коэффициентов приближения
function fw=w2transform(f,varargin)
if ~isempty(varargin)
    nargs=length(varargin);
    if nargs==1
        p=varargin{1};
    else
        error('Слишком много аргументов');
    end;
else
    p=-1;  % ВП все шаги до конца
end
if mod(p,1)~=0, error('2-й аргумент должен быть целым числом');
end
if (p<-1) , error('Неверно задан 2-й аргумент'); end
if p==0, fw=f; return; end

[M,N]=size(f);
if M~=N , error('Изображение f должно быть квадратным'); end
nn=log2(N);
if mod(nn,1)~=0, error('Размер изображения f должен быть степенью
двойки'), end
if p>nn , error('Максимальное значение для 2-го аргумента =
%d',nn); end
if p==-1,  p=nn; end

f1=f;
for n=nn:-1:nn-(p-1)
    % один шаг ВП по строкам
    a=zeros(N,N/2);  d=zeros(N,N/2); % выделяем место
```

```
    for i=1:N
        [a(i,:),d(i,:)]=wtransform(f1(i,:),n-1);
    end
    ff=cat(2,a,d)';  % матрица коэффициентов [a,d]
    % один шаг ВП по столбцам матрицы коэффициентов [a,d]
    fa=zeros(N,N/2); fd=zeros(N,N/2);
    for i=1:N
        [fa(i,:),fd(i,:)]=wtransform(ff(i,:),n-1);
    end
    f1=cat(2,fa,fd)';
    fw(1:N,1:N)=f1;    % запоминаем результат
    % готовимся к следующему шагу цикла
    N=N/2;
    f1=fw(1:N,1:N);
end
```

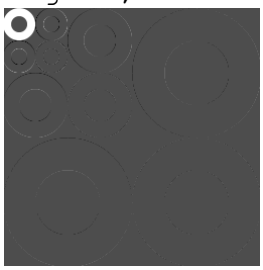
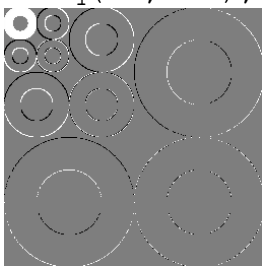
Заметим, что здесь мы дважды использовали функцию `wtransform` одномерного ВП, вначале применив ее к строкам исходного изображения, а затем к столбцам полученной матрицы коэффициентов.

В некоторых последующих примерах мы будем использовать изображение кольца. Нам будет удобно создавать это изображение различного размера с помощью функции `makering(n)`, где параметр `n` определяет размер изображения `N x N` кольца при $N=2^n$ (ch05\ makering.m).

```
function f=makering(n)
M=2^(n-1);
N=2*M;   % N размер изображения
[X,Y]=meshgrid(1:N,1:N);
X=X-M; Y=Y-M;
D=((X.^2+Y.^2)<=M^2) & ((X.^2+Y.^2)>=(M/2)^2);
f=im2double(D);
```

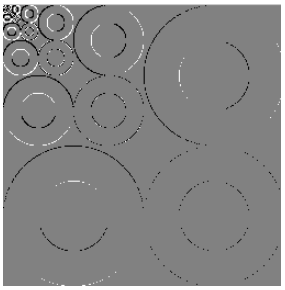
Пример. Выполним несколько шагов двумерного ВП к изображению кольцевой области (ch05\ch05ex17.m), используя созданную выше функцию.

```
% Несколько шагов двумерного ВП над изображением круга
close all; clear all;
f=makering(8);  % изображение круга
imshow(f);
p=3;
fw=w2transform(f,p);  % выполняем 3 шага двумерного ВП
figure,imshow(fw,[]);
% строим обработанное изображение
f4=(fw-min(min(fw)))/(max(max(fw))-min(min(fw)));
figure,imshow(histeq(f4,256),[]);
```

□

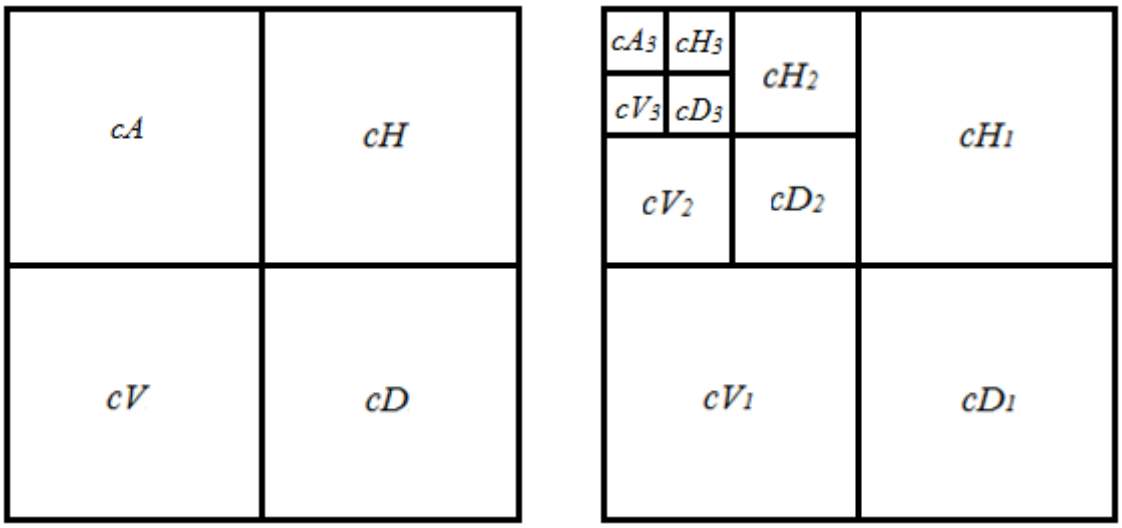
Пример. Если в сценарии ch05\ch05ex17.m положить p=8, то будут выполнены все шаги ВП. Изображение матрицы коэффициентов, построенное последней строкой кода приведено ниже.



Заметим, что само кольцо не несет детальной информации, информацию несет только его граница и поэтому нет никаких детализирующих коэффициентов, соответствующих закрашенной части кольца.

5.6 Упрощенная схема вейвлет сжатия.

Простейший способ вейвлет сжатия состоит в обнулении вейвлет коэффициентов по квадрантам. Мы видели, что двумерное ВП представляет операции усреднения и выделения деталей. Один шаг двумерного ВП применяет одномерное ВП к каждой строке, а затем к каждому столбцу результирующей матрицы коэффициентов. В результате матрица коэффициентов разбивается (условно) на 4 матрицы – на матрицу приближения `cA`, и 3 матрицы коэффициентов детализации `cH, cV, cD`. Следующий шаг двумерного ВП состоит в применении к матрице `cA` тех же преобразований и т.д. Приведенная ниже схема поясняет, как это выглядит.



Блок коэффициентов в правом нижнем углу схемы (блок cD_1) содержит детализирующую информацию самого низкого уровня Блоки cH_1 и cV_1 также получаются из детализирующих коэффициентов при одном шаге одномерного ВП по строкам или столбцам. Обнуляя коэффициенты блоков cD_1, cH_1 и cV_1 мы уменьшаем объем хранимой информации в 4 раза (до 25%). При этом качество изображения несколько снижается. Оставшийся блок `cA` обрабатывается одномерным ВП по строкам, а затем по столбцам. Блоки cD_2, cH_2 и cV_2 хранят детализирующие коэффициенты этого оставшегося блока.

Обнуляя коэффициенты этих блоков cD_2, cH_2 и cV_2 мы уменьшаем объем хранимой информации еще в 4 раза , т.е. всего в 16 раз по сравнению с исходным объемом. При этом снижается качество изображения. Продолжая этот процесс далее, можно обнулять коэффициенты блоков cD_k, cH_k и cV_k (k = 1,2,...) уменьшая объем изображения, но ухудшая его качество.

Однако лучшее качество сжатого изображения при том же объеме хранимой информации достигается при следующей схеме вейвлет сжатия. После вычисления вейвлет преобразования, вейвлет коэффициенты сортируются по убыванию их абсолютных значений. Запоминаются только x% наибольших по модулю коэффициентов, а оставшиеся (100-x)% коэффициентов полагаются равными нулю. Значение x выбирается пользователем. Восстановление изображения производится путем применения обратного ВП к прореженной матрице коэффициентов.

Вначале рассмотрим пример, который содержит и поясняет код, выполняющий прореживание матрицы

Пример. Прореживание матрицы коэффициентов (ch05\ch05ex15.m).

```
% оставляем pr самых больших элементов матрицы a,
% остальные обнуляем
a=magic(3)    % тестовая матрица
b=a(:);b'     % преобразуем матрицу в вектор
% сортируем b по убыванию и запоминаем индексы элементов
[c,ic]=sort(b,'descend'); ic'
pr=0.5; % доля оставляемых коэффициентов
% оставляем pr-ю часть индексов из ic
icc=ic(1:floor(length(ic)*pr)+1); icc'
sc=zeros(1, length(ic));
% первые pr больших элементов переносим в массив sc
sc(icc)=b(icc)
% преобразуем sc в матрицу исходного размера
aa=reshape(sc,size(a))
```

Результаты счета

```
a  =   8     1     6
       3     5     7
       4     9     2
b'  =  8     3     4     1     5     9     6     7     2
ic' =  6     1     8     7     5     3     2     9     4
icc' = 6     1     8     7     5
sc =   8     0     0     0     5     9     6     7     0
aa =   8     0     6
       0     5     7
       0     9     0
```

Комментарии в коде сценария и результаты счета поясняют алгоритм обнуления `pr` - й доли меньших элементов исходной матрицы.

□

Прежде, чем рассматривать примеры «сжатия» изображения, напишем код функции, выполняющей обратное двумерное ВП (ch05\iw2transform.m). При этом первая половина кода функции содержит проверки правильности задания аргументов функции.

```
% Многошаговое двумерное обратное вейвлет преобразование
% Аргументы: квадратная матрица fw с size(fw)=2^n x 2^n
% Рассматривается как матрица коэффициентов ВП
% Необязательный аргумент: p количество шагов ОВП необходимых
% для получения изображения
% Возвращает матрицу изображения f
function f=iw2transform(fw,varargin)
if ~isempty(varargin)
    nargs=length(varargin);
    if nargs==1
        p=varargin{1};
    else
        error('Слишком много аргументов');
    end;
else
    p=-1;  % ВП все шаги до конца
end
if mod(p,1)~=0, error('2-й аргумент должен быть целым числом');
end
if (p<-1) , error('Неверно задан 2-й аргумент'); end
if p==0, f=fw; return; end

[M,N]=size(fw);
if M~=N , error('Матрица коэффициентов fw должнf быть
квадратной'); end
nn=log2(N);
if mod(nn,1)~=0, error('Размер матрицы коэффициентов fw должен
быть степенью двойки'), end
if p>nn , error('Максимальное значение для 2-го аргумента =
%d',nn); end
if p==-1,  p=nn; end

N=2^(nn-(p-1));   % размер внутренней матрицы коэффициентов
% внутренняя матрица коэффициентов содержит минимальную
% усредняющую матрицу LL и обрамляющие ее матрицы LH, HL, HH
f1=fw(1:N, 1:N);  % создаю внутреннюю матрицу коэффициентов
for n=1:p
    % один шаг ОВП по столбцам внутренней матрицы коэффициентов
    aa=zeros(N/2,N); a=zeros(N,N);
    dd=zeros(N/2,N); %d=zeros(0,N);
    for i=1:N
        aa(:,i)=f1(1:N/2,i);
        dd(:,i)=f1(N/2+1:N,i);
        [a(:,i),d]=iwtransform(aa(:,i),dd(:,i),1); % d() пусто
    end
    ff=a';
    % один шаг ОВП по строкам  матрицы коэффициентов ff
    aa=zeros(N/2,N); a=zeros(N,N);
    dd=zeros(N/2,N); %d=zeros(0,N);
    for i=1:N
        aa(:,i)=ff(1:N/2,i);
        dd(:,i)=ff(N/2+1:N,i);
        [a(:,i),d]=iwtransform(aa(:,i),dd(:,i),1); % d()пусто
```

```
    end
    f2=a';  % усредненное изобр. из внутренней матрицы коэф.
    N=N*2;
    if N>M, break; end
    f1=fw(1:N, 1:N); % внутр. матрица коэфф. на уровень выше
    f1(1:N/2,1:N/2)=f2;  % ее усредненное изображение
end
f=f2;
```

Пример. Упрощенные алгоритмы сжатия, использующие вейвлет преобразование (cp05\imZip02.m).

```
% примеры вейвлет сжатия изображения
close all; clear all;
% 1 - читаем изображение
f=im2double(imread('Lobachevsky01.tif'));   % 256 x 256
imshow(f,[]);
% 2 - Выполняем двумерное ВП
fw=w2transform(f);

% ================== Блок 1=====================
% в матрице f оставляем pr самых больших элементов, остальные 0
b=fw(:);
% 3. Сортируем коэффициенты по убыванию модулей
%    и запоминаем их индексы
[c,ic]=sort(abs(b),'descend');  % сортируем модули
pr=0.1; % доля оставляемых коэффициентов
icc=ic(1:floor(length(ic)*pr));
sc=zeros(1,numel(b));
sc(icc)=b(icc);
fa=reshape(sc,size(fw)); % корректн. матр. вейвлет коэф.
% --------------- конец блока 1 --------------------
% ================== Блок 2 ======================
% обнуляем квадранты
% [M,N]=size(fw);  % предполагаем, что M=N
% fa=fw;
% for p=1:2
%    fa(1:N/2^p,N/2^p+1:N/2^(p-1))=0;
%    fa(N/2^p+1:N/2^(p-1),1:N/2^p)=0;
%    fa(N/2^p+1:N/2^(p-1),N/2^p+1:N/2^(p-1))=0;
% end
% --------------- конец блока 2 --------------------
% 4 - обратное двумерное ВП
f0=iw2transform(fa);
figure,imshow(f0,[]);
```


pr=0.1


pr=0.05


pr=0.01


p=1:1


p=1:2


p=1:3

Изображения в первой строке получаются в сценарии изменением переменной `pr`, значения которой приведены под рисунками.

Изображения во второй строке создаются сценарием, если закомментировать блок 1, раскомментировать блок 2 и в строке кода `for p=1:2`
изменять диапазон параметра `p`. Соответствующий диапазон приведен под рисунками второй строки изображений. Обратите внимание, что третье изображение в первой строке, «сжатое» в 100 раз смотрится лучше, чем третье изображение во второй строке, «сжатое» в 4^3=64 раза.

Хотя изображения в первой строке используют 10%, 5% и 1% вейвлет коэффициентов, они не представляют сжатие соответственно 10:1, 20:1, 100:1. Это объясняется тем, что информацию о расположении оставшихся коэффициентов тоже нужно хранить вместе с самими значениями коэффициентов.

5.7 Вейвлеты Хаара в Matlab

5.7.1 Функции Matlab работы с одномерными вейвлетами

В Wavelet Toolbox Matlab можно использовать большое количество типов вейвлетов. Для получения справочной информации о каком либо типе вейвлетов можно подать команду `waveinfo` с аргументом – именем типа вейвлетов.

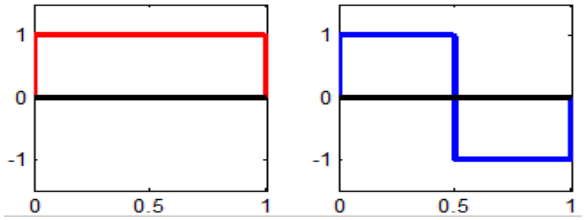
```
waveinfo('haar')
     ...
    Family                  Haar
    Short name              haar
    Examples                haar is the same as db1
    Orthogonal              yes
     ...
```

Функция `wavefun('haar',n)` возвращает вектор значений масштабирующей $\phi(x)$ и вейвлет функций Хаара $\psi(x)$ в 2^{n+1} точках отрезка $\left[0,1+\frac{1}{2^n}\right]$, включая конечные, например

```
[phi,psi,xval]=wavefun('haar',2)
phi =    0    1.0000    1.0000    1.0000    1.0000         0
psi =    0    1.0000    1.0000   -1.0000   -1.0000         0
xval =   0    0.2500    0.5000    0.7500    1.0000    1.2500
```

Пример. Построим графики функций $\phi(x)$ и $\psi(x)$.

```
[phi,psi,xval]=wavefun('haar',8);
xaxis=zeros(size(xval));
subplot(121),plot(xval,phi,'r',xval,xaxis,'k','LineWidth',3);
axis([min(xval) max(xval) -1.5 1.5]);
subplot(122),plot(xval,psi,'b',xval,xaxis,'k','LineWidth',3);
axis([min(xval) max(xval) -1.5 1.5]);
```



□

Один шаг ВП Хаара можно выполнить командой `dwt`, которая имеет следующий синтаксис

```
[cA,cD] = dwt(X,'haar');
```

Функция `dwt` вычисляет аппроксимирующие коэффициенты `cA` и детализирующие коэффициенты `cD` вектора `X`. Второй аргумент является строкой имени типа используемых вейвлетов.

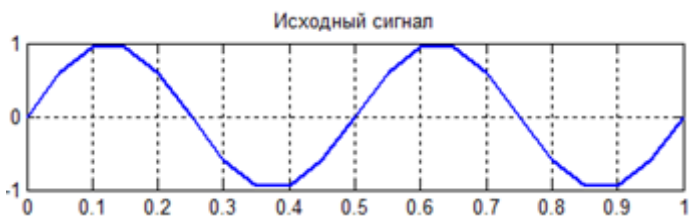
Функция `idwt` выпоняет одноуровневое обратное дискретное одномерное ВП. Команду `idwt` можно употреблять в следующей форме:

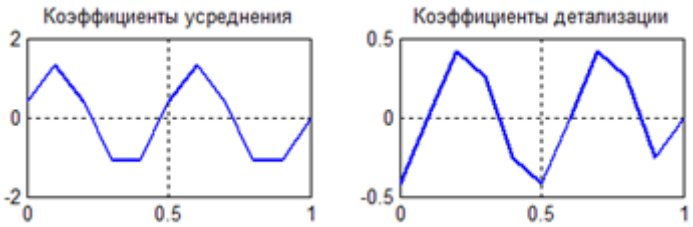
```
s = idwt(cA,cD,'haar');
```

Она по известным аппроксимирующим коэффициентам `cA` и детализирующим коэффициентам `cD` первого уровня вычисляет значения сигнала `s`, используя имя вейвлета.

Пример. Разложение функции $\sin(4\pi x)$ по вейвлетам Хаара (ch05\ch05ex22.m).

```
% разложение функции sin(4*pi*x) по вейвлетам Хаара
close all; clear all;
x=0:0.05:1+0.05; y=sin(4*pi.*x);
subplot(211); plot(x,y,'LineWidth',2);
title('Исходный сигнал'); grid on;

[ca,cd]=dwt(y,'haar');
xval=0:0.1:1;
subplot(223); plot(xval,ca,'LineWidth',2);
title('Коэффициенты усреднения'); grid on;
subplot(224); plot(xval,cd,'LineWidth',2);
title('Коэффициенты детализации'); grid on;
% Обратное ДВП
s=idwt(ca,cd,'haar');
err=norm(y-s)
```





Норма разницы исходного вектора `y` и восстановленного `s` очень мала

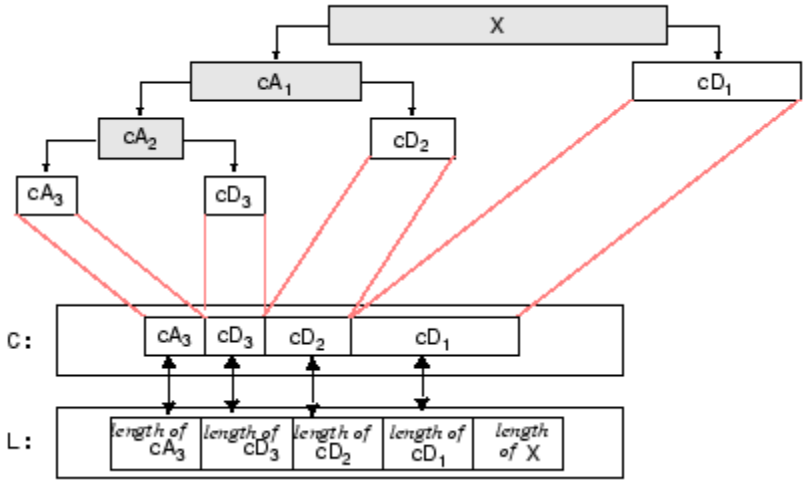
```
err =  6.5910e-016
```

□

Одномерное многоуровневое ДВП в Matlab выполняет функция `wavedec(f,k,'haar')`. Она принимает вектор `f`, количество шагов ВП `k` (в терминологии Matlab `k` – уровень разложения) и имя типа вейвлетов (в нашем случае `'haar'`). Возвращает пару векторов `c` и `L`.

```
[C,L] = wavedec(f,k,'haar');
```

Смысл возвращаемых параметров объясняет следующий рисунок, взятый нами из справочной системы Matlab на странице справки функции `wavedec`).



Все коэффициенты одномерного ВП возвращаются в векторе `C`. При этом длина усредняющего вектора **a**, длины одномерных детализирующих векторов $\mathbf{d}^i$ и длина исходного вектора `x` (совпадает с длиной вектора `c`) возвращаются в векторе L.

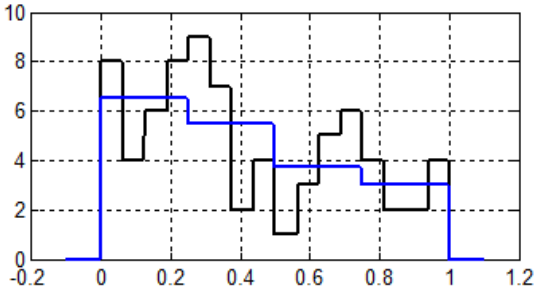
Обратите внимание, что индексация шагов вейвлет разложения, которая нами использовалась в предыдущих параграфах, отличается от нумерации уровней разложения, используемых в Matlab. Так, если вспомнить индексацию коэффициентов в последовательности (3.11)

$$\left\{a_{0,0}, d_{0,0}, d_{1,0}, d_{1,1}, d_{2,0}, \ldots, d_{n-1,0}, \ldots, d_{n-1,2^{n-1}-1}\right\},$$

то первый из индексов у нас показывал номер многошагового процесса и увеличивался слева направо. В то же время в Matlab уровень разложения в векторе `c`, возвращаемом функцией `wavedec`, увеличивается справа налево.

Пример. Продемонстрируем использование функции `wavedec` (ch05\ch05ex12.m)

```
close all; clear all;
f=[8 4 6 8 9 7 2 4 1 3 5 6 4 2 2 4];% длина f = степень двойки
N=length(f);
f=f./sqrt(N);  % преобразуем f к нормированным psi
x=-0.1:0.005:1.1;
[c,L] = wavedec(f,2,'haar');% изменяем второй аргумент от 0 до 4
a=c(1:L(1));
ym=wcalc(x,a,[]);  % усредняющая функция
%figure,plot(x,ym,'k-','LineWidth',2);   % 1
%axis([min(x) max(x) min(ym)-0.5 max(ym)+0.5]); grid on;
d=c(L(1)+1:L(end));
y=wcalc(x,a,d);  % исходная функция
%figure,plot(x,y,'k-','LineWidth',2);   % 2
%axis([min(x) max(x) min(y)-0.5 max(y)+0.5]); grid on;
figure,plot(x,y,'k-',x,ym,'LineWidth',2); % оба графика
grid on; set(gcf,'Color',[1 1 1]);
```



Для отображения усредняющей и исходной функций мы использовали созданную нами ранее функцию `wcalc`.

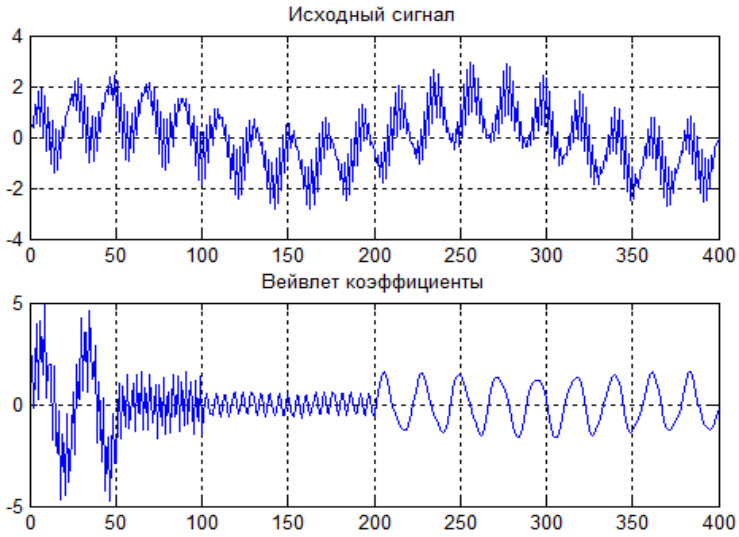
□

Функция `waverec` – многоуровневое восстановление одномерного сигнала. Она производит восстановление сигнала `s`, используя полученные ранее векторы многоуровневого разложения `[C,L]`. При восстановлении указывается имя используемого вейвлета, в нашем случае `'haar'`. Один из вариантов использования этой функции следующий:

```
s = waverec(C,L,'haar');
```

Пример. Использование функции `wavedec` и `waverec` (ch05\ch05ex23.m)

```
% разложение тестового сигнала по вейвлетам Хаара
% и его реконструкция
close all; clear all;
load sumsin;   % тестовый сигнал длиной 1000 точек
s=sumsin(1:400);
[c,l]=wavedec(s,3,'haar');
subplot(211);  plot(s,'LineWidth',1);  title('Исходный  сигнал');
grid on;
subplot 212; plot(c); title('Вейвлет коэффициенты'); grid on;

sr=waverec(c,l,'haar');       % реконструкция сигнала
err=norm(s-sr)                % норма разности сигналов
```



По нижнему рисунку хорошо видно, что длина исходного сигнала такая же как и у вектора коэффициентов, который составлен из компонент `cA3`, `cD3`,`cD2`,`cD1`. На нижнем графике хорошо различимы первая компонента вектора `C` длины 50 (усредняющая часть) и три детализирующих участка.

Последние две строки кода сценария показывают, что исходный и восстановленный сигналы практически совпадают. Получена следующая ошибка

```
err = 9.2579e-015
```

□

Команда `appcoef` вычисляет аппроксимирующие коэффициенты `cA` одномерного сигнала уровня `N`, используя полученные ранее векторы многоуровневого разложения `[C,L]`. При этом уровень `N` должен быть не выше, чем уровень N_0 полученного разложения [C,L]. Например, если вектор `C` имеет структуру [A(N_0), D(N_0), ..., D(N),...,D(1)], то должно быть $0 \le N \le N_0$. Обычно команда применяется в следующей форме:

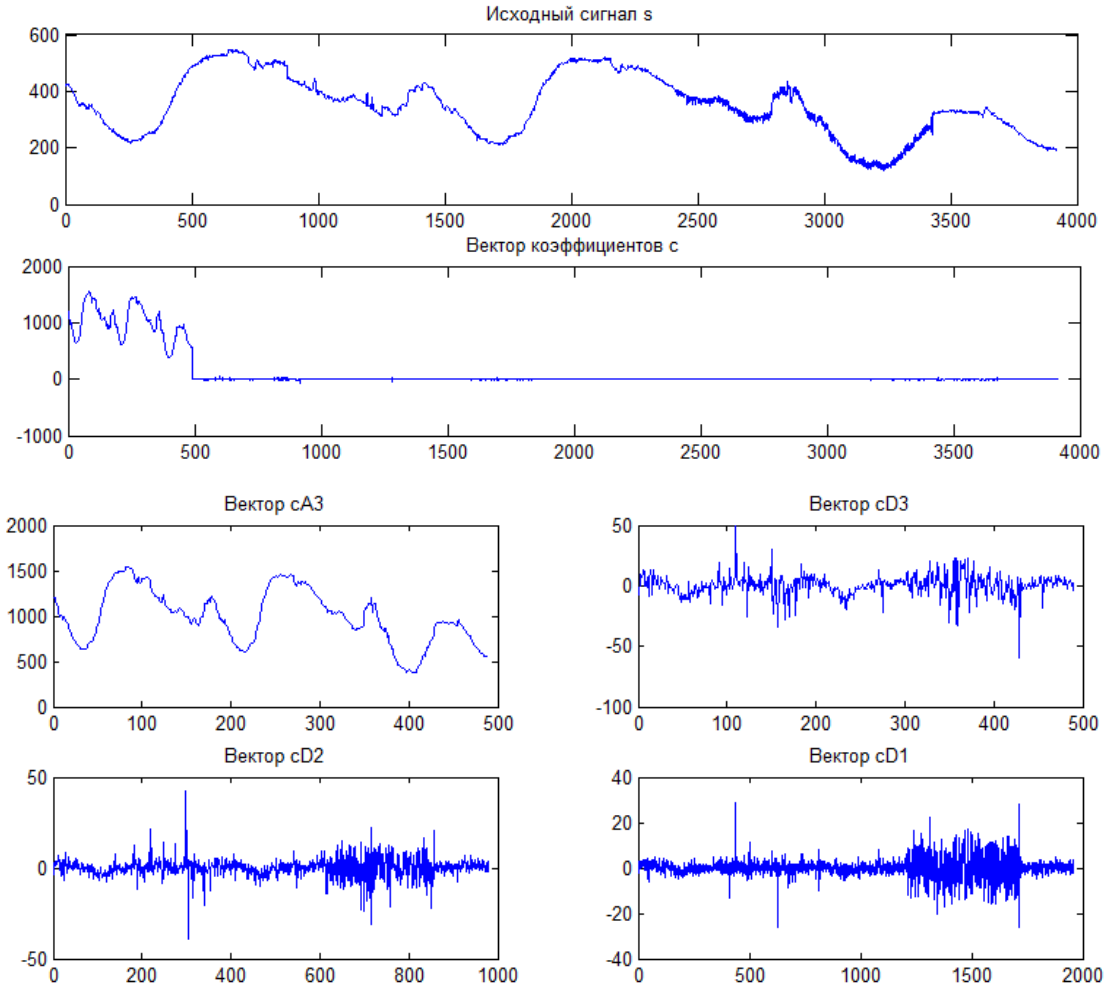
```
A = appcoef(C,L,'haar',N);
```

Команда `detcoef` находит детализирующие коэффициенты одномерного сигнала уровня N, используя полученные ранее векторы многоуровневого разложения `[C,L]`. Уровень N должен быть не выше, чем уровень полученного разложения `[C,L]`, т.е. $0 \le N \le N_0$. Ее использование показано в примере.

Пример. Тестовый сигнал раскладываем до уровня 3, а затем извлекаем аппроксимирующие и детализирующие коэффициенты уровней 1, 2, 3 (ch05\ch05ex24.m).

```
close all; clear all;
load leleccum;           % тестовый сигнал
s=leleccum(1:4000);
subplot(411); plot(s); title('Исходный сигнал s');
[c,l]=wavedec(s,3,'haar');
subplot 412; plot(c); title('Вектор коэффициентов c');
ca3=appcoef(c,l,'haar',3);
subplot(425); plot(ca3); title('Вектор cA3');
```

```
[cd1,cd2,cd3] = detcoef(c,l,[1 2 3]);
subplot(426); plot(cd3); title('Вектор cD3');
subplot(427); plot(cd2); title('Вектор cD2');
subplot(428); plot(cd1); title('Вектор cD1');
```



Заметим, что команда `appcoef(c,l,'haar',0)` возвращает исходный сигнал.

```
ca0=appcoef(c,l,'haar',0);
figure; plot(ca0-10,'r'); % смещение графика вниз
hold on; plot(s); hold off;
```

□

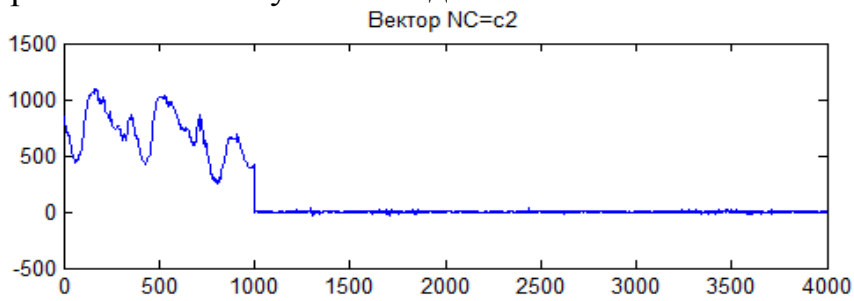
Функция `upwlev` выполняет одноуровневую реконструкцию одномерного ВП. Если заданы векторы вейвлет разложения `[C,L]` уровня разложения N_0, то команда `upwlev` находит векторы вейвлет разложения `[NC, NL]` уровня $N_0 - 1$, т. е. на один уровень выше. Извлекаются также последние аппроксимирующие коэффициенты `cA` уровня N_0.

```
[NC,NL,cA] = upwlev(C,L,'haar');
```

Пример. В продолжении предыдущего сценария добавляем команды

```
[c2,l2]=wavedec(s,2,'haar');
[NC,NL,CA]=upwlev(c,l,'haar');
figure;
subplot 211; plot(c2); title('Вектор коэффициентов c2');
subplot(212); plot(NC); title('Вектор NC=c2');
```

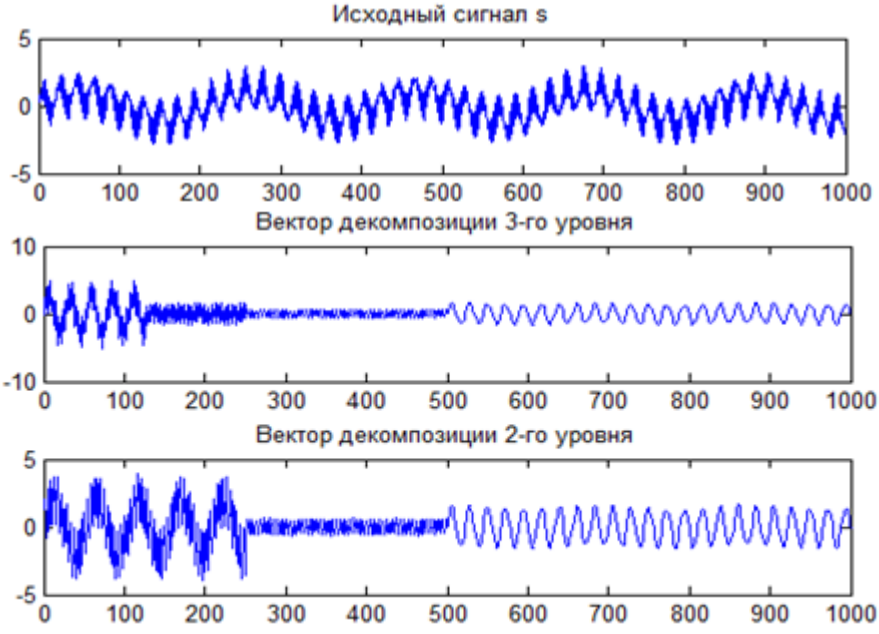
Здесь мы строим график вектора коэффициентов ДВП 2-го уровня с помощью команды `[c2,l2]=wavedec(s,2,'haar')`. Затем берем вектор коэффициентов `[c,l]`, полученный нами ранее. Он представляет 3 –й уровень ДВП, к которому применяем функцию `[NC,NL,CA]=upwlev(C,L,'haar')` и получаем вектор коэффициентов 2 – го уровня. Как и следовало ожидать, графики векторов `c2` и `NC` получились идентичными.



□

Пример. Одноуровневая реконструкция с использованием функции `upwlev` (ch05\ch05ex25.m).

```
close all; clear all;
load sumsin; % тестовый сигнал
s=sumsin(1:1000);
subplot(311); plot(s); title('Исходный сигнал s');
[c,l]=wavedec(s,3,'haar');
subplot 312; plot(c); title('Вектор декомпозиции 3-го уровня');
[nc,nl]=upwlev(c,l,'haar');
subplot 313; plot(nc); title('Вектор декомпозиции 2-го уровня');
set(gcf,'Color',[1 1 1]);
% сравнение результатов wavedec(s,2,'haar') и
% одноуровневой реконструкции upwlev(c,l,'haar');
[c2,l2]=wavedec(s,2,'haar');
figure, plot(c2); hold on; plot(nc-0.2,'r'); hold off;
```



□

Функция wrcoef восстанавливает сигнал по одному усредняющему вектору cA_N или детализирующему вектору cD_N массива вейвлет коэффициентов. В первом случае все детализирующие коэффициенты уровня N и ниже обнуляются и восстанавливается «приближенно» исходная функция. Во втором – обнуляется усредняющий вектор и все детализирующие коэффициенты ниже уровня N. Фактически значения вектора cA_N или cD_N «растягиваются» на всю длину исходного сигнала (увеличивается количество элементов вектора). Кроме того, функция учитывает нормирующие коэффициенты. Функция имеет следующий синтаксис:

```
X=wrcoef('type',c,l,'haar',N);
```

Здесь 'type' определяет тип ветви:

'type'='a'– восстановление из аппроксимирующих коэффициентов;

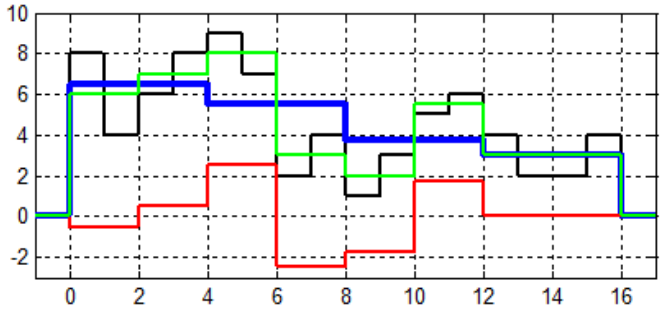
'type'='d' – восстановление из детализирующих коэффициентов.

Если N не указано, то имеется в виду максимальный уровень.

Пример. (ch05\ch05ex28.m).

```
close all; clear all; %clc;
s=[8 4 6 8 9 7 2 4 1 3 5 6 4 2 2 4]; % вектор сигнала
% строим график исходной кусочной функции
x=-1:0.01:17;
n=length(s);
y=0;
for i=1:n
    y=y+s(i).*psiHaar(x,0,i-1);
end
plot(x,y,'k-','LineWidth',2);   % график (черная ломаная)
axis([min(x) max(x) min(y)-1 max(y)+1]); grid on;
hold on;

[c,l] = wavedec(s,2,'haar');%вейвлет декомпозиция до 2-го уровня

c2=c(1:4)%четыре значения делим на 2 (два раза по корню из двух)
% Сравнение с работой функции appcoef
ca2=appcoef(c,l,'haar',2);
ca2  % те же значения ca2=c2

% Приближенная реконструкция  сигнала
% Из структуры [c,l] из вектора аппроксимации ca2
awr2 = wrcoef('a',c,l,'haar',2)

% график приближения по коэффициентам awr2
yr=0;
for i=1:n
    yr=yr+awr2(i).*psiHaar(x,0,i-1);
end
plot(x,yr,'b-','LineWidth',3); % синяя

% график приближения по коэффициентам awd2
awd2 = wrcoef('d',c,l,'haar',2);
yd=0;
```

```
for i=1:n
    yd=yd+awd2(i).*psiHaar(x,0,i-1);
end
plot(x,yd,'r-','LineWidth',2);  % красная
% сумма приближений A и D
plot(x,yd+yr,'g-','LineWidth',2); % зеленая
```



На графике приведена исходная кусочно постоянная функция (черная кривая) и ее приближение (синий цвет), построенное по аппроксимирующим вейвлет коэффициентам 2-го уровня. Красная ломаная представляет приближение по коэффициентам cD_2 , а зеленая – сумма двух приближений (сумма синей и красной кривых). Заметим, что длины векторов `awr2` и `awd2` совпадают с длиной исходного вектора `s`.

Значения векторов:

```
c2 =  13.0000   11.0000    7.5000    6.0000
ca2 = 13.0000   11.0000    7.5000    6.0000
awr2 =    6.5000    6.5000    6.5000    6.5000    5.5000 ...
          5.5000    5.5000    5.5000    3.7500    3.7500 ...
          3.7500    3.7500    3.0000    3.0000    3.0000 ...
          3.0000
```

□

Функция `upcoef` принимает вектор `X` и восстанавливает вектор `Y` из коэффициентов ДВП, применяя `N` раз процедуру обратного вейвлет преобразования. Она имеет следующий синтаксис

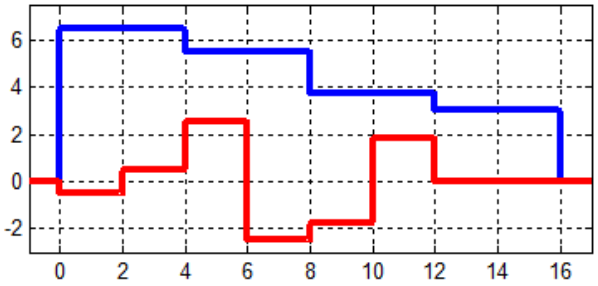
```
Y = upcoef('type',X,'haar',N);
```

Вектор `X` считается детализирующими коэффициентами, если `'type'='d'` и аппроксимирующие коэффициенты полагаются нулевыми. Наоборот, если `'type' = 'a'`, то детализирующие коэффициенты полагаются нулевыми, а вектор `X` рассматривается как аппроксимирующий.

Пример.

```
close all; clear all;
x=-1:0.01:17;
ca2 = [13 11 7.5 6];
cd2 =[ -1 5 -3.6 0];
rc = upcoef('a',ca2,'haar',2);
rd = upcoef('d',cd2,'haar',2);
yr=0; yd=0;
for i=1:16
    yr=yr+rc(i).*psiHaar(x,0,i-1);
    yd=yd+rd(i).*psiHaar(x,0,i-1);
end
```

```
plot(x,yr,'b-',x,yd,'r-','LineWidth',3);
axis([min(x) max(x) min(yr)-3 max(yr)+1]); grid on;
```



В этом примере мы взяли вектор `ca2` таким же как получен в предыдущем сценарии. Рассматривая его как аппроксимирующий вектор 2 – го уровня, мы получили приближение кривой (синяя ломаная) такой же как в предыдущем сценарии. Если в предыдущем сценарии выполнить команду

```
cd2=detcoef(c,l, 2 )
```

то она вернет вектор детализирующих коэффициентов 2 – го уровня

```
cd2 = -1.0000    5.0000   -3.5000       0
```

Их мы использовали в качестве вектора значений `cd2` при получении вектора `rd` в текущем сценарии. Используя его, мы получили приближение кривой (красная ломаная) такой же, как в предыдущем сценарии.

□

Перечислим функции Wavelet Toolbox Matlab, которые мы описали в этом параграфе и которые связаны с одномерным дискретным ВП.

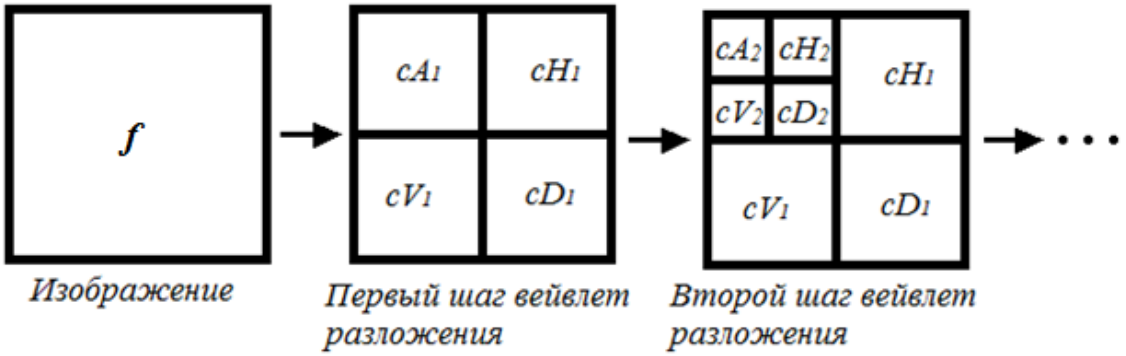
```
waveinfo('haar')
wavefun('haar',n)
[cA,cD] = dwt(X,'haar');
s = idwt(cA,cD,'haar');
[C,L] = wavedec(f,k,'haar');
s = waverec(C,L,'haar');
A = appcoef(C,L,'haar',N);
[cd1,cd2,cd3] = detcoef(c,l,[1 2 3]);
[NC,NL,cA] = upwlev(C,L,'haar');
X=wrcoef('type',c,l,'haar',N);
Y = upcoef('type',X,'haar',N);
```

У этих функций есть и другие варианты использования. С ними вы можете познакомится по справочной системе Matlab. Кроме того, мы везде использовали тип вейвлетов `'haar'`, который конечно может быть другим.

5.7.2 Функции Matlab работы с изображениями

Как мы видели в п.5, функция $f(x,y)$ раскладывается по функциям $\phi_{k,j}(x)\phi_{k,i}(y)$, $\phi_{k,j}(x)\psi_{k,i}(y)$, $\psi_{k,j}(x)\phi_{k,i}(y)$ и $\psi_{k,j}(x)\psi_{k,i}(y)$ (см. формулу (5.8)). Коэффициенты разложения по вейвлет базису $\phi_{k,j}(x)\phi_{k,i}(y)$ называются аппроксимирующими коэффициентами, а матрица cA_k, которую они образуют, называется матрицей приближения. Коэффициенты разложения по функциям $\psi_{k,j}(x)\phi_{k,i}(y)$,

$\phi_{k,j}(x)\psi_{k,i}(y)$, $\psi_{k,j}(x)\psi_{k,i}(y)$ называются горизонтальными, вертикальными и диагональными детализирующими коэффициентами соответственно, а матрицы, которые они образуют, обозначаются cH_k, cV_k и cD_k. На следующем рисунке показана схема обозначений матриц коэффициентов при вейвлет разложении двумерной функции (изображения) ***f***.



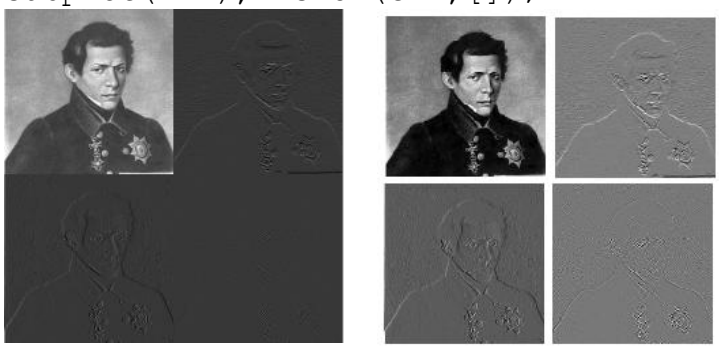
Один шаг дискретного ВП выполняет в Matlab функция `dwt2`. Она вызывается следующей командой

```
[cA,cH,cV,cD] = dwt2(f,'haar');
```

Аргумент `f` – это изображение, подвергаемое ВП. Второй аргумент указывает имя типа вейвлетов. Возвращает функция четыре матрицы коэффициентов ВП: матрицу приближения `cA` и три матрицы деталей `cH`, `cV`, `cD` (горизонтальную, вертикальную и диагональную).

Пример. (ch05\ch05ex20.m)

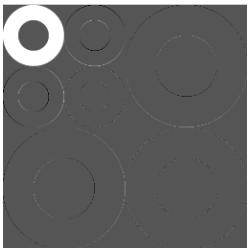
```
% один шаг 2D ВП
close all; clear all;
f=im2double(imread('Lobachevsky01.tif'));
[cA1,cH1,cV1,cD1] = dwt2(f,'haar');  % один шаг ВП
fw1=[cA1, cH1; ...
    cV1, cD1];
figure,imshow(fw1,[]);   % левое графическое окно
figure;                  % правое графическое окно
subplot(221),imshow(cA1,[]);
subplot(222),imshow(cH1,[]);
subplot(223),imshow(cV1,[]);
subplot(224),imshow(cD1,[]);
```



□

Пример. Два шага ВП (ch05\ch05ex19.m)

```
close all; clear all;
f=makering(8);   % изображение кольца
imshow(f,[]);
[cA1,cH1,cV1,cD1] = dwt2(f,'haar');      % первый шаг ВП
[cA2,cH2,cV2,cD2] = dwt2(cA1,'haar');    % второй шаг ВП
% собираем матрицы коэффициентов в одну матрицу
fw2=[cA2, cH2; cV2, cD2];
fw1=[fw2, cH1; cV1, cD1];
figure,imshow(fw1,[]);
```



□

Функции `image` и `imshow` могут быть применены для визуализации к любой матрице. Если использовать эти команды для вывода на экран матрицы палитрового изображения, то изображение может плохо соответствовать реальному. Для того чтобы получить на экране более качественное изображение можно применить к матрице `X` псевдоцветное матричное масштабирование командой `wcodemat`. Результатом команды `Y = wcodemat(X, n)` будет матрица `Y` из натуральных чисел от 1 до `n` в соответствии с величиной элементов матрицы `X`. Полный формат команды:

`Y = wcodemat(X, n,'mat',ABSOL),`

если `ABSOL = 0`, то масштабируются значения элементов матрицы `X`; если `ABSOL = 1`, то масштабируются абсолютные значения матрицы `X`. При третьем параметре `'mat'` диапазон значений матрицы от минимального до максимального делится на n участков. Значения матрицы, попавшие в первый участок получают значение 1, попавшие во второй участок – значение 2, и т.д. Третий аргумент также может принимать значение `'col'` или `'row'`. Тогда масштабируются элементы матрицы по отдельности в каждом столбце или строке. Значения по умолчанию: `ABSOL = 1, n = 16` и `'mat'`. Например, если `Y = wcodemat(X)`, то `image(Y)` – это изображение с использованием 16 градаций.

Пример.

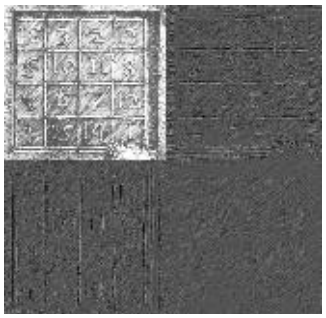
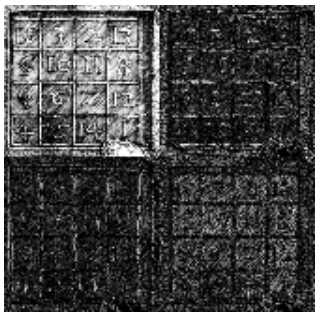
```
a=magic(4)
a = 16     2     3    13
     5    11    10     8
     9     7     6    12
     4    14    15     1
b=wcodemat(a,4,'mat')
b =  4     1     1     4
     2     3     3     2
     3     2     2     3
     1     4     4     1
```

```
a=rand(4,4)
a = 0.8147    0.6324    0.9575    0.9572
    0.9058    0.0975    0.9649    0.4854
    0.1270    0.2785    0.1576    0.8003
    0.9134    0.5469    0.9706    0.1419
b10=wcodemat(a,10)
b10 = 9     7    10    10
      10     1    10     5
      1     3     1     9
      10     6    10     1
```

□

Пример. Загружаем тестовый файл `detail` (магический квадрат Дюрера). При этом `X` содержит матрицу изображения, а `map` – цветовую карту. Выполняем один шаг ВП. Использование функции `wcodemat` значительно улучшает вид результата (ch05\ch05ex30.m).

```
close all; clear all;
load detail;
imshow(X,[]);
n=size(map,1);
[cA1,cH1,cV1,cD1] = dwt2(X,'haar');
fw1=[cA1, cH1; cV1, cD1];
figure,imshow(fw1,[]);  % изображение в середине
fwc1=[wcodemat(cA1,n),wcodemat(cH1,n);
      wcodemat(cV1,n), wcodemat(cD1,n)];
figure,imshow(fwc1,[]);  % изображение справа
```

□

Функция `idwt2` выполняет одноуровневое обратное двумерное преобразование. По известным аппроксимирующим коэффициентам `cA` и детализирующим коэффициентам `cH`, `cV`, `cD` команда `idwt2` восстанавливает значения матрицы X. Один из вариантов ее использования имеет вид

```
X = idwt2(cA,cH,cV,cD,'haar');
```

Пример. Продолжение сценария ch05\ch05ex30.m

```
% один шаг обратного ВП
X = idwt2(cA1,cH1,cV1,cD1,'haar');
figure,imshow(X,[]);
```

В результате будет построено исходное изображение.

□

Многие функции вейвлет преобразований из пакета Wavelet Tools порождают невоспроизводимые на экране «структуры» данных вида [C,S], где C — это вектор коэффициентов преобразования, а S — управляющая матрица. Ниже дается определение «структуры» [C,S] и рассматриваются некоторые функции Wavelet Tools для обращения с этими данными. Некоторым функциям, таким как приведены выше dwt2 и idwt2, эта структура данных не требуется.

Двумерное многоуровневое ВП Хаара выполняет функция wavedec2(X,N,'haar'), где X матрица изображения, N – число шагов ВП. Выходом является пара массивов: C – вектор строка коэффициентов ВП, S – вспомогательная управляющая матрица. Их смысл поясним на следующем примере.

Пример.

```
X=magic(4)
X = 16     2     3    13
     5    11    10     8
     9     7     6    12
     4    14    15     1
[C,S]=wavedec2(X,1,'haar')
C =  17.0000   17.0000   17.0000   17.0000  1.0000    -1.0000
     -1.0000    1.0000    4.0000   -4.0000  -4.0000   4.0000
     10.0000    6.0000   -6.0000  -10.0000
S =  2     2
     2     2
     4     4
```

Здесь матрица X преобразуется в вектор C длиной 1 x 16 и управляющую матрицу S размером 3 x 2. За один шаг ВП, выполненный в этом примере, мы должны получить четыре выходных матрицы размером 2 x 2: одна прореженное приближение cA и три матрицы деталей – горизонтальная, вертикальная и диагональная. В участке вектора C с 1-й по 4-ю позиции записаны коэффициенты cA, с 5-й по 8-ю записаны коэффициенты cH, элементы вектора C(9) – C(12) содержат коэффициенты cV и элементы вектора C(13)–C(16) содержат коэффициенты cD. В управляющей матрице S записаны размеры вычисленных матриц, которые по столбцам помещены в вектор C, плюс размер исходного изображения X. Таким образам, вектор S(1,:) содержит размер матрицы приближения (количество строк и столбцов), вектор S(2,:) содержит размер трех матриц деталей, вектор S(end,:) – размер исходного изображения X.

Чтобы в нашем примере для матрицы X выполнить ВП до конца (нулевой масштаб приближения) необходимо выполнить два шага командой

```
[C2,S2]=wavedec2(X,2,'haar')
C2 =  Columns 1 through 10
     34.0000         0         0    0.0000    1.0000
     -1.0000   -1.0000    1.0000    4.0000   -4.0000
     Columns 11 through 16
     -4.0000    4.0000   10.0000    6.0000   -6.0000
     -10.0000
```

```
S2 = 1     1
     1     1
     2     2
     4     4
```

Управляющая матрица `S` была обновлена. Ее первая строка в `S2` заменилась двумя строками, которые отражают факт того, что прежняя матрица приближения `2x2` в `C` была заменена четырьмя матрицами: матрицей приближения (размера `1x1`) и тремя матрицами деталей также размера `1x1`. В результате в `S2(1,:)` находится размер матрицы приближения, а `S2(2,:)` – размер матриц деталей масштаба 0; в `S2(3,:)` остался размер матриц деталей масштаба 1, полученных при первом шаге ВП, в `S2(end,:)` остался размер исходно изображения.

□

После всех шагов ВП вектор `C` будет организован следующим образом

$$C = \lfloor A_n \mid H_n \mid V_n \mid D_n \mid H_{n-1} \mid V_{n-1} \mid D_{n-1} \mid \ldots \mid H_1 \mid V_1 \mid D_1 \rfloor$$

где `A,H,V,D` являются вектор – строками, содержащими:

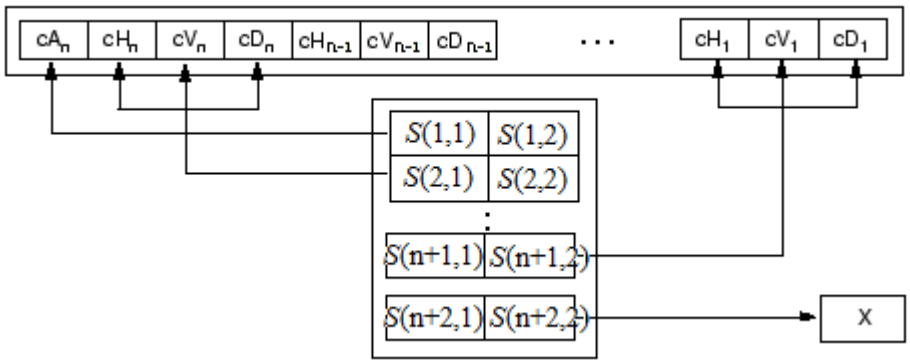
`A` – элементы матрицы приближения;

`H` – элементы горизонтальной матрицы деталей;

`V` – элементы вертикальной матрицы деталей;

`D` – элементы диагональной матрицы деталей.

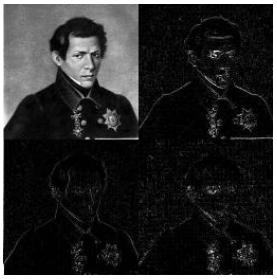
Каждый вектор содержит матрицы соответствующего уровня, записанные в него по столбцам. Следующий рисунок поясняет сказанное.



Пример. Построение матриц приближения и детализации одноуровневого ВП с использованием функции `wavedec2` и «структуры» [`C,S`] (ch05\ch05ex31.m).

```
close all; clear all;
X=im2double(imread('Lobachevsky01.tif'));
figure,imshow(X,[]);
[C,S]=wavedec2(X,1,'haar');
m1=S(1,1); n1=S(1,2);
XA=reshape(C(1:m1*n1),m1,n1);  % размещает вектор по столбцам в
матрицу
m2=S(2,1); n2=S(2,2);
XH=reshape(C(m1*n1+1:m1*n1+m2*n2),m2,n2);
XV=reshape(C(m1*n1+m2*n2+1:m1*n1+2*m2*n2),m2,n2);
XD=reshape(C(m1*n1+2*m2*n2+1:m1*n1+3*m2*n2),m2,n2);
n=255;
```

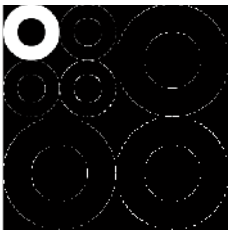
```
XCC=[wcodemat(XA,n), wcodemat(XH,n);...
     wcodemat(XV,n), wcodemat(XD,n)];
figure,imshow(XCC,[]);
```



□

Пример. Построение матриц приближения и детализации двухуровневого ВП и «структуры» [C,S] (ch05\ch05ex32.m).

```
close all; clear all;
X=makering(8);   % изображение кольца
figure,imshow(X,[]);
n=255;  % размер палитры
[C,S]=wavedec2(X,2,'haar'); % два шага ВП
m1=S(1,1); n1=S(1,2);
XA2=reshape(C(1:m1*n1),m1,n1); % вектор по столбцам в матрицу
m2=S(2,1); n2=S(2,2);
XH2=reshape(C(m1*n1+1:m1*n1+m2*n2),m2,n2);
XV2=reshape(C(m1*n1+m2*n2+1:m1*n1+2*m2*n2),m2,n2);
XD2=reshape(C(m1*n1+2*m2*n2+1:m1*n1+3*m2*n2),m2,n2);
XC2=[wcodemat(XA2,n), wcodemat(XH2,n); ...
     wcodemat(XV2,n), wcodemat(XD2,n)];
MN=m1*n1+3*m2*n2;
m3=S(3,1); n3=S(3,2);
XH1=reshape(C(MN+1:MN+m3*n3),m3,n3);
XV1=reshape(C(MN+m3*n3+1:MN+2*m3*n3),m3,n3);
XD1=reshape(C(MN+2*m3*n3+1:MN+3*m3*n3),m3,n3);
XCC=[XC2, wcodemat(XH1,n); wcodemat(XV1,n), wcodemat(XD1,n)];
figure,imshow(XCC,[]);
```



□

Чтобы не выполнять утомительные преобразования из вектор – строки в матрицы приближения и деталей, в IPT пакете есть функции `appcoef2` и `detcoef2`, предназначенные для получения матриц приближения и детализации.

Функция `appcoef2` возвращает матрицу коэффициентов приближения `cA` уровня N, используя матрицы C и S, полученные при декомпозиции изображения с помощью функции `wavedec2`. Если N не указан, то N=1. Функция имеет следующий синтаксис

```
A = appcoef2(C,S,'wname');
```

или

```
A = appcoef2(C,S,'wname',N);
```

Функция `detcoef2` возвращает матрицы коэффициентов детализации `cH, cV, cD` уровня N, используя матрицы C и S, полученные при декомпозиции изображения с помощью функции `wavedec2`. Если N не указан, то N=1. Функция имеет следующий синтаксис

```
D = detcoef2(O,C,S,N);
```

Если параметр `O = 'h'` (или `'v'` или `'d'`), то возвращаются матрицы `cH, cV, cD` уровня `N` ($1 \le N \le$ size(S,1)-2) соответственно. При вызове

```
[H,V,D] = detcoef2('all',C,S,N) ;
```

возвращаются все 3 матрицы коэффициентов детализации уровня `N`.

Пример. Построение матриц приближения и детализации одноуровневого ВП (ch05\ch05ex33.m).

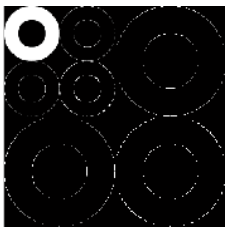
```
close all; clear all;
X=im2double(imread('Lobachevsky01.tif'));
figure,imshow(X,[]);
[C,S]=wavedec2(X,1,'haar');  % один шаг ВП
XA=appcoef2(C,S,'haar');
[XH,XV,XD] = detcoef2('all',C,S,1);
n=255;
XCC=[wcodemat(XA,n),wcodemat(XH,n); ...
     wcodemat(XV,n), wcodemat(XD,n)];
figure,imshow(XCC,[]);
```

Сравните со сценарием ch05ex31.m. Результат тот же, но код короче.

□

Пример. Построение матриц приближения и детализации двухуровневого ВП (ch05\ch05ex34.m).

```
close all; clear all;
X=makering(8);   % изображение кольца
figure,imshow(X,[]);
n=255;  % размер палитры
[C,S]=wavedec2(X,2,'haar'); % два шага ВП
% восстановление
XA2=appcoef2(C,S,'haar',2);
[XH2,XV2,XD2] = detcoef2('all',C,S,2);
[XH1,XV1,XD1] = detcoef2('all',C,S,1);
XC2=[wcodemat(XA2,n), wcodemat(XH2,n); ...
     wcodemat(XV2,n), wcodemat(XD2,n)];
XCC=[XC2, wcodemat(XH1,n); wcodemat(XV1,n), wcodemat(XD1,n)];
figure,imshow(XCC,[]);
```



Сравните со сценарием ch05ex32.m. Результат тот же, но код значительно короче.

Заметим, что матрицы деталей можно получать по отдельности

```
XH2=detcoef2('h',C,S,2);  % горизонт. матрица деталей уровня 2
XV2=detcoef2('v',C,S,2);  % верт. матрица деталей уровня 2
XD2=detcoef2('d',C,S,2);   % диаг. матрица деталей уровня 2
XH1=detcoef2('h',C,S,1);  % горизонт. матрица деталей уровня 1
XV1=detcoef2('v',C,S,1);  % верт. матрица деталей уровня 1
XD1=detcoef2('d',C,S,1);  % диаг. матрица деталей уровня 1
```

□

Функция `waverec2` предназначена для реконструкции изображения X из матриц `C` и `S`, полученных при декомпозиции изображения с помощью функции `wavedec2`.

```
X = waverec2(C,S,'haar');
```

Отметим, что эта команда эквивалентна вызову функции

```
X = appcoef2(C,S,'wname',0);
```

Пример. Здесь мы выполняем 5 шагов ВП. Затем, используя полученные матрицы декомпозиции C и S, восстанавливаем изображение (ch05\ch05ex35.m).

```
close all; clear all;
X=im2double(imread('Lobachevsky01.tif'));
figure,imshow(X,[]);
n=255;  % размер палитры
[C,S]=wavedec2(X,5,'haar');  % два шага ВП
XR = waverec2(C,S,'haar');   % восстановление
figure,imshow(XR,[]);
```

□

Функция `wrcoef2` восстанавливает изображение по одной ветви вейвлет коэффициентов. Если известно вейвлет разложение `[C,S]` некоторого уровня N_0, то функция восстанавливает матрицу `X` по одной ветви вейвлет коэффициентов более низкого уровня $N \le N_0$.

```
XR = wrcoef2('type',C,S,'haar',N);
XR = wrcoef2('type',C,S,'haar');
```

Если `'type'='a'`, то выполняется восстановление по аппроксимирующим коэффициентам; если `'type'='h'` или `'v'`, или `'d'`, то восстановление

выполняется по детализирующим коэффициентам – горизонтальным, вертикальным или диагональным. Если N не указано, то имеется в виду уровень N_0. Результирующая матрица изображения XR имеет те же размеры, что и исходная матрица X, которая использовалась для получения структуры декомпозиции [C,S].

Пример. Примеры использования функции wrcoef2 (ch05\ex03.m).

```
close all; clear all;
load woman;
n=255;  % размер палитры
[c,s] = wavedec2(X,2,'haar');
% Реконструкция изображения из матриц приближения уровней 1 и 2
a2 = wrcoef2('a',c,s,'haar',2);
a1 = wrcoef2('a',c,s,'haar',1);
% Реконструкция изображения из матриц деталей уровня 2
hd2 = wrcoef2('h',c,s,'sym5',2);
vd2 = wrcoef2('v',c,s,'sym5',2);
dd2 = wrcoef2('d',c,s,'sym5',2);
figure;
subplot(231),imshow(X,[]);
subplot(232),imshow(a2,[]);
subplot(233),imshow(a1,[]);
subplot(234),imshow(wcodemat(hd2,n),[]);
subplot(235),imshow(wcodemat(vd2,n),[]);
subplot(236),imshow(wcodemat(dd2,n),[]);
```

О качестве восстановления изображения вы можете судить по приведенным картинкам.

□

Функция upcoef2 выполняет реконструкцию изображения из матриц приближения или детализации. Она похожа на функцию wrcoef2 , которая использует структуру [C,S]. Но для upcoef2 используются матрицы cA или cH, cV, cD. Она имеет следующий синтаксис

```
XR = upcoef2(O,X,'haar',N);
```

Функция `XR = upcoef2(O,X,'haar',N)` восстанавливает матрицу `XR` из коэффициентов матрицы `X` заданного уровня `N`, применяя `N` раз процедуру обратного вейвлет преобразования. При этом в зависимости от значения параметра O детализирующие (или аппроксимирующие коэффициенты полагаются нулевыми). Если `O='a'`, то реконструируются аппроксимирующие коэффициенты `X`. Если `O='h','v'` или `'d'`, то восстанавливаются детализирующие коэффициенты `X`. Команда

```
XR=upcoef2(O,X,'haar')
```

эквивалентна

```
XR=upcoef2(O,X,'haar',1).
```

Пример. Примеры использования функции `upcoef2` (ch05\ex04.m).

```
close all; clear all;
load woman;
figure, imshow(X,[]);  % матрица X содержит изображение
n=255;  % размер палитры
[c,s] = wavedec2(X,2,'haar');
ca1 = appcoef2(c,s,'haar',1);
a1 = upcoef2('a',ca1,'haar',1);
figure, imshow(ca1,[]);
figure, imshow(a1,[]);
chd1 = detcoef2('h',c,s,1);
hd1 = upcoef2('h',chd1,'haar',1);
figure, imshow(chd1,[]);
figure, imshow(hd1,[]);
cvd1 = detcoef2('v',c,s,1);
vd1 = upcoef2('v',cvd1,'haar',1);
figure, imshow(cvd1,[]);
figure, imshow(vd1,[]);
cdd1 = detcoef2('d',c,s,1);
dd1 = upcoef2('d',cdd1,'haar',1); %,siz);
figure, imshow(cdd1,[]);
figure, imshow(dd1,[]);
```

□

Функция `upwlev2` выполняет одноуровневую реконструкция 2D вейвлет разложения. Имея структуру `[C,S]` уровня N, она строит такую же структуру уровня на единицу ниже. Вариант ее вызова следующий

```
[NC,NS,cA] = upwlev2(C,S,'wname');
```

Ее можно также вызывать в варианте

```
[NC,NS,cA] = upwlev2(C,S,'wname') ;
```

в котором она возвращает не только структуру декомпозиции `[NC,NS]` уровня на единицу ниже, но возвращает также матрицу приближения текущего уровня. Если структура `[C,S]` имеет уровень `n`, структура `[NC,NS]` имеет уровень `n-1`, то `cA` является матрицей приближения уровня `n`.

Пример. Использование функции upwlev2 (ch05\ex05.m).

```
close all; clear all;
load woman;
figure, imshow(X,[]);  % матрица X содержит изображение
n=255;  % размер палитры
[c,s] = wavedec2(X,2,'haar');
% Размер вектора декомпозиции c = к-во пикселей изображения X
sc = size(c)
s % размеры матриц декомпозиции
[nc,ns] = upwlev2(c,s,'db1');% Шаг реконструкции структуры [c,s]
snc = size(nc)
ns
figure,imshow(cA,[]);
scA=size(cA)
```

Размеры векторов c и nc одинаковы, а матрица размеров s содержит коэффициентов на одну строку больше, чем ns.

```
sc =  1       65536
s =  64    64
     64    64
     128   128
     256   256
snc = 1       65536
ns = 128   128
     128   128
     256   256
scA = 64    64
```

□

Вспомогательные функции.

Функция wmaxlev принимает размер сигнала (число) или изображения (вектор из двух чисел) и возвращает максимально допустимый уровень вейвлет разложения (число).

```
wmaxlev(256,'haar')
ans =    8
s = [2^9 2^7];
s
s = 512   128
wmaxlev(s,'haar')
ans =  7
```

□

Когда изображение имеет размер отличный от 2^nx2^n, то при выполнении ВП изображение приходится расширять до квадратного с размером равным степени двойки. Значения, которые используются при расширении, определяются глобальным параметром dwtmode. Чтобы узнать текущее значение этого параметра, выполните команду dwtmode('status') или просто dwtmode.

```
dwtmode('status')
*************************************************
**  DWT Extension Mode: Symmetrization (half-point)  **
*************************************************
```

Перечислим функции Wavelet Toolbox Matlab, которые мы описали в этом параграфе и которые связаны с двумерным дискретным ВП.

`dwt2`	один шаг дискретного двумерного ВП
`idwt2`	одноуровневое обратное двумерное ВП
`wavedec2`	двумерное многоуровневое ВП, возвращает структуру `[C,S]`
`waverec2`	реконструкции изображения из структуры `[C, S]`
`appcoef2`	возвращает матрицу приближения из `[C, S]`
`detcoef2`	возвращает матрицы детализации из `[C, S]`
`upwlev2`	одноуровневая реконструкция структуры [C,S]
`wrcoef2`	восстанавливает изображение по одной ветви вейвлет разложение `[C,S]`
`upcoef2`	восстановление изображения из матриц приближения или детализации
`wmaxlev`	максимальный уровень вейвлет разложения
`dwtmode`	глобальный параметр расширения изображений.

У этих функций есть и другие варианты использования. С ними вы можете познакомится по справочной системе Matlab. Кроме того, мы везде использовали тип вейвлетов `'haar'`, который конечно может быть другим. Другие типы вейвлетов мы еще не рассматривали.

Литература.

1. Р. Гонсалес, Р. Вудс Цифровая обработка изображений. М.: Техносфера, 2005.
2. Р. Гонсалес, Р. Вудс С. Эддинс Цифровая обработка изображений в среде Matlab. М.: Техносфера, 2006.
3. Штарк Г. Г. Применение вейвлетов для ЦОС. – М.: Техносфера, 2007. – 192 с.
4. С. Уэлстид Фракталы и вейвлеты для сжатия изображений в действии. Учебное пособ. – М.: ИздательствоТриумы, 2003. – 320с.
5. Смоленцев Н.К. Основы теории вейвлетов. Вейвлеты в Matlab. – М.: ДМК Пресс, 2005. – 304 с.
6. M. Misiti, Y. Misiti, G. Oppenheim, J-M Poggi Wavelet Toolbox. User's Guide. R2014a. The MathWorks, Inc.