Министерство образования и науки Российской Федерации

Государственное образовательное учреждение высшего
профессионального образования
Омский государственный университет им. Ф.М. Достоевского
Математический факультет
Кафедра “Прикладная и вычислительная математика”

ДИПЛОМНАЯ РАБОТА

СЖАТИЕ ЗВУКОВОГО СИГНАЛА С ПОМОЩЬЮ ВЕЙВЛЕТ-АНАЛИЗА

Выполнил студент группы
МП-003 Акчурин А.Р.

_____

Научный руководитель:
к.ф-м.н. Еремеев А.В.

_____

Омск-2005

Введение

В наше время активного развития электронных технологий и внедрения их в бытовые изделия широкого потребления, в частности, в мультимедийную технику: цифровые плееры, фотоаппараты, камеры, остро встает вопрос об удобстве хранения цифровой информации, передачи по различным интерфейсам и протоколам и, естественно, о ее сжатии. Существует достаточно много форматов сжатия аудио сигнала. Среди них наиболее известные:

- MP3 (MPEG-1 layer 3);
- Ogg Vorbis;
- WMA (Windows Media);
- RealAudio.

Безусловно, MP3 сейчас является самым распространённым форматом сжатия аудио. Но вовсе не потому, что он – лучший по качеству звучания или компрессии. Просто исторически сложилось так, что он появился на рынке намного раньше других форматов. Еще одна весомая причина – это инертность производителей разнообразной мультимедийной техники, не желающих вводить новые аудиоформаты. Если судить по проведенным тестам независимых исследователей, то выявляется другой лидер – Ogg Vorbis [1,2]. Мы исследовали этот формат кодирования, и учли тенденции его развития при постановке задачи.

И MP3 и Vorbis основаны на разбиении сигнала на сегменты и применении к каждому сегменту дискретного преобразования Фурье с использованием психоаккустического фильтра и алгоритмов кодирования (Huffman, VQ), но отличаются реализацией. В перспективе развития разработчики кодека Ogg Vorbis планируют привлечь вэйвлет-технологию сжатия цифровых аудиоданных.

Естественным образом возникла идея изучить вэйвлет-анализ (анализ всплесков), активно внедряющийся в сферы обработки одномерных и двумерных сигналов, научиться использовать его на практике, а также

попробовать применить оптимизационный метод в целях решения поставленной задачи.

Для сжатия звука, можно использовать следующие методы или их комбинации.

1) Трешолдинг – отбрасывание близких к нулю коэффициентов в разложении сигнала.
2) Округление коэффициентов. Естественно, за сжатие надо «платить» потерей качества, которая возникает из-за округления. Желательно, чтобы восстановленный сигнал как можно меньше отличался от исходного. Поэтому, одной из главных задач становится оценка значимости того или иного коэффициента в разложении относительно остальных и, соответственно, определение точности его хранения.
3) Подбор преобразования, обеспечивающего наименьший суммарный объем хранимого числа разрядов.
4) Подбор разбиения сигнала, обеспечивающего наименьший суммарный объем хранимого числа разрядов. Есть гипотеза, о том, что можно сэкономить суммарное количество разрядов, подбирая сегменты разбиения так, чтобы сигнал совпадал по фазе с некоторыми базисными функциями. Следовательно, коэффициенты при других базисных функциях будут близкими к нулю. Для нахождения оптимального разбиения можно использовать подходящий оптимизационный метод, например, динамическое программирование.

Таким образом, в данной работе рассмотрен способ сжатия сигнала, основанный на различных вариантах дискретного вэйвлет-преобразования (Discrete Wavelet Transform, DWT), а также дискретного преобразования Фурье (Discrete Fourier Transform, DFT), оценке разрядности коэффициентов, PPM-кодировании, битовой упаковке и динамическом программировании.

1. Алгоритм сжатия сегмента сигнала

Рассмотрим процедуру сжатия одного сегмента сигнала. В зависимости от количества каналов в исходном звуковом файле, на вход подается один (моно) или два (стерео) массива чисел длины N. Эти числа будем называть сэмплами (от англ. sample). На выходе, после обработки, мы получаем N_b байтов закодированного сигнала. Сама же обработка массивов состоит из последовательности шагов, каждый из которых рассмотрим отдельно.

1.1. Дискретное вэйвлет-преобразование

Для понимания техники вэйвлет-разложения введем некоторые понятия. С более подробным изложением основных понятий вэйвлет-анализа можно ознакомиться в приложении или в [7]. Рассмотрим две функции ϕ и ψ из $L^2(\mathbf{R})$, удовлетворяющие некоторым свойствам, описанным в приложении, где ϕ – *масштабирующая функция*, а ψ – *вэйвлет*. Каждую функцию f из $L^2(\mathbf{R})$ можно приблизить функциями $\{\phi(2^p - k) : k \in \mathbf{Z}\}$ с любой наперед заданной точностью, выбрав достаточно большое значение p:

$$f(x) = \sum_{k \in \mathbf{Z}} A_k^p \phi(2^p x - k). \qquad (1.1.1)$$

Коэффициенты $\{A_k^p\}$ называются *аппроксимирующими коэффициентами*.

Будем называть функции $\phi(2^{p-1}x - k)$, соответствующие фиксированному p функциями p-го уровня. Функции ϕ и ψ обладают таким свойством, что масштабирующие функции p-го уровня выражаются через масштабирующие функции и вэйвлет-функции p-1-го уровня и наоборот:

$$\phi(2^p x - l) = \sum_k [a_{l-2k}\phi(2^{p-1}x - k) + b_{l-2k}\psi(2^{p-1}x - k)], \quad l, p \in \mathbf{Z}, \qquad (1.1.2)$$

$$\phi(2^{p-1}x - l) = \sum_k p_{k+2l}\phi(2^p x - k), \quad l, p \in \mathbf{Z}, \qquad (1.1.3)$$

$$\psi(2^{p-1}x - l) = \sum_k q_{k+2l}\psi(2^p x - k), \quad l, p \in \mathbf{Z}, \qquad (1.1.4)$$

где последовательности $\{a_k\}$ и $\{b_k\}$ называются *последовательностями разложения*, а $\{p_k\}$ и $\{q_k\}$ – *последовательности восстановления*.

Следовательно, используя (1.1.2) мы можем переписать (1.1.1) как

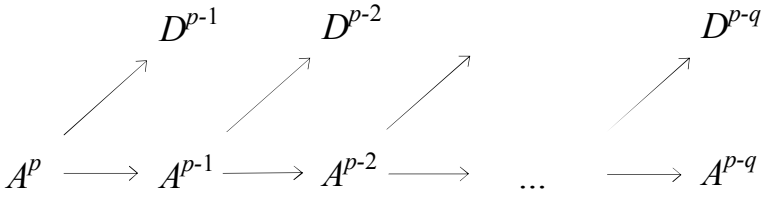
$$f_p(x) = \sum_{k\in\mathbf{Z}} A_k^p \phi(2^p x - k) = \sum_{k\in\mathbf{Z}} A_k^{p-1} \phi(2^{p-1} x - k) + \sum_{k\in\mathbf{Z}} D_k^{p-1} \psi(2^{p-1} x - k).$$

$$\begin{cases} \overline{A}_k^{p-1} = \sum_l a_{l-k} A_l^p, \\ \overline{D}_k^{p-1} = \sum_l b_{l-k} A_l^p, \end{cases} \tag{1.1.5}$$

$$\begin{cases} A_k^{p-1} = \overline{A}_{2k}^{p-1}, \\ D_k^{p-1} = \overline{D}_{2k}^{p-1}, \end{cases} \tag{1.1.6}$$

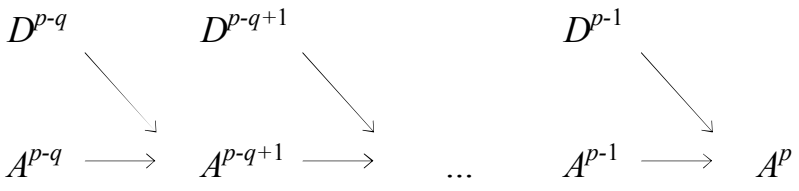
где коэффициенты $\{D_k^{p-1}\}$ называются *детализирующими коэффициентами*.

Отметим, что коэффициенты $A^{p\text{-}1}$ и коэффициенты $D^{p\text{-}1}$ можно получить из коэффициентов A^p, используя (1.1.5) и (1.1.6), причем применение (1.1.6) называется *сгущающей выборкой*, то есть, применив (1.1.5), мы оставляем лишь те коэффициенты, которые имеют четный индекс. Продолжая этот процесс, получаем следующую схему расчетов коэффициентов:



В итоге мы получили *вэйвлет-разложение,* то есть набор последовательностей коэффициентов $D^{p\text{-}1}$, $D^{p\text{-}2}$, …, $D^{p\text{-}q}$, $A^{p\text{-}q}$.

Используя обратные соотношения между функциями ϕ и ψ (1.1.3) и (1.1.4) мы можем построить обратный процесс:



Условимся называть аппроксимирующие и детализирующие коэффициенты A^j и D^j коэффициентами j-го уровня. Следуя теории вэйвлетов, для вэйвлет-разложения, сначала необходимо аппроксимировать сигнал с требуемой точностью на верхнем уровне. Пусть этот уровень имеет индекс 0. Предположим, нам дано N сэмплов, и необходимо получить N аппроксимирующих коэффициентов для дальнейшего разложения по алгоритму, описанному выше. Аппроксимировав сигнал на нулевом уровне, мы получим аппроксимирующие сигнал коэффициенты нулевого уровня $\{A_k^0\}$.

Предположим для простоты, что $N = 2^q$, $q \in \mathbf{N}$. В соответствии с алгоритмом разложения, описанным выше, получаем A^{-1} и D^{-1}. Заметим, что в алгоритме используется сгущающая выборка (берутся коэффициенты с четными номерами), поэтому в массивах A^{-1} и D^{-1} содержится по $N/2$ элементов. Аналогично, массивы A^{-2} и D^{-2} будут содержать по $N/4$ элементов. На последнем шаге A^{-q} и D^{-q} содержат по одному элементу. Таким образом, мы получили вэйвлет-разложение A^{-q}, D^{-q}, D^{-q+1}, …, D^{-2}, D^{-1}, в котором по-прежнему N элементов ($1+1+2+4+8+\ldots+2^{q-1} = 2^q = N$).

Заметим, что последовательности разложения $\{a_k\}$ и $\{b_k\}$ могут иметь более двух элементов, и в процессе вычисления коэффициентов нижестоящего уровня соответственно (1.1.5) и (1.1.6) они могут «выходить» за пределы массива A^{-j} $j = 0, 1, \ldots, q\text{-}1$ (рис 1.1):

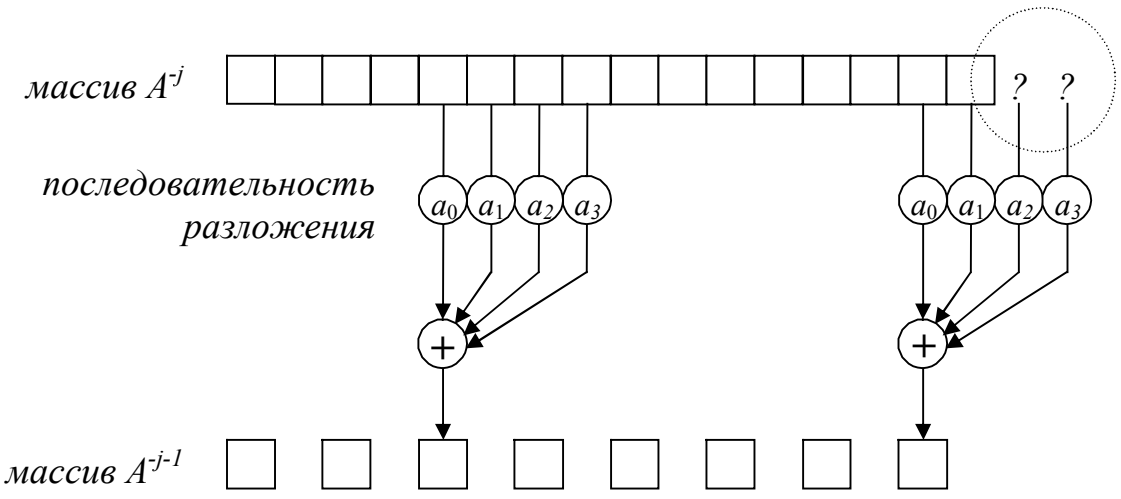


Рис. 1.1. Алгоритм разложения.

Один из способов разрешения ситуации – это замыкание массива A^{-j} в кольцо. Для этого достаточно взять остаток от деления индекса на длину массива. Так, для получения последнего элемента массива A^{-j-1} будут использованы два последних и два первых элемента массива A^{-j}. Этот метод позволяет взаимно-однозначно преобразовывать A^{-j} в A^{-j-1} и D^{-j-1}, и обратно, $j = 0, 1, \ldots, j\text{-}1$.

Мы рассмотрели тот случай, когда $N = 2^q$. Если же N нечетное, то на первом же шаге у нас не будет взаимно-однозначного соответствия между A^0, где N элементов и (A^{-1}, D^{-1}) где N-1 элемент. Решение следующее: Если на каком-то шаге j массив A^{-j} имеет нечетное число элементов, то расширим массив путем добавления в конец одного элемента с произвольным значением, например, равным значению последнего элемента A^{-j} для обеспечения непрерывного расширения. Здесь возникает избыточность информации, но она минимальна, так как число дополнительных коэффициентов не превышает $\log_2(N)$.

Отметим также, что трудоемкость алгоритмов вэйвлет-разложения и вэйвлет-восстановления равна $O(N)$. Даже по сравнению с быстрым преобразованием Фурье (БПФ), трудоемкость которого равна $O(N\log_2 N)$, описанный алгоритм работает значительно быстрее.

1.2. Дискретное преобразование Фурье

Определение 1.1. *Дана конечная последовательность* $x_0, x_1, x_2, \ldots, x_{N\text{-}1}$ *комплексных* чисел. *Дискретное преобразование Фурье (ДПФ) заключается в поиске последовательности* $X_0, X_1, X_2, \ldots, X_{N\text{-}1}$, *элементы которой вычисляются по формуле:*

$$X_k = \sum_{n=0}^{N-1} x_n e^{-i\frac{2\pi kn}{N}}.$$

Определение 1.2. *Дана конечная последовательность* $X_0, X_1, X_2, \ldots, X_{N\text{-}1}$ *комплексных чисел. Обратное дискретное преобразование Фурье заключается*

в поиске последовательности x_0, x_1, x_2, ..., x_{N-1}, элементы которой вычисляются по формуле:

$$x_k = \sum_{n=0}^{N-1} X_n e^{i\frac{2\pi kn}{N}} .$$

При помощи дискретного преобразования Фурье можно получить спектр сигнала, то есть коэффициенты при синусах и косинусах в разложении Фурье. Основным свойством преобразования Фурье (см., например, [7]) является обратимость, то есть если из последовательности $\{x_k\}$ при прямом преобразовании получается последовательность $\{X_k\}$, то при обратном преобразовании из $\{X_k\}$ получится исходная последовательность $\{x_k\}$.

Отметим, что данное преобразование имеет трудоемкость $O(N^2)$ для набора чисел длины N, однако его можно оптимизировать так, что трудоемкость составит $M^2 2^T + NT$, где $N = M2^T$ (см. [8]).

1.3. Преобразование и оценка коэффициентов

Пусть мы имеем массив коэффициентов $\{c_k\}$, $k = 0, 1, \ldots, N\text{-}1$, и этот массив *нормализован*, в том смысле, что $|c_k| < 2$, $k = 0, 1, \ldots, N\text{-}1$. Преобразуем этот массив в два новых массива $\{e_k\}$ и $\{m_k\}$ так, что

$$c_k = m_k \cdot 2^{-e_k}, \qquad \text{где } m_k \in [1,2),\ e_k \in \mathbf{Z}^+,\ \ k = 0, 1, \ldots, N\text{-}1. \qquad (1.3.1)$$

Заметим, что на ЭВМ стандартное представление числа с плавающей точкой в виде *мантиссы* и *экспоненты*, аналогично (1.3.1) с той лишь разницей, что в (1.3.1) экспонента берется со знаком ‘-‘.

Теперь нам нужно оценить точность хранения коэффициентов. Отметим во-первых, что коэффициенты с большим значением e_k можно отбросить, как близкие к нулю. Во-вторых, при огрублении массива экспонент $\{e_k\}$ на n бит мы можем получить ошибку, равную $c_k 2^n$. Ясно, что при больших c_k, мы получаем неприемлемую ошибку, то есть массив экспонент необходимо хранить точно. А вот при округлении мантиссы m_k на n бит мы получаем ошибку $2^{n-(e_k+l)}$, где l – это исходная разрядность мантиссы.

Будем оценивать разрядность хранения каждого элемента m_k во-первых, в зависимости от совокупности значений $\{e_k\}$, и во-вторых, от порядкового номера k в массиве. Поясним это на примере:

$$\mathrm{BitCount}(k) := \left[K \cdot \exp\{P \cdot e_{\min} - Q \cdot e_k\} \cdot \mathrm{F}(k)\right], \qquad \text{где } e_{\min} := \min_{0 \le k \le N-1} e_k .$$

Здесь функция BitCount(k) определяет разрядность хранения мантиссы m_k. Параметр K задает верхнюю границу разрядности хранения (например K = 16), а функции exp и F, могут уменьшить разрядность, т.к. 0≤P≤Q, 0≤F(k)≤1, где P и Q – константы.

Отметим некоторые моменты. Чем больше максимальный коэффициент, тем меньше у него $e_{\min}$, следовательно, тем меньше бит для хранения будет выделено для остальных коэффициентов. Это отражает тот факт, что на фоне доминирующей частоты, остальные частоты менее слышны, нежели в ее отсутствие.

Функция F отражает зависимость разрядности коэффициента от его положения в разложении, например, если рассматриваемые коэффициенты – это спектр разложения Фурье, то мы можем применить частотную фильтрацию. Допустим, если мы a priori знаем, что исходный сигнал содержит только низкие частоты, а остальные не важны или являются помехами, то положив

$$\mathrm{F}(k) := \begin{cases} 1, & \text{если } k < H, \\ 0, & \text{если } k \ge H, \end{cases}$$

мы зададим тем самым *низкочастотный фильтр* (*low-pass filter*), то есть фильтр, пропускающий только низкочастотную составляющую сигнала, это позволит существенно сократить объем выходных данных. Отметим, что описанная оценка разрядности называется *психоакустическим фильтром*.

1.4. Кодирование с предсказыванием по частичному совпадению

В ходе экспериментов было выявлено, что для обоих используемых преобразований (DWT, DFT) величины $\{e_k\}$ имеют стабильное распределение, близкое к нормальному. Это натолкнуло на мысль использовать какой-либо вероятностный метод кодирования. Мы провели анализ методов кодирования и

выбрали контекстный метод, основанный на предсказывании по частичному совпадению (Prediction by Partial Matching, PPM), который является надстройкой над методом арифметического кодирования. Описание арифметического кодирования можно найти в [9]. Описание PPM можно найти в [10].

1.5. Стерео соединение

Чаще всего, стерео сигнал содержит избыточную информацию, так как часть звуковой информации дублируется. В данной работе используется метод соединения каналов, с целью устранить избыточность информации.

Итак, пусть мы имеем два массива коэффициентов $\{c_k^l\}$ и $\{c_k^r\}$, $k = 0, 1, \ldots, N\text{-}1$. Преобразуем каждую пару (c_k^l, c_k^r) в другую пару (c_k, α_k) при помощи полярного преобразования координат. Далее, применим рассмотренную в пункте 1.3 схему для преобразования коэффициентов c_k. А для α_k применим следующий прием: мы масштабируем интервал изменения α_k к интервалу [1,2), и теперь α_k в записи (m_k, e_k) имеет значение $(\alpha_k, 0)$, где 0 нам вовсе не обязательно хранить. То есть в итоге мы получили три массива величин – $\{e_k\}$, $\{m_k\}$ и $\{\alpha_k\}$ – это массив экспонент, массив мантисс и *массив фаз*.

1.6. Битовые потоки

ЭВМ позволяет эффективно манипулировать данными разрядности 8, 16, 32 бита, но этот формат невыгоден для хранения коэффициентов разложения, так как у нас появляются не используемые разряды. Например, если функция оценки разрядности коэффициента m_k выдает значение 10, то записывая этот коэффициент в 16-разрядную ячейку, мы не используем оставшиеся 6 разрядов. Проблема решается путем введения битовых потоков, в которых числа разной разрядности хранятся «без зазоров». Это достигается при помощи арифметических сдвигов и логических операций.

1.7. Общая схема

Итак, рассмотрев по отдельности все шаги, поясним общую схему алгоритма сжатия сегмента звукового сигнала.

- На вход процедуры сжатия подается два массива $\{x_k^l\}$ и $\{x_k^r\}$, $k = 0$, 1, …, N-1 (рассмотрим случай со стерео сигналом).
- Применяем к каждому из массивов выбранное нами преобразование (будь то DWT или DFT) и получаем массивы коэффициентов $\{c_k^l\}$ и $\{c_k^r\}$.
- Применяем стерео соединение коэффициентов $\{c_k^l\}$ и $\{c_k^r\}$ и получаем три массива $\{e_k\}$, $\{m_k\}$ и $\{\alpha_k\}$.
- Оцениваем разрядность хранения величин $\{m_k\}$ и $\{\alpha_k\}$, используя массив экспонент $\{e_k\}$ (см. пункт 1.3).
- Выполняем PPM-кодирование массива $\{e_k\}$.
- Выполняем битовую упаковку массивов $\{m_k\}$ и $\{\alpha_k\}$ с нужным числом разрядов.

В результате получается блок, состоящий из 3-х подблоков.

PPM-код массива экспонент	битовый массив мантисс	битовый массив фаз

Алгоритм восстановления сигнала симметричен алгоритму сжатия, но с использованием обратных преобразований.

1.8. Устранение помех на границах сегментов

Так как в качестве базиса, как правило, берутся непрерывные функции, то, несмотря на округление коэффициентов, при восстановлении сегмента мы получаем непрерывную функцию. Но между сегментами может возникнуть «скачек», т.к. в конце текущего сегмента восстановленный сигнал является суперпозицией одних *компонент*, а в начале следующего сегмента – суперпозицией других компонент. Под компонентами здесь понимаются базисные функции, умноженные на соответствующие коэффициенты

разложения. И если значения сигнала на стыке двух сегментов совпадали до округления, то после округления они могут не совпадать. Таким образом, непрерывность исходного сигнала может быть утеряна.

Если рассматривать спектр восстановленного сигнала, то разрыв интерпретируется как высокочастотная осцилляция, и на слух он воспринимается, как щелчок. Такие щелчки сильно выделяются на фоне остальных погрешностей и выливаются в шум на протяжении всего восстановленного сигнала.

Эта проблема решается перекрыванием сегментов. При сжатии сегменты берутся с некоторым заступом на следующие, а при восстановлении применяется следующий прием: на протяжении участка перекрывания, амплитуда текущего восстановленного сегмента непрерывно уменьшается до нулевой, а амплитуда следующего восстановленного сегмента непрерывно увеличивается от нулевой амплитуды до исходного значения. Чтобы описать этот процесс, возьмем неубывающую функцию

$$w(x) \in C[0,1], \qquad \text{такую, что } w(0) = 0,\ w(1) = 1.$$

Пусть область перекрывания на временной оси – есть интервал $[a, b]$. Будем умножать амплитуду сигнала текущего сегмента в области перекрывания на $1 - w\left(\frac{x-a}{b-a}\right)$, а амплитуду сигнала следующего сегмента – на $w\left(\frac{x-a}{b-a}\right)$, после чего, эти амплитуды складываются. Этот процесс называется *кроссфэйдингом* (crossfading).

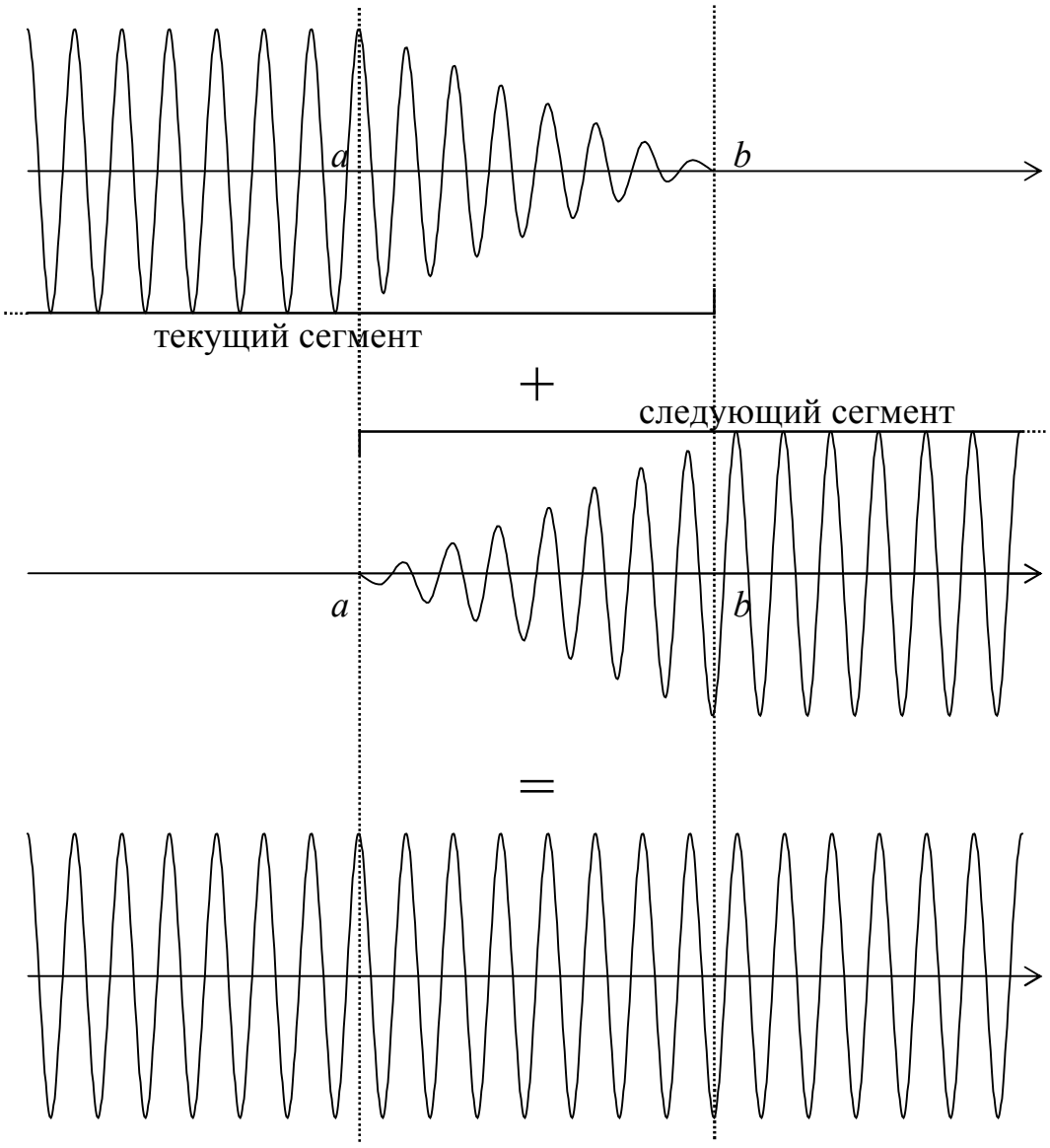


Рис. 1.2. Кроссфэйдинг в области перекрывания сегментов.

На рисунке 1.2 изображен восстановленный сигнал, который был сжат с использованием метода перекрывания сегментов; сигнал представляет собой синусоидальную волну, а в качестве $w(x)$ взята линейная функция. Таким образом, возможный разрыв устраняется, так как сигнал сегмента теперь представляет собой непрерывную функцию на временной оси, а восстановленный сигнал является композицией сигналов сегментов.

2. Деление сигнала на сегменты

2.1. Статическая реализация

Первый и самый простой способ заключается в выборе фиксированной длины сегмента, то есть на вход процедуре сжатия последовательно подаются сегменты одинаковой длины, а затем сегмент-остаток.

2.2. Адаптивная реализация

Второй способ заключается в выборе длины текущего сегмента на основе характеристик предыдущего. В качестве таких характеристик можно взять коэффициенты разложения предыдущего сегмента. В случае преобладания высоких частот можно уменьшать длину текущего сегмента для лучшей локализации высоких частот. В случае преобладания низких частот мы, наоборот, увеличиваем длину текущего сегмента для лучшей локализации низких частот. Этот метод используется почти во всех доминирующих форматах сжатия звука на сегодняшний день, таких как MP3 и Ogg Vorbis.

2.3. Динамическая реализация

Третий способ заключается в подборе такого разбиения, которое минимизирует суммарное количество байтов закодированного сигнала, то есть предоставление программе возможности «адаптироваться» к структуре сигнала. Это возможно в случае, если преобразование «чувствительно» к разбиению сигнала на сегменты. Поясним это свойство на примере.

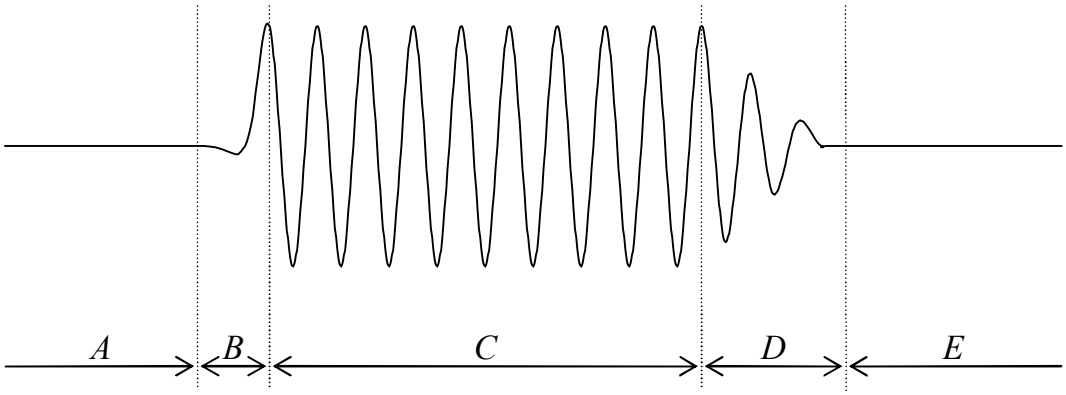


Рис. 2.1. Разбиение на сегменты.

На рисунке 2.1 изображена часть сигнала, разбитая на сегменты. Отметим, что для преобразования Фурье этот вариант разбиения будет самым эффективным среди всех возможных с точки зрения сжатия, так как сегмент *C* будет иметь всего один ненулевой коэффициент разложения, в *A* и *E* будут нули, и лишь сегментах *B* и *D* будут присутствовать разные составляющие спектра. Чтобы определить наиболее эффективное с точки зрения сжатия *разбиение*, можно использовать динамическое программирование.

Итак, нам необходимо определить разбиение, минимизирующее объем выходных данных – такую последовательность точек, что интервалы между соседними точками принимаются за сегменты звукового сигнала, подлежащие сжатию. Во-первых, отметим, что слишком большие длины сегментов брать бессмысленно, так как в случае преобразования Фурье, с ростом длины сегмента растут трудоемкость и погрешность (как отмечалось в пункте 1.2). Во-вторых, мы должны зафиксировать шаг для динамического программирования, который определяет длину минимального возможного сегмента в разбиении так, что длина каждого сегмента кратна этому шагу. Если фиксировать шаг равный 1, то при сжатии больших объемов аудио информации алгоритм становится очень трудоемким.

Введем некоторые обозначения:

h – длина минимально возможного сегмента.

n – количество шагов в сегменте максимальной возможной длины.

Таким образом, мы получили равномерную сетку на сигнале с шагом *h*:

$$w_h = \{x_k := kh |\ k = 0,1,...,m\}, \quad \text{где } m = \left[\frac{N}{h}\right].$$

Введем понятие пути P^k из x_0 в x_k:

$$P^k = \{p_i \in w_h | i = 0,1,...,M\}, \quad \text{причем } \begin{cases} p_0 = x_0, \\ h \le p_j\text{-}p_{j\text{-}1} \le nh, j = 1,...,M \\ p_M = x_k, \end{cases}$$

где $M = |P^k| - 1$ – число сегментов в пути P^k. Обратим внимание на то, что в определении пути присутствуют три свойства: длина каждого сегмента (p_{j-1}, p_j) пути P^k кратна шагу h, не равна нулю и не превышает длины максимально возможного сегмента nh.

Пусть вес сегмента (x_i, x_j) – это количество байт после сжатия сегмента сигнала, ограниченного этими точками, обозначим вес сегмента за $d(x_i, x_j)$. Тогда вес пути P^k равен сумме весов всех его сегментов:

$$d(P^k) = \sum_{i=1}^{M} d(p_{i-1}, p_i), \quad \text{где } M = |P^k| - 1.$$

Путь P^m из x_0 в конечную точку x_m можно интерпретировать как разбиение исходного сигнала. Отметим, что путь из x_0 в x_m минимального веса и будет искомым разбиением. Итак, можно ввести эквивалентную задачу на следующем графе:

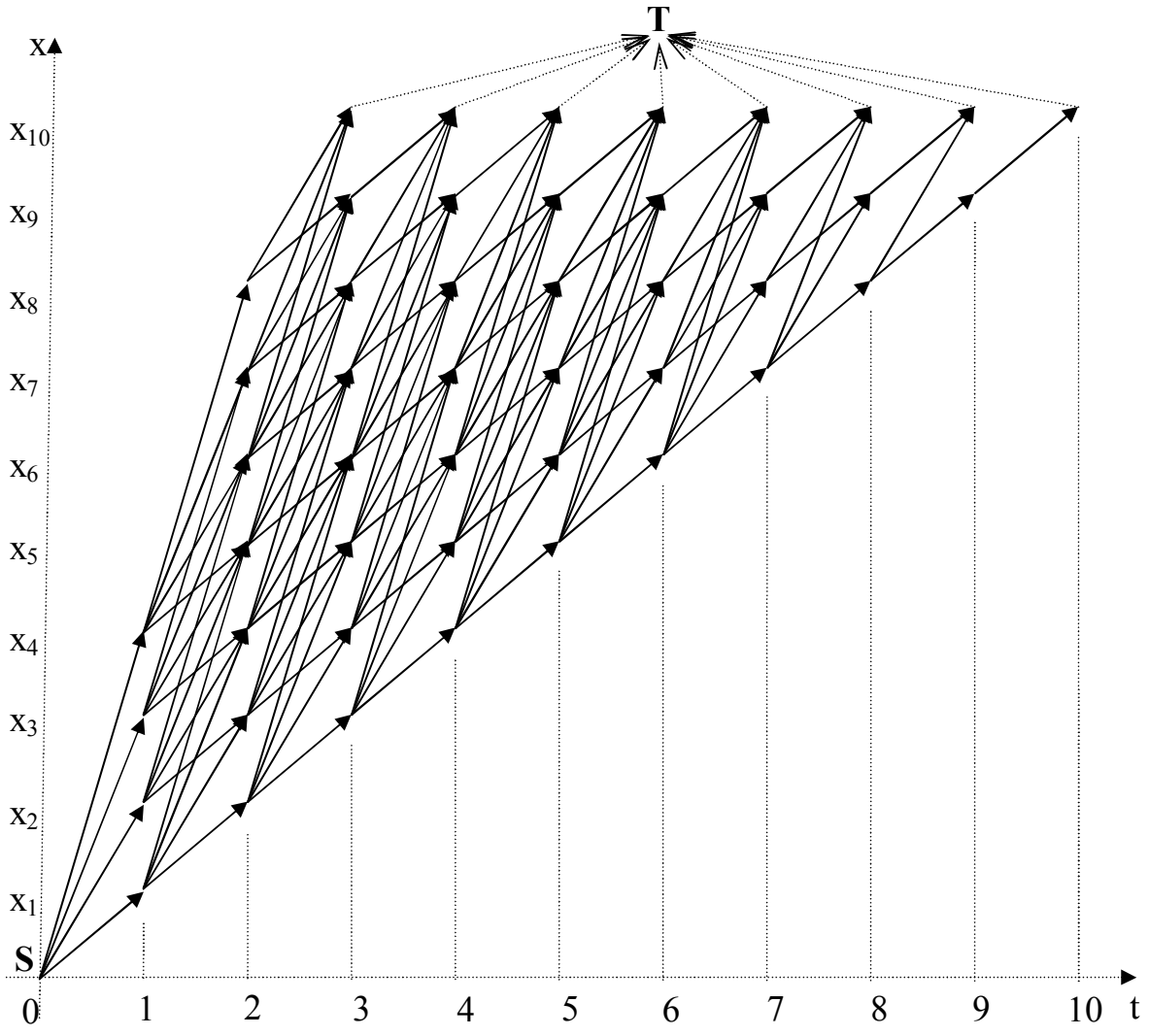


Рис. 2.2. Граф состояний.

На рисунке 2.2 изображен *граф состояний*, соответствующий конкретной задаче сжатия, в которой число шагов $m = 10$, количество шагов в максимально возможном сегменте $n = 4$.

Каждая вершина графа (x_k, t) характеризует множество путей из x_0 в x_k, которые состоят из t сегментов. Вес любого ребра $[(x_{k_1}, t_1), (x_{k_2}, t_2)]$ в графе определим как вес сегмента (x_{k_1}, x_{k_2}):

$$c[(x_{k_1}, t_1), (x_{k_2}, t_2)] = d(x_{k_1}, x_{k_2}).$$

Нам необходимо найти путь минимального веса до любой из вершин (x_{10}, t). Можно ввести фиктивные ребра, вес которых равен нулю (изображены пунктиром) до фиктивной вершины T, тогда задача сводится к нахождению пути минимального веса на графе (рис. 2.2) из S в T.

Вес ребер не зависит от числа сегментов t, и, по сути, нам не важно, сколько сегментов в пути, а важен его вес, поэтому все состояния, соответствующие фиксированному x_k и разным t можно считать тождественными, таким образом, можно рассматривать проекцию графа (рис. 2.2) на ось x:

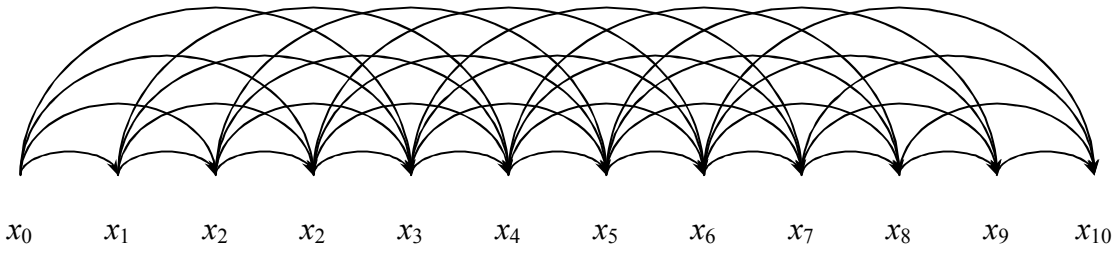


Рис. 2.3. Проекция графа состояний на ось x.

Задача сводится к нахождению пути минимального веса на графе (рис. 2.3) из x_0 в x_m. Будем решать задачу методом динамического программирования. Разобьем задачу на этапы. На k-м этапе необходимо найти путь минимального веса из x_0 в x_k, обозначим его через P^{k^*}, при условии, что P^{0^*}, P^{1^*}, …, P^{k-1^*} уже найдены. Запишем *уравнение Бэллмана*:

$$d(P^{k^*}) = \min_{i=1,\dots,n}\{d(P^{k-i^*}) + d(x_{k-i},x_k)\},$$

начальные условия:

$$\begin{cases} d(P^{0^*}) := 0, \\ d(P^{i^*}) := d(x_i,x_k) := +\infty, \quad \textit{для всех } i < 0, k \in \mathbf{Z}. \end{cases}$$

На k-м этапе у нас имеется n управлений, где i-е управление отвечает за выбор пути $P^{k^*} := P^{k-i^*} \Upsilon x_k$. Выбор оптимального управления осуществляется очевидным образом, в соответствии с уравнением Бэллмана, то есть

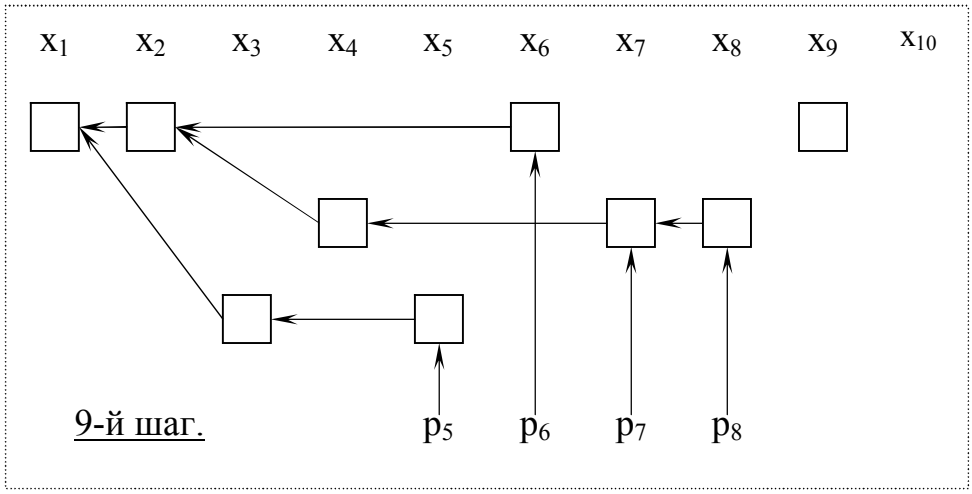
$$i = \arg\min_{j=1,\dots,n}\{d(P^{k-j^*}) + d(x_{k-j},x_k)\}.$$

Для определения величин $d(x_{k-j}, x_k)$, где $j = 1, 2, \dots, n$, нам необходимо провести сжатие n сегментов (x_{k-1}, x_k), (x_{k-2}, x_k), …, (x_{k-n}, x_k). Заметим, что на каждом шаге алгоритма Дейкстры при добавлении вершны x_k мы также должны сжать n сегментов (x_k, x_{k+1}), (x_k, x_{k+2}), …, (x_k, x_{k+n}). Следовательно, так как количество шагов алгоритма Дейкстры и описанного алгоритма равно m, то трудоемкости этих алгоритмов совпадают. Но описанный алгоритм более удобен в реализации, так как в алгоритме Дейкстры вершины могут добавляться непоследовательно.

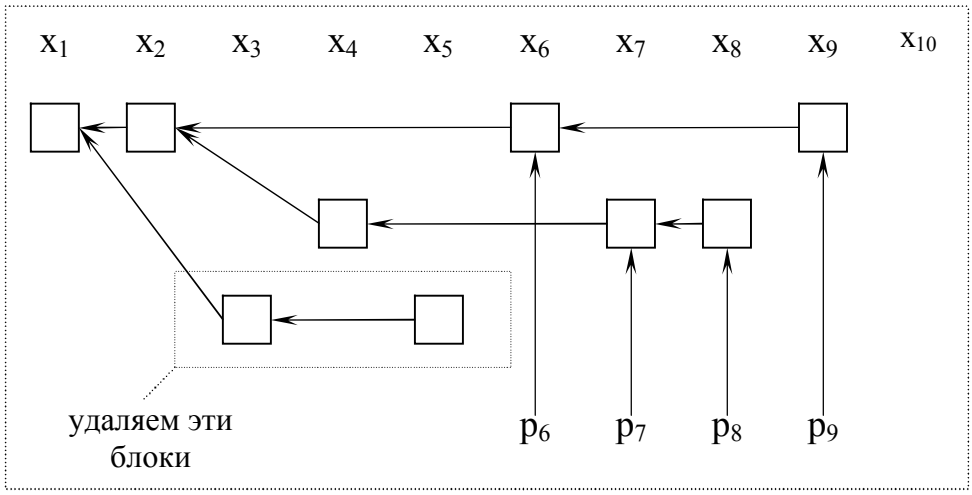
Сделаем несколько замечаний относительно реализации описанного алгоритма динамического программирования:

1. При сжатии сегментов, на каждом шаге формируются n блоков, содержащих сжатые сегменты сигнала, из которых выбирается один, а остальные удаляются.
2. В пути, фактически, не содержится точек разбиения, а содержится несколько таких блоков, связанных указателями, и мы храним только указатели на последние блоки путей.

3. При формировании нового пути, нам не нужно копировать все блоки из *k-j*-го пути, нам достаточно установить указатель *i*-го блока, на последний блок *k-j*-го пути, а указатель *k*-го пути на *j*-й блок.



4. На *k*-м шаге нам уже не нужны пути P^{0^*}, P^{1^*}, …, P^{k-n-1^*}, и после *k*-го шага, путь P^{k-n^*} уже не понадобится. Удалим те его блоки, на которые нет ссылок из других путей. Следовательно, на *k*-м шаге можно хранить только указатели на последние блоки последних *n* путей.



5. На каждом шаге блоки со ссылками образуют дерево и являются его узлами.

3. Программная реализация и вычислительный эксперимент

3.1. Программная реализация

В ходе решения поставленной задачи, были разработаны две библиотеки на C++.

- *WaveLib* – библиотека, позволяющая работать с файлами формата WAV. Реализована загрузка, сохранение, проигрывание, отображение контура волны и навигация.
- *ComprLib* – библиотека, предоставляющая набор средств для сжатия WAV-файлов, а также позволяющая разрабатывать собственные алгоритмы и совершенствовать имеющиеся, используя наследование и полиморфизм объектно-ориентированного программирования.

Библиотека **WaveLib**.

Файл "Wave.h":

CWave – класс обеспечивающий манипуляцию с WAV-файлами.

Файл "Player.h":

CPlayer – абстрактный класс, характеризующий интерфейс проигрывателя.

Файл "WavePlay.h":

CWavePlayer(CPlayer) – класс, реализующий проигрывание объектов типа CWave средствами DirectSound 8.0.

Файл "WaveView.h":

CWaveView(CView) – класс, производный от класса MFC-библиотеки, спроектированный для одновременного отображения нескольких объектов класса CWave, навигации, масштабирования и редактирования.

Файл “PeakData.h”:

CPeakData – вспомогательный класс, который используется в реализации CWaveView, предназначен для быстрого отображения (за логарифмическое время) звуковых файлов большого объема, основан на построении дополнительной информации об объекте класса CWave.

Библиотека **ComprLib**.

Файл “compression.h”:

CDecomposition – класс, хранящий сжатый сигнал в виде списка блоков, каждый из которых является прообразом сегмента при восстановлении.

CCompressManager – класс, отвечающий за разбиение сигнала на сегменты при сжатии.

CCompressor – абстрактный класс, отвечающий за сжатие и восстановление одного сегмента сигнала.

CBuffCompressor(CCompressor) – абстрактный класс поддерживающий рабочий буфер.

CTransform – абстрактный класс, характеризующий преобразование сегмента сигнала и оценку разрядности коэффициентов.

CTransformCompressor(CBuffCompressor) – класс, поддерживающий сжатие с помощью преобразования описанного в виде класса, реализующего интерфейс абстрактного класса CTransform, как это описано в пункте 1.7.

CStaticOverlapCompressor(CTransformCompressor) – класс, поддерживающий перекрывание сегментов, описанного в пункте 1.8.

Файл “FourierTransform.h”:

CFourierTransform(CTransform) – класс поддерживающий оптимизированное преобразование Фурье (см. пункт 1.2).

Файл “WaveletTransform.h”:

CWaveletTransform(CTransform) – абстрактный класс, характеризующий вэйвлет-преобразование и оценку разрядности вэйвлет-коэффициентов.

COrthoWaveletTransform(CWaveletTransform) – класс, оптимизированный под ортогональное вэйвлет-разложение.

CEBiorWaveletTransform(CWaveletTransform) – класс, оптимизированный под биортогональное вэйвлет-разложение, учитывающий симметрию последовательностей разложения и восстановления с четным числом элементов.

COBiorWaveletTransform(CWaveletTransform) – класс, оптимизированный под биортогональное вэйвлет-разложение, учитывающий симметрию последовательностей разложения и восстановления с нечетным числом элементов.

Файл “Managers.h”:

CLinearCompressManager(CCompressManager) – класс, обеспечивающий разбиение сигнала на сегменты по методу, описанному в пункте 2.1.

CDynamicCompressManager(CCompressManager) – класс, обеспечивающий разбиение сигнала на сегменты по методу, описанному в пункте 2.3.

Файл “PPMCoder.h”:

CPPMCoder – класс, реализующий PPM-кодирование последовательности целых чисел.

Файл “RangeCoder.h”:

CRangeCoder – класс, реализующий арифметическое кодирование целых чисел.

Файл “bitstream .h”:

bitstream – класс, реализующий битовую упаковку целых чисел.

Как было сказано ранее, библиотека ComprLib допускает расширение за счет

- введения новых преобразований – классов, производных от CTransform
- введения новых компрессоров – классов, производных от CCompressor, CBuffCompressor, CTransformCompressor или CStaticOverlapCompressor
- введение новых классов, отвечающих за разбиение сигнала на сегменты (производных от CCompressManager).

Можно также получать разные способы сжатия, комбинируя нововведения с возможностями, предоставляемыми библиотекой, так как любая тройка объектов классов, производных соответственно от CTransformCompressor, CCompressManager и CTransform может быть использована для сжатия объектов класса CWave в объекты класса CDecomposition и восстановления объектов класса CDecomposition до объектов класса CWave.

Разработана программа, которая имеет удобный пользовательский интерфейс, реализованный средствами библиотеки WaveLib, и позволяет сжимать WAV-файлы с помощью библиотеки ComprLib (см. рис. 3.1).

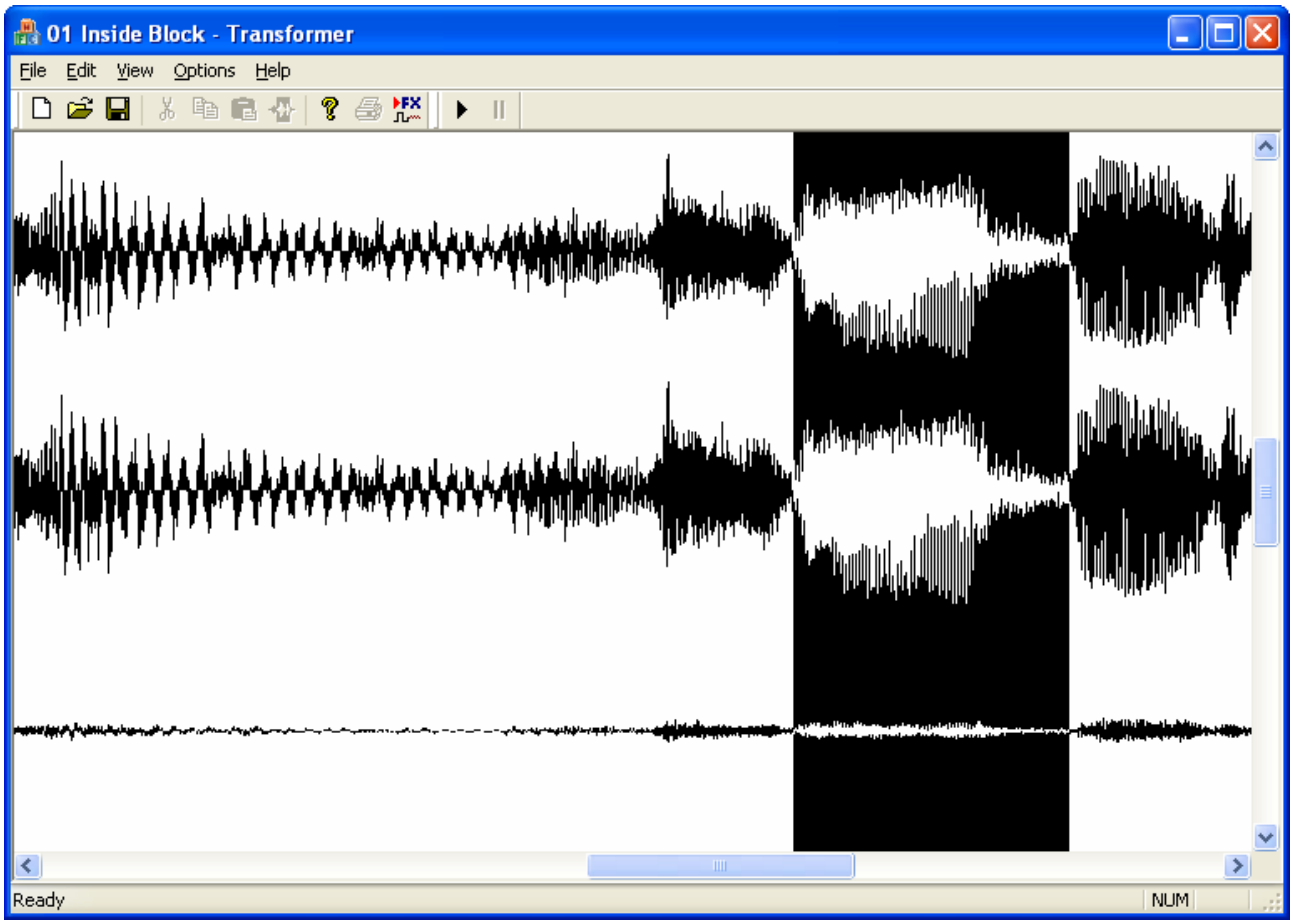


Рис. 3.1. Программа, реализующая интерфейс средствами библиотеки WaveLib.

3.2. Вычислительный эксперимент

Для проверки эффективности различных способов сжатия были взяты два звуковых файла формата WAV. Причем для обоих файлов использовалась одна и та же функция оценки разрядности хранения коэффициентов.

В первом эксперименте был взят сильно насыщенный сигнал, в том смысле, что в спектре сигнала присутствуют частоты широкого диапазона (часть композиции Inside – *Sting (2003) Sacred Love*) . В таблице приведены результаты проведенного эксперимента:

Преобразование	Динамическое программирование	Коэффициент сжатия (от исходного объема)
DWT	не использовалось	22.1%
DWT	использовалось	22.0%
DFT	не использовалось	23.7%
DFT	использовалось	21.5%

Во втором эксперименте был взят несильно насыщенный сигнал, в том смысле, что в спектре сигнала присутствует только узкий диапазон частот. (часть композиции Falling – *Alicia Keys (2001) Songs in A minor*).

Преобразование	Динамическое программирование	Коэффициент сжатия (от исходного объема)
DWT	не использовалось	19.9%
DWT	использовалось	19.8%
DFT	не использовалось	20.7%
DFT	использовалось	18.9%

Как видно из экспериментов, для вэйвлет-разложения динамическое программирование не дает большого выигрыша в объеме и качестве, так как сама структура вэйвлет-преобразования такова, что осцилляции сигнала локализуются независимо от разбиения на сегменты. Разложение Фурье, как это было описано в пункте 2.3, зависит от разбиения сигнала на сегменты; поэтому динамическое программирование дает некоторый выигрыш, как в объеме, так и в качестве звучания.

Укажем некоторые замечания относительно реализации:

- Массив экспонент достаточно хранить с точностью 3-4 бита. Если отводить под элемент массива 4 бита и хранить массив экспонент несжатым, то его объем составляет 17.5% от объема исходного сигнала. Однако PPM-код массива занимает в среднем 8% от объема исходного сигнала. Следовательно, мы добились сжатия массива экспонент «без потерь» более чем в два раза.
- Заметим, что можно подобрать такие допустимые оценки точности элементов массивов мантисс и фаз, что суммарный объем, занимаемый этими массивами составит 16-17% от исходного. Причем эти массивы хранятся в несжатом виде, а подвергаются лишь битовой упаковке. Если мы допустим, что эти массивы можно сжать каким-либо алгоритмом кодирования в два раза, как и массив экспонент, то сжатый сигнал будет занимать около 15% от исходного объема. Этот результат близок к результатам сжатия лучших кодировщиков на сегодняшний день, таких, как Ogg Vorbis и MPEG Layer-3. Например, если сжать WAV-файл в MP3 с качеством 192kbps, то объем MP3-файла составит 14% от исходного.

Заключение

В данной работе проведен обзор основных концепций вэйвлет-анализа, запрограммированы алгоритмы разложения и восстановления, исследованы особенности вэйвлет-разложений.

Реализован способ разбиения сигнала на сегменты основанный на методе динамического программирования. Предложена схема сжатия сегмента сигнала, особенность которой состоит в разложении массива коэффициентов на массив мантисс и массив экспонент.

Разработаны программные библиотеки для манипуляции с WAV-файлами и их сжатия, а также Windows-приложение, реализующее возможности этих библиотек. Достигнутый коэффициент сжатия при допустимой потере качества составляет 20-25% от исходного объема.

Приложение. Введение в вэйвлет-анализ

Обозначим через $L^2(0,2\pi)$ множество всех интегрируемых с квадратом измеримых функций, определенных на интервале $(0,2\pi)$. Любую функцию из $L^2(0,2\pi)$ можно представить рядом Фурье

$$f(x) = \sum_{n=-\infty}^{\infty} c_n e^{inx}, \tag{П.1}$$

где константы c_n, называемые коэффициентами Фурье, определяются формулой

$$c_n = \frac{1}{2\pi}\int_0^{2\pi} f(x)e^{-inx}dx. \tag{П.2}$$

Имеются две явные особенности разложений в ряды Фурье (П.1). Первая особенность состоит в том, что f разлагается в бесконечную сумму ортогональных компонент, т.к. функции

$$\omega_n(x) := e^{inx}, \qquad n \in \mathbf{Z}$$

образуют ортонормированный (о.н.) базис в $L^2(0,2\pi)$. Ортогональность означает, что

$$\langle \omega_m, \omega_n \rangle^* = 0, \qquad \text{для всех} \quad m \neq n$$

со скалярным произведением, определенным формулой:

$$\langle \omega_m, \omega_n \rangle^* = \frac{1}{2\pi}\int_0^{2\pi} \omega_m(x)\overline{\omega_n(x)}dx$$

Вторая особенность разложения в ряд Фурье (П.1) состоит в том, что о.н. базис $\{\omega_n\}$ порождается растяжением единственной функции

$$\omega(x) := e^{ix},$$

так что $\omega_n(x) := \omega(nx)$ для всех целых n. Это будет называться в дальнейшем *целочисленным растяжением*.

Подводя итог, можно сказать, что *каждая функция из* $L^2(0,2\pi)$ *порождается «суперпозицией» целочисленных растяжений базисной функции* $\omega(x) = e^{ix}$.

Обратим внимание на то, что базисная функция

$$\omega(x) = e^{ix} = \cos x + i \sin x$$

является синусоидальной волной. Чем больше абсолютная величина n, тем более высокую частоту имеет волна $\omega_n(x) := \omega(nx)$. Таким образом, каждая функция из $L^2(0,2\pi)$ состоит из волн различных частот.

Далее рассмотрим пространство $L^2(\mathbf{R})$ интегрируемых с квадратом измеримых функций, определенных на вещественной оси **R**. Ясно, что два пространства $L^2(0,2\pi)$ и $L^2(\mathbf{R})$ совершенно различны. В частности, каждая функция (ее локальное среднее значение) из $L^2(\mathbf{R})$ должна «затухать» до нуля при x, стремящемся к $\pm\infty$, но синусоидальные «волны» функции $\omega_n(x)$ не принадлежат $L^2(\mathbf{R})$. В сущности, если мы хотим использовать «волны», порождающие $L^2(\mathbf{R})$, то эти волны должны были бы затухать до нуля при $x \to \pm\infty$, и из всех практических соображений это затухание должно быть очень быстрым. Так, мы приходим к рассмотрению малых волн, или *вэйвлетов*, для порождения $L^2(\mathbf{R})$. Так же, как и в случае с $L^2(0,2\pi)$, где одна функция $\omega(x) = e^{ix}$ порождает целое пространство, мы предпочитаем иметь одну функцию для порождения всего $L^2(\mathbf{R})$ и будем обозначать ее через ψ. Но если вэйвлет ψ имеет очень быстрое затухание, то как он может покрыть всю вещественную ось? Очевидным способом является сдвиг ψ вдоль **R**.

Простейший способ для ψ покрыть все множество **R** состоит в рассмотрении *целочисленных сдвигов* ψ, а именно,

$$\psi(x-k), \quad k \in \mathbf{Z}.$$

Затем, так же как и в синусоидальном случае, мы должны рассматривать волны различных частот. Ради вычислительной эффективности мы будем использовать для частотного разбиения целые степени 2. В результате мы рассматриваем малые волны

$$\psi(2^j x - k), \qquad j, k \in \mathbf{Z}.$$

Заметим, что $\psi(2^j x - k)$ получена из одной «вэйвлет-функции» ψ в результате *двоичного растяжения* (т.е. растяжения в 2^j раз) и *двухпараметрического сдвига* (на $k / 2^j$).

Итак, нас интересуют «вэйвлет-функции» ψ, двоичные растяжения и двухпараметрические сдвиги которых достаточны для представления любой функции из $L^2(\mathbf{R})$.

Далее будем использовать следующие обозначения для *скалярного произведения и нормы* в пространстве $L^2(\mathbf{R})$:

$$\langle f, g\rangle := \int_{-\infty}^{\infty} f(x)\overline{g(x)}dx;$$

$$\| f \|_2 := \langle f, f\rangle^{1/2},$$

где $f, g \in L^2(\mathbf{R})$. Заметим, что для любых $j, k \in \mathbf{Z}$ мы имеем

$$\| f(2^j \cdot - k) \|_2 = \left[\int_{-\infty}^{\infty} | f(2^j x - k) |^2 \, dx\right]^{1/2} = 2^{-j/2} \| f \|_2 .$$

Следовательно, если функция $\psi \in L^2(\mathbf{R})$ имеет единичную норму, то все функции $\psi_{j,k}$, определенные формулой

$$\psi_{j,k}(x) := 2^{j/2}\psi(2^j x - k), \qquad j, k \in \mathbf{Z}, \tag{П.3}$$

также имеют единичную норму, то есть

$$\| \psi_{j,k} \|_2 = \| \psi \|_2 = 1, \qquad j, k \in \mathbf{Z},$$

Определение 1.1. *Функция $\psi \in L^2(\mathbf{R})$ называется R-функцией, если $\{\psi_{j,k}\}$, как это определено в (П.3), является базисом Рисса в $L^2(\mathbf{R})$, в том смысле, что линейная оболочка $\psi_{j,k}$, $j, k \in \mathbf{Z}$, плотна в $L^2(\mathbf{R})$ и что существуют положительные константы A и B, $0 < A \le B < \infty$ такие, что*

$$A \| \{c_{j,k}\} \|_{l^2}^2 \le \left\| \sum_{j=-\infty}^{\infty} \sum_{k=-\infty}^{\infty} c_{j,k}\psi_{j,k} \right\|_2^2 \le B \| \{c_{j,k}\} \|_{l^2}^2$$

для всех бесконечных суммируемых с квадратом последовательностей $\{c_{j,k}\}$, то есть

$$\| \{c_{j,k}\} \|_{l^2}^2 := \sum_{j=-\infty}^{\infty} \sum_{k=-\infty}^{\infty} | c_{j,k} |^2 < \infty .$$

Определение 1.2. *Пусть $\psi \in L^2(\mathbf{R})$ является R-функцией, которая порождает $\{\psi_{j,k}\}$, определенное формулой (П.3). Тогда ψ называется*

ортогональным вэйвлетом (или о.н. вэйвлетом), если семейство $\{\psi_{j,k}\}$, *удовлетворяет условию ортогональности:*

$$\langle \psi_{j,k}, \psi_{l,m} \rangle = \delta_{j,l} \cdot \delta_{k,m}, \qquad j, k, l, m \in \mathbf{Z};$$

это означает, что любая $f \in L^2(\mathbf{R})$ *может быть представлена как*

$$f(x) = \sum_{j,k=-\infty}^{\infty} c_{j,k} \psi_{j,k}(x), \tag{П.4}$$

где ряд (П.4) сходится в $L^2(\mathbf{R})$, *а именно*

$$\lim_{M_1, N_1, M_2, N_2 \to \infty} \left\| f - \sum_{j=-M_2}^{N_2} \sum_{k=-M_1}^{N_1} c_{j,k} \psi_{j,k} \right\|_2 = 0.$$

Простейшим примером ортогонального вейвлета является функция Хаара ψ_H, определенная формулой

$$\psi_H(x) := \begin{cases} 1 & для\ 0 \le x < \frac{1}{2}, \\ -1 & для\ \frac{1}{2} \le x < 1, \\ 0 & в\ других\ случаях. \end{cases}$$

Ряды, представляющие функции f в (П.4), называются *вэйвлет-рядами*. Аналогично обозначению коэффициентов Фурье в (П.2) вэйвлет-коэффициенты определяются формулой

$$c_{j,k} = \langle f, \psi_{j,k} \rangle \qquad j, k \in \mathbf{Z}. \tag{П.5}$$

Однако чтобы получить вэйвлет-ряд функции f, условие ортогональности вэйвлета ψ не обязательно.

Определение 1.3. *R-функция* $\psi \in L^2(\mathbf{R})$ *называется R-вэйвлетом (или вэйвлетом), если существует функция* $\tilde{\psi} \in L^2(\mathbf{R})$ *такая, что соответствующие семейства* $\{\psi_{j,k}\}$ *и* $\{\tilde{\psi}_{j,k}\}$ *удовлетворяют условию биортогональности:*

$$\langle \psi_{j,k}, \tilde{\psi}_{l,m} \rangle = \delta_{j,l} \cdot \delta_{k,m}, \qquad j, k, l, m \in \mathbf{Z};$$

Если ψ *-* ***R****-вэйвлет, то* $\widetilde{\psi}$ *называют двойственным вэйвлетом, соответствующим* ψ.

Заметим, что если ψ - ортогональный вэйвлет, то он является двойственным самому себе, в том смысле, что $\widetilde{\psi} = \psi$. Если ψ - вэйвлет с двойственным $\widetilde{\psi}$, то по определению базиса Рисса, каждая $f \in L^2(\mathbf{R})$ может быть записана как

$$f(x) = \sum_{j,k\in\mathbf{Z}} c_{j,k}\psi_{j,k}(x) = \sum_{j,k\in\mathbf{Z}} d_{j,k}\widetilde{\psi}_{j,k}(x).$$

Эти вэйвлет-ряды сходятся в $L^2(\mathbf{R})$. Из условия биортогональности следует, что

$$\begin{cases} c_{j,k} = \langle f, \widetilde{\psi}_{j,k}\rangle, \\ d_{j,k} = \langle f, \psi_{j,k}\rangle. \end{cases}$$

Пусть ψ - любой вэйвлет, рассмотрим порожденный им базис Рисса $\{\psi_{j,k}\}$. Для каждого $j \in \mathbf{Z}$ обозначим через W_j замыкание линейной оболочки $\{\psi_{j,k} : k \in \mathbf{Z}\}$, а именно:

$$W_j := \operatorname{clos}_{L^2(\mathbf{R})}(\psi_{j,k} : k \in \mathbf{Z}). \qquad (\text{П.6})$$

Очевидно, $L^2(\mathbf{R})$ может быть разложено в *прямую сумму* подпространств W_j:

$$L^2(\mathbf{R}) = \dot{\sum_{j\in\mathbf{Z}}} W_j := \ldots \dot{+} W_{-1} \dot{+} W_0 \dot{+} W_1 \dot{+} \ldots \qquad (\text{П.7})$$

в том смысле, что любую $f \in L^2(\mathbf{R})$ можно единственным образом представить в виде суммы

$$f(x) = \ldots + g_{-1}(x) + g_0(x) + g_1(x) + \ldots,$$

где $g_i \in W_j$ для всех $j \in \mathbf{Z}$. Точки над знаком суммирования указывают на то, что берутся «прямые суммы».

Если ψ - ортогональный вэйвлет, то подпространства W_j из $L^2(\mathbf{R})$ взаимно ортогональны и прямая сумма в (П.7) становится *ортогональной суммой*. Однако, любой вэйвлет, ортогональный или нет, порождает разложение $L^2(\mathbf{R})$ в прямую сумму подпространств (П.7). Для каждого $j \in \mathbf{Z}$ будем рассматривать замкнутые подпространства

$$V_j := \ldots \dot{+} W_{j-2} \dot{+} W_{j-1}, \qquad j \in \mathbf{Z},$$

в $L^2(\mathbf{R})$. Ясно, что эти пространства обладают следующими свойствами:

(1°) $\ldots \subset V_{-1} \subset V_0 \subset V_1 \subset \ldots$,

(2°) $clos_{L2}\,\mathrm{clos}_{L^2(\mathbf{R})}\left(\bigcup_{j\in\mathbf{Z}} V_j\right) = L^2(\mathbf{R})$,

(3°) $\bigcap_{j\in\mathbf{Z}} V_j = \{0\}$,

Следовательно, в противоположность пространствам W_j, которые удовлетворяют соотношению

$$W_j \cap W_l = \{0\}, \qquad j \neq l,$$

подпространства V_j вложены друг в друга, как это описано в условии (1°) и обладают тем свойством, что любая функция f из $L^2(\mathbf{R})$ может быть приближена с произвольной точностью ее проекциями $P_j f$ на V_i, что следует из условия (2°). С другой стороны, с уменьшением j проекции $P_j f$ могут иметь сколь угодно малую норму в $L^2(\mathbf{R})$, что обусловлено условием (3°).

Предположим, что по аналогии с (П.6) пространства V_j порождены единственной функцией $\phi \in L^2(\mathbf{R})$ в смысле, что

$$V_j := \mathrm{clos}_{L^2(\mathbf{R})}(\phi_{j,k} : k \in \mathbf{Z}) \qquad (\text{П.8})$$

где

$$\phi_{j,k}(x) := 2^{j/2}\phi(2^j x - k).$$

Определение 1.4. *Функция $\phi \in L^2(\mathbf{R})$ называется масштабирующей функцией, если она порождает подпространства V_j в смысле (П.8), которые удовлетворяют условиям* (1°) - (3°)*, и $\{\phi_{0,k}\}$ является базисом Рисса пространства V_0.*

Как было упомянуто выше, ввиду условия (2°) каждая f из $L^2(\mathbf{R})$ с любой желаемой точностью может быть приближена функцией $f_N \in V_N$ при некотором $N \in \mathbf{Z}$. Так, как $V_j = V_{j-1} \dot{+} W_{j-1}$, для любого $j \in \mathbf{Z}$, f_N имеет единственное разложение:

$$f_N = f_{N-1} + g_{N-1},$$

где $f_{N-1} \in V_{N-1}$ и $g_{N-1} \in W_{N-1}$. Повторяя этот процесс, имеем:

$$f_N = g_{N-1} + g_{N-2} + ... + g_{N-M} + f_{N-M}\,, \qquad \text{(П.9)}$$

где $f_j \in V_j$ и $g_j \in W_j$ для любого j. Один из критериев остановки процесса состоит в требовании, чтобы $\|f_{N-M}\|$ было меньше некоторого порога.

Так как масштабирующая функция $\phi \in V_0$ и вэйвлет $\psi \in W_0$ принадлежат V_1, а V_1 порождено $\phi_{1,k}(x) = 2^{1/2}\phi(2x-k)$, $k \in \mathbf{Z}$, то существуют две последовательности $\{p_k\}$ и $\{q_k\} \in l^2$ такие, что

$$\phi(x) = \sum_k p_k \phi(2x-k), \qquad \text{(П.10)}$$

$$\psi(x) = \sum_k q_k \psi(2x-k) \qquad \text{(П.11)}$$

для всех $x \in \mathbf{R}$. Формулы (П.10) и (П.11) называются *двухмасштабными соотношениями* масштабирующей функции и вэйвлета, соответственно. С другой стороны, так как $\phi(2x)$ и $\phi(2x\text{-}1)$ принадлежат V_1 и $V_1 = V_0 \dot{+} W_0$ то существуют четыре последовательности из l^2, которые мы обозначаем как $\{a_{-2k}\}$, $\{b_{-2k}\}$, $\{a_{1-2k}\}$, $\{b_{1-2k}\}$, $k \in \mathbf{Z}$, такие что

$$\phi(2x) = \sum_k [a_{-2k}\phi(x-k) + b_{-2k}\psi(x-k)], \qquad \text{(П.12)}$$

$$\phi(2x-1) = \sum_k [a_{1-2k}\phi(x-k) + b_{1-2k}\psi(x-k)] \qquad \text{(П.13)}$$

для всех $x \in \mathbf{R}$. Две формулы (П.12) и (П.13) могут быть объединены в одну:

$$\phi(2x-l) = \sum_k [a_{l-2k}\phi(x-k) + b_{l-2k}\psi(x-k)], \quad l \in \mathbf{Z}, \qquad \text{(П.14)}$$

которая называется соотношением разложения для ϕ и ψ. Теперь мы имеем две пары последовательностей $(\{p_k\},\{q_k\})$ и $(\{a_k\},\{b_k\})$, которые единственны для данных ϕ и ψ. Эти последовательности используются для последующих алгоритмов разложения и восстановления. Поэтому, $\{p_k\}$ и $\{q_k\}$ называются последовательностями восстановления, тогда как $\{a_k\}$ и $\{b_k\}$ – последовательности разложения.

Итак, пусть $f_N \in V_N$ аппроксимирует $f \in L^2(\mathbf{R})$ с нужной нам точностью. С одной стороны

$$f_N(x) = \sum_{k\in\mathbf{Z}} A_k^N \phi_{N,k}(x), \qquad \text{(П.15)}$$

с другой стороны, так как $V_N = V_{N-1} \dot{+} W_{N-1}$,

$$f_N(x) = f_{N-1}(x) + g_{N-1}(x) = \sum_{k\in\mathbf{Z}} A_k^{N-1}\phi_{N-1,k}(x) + \sum_{k\in\mathbf{Z}} D_k^{N-1}\psi_{N-1,k}(x)$$

где $f_{N-1} \in V_{N-1}$ и $g_{N-1} \in W_{N-1}$. Связь между последовательностями коэффициентов A^N, A^{N-1} и D^{N-1} найдем из соотношения разложения (П.14):

$$\sum_{l\in\mathbf{Z}} A_l^N \phi_{N,l}(x) = \sum_{l\in\mathbf{Z}} A_l^N \sum_k [a_{l-2k}\phi_{N-1,k}(x) + b_{l-2k}\psi_{N-1,k}(x)] =$$

$$= \sum_{k\in\mathbf{Z}} \left(\sum_l a_{l-2k} A_l^N \right) \phi_{N-1,k}(x) + \sum_{k\in\mathbf{Z}} \left(\sum_l b_{l-2k} A_l^N \right) \psi_{N-1,k}(x) =$$

$$= \sum_{k\in\mathbf{Z}} A_k^{N-1}\phi_{N-1,k}(x) + \sum_{k\in\mathbf{Z}} D_k^{N-1}\psi_{N-1,k}(x).$$

Теперь видно, что

$$\begin{cases} A_k^{N-1} = \sum_l a_{l-2k} A_l^N; \\ D_k^{N-1} = \sum_l b_{l-2k} A_l^N. \end{cases} \quad \text{или} \quad \begin{cases} A_k^{N-1} = \sum_l a_l A_{2k+l}^N; \\ D_k^{N-1} = \sum_l b_l A_{2k+l}^N. \end{cases} \qquad \text{(П.16)}$$

Продолжаем этот процесс для A_{N-1}, потом для A_{N-2} и так далее до A_{N-M+1}.

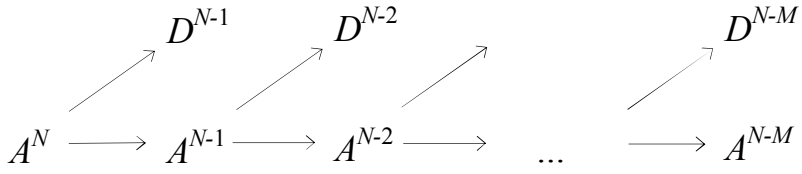


Рис. П.1. Вэйвлет-разложение.

Заметим, что обе последовательности A^{N-1} и D^{N-1} получаются из A^N по схеме *скользящего среднего*, с использованием последовательностей разложения в качестве «весов» с той особенностью, что эти скользящие средние вычисляются только в четных точках, что видно из (П.16). Такое преобразование называется сгущающей выборкой. Поэтому каждая стрелка на рис. П.1 указывает на скользящее среднее со сгущающей выборкой.

Рассмотрим алгоритм восстановления. Мы имеем последовательности коэффициентов A^{N-M}, D^{N-M}, D^{N-M+1}, …, D^{N-1}. Нам необходимо восстановить последовательность коэффициентов A^N. Для того чтобы найти обратную связь между A^{N-M}, D^{N-M} и A^{N-M+1} обратимся к двухмасштабным соотношениям (П.10) и (П.11). Для любого j, $V_j \dot{+} W_j = V_{j+1}$, поэтому

$$\sum_{l\in\mathbf{Z}} A_l^{j-1}\phi_{j-1,l}(x) + \sum_{l\in\mathbf{Z}} D_l^{j-1}\psi_{j-1,l}(x) =$$

$$= \sum_{l\in\mathbf{Z}} A_l^{j-1}\sum_k p_{k-2l}\phi_{j,k}(x) + \sum_{l\in\mathbf{Z}} D_l^{j-1}\sum_k q_{k-2l}\phi_{j,k}(x) =$$

$$= \sum_{k\in\mathbf{Z}}\left(\sum_l p_{k-2l}A_l^{j-1}\right)\phi_{j,k}(x) + \sum_{k\in\mathbf{Z}}\left(\sum_l q_{k-2l}D_l^{j-1}\right)\phi_{j,k}(x) = \sum_{k\in\mathbf{Z}} A_k^j\phi_{j,k}(x),$$

откуда видно, что

$$A_k^j = \sum_l [p_{k-2l}A_l^{j-1} + q_{k-2l}D_l^{j-1}].$$

В алгоритме восстановления j пробегает от N-M+1 до N.

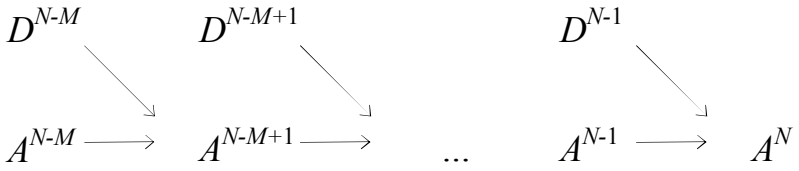


Рис. П.2. Вэйвлет-восстановление.

Здесь A^j получается из A^{j-1} и D^{j-1} с помощью двух скользящих средних, использующих последовательности восстановления в качестве «весов», с той особенностью, что разрежающая выборка должна быть выполнена до реализации скользящих средних.

Мы заканчиваем этот параграф несколькими замечаниями о двух приведенных выше алгоритмах. Во-первых, если весовая последовательность ($\{a_k\}$,$\{b_k\}$,$\{p_k\}$ или $\{q_k\}$) конечна, то соответствующий алгоритм скользящего среднего – это очень простой фильтр с конечным импульсным откликом (КИО-фильтр). Если, однако, весовая последовательность бесконечна, то соответствующий алгоритм скользящего среднего есть фильтр с бесконечным

импульсным откликом (БИО-фильтр). В англоязычной литературе эти фильтры называются соответственно FIR-filter (Finite Impulse Response filter) и IIR-filter (Infinite Impulse Response filter). Для практического использования БИО-фильтра в алгоритмах, он должен быть усечен до КИО-фильтра. Во-вторых, если весовая последовательность состоит из иррациональных чисел или чисел, с большим числом знаков после запятой, то необходимо их округление («квантование»). Конечно, усечение и квантование влекут за собой погрешности, которые должны быть оценены a priori.

Список литературы

1) **Нагорный А.** Vorbis против всех? Или какой кодек выбрать для сжатия аудио // портал Hardvision, 2004 (http://www.hardvision.ru/?dir=soft&doc=ogg_vorbis).
2) **Alexander C., Strauss N.,** The Ogg Vorbis CODEC project, Xiph.Org., 2003 (http://www.xiph.org/ogg/vorbis/).
3) **Hardle W., Kerkyacharian G., Picard D., Tsybakov. A.** Wavelets, Approximation, and Statistical Applications (Lecture Notes in Statistics, Vol 129). New York: Springer-Verlag, 1997.
4) **Алексеев К.А.** Теория и практика шумоподавления в задаче обработки сейсмоакустических сигналов // Обработка сигналов и изображений. Wavelet Toolbox, Консультационный центр Matlab, 2004. (http://matlab.exponenta.ru/wavelet/book5/index.php).
5) **Алексеев К.А.** Вейвлеты, аппроксимация и статистические приложения // Обработка сигналов и изображений. Wavelet Toolbox, Консультационный центр Matlab, 2004. (http://matlab.exponenta.ru/wavelet/book6/index.php).
6) **Таха Х.** Введение в исследование операций. М.: изд. дом “Вильямс”, 2001.
7) **Чуи К.** Введение в вейвлеты М. «Мир», 2001.
8) **Войнаровский М.** Психологика // Быстрое преобразование Фурье, 2002-2003 (http://psi-logic.narod.ru/fft/fft.htm).
9) **Кантор И.** Алгоритмы и методы (http://algolist.manual.ru/compress/standard).
10) **Смирнов М.** Введение в PPM (http://www.compression.ru/download/articles/ppm/smirnov_2000_ppm_faq.html).