УДК 004.67, 004.021

ВСТРАИВАНИЕ ЦИФРОВОГО ВОДЯНОГО ЗНАКА В АУДИОСИГНАЛ С ПРИМЕНЕНИЕМ БЫСТРОГО ПРЕОБРАЗОВАНИЯ ФУРЬЕ

Светов Леонид Андреевич

Студент;
ГБОУ ВПО «Международный Университет природы, общества и человека «Дубна»,
Институт системного анализа и управления;
141980, Московская обл., г. Дубна, ул. Университетская, 19;
e-mail: lenzo@list.ru.

Статья посвящена вопросам внедрения цифровых водяных знаков в аудиосигналы с применением быстрого преобразования Фурье. В ней рассматриваются методы вычисления быстрого преобразования Фурье, быстрого косинусного преобразования и особенности их программной реализации на языке С++. На основе этих преобразований выбрана методика добавления водяного знака в звуковой сигнал. Статья содержит описание и анализ результатов ее работы.

Ключевые слова: цифровой водяной знак, аудиосигнал, быстрое преобразование Фурье, быстрое косинусное преобразование.

DIGITAL AUDIO WATERMARKING USING FAST FOURIER TRANSFORM

Svetov Leonid[1]

Student;
Dubna International University of Nature, Society and Man,
Institute of system analysis and management;
141980, Dubna, Moscow reg., Universitetskaya str., 19;
e-mail: lenzo@list.ru.

This article is devoted to issues of digital audio watermarking based on the fast Fourier transform. It discusses methods of calculating the fast Fourier transform, the fast cosine transform and features of their software implementation in C++ language. A technique of audio watermarking was chosen based on these transformations. The article contains a description and analysis of results of its work.

Keywords: *digital watermark, audio signal, fast Fourier transform, fast discrete cosine transform*.

Введение

Задача защиты информации от несанкционированного доступа решалась во все времена на протяжении истории человечества. Уже в древнем мире выделилось два основных направления решения этой задачи, существующие и по сегодняшний день: криптография и стеганография. Целью криптографии является скрытие содержимого сообщений за счет их шифрования. В отличие от этого, при стеганографии скрывается сам факт существования тайного сообщения.

Для защиты персональной информации люди использовали самые разнообразные приемы. Сначала они были довольно примитивными, но с развитием технологий постепенно усложнялись. В век информационных технологий и вычислительной техники развитие стеганографии вышло на принципиально новый этап, который называют компьютерной стеганографией. Скрываемые сообщения стали встраивать в цифровые данные, как правило, мультимедийного характера (изображения, звук, видео).

За последние несколько десятилетий были написаны множество работ в области так называемых цифровых водяных знаков (ЦВЗ). ЦВЗ – специальная метка, незаметно внедряемая в изображение или другой сигнал с целью тем или иным образом контролировать его использование. Это

направление стеганографии помогает решать проблемы защиты авторских прав на произведения. Чтобы файл, представляющий собой объект авторского права, не мог быть изменен без ведома автора, чтобы он содержал всю необходимую информацию о правомерном использовании, в него добавляется ЦВЗ. Если произведение подвергается какому-то изменению, то вместе с ним изменяется и водяной знак. Главные требования к ЦВЗ – это его невидимость и устойчивость к внешним воздействиям или помехами (робастность). Эти требования необходимы для того, чтобы усложнить задачу злоумышленнику, стремящемуся уничтожить ЦВЗ.

Существует множество способов добавления ЦВЗ в защищаемые файлы. Когда целью защиты является файл, содержащий оцифрованный аналоговый сигнал, например, звук, то один из возможных подходов в таком случае – применение аддитивных алгоритмов встраивания ЦВЗ. Эти алгоритмы основываются на линейной модификации исходного сигнала, а извлечение ЦВЗ производится корреляционными методами. При этом ЦВЗ обычно складывается с сигналом-контейнером, либо «вплавляется» (*fusion*) в него.

Попытки удаления ЦВЗ из аудиосигнала могут быть основаны на его перекодировании в другие форматы со сжатием или выборочном изменении частот. Смысл в таких действиях имеется, если, конечно, после них контейнер знака похож на исходный вариант, т.е. различия между исходным сигналом и сигналом с удаленным ЦВЗ незаметны [1].

Теоретические основы

В статье рассматривается аддитивный алгоритм внедрения ЦВЗ, основанный на преобразовании Фурье. На сегодняшний день оно является одним из широко распространенных инструментов анализа, применяющимся во всех отраслях науки и техники. В основе преобразования Фурье (ПФ) лежит довольно простая и полезная идея – всякой функции вещественной переменной можно сопоставить другую функцию вещественной переменной, которая будет описывать коэффициенты разложения исходной функции на гармонические колебания с различными частотами. Иными словами, любую функцию вещественной переменной можно представить в виде суммы синусоид с кратными амплитудами, периодами и частотами (рис. 1). Классическое преобразование Фурье задается следующим интегралом [1]:

$$f(u) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} f(x) e^{-jxu} dx .$$

Это преобразование идентифицирует частоты и амплитуды тех комплексных синусоид, на которые разлагается некоторое колебание. Оно определено для функций из пространства L_2.

Помимо обычного ПФ, также существуют различные его модификации. Одна из них – это дискретное преобразование Фурье. ДПФ широко применяется в алгоритмах цифровой обработки сигналов (его модификации применяются при сжатии звука в формат *mp3*, сжатии изображений в формат *JPEG* и др.), а также в других областях, связанных с анализом частот в дискретном (к примеру, оцифрованном аналоговом) сигнале. ДПФ на входе принимает дискретную функцию, которой ставит в соответствие другую дискретную функцию. Такие функции могут быть получены путем выборки значений из непрерывных функций.

Преобразование Фурье оперирует с комплексными числами, однако в реальных задачах обычно требуется работать с действительными числами. Очевидно, что прямое и обратное преобразования действительного сигнала вернут в итоге этот действительный сигнал. Однако после внесения изменений в частотную область сигнала обратное преобразование может дать комплексные коэффициенты с ненулевой мнимой частью. Эти изменения влекут в общем случае ненулевую мнимую часть преобразования. Таким образом, нам необходимо такое преобразование, которое не выходило бы за рамки действительных чисел.

Обратим внимание на два важных свойства преобразования Фурье: если исходная функция четная, то коэффициенты ее разложения имеют нулевые мнимые части, и выражение идет по косинусам

$$f(x)=\sqrt{\frac{2}{\pi}}\int_0^{\infty} f(t)\cos(xt)dt,$$

а если функция нечетная, то разложение идет по синусам, и коэффициенты имеют нулевые действительные части:

$$f(x)=\sqrt{\frac{2}{\pi}}\int_0^{\infty} f(t)\sin(xt)dt.$$

Эти выражения называется соответственно косинусным и синусным преобразованием. Рассмотрим подробнее дискретное косинусное преобразование (ДКП, *Discrete cosine transform*, *DCT*). Данное преобразование раскладывает сигнал на косинусы с различной амплитудой и частотой. Одна из особенностей ДКП состоит в том, что некоторые локальные участки сигнала можно охарактеризовать небольшим количеством коэффициентов преобразования. Это свойство очень часто используется при разработке методов сжатия данных (например, изображений). Так, ДКП является основой международного стандарта, который используется в алгоритме сжатия изображений с потерями *JPEG*.

Разделяют несколько разновидностей ДКП [2]. Наиболее распространенные из них – *DCT*2 и *DCT*3. Это обуславливается тем, что им присуще свойство сохранения основной информации в небольшом числе коэффициентов. Это свойство называют «уплотнением» энергии.

$$DCT2: \quad X[k]=\sum_{n=0}^{N-1} x[n]\cos\left(\frac{\pi(2n+1)k}{2N}\right), \quad k=0,1,...,N-1,$$

$$DCT3: \quad X[k]=\frac{1}{2}x[0]+\sum_{n=1}^{N-1} x[n]\cos\left(\frac{\pi n}{N}\left(k+\frac{1}{2}\right)\right), \quad k=0,1,...,N-1.$$

*DCT*3 есть не что иное как обратное преобразование для *DCT*2 (*Inverse discrete cosine transform*, *IDCT*).

Итак, прямое одномерное ДКП имеет вид:

$$X[k]=a[k]\sum_{n=0}^{N-1} x[n]\cos\left(\frac{\pi(2n+1)k}{2N}\right), \quad k=0,1,...,N-1,$$

обратное ДКП:

$$x[n]=\sum_{k=0}^{N-1} a[k]X[k]\cos\left(\frac{\pi(2n+1)k}{2N}\right), \quad n=0,1,...,N-1,$$

где

$$a[k]=\begin{cases}\sqrt{\frac{1}{N}} & k=0\\ \sqrt{\frac{2}{N}} & k=1,2,...,N-1\end{cases}.$$

Данное преобразование можно определить в терминах $2N$ –точечного ДПФ. Для этого введем новую последовательность $x'[m]$ длины $2N$, которая получается из исходного сигнала $\{x[m], \quad m=0,1,...,N-1\}$ следующим образом:

$$x'[m]=\begin{cases}x[m] & 0\le m\le N-1\\ x[-m-1] & -N\le m\le -1\end{cases}.$$

Предполагается, что последовательность $x'[m]$ повторяется за пределами интервала $-N \le n \le N-1$, т.е. является периодической с $T = 2N$ и симметрична относительно точки $m = -1/2$:

$$x'[m] = x'[-m-1] = X'[2N-m-1].$$

Если теперь сдвинуть $x'[m]$ на $1/2$ вправо, что эквивалентно сдвигу m на $1/2$ влево путем объявления новой переменной $m' = m + 1/2$, то $x'[m] = x'[m' - 1/2]$, что делает последовательность $x'[m]$ симметричной относительно $m' = 0$. Наглядно такой переход показан на рис. 1. Для простоты в дальнейшем будем понимать под $x[m]$ нашу новую последовательность $x'[m]$.

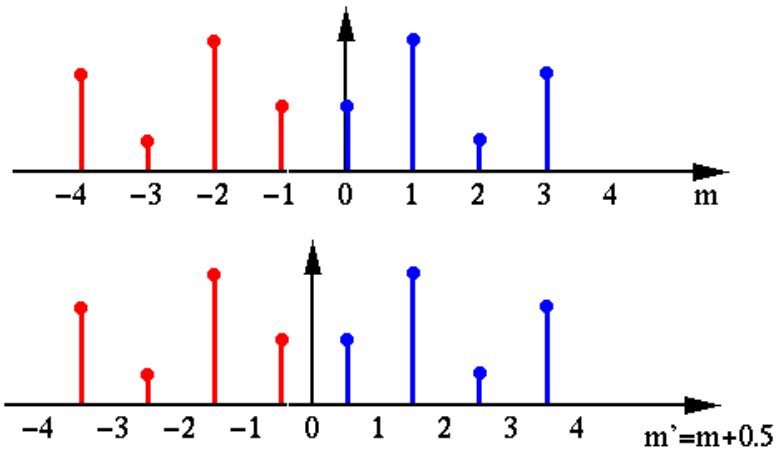

Рис. 1. Отражение сигнала и его сдвиг

ДПФ от этой четно-симметричной последовательности может быть найдена так:

$$X[n] = \frac{1}{\sqrt{2N}} \sum_{m'=-N+1/2}^{N-1/2} x\left[m' - \frac{1}{2}\right] e^{\frac{-j2\pi m' n}{2N}} =$$

$$= \frac{1}{\sqrt{2N}} \sum_{m'=-N+1/2}^{N-1/2} x\left[m' - \frac{1}{2}\right] \cos\left(\frac{2\pi m' n}{2N}\right) - \frac{j}{\sqrt{2N}} \sum_{m'=-N+1/2}^{N-1/2} x\left[m' - \frac{1}{2}\right] \sin\left(\frac{2\pi m' n}{2N}\right) =$$

$$= \frac{1}{\sqrt{2N}} \sum_{m'=-N+1/2}^{N-1/2} x\left[m' - \frac{1}{2}\right] \cos\left(\frac{2\pi m' n}{2N}\right), \quad n = 0,1,\ldots,2N-1.$$

Т.к. функция $x[m]$ четная, $\cos\left(\frac{2\pi m' n}{2N}\right)$ и $\sin\left(\frac{2\pi m' n}{2N}\right)$ соответственно четная и нечетная функции относительно $m' = 0$ или $m = -1/2$, вторая сумма равна нулю. Также следует отметить, что функция $X[n]$ действительная и четная, т.е. $X[n] = X[-n]$.

Заменяя m' на $m + 1/2$ получаем выражение для ДКП:

$$X[n]=\frac{1}{\sqrt{2N}}\sum_{m'=-N+1/2}^{N-1/2}x\left[m'-\frac{1}{2}\right]\cos\left(\frac{2\pi m' n}{2N}\right)=$$
$$=\frac{1}{\sqrt{2N}}\sum_{m=-N}^{N-1}x[m]\cos\left(\frac{\pi(2m+1)n}{2N}\right)=$$
$$=\sqrt{\frac{2}{N}}\sum_{m=0}^{N-1}x[m]\cos\left(\frac{\pi(2m+1)n}{2N}\right),\quad n=0,1,...,2N-1.$$

Т.к. все суммируемые элементы четно симметричны, достаточно использовать только половину точек. Более того, т.к. косинус есть четная функция, $X[n]=X[-n]$ также четная последовательность с периодом $2N$, получаем:

$$X[N+n]=X[N+n-2N]=X[N-n]=X[n-N],$$

заметив, что точки $X[N+n],\quad n=0,1,...,N-1$ из второй половины совпадают с точками $X[N-n]$ из первой половины, т.е. вторая половина не нужна и поэтому может быть отброшена.

Таким образом, получаем выражение для ДКП:

$$X[n]=\sqrt{\frac{2}{N}}\sum_{m=0}^{N-1}x[m]\cos\left(\frac{\pi(2m+1)n}{2N}\right)=\sum_{m=0}^{N-1}x[m]c[n,m],\quad n=0,1,...,N-1.$$

Коэффициент $c[n,m]$ есть элемент матрицы косинусного преобразования:

$$c[n,m]=\sqrt{\frac{2}{N}}\cos\left(\frac{(2m+1)n\pi}{2N}\right),\quad m,n=0,1,...,N-1.$$

Все векторы этой матрицы, составленные из ее строк, ортогональны и нормированы кроме первого вектора ($n=0$):

$$\sqrt{\sum_{m=0}^{N-1}c^2[n,m]}=\sqrt{\frac{2}{N}\sum_{m=0}^{N-1}\cos^2\left(\frac{(2m+1)n\pi}{2N}\right)}=\begin{cases}\sqrt{2} & n=0\\ 1 & n=1,2,...,N-1\end{cases}.$$

Чтобы сделать ДКП ортонормированным введем дополнительный коэффициент:

$$a[n]=\begin{cases}\sqrt{\frac{1}{N}} & n=0\\ \sqrt{\frac{2}{N}} & n=1,2,...,N-1\end{cases}.$$

Тогда выражение для ДКП примет вид:

$$X[n]=a[n]\sum_{m=0}^{N-1}x[m]\cos\left(\frac{(2m+1)n\pi}{2N}\right)=a[n]\sum_{m=0}^{N-1}x[m]c[n,m],\quad n=0,1,...,N-1,$$

где коэффициенты $c[n,m]$ включают в себя $a[n]$ и являются элементами матрицы

$$\begin{bmatrix}\dots & \dots & \dots\\ \dots & c[n,m] & \dots\\ \dots & \dots & \dots\end{bmatrix}=\begin{bmatrix}\mathbf{c}_0^T\\ \vdots\\ \mathbf{c}_{N-1}^T\end{bmatrix}=\mathbf{C}^T.$$

Здесь $\mathbf{c}_i^T = [c[i,0], c[i,1], ..., c[i,N-1]]$ – i-я строка матрицы $\mathbf{C}$. Так как эти строки (векторы) ортогональны, т.е.

$$(c_i, c_j) = c_i^T c_j = \delta_{ij} = \begin{cases} 1 & i = j \\ 0 & i \neq j \end{cases},$$

то матрица $\mathbf{C}$ ортогональна:

$$\mathbf{C}^{-1} = \mathbf{C}^T, \quad \mathbf{C}^T\mathbf{C} = \mathbf{E}.$$

Таким образом, ДКП может быть записано в матричной форме:

$$\mathbf{X} = \mathbf{C}^T\mathbf{x}.$$

Обратное ДКП имеет вид:

$$x[m] = \sum_{n=0}^{N-1} X[n]c[m,n] = \sum_{n=0}^{N-1} a[n]X[n]\cos\left(\frac{(2m+1)n\pi}{2N}\right) =$$
$$= \sum_{n=0}^{N-1} X[n]c[m,n], \quad m = 0,1,...,N-1,$$

что в матричном виде выглядит следующим образом:

$$\mathbf{x} = \left(\mathbf{C}^T\right)^{-1}\mathbf{X} = \mathbf{C}\mathbf{X}.$$

Как видно из формул, сложность алгоритма, вычисляющего ДКП и обратное ДКП, можно оценить в $\Theta(N^2)$. Такой сложностью можно пренебречь только при малых значениях N. Решающим фактором применимости таких преобразований является существование быстрого алгоритма, который позволяет снизить вычислительную сложность с $\Theta(N^2)$ операций, которые необходимы для умножения вектора на матрицу.

Задача вычисления ДКП по традиционным формулам довольно проста и не требует дополнительного объяснения. За счет высокой сложности этого алгоритма скорость его работы резко падает при увеличении объема входных данных. Поэтому необходим способ, реализующий эти выражения за меньшее время – быстрое косинусное преобразования (БКП). Подсчет БКП – нетривиальная задача, и она имеет несколько вариантов реализаций. В статье рассматривается методика вычисления БКП, имеющая лучшие оценки скорости выполнения среди остальных алгоритмов [4, 5, 6].

Один из способов вычислить ДКП – взять за основу алгоритм, вычисляющий ДПФ. Такая схема имеет смысл в том случае, когда сложность нахождения ДПФ меньше $\Theta(N^2)$. В противном случае мы не получим выгоды, т.к. формулы явного ДКП также имеют сложность $\Theta(N^2)$. Выражения для классического ДПФ нам не подходят из-за большого времени выполнения. Нас интересует алгоритм, требующий меньшего времени. Таковым является быстрое преобразования Фурье или БПФ (*Fast Fourier transform*, *FFT*). Оно имеет вычислительную сложность $\Theta(N\log_2 N)$. Соответственно, чтобы быстро выполнить ДКП (сделать быстрое косинусное преобразование или БКП) нужно взять за основу БПФ. Сложность подсчета ДКП имея найденный вектор Фурье-коэффициентов равна $\Theta(N)$. Суммарная сложность подсчета ДКП равна $\Theta(N\log_2 N) + \Theta(N) = \Theta(N\log_2 N)$. Таким образом, используя БПФ для нахождения БКП суммарная сложность нахождения последнего напрямую зависит от сложности БПФ.

В статье рассматривается способ, основанный на алгоритме Кули-Тьюки (*Cooley-Tukey algorithm*), который использует свойства комплексных корней из единицы. Впрочем, это не единственный метод подсчета БПФ. Так, например, относительно недавно был предложен новый метод вычислений называемый *sFFT* (*Sparse Fast Fourier Transform*). Этот алгоритм быстро отыскивает

фрагменты с «разреженным» сигналом (*sparse signal*) и определяет исходную амплитуду в каждом из них. Сигнал разбивается на фрагменты до тех пор, пока не останется разреженный сигнал с единственной амплитудой. А уже там новый алгоритм выявляет её во много раз быстрее классического БПФ [7].

Ниже представлен код процедуры *fftRecursive*, вычисляющей БПФ [8, 9]. Флаг *inverse* показывает какое преобразование необходимо вычислить (обратное или прямое).

```
1       vector<cmplx> fftRecursive(vector<cmplx> &a, const bool
2       inverse)
3       {
4               int N=(int)a.size();
5               if (N==1)
6                       return a;
7
8               int N2=N/2;
9               double f=2*pi/N*(inverse?-1:1);
10              cmplx w(1),wn(cos(f),sin(f));
11
12              vector<cmplx> a0(n2),a1(n2);
13              for (int i=0;i<N2;i++)
14              {
15                      a0[i]=a[i<<1];
16                      a1[i]=a[(i<<1)+1];
17              }
18              vector<cmplx> y(N);
19              vector<cmplx> y0=fftRecursive(a0,inverse);
20              vector<cmplx> y1=fftRecursive(a1,inverse);
21
22              double ang=2*pi/N*(inverse?-1:1);
23              for (int k=0;k<N2;k++)
24              {
25                      y[k]=y0[k]+w*y1[k];
26                      y[k+n2]=y0[k]-w*y1[k];
27                      if (inverse)
28                      {
29                              y[k]/=2;
30                              y[k+n2]/=2;
31                      }
32                      w*=wn;
33              }
34              return y;
        }
```

Введем обозначение. Пусть $A(x)$ — многочлен степени меньше N:

$$A(x)=\sum_{i=0}^{N-1} a_i x^i .$$

Выделим отдельно члены четных и нечетных степеней

$$A(x)=A^{[0]}(x^2)+xA^{[1]}(x^2),$$

где

$$A^{[0]}(x)=\sum_{i=0}^{N/2-1} a_{2i} x^i ,$$

$$A^{[1]}(x)=\sum_{i=0}^{N/2-1} a_{2i+1} x^i .$$

В строках 3–5 формируется базис рекурсии. ДПФ для вектора длины 1 есть сам этот вектор, т.к.

$$y_0 = a_0 \omega_1^0 = a_0 1 = a_0 ,$$

где

$$\omega_x = e^{j2\pi/x},$$

а ω_x^n — комплексный корень степени n из x.

В строках 12–16 формируются два новых вектора, первый из которых содержит элементы исходного вектора с четными индексами, а второй – с нечетными.

К моменту выполнения строк 28 и 29, величина ω_n^i уже посчитана. За счет строки 9 ее не нужно вычислять каждый раз заново; отсюда экономия по времени.

В строках 18–19 рекурсивно вычисляются значения:

$$y_k^{[0]} = A^{[0]}\left(\omega_{N/2}^k\right),$$
$$y_k^{[1]} = A^{[1]}\left(\omega_{N/2}^k\right).$$

Используя свойство

$$\omega_{N/2}^k = \omega_N^{2k}$$

получаем

$$y_k^{[0]} = A^{[0]}\left(\omega_N^{2k}\right),$$
$$y_k^{[1]} = A^{[1]}\left(\omega_N^{2k}\right).$$

В строках 24–25 собираются результаты рекурсивных вычислений из строк 18–19. Для $y_0, y_1, ..., y_{N/2-1}$ и $k = 0,1,...,N/2-1$ получается:

$$y_k = y_k^{[0]} + \omega_N^k y_k^{[1]} = A^{[0]}\left(\omega_N^{2k}\right) + \omega_N^k A^{[1]}\left(\omega_N^{2k}\right) = A\left(\omega_N^k\right).$$

Для $y_{N/2}, y_{N/2+1}, ..., y_{N-1}$ и $k = 0,1,...,N/2-1$, используя свойства $\omega_N^{k+N/2} = -\omega_N^k$, $\omega_N^N = 1$, $\omega_N^{2k} = \omega_N^{2k+N}$, получается:

$$\begin{aligned} y_{k+(N/2)} &= y_k^{[0]} + \omega_N^k y_k^{[1]} = y_k^{[0]} + \omega_N^{k+N/2} y_k^{[1]} = \\ &= A^{[0]}\left(\omega_N^{2k}\right) + \omega_N^{k+N/2} A^{[1]}\left(\omega_N^{2k}\right) = \\ &= A^{[0]}\left(\omega_N^{2k+N}\right) + \omega_n^{k+N/2} A^{[1]}\left(\omega_N^{2k+N}\right) = \\ &= A\left(\omega_N^{k+N/2}\right). \end{aligned}$$

Время, за которое процедура *fftRecursive* вычисляет БПФ, можно оценить как $T(N) = 2T(N/2) + \Theta(N) = \Theta(N \log N)$.

Рассмотренный метод можно оптимизировать, заменив рекурсивный алгоритм итеративным. Последний имеет лучшие константы в асимптотической оценке $\Theta(N \log_2 N)$.

Для начала можно убрать лишнее вычисление $\omega_N^k y_k^{[1]}$ введя дополнительную переменную. Получим:

```
1       for (int k=0;k<N/2-1;k++)
2       {
3               t=w*y1[k];
4               y[k]=y0[k]+t;
5               y[k+n/2]=y0[k]-t;
6       }
```

Подобное преобразование часто называют преобразованием бабочки из-за внешнего вида схемы вычислений (рис. 2).

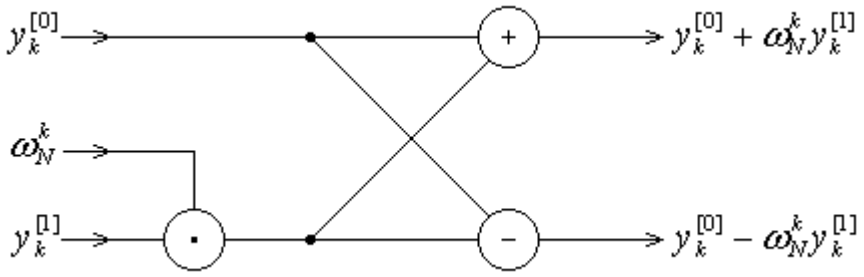

Рис. 2. Преобразование бабочки

Теперь можно избавиться от рекурсии, заменив ее вычислением снизу вверх. Для этого расположим элементы в таком порядке, как в нижних листьях дерева на рис. 3.

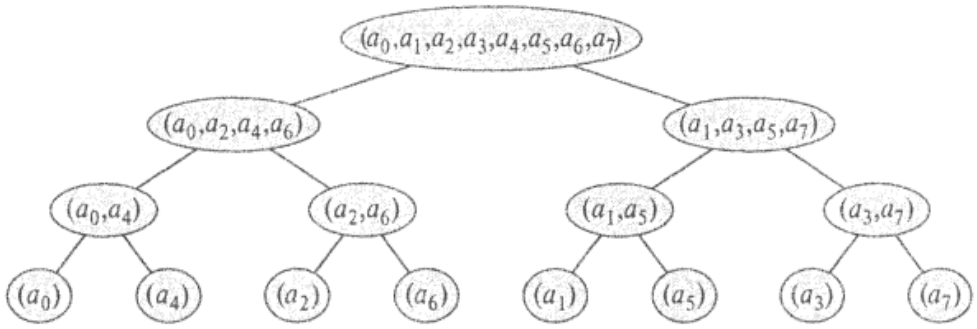

Рис. 3. Дерево входных веторов для процедуры fftRecursive при N = 8

Затем для пар элементов, стоящих на нижней строке дерева, вычислим ДПФ с помощью преобразования бабочки. Результатом будут значения второй снизу строки (полученные элементы можно записать на место исходного вектора, т.к. он больше не понадобится). Теперь из каждых двух пар векторов собираем две четверки и для каждой выполняем преобразование бабочки. Далее из полученных четверок собираются восьмерки и т.д. В общем случае на последнем шаге из двух векторов длины $N/2$, являющихся преобразованиями Фурье четной и нечетной части вектора a, получается преобразование Фурье всего вектора a.

Перед началом реализации нового алгоритма, необходимо перегруппировать элементы исходного вектора в следующем порядке: 0, 4, 2, 6, 1, 5, 3, 7 (при $N = 8$). Заметим, что эти индексы могут быть получены инверсией (реверсом) битов нормальной последовательности $0,1,...,N-1$ (см. табл. 1).

Таблица 1. Соответствие индексов с обычным и обратным порядком битов

0 — 000	000 — 0
1 — 001	100 — 4
2 — 010	010 — 2
3 — 011	110 — 6
4 — 100	001 — 1
5 — 101	101 — 5
6 — 110	011 — 3
7 — 111	111 — 7

Функцию, инвертирующую последовательность битов в исходном числе, можно записать так:

```
1       inline int reverseBits(int x, const int l)
2       {
3               int res=0;
4               for (int j=0;j<l;j++)
5               {
6                       res<<=1;
7                       res|=x&1;
8                       x>>=1;
9               }
10              return res;
11      }
```

где x – исходное число, а l – «ширина» инверсии. Заметим, что $l = \log_2 N$.

Для дополнительной оптимизации можно выполнить следующие действия [14]:

- вычислить заранее реверс битов для всех чисел (т.к. в программе длина вектора, над которым выполняется преобразование, есть константа, это легко сделать);
- вычислить заранее все степени ω;
- использовать собственный класс для представления комплексных чисел;
- перейти от хранения данных в vector к хранению в обычном массиве (эффект от замены зависит от компилятора).

Также можно отдельно реализовать БПФ в виде явных формул для малых значений, например для $N = 4$.

Как правило, изобретение этого метода в начале 1960-х годов приписывают ученым Кули и Тьюки. В действительности этот метод неоднократно встречался и раньше, однако его важность стала ощутима только с появлением современных вычислительных машин. Считается, что этот алгоритм знали Рунге и Кёниг еще в 1924 году.

Быстрое косинусное преобразование

Научившись вычислять БПФ, можно составить алгоритм для подсчета БКП. Два этих преобразования тесно связаны между собой, и БКП вектора $\{x[m], \quad m = 0,1,...,N-1\}$ может быть получено через его БПФ.

Для начала введем новый вектор $\{y[m], \quad m = 0,1,...,N-1\}$ следующим образом:

$$\begin{cases} y[m] = x[2m] \\ y[N-1-m] = x[2m+1] \end{cases} \quad (m = 0,1,...,N/2-1).$$

Тогда БКП для вектора $x[m]$ можно записать так (для простоты опустим коэффициенты нормировки):

$$X[n] = \sum_{m=0}^{N-1} x[m] \cos\left(\frac{(2m+1)n\pi}{2N}\right) =$$

$$= \sum_{m=0}^{N/2-1} x[2m] \cos\left(\frac{(4m+1)n\pi}{2N}\right) + \sum_{m=0}^{N/2-1} x[2m+1] \cos\left(\frac{(4m+3)n\pi}{2N}\right) =$$

$$= \sum_{m=0}^{N/2-1} y[m] \cos\left(\frac{(4m+1)n\pi}{2N}\right) + \sum_{m=0}^{N/2-1} y[N-1-m] \cos\left(\frac{(4m+3)n\pi}{2N}\right),$$

где первая сумма идет по четным индексам, а вторая по нечетным. Для второй суммы введем $m' = N-1-m$, тогда пределы суммирования 0 и $N/2-1$ для m становятся равными соответственно $N-1$ и $N/2$. Таким образом, вторая сумма принимает вид:

$$\sum_{m'=N/2}^{N-1} y[m']\cos\left(2n\pi - \frac{(4m'+1)n\pi}{2N}\right) = \sum_{m'=N/2}^{N-1} y[m']\cos\left(\frac{(4m'+1)n\pi}{2N}\right).$$

Равенство возможно благодаря свойству

$$\cos(\alpha-\beta)=\cos(\alpha)\cos(\beta)+\sin(\alpha)\sin(\beta).$$

Теперь две суммы в выражении для $X[n]$ можно объединить в одну:

$$X[n]=\sum_{m=0}^{N-1} y[m]\cos\left(\frac{(4m+1)n\pi}{2N}\right).$$

Заметим, что ДПФ от $y[m]$ есть

$$Y[n]=\sum_{m=0}^{N-1} y[m]e^{-\frac{j2\pi}{N}mn}=\sum_{m=0}^{N-1} y[m]\left[\cos\left(\frac{2\pi mn}{N}\right)-j\sin\left(\frac{2\pi mn}{N}\right)\right].$$

Если умножить обе части равенства на $e^{\frac{-jn\pi}{2N}}=\cos\left(\frac{n\pi}{2N}\right)-j\sin\left(\frac{n\pi}{2N}\right)$ и взять действительные части от результата (учитывая, что $x[m]$ и $y[m]$ вещественные), получаем:

$$\mathrm{Re}\left[e^{\frac{-jn\pi}{2N}}Y[n]\right]=\sum_{m=0}^{N-1} y[m]\left[\cos\left(\frac{2\pi mn}{N}\right)\cos\left(\frac{n\pi}{2N}\right)-\sin\left(\frac{2\pi mn}{N}\right)\sin\left(\frac{n\pi}{2N}\right)\right]=$$

$$=\sum_{m=0}^{N-1} y[m]\cos\left(\frac{4m+1}{2N}n\pi\right).$$

Последнее равенство получено благодаря свойству

$$\cos(\alpha-\beta)=\cos(\alpha)\cos(\beta)+\sin(\alpha)\sin(\beta).$$

Выражение для $\mathrm{Re}\left[e^{-jn\pi/2N}Y[n]\right]$ идентично последнему выражению для $X[n]$ откуда следует:

$$X[n]=\mathrm{Re}\left[e^{\frac{-jn\pi}{2N}}Y[n]\right],$$

где $Y[n]$ – ДПФ от последовательности $y[m]$, определенной через $x[m]$, которое может быть вычислено через БПФ за время $\Theta(N\log_2 N)$.

В итоге БКП можно выполнить в четыре шага:

- сформировать последовательность $y[m]$ из $x[m]$

$$\begin{cases} y[m]=x[2m] \\ y[N-1-m]=x[2m+1] \end{cases} \quad (m=0,1,\ldots,N/2-1);$$

- выполнить БПФ $Y[n]$ для $y[m]$;
- учесть коэффициент нормировки

$$Y[n]=C[n]Y[n],$$

$$C[n]=\begin{cases} \sqrt{\frac{1}{N}}, & n=0 \\ \sqrt{\frac{2}{N}}, & n=1,2,\ldots,N-1 \end{cases};$$

- получить БКП $X[n]$ из $Y[n]$ через выражение

$$X[n] = \mathrm{Re}\left[e^{\frac{-j\pi n}{2N}} Y[n] \right].$$

При нахождении обратного БКП все действия выполняются в обратном порядке:

- сформировать последовательность $Y[n]$ из $X[n]$ учитывая коэффициент нормировки (На этом шаге также необходимо умножить все элементы на коэффициент $1/N$, который появляется в обратном БПФ в строках 28–29. Однако поскольку вычисление обратного БПФ идет только в контексте обратного БКП, то этот коэффициент можно не учитывать ни там, ни сейчас):

$$Y[n] = X[n]C[n]e^{\frac{j\pi n}{2N}}, \quad n = 0,1,...,N-1;$$

- получить $y[m]$ из $Y[n]$ через обратное БПФ и взять действительную часть от полученного результата;
- сформировать $x[m]$ из $y[m]$

$$\begin{cases} x[2m] = y[m] \\ x[2m+1] = y[N-1-m] \end{cases} \quad (m = 0,1,...,N/2-1).$$

Программная реализация

Перейдем к рассмотрению практической стороны вопроса. В качестве аудиосигнала выберем сигнал, записанный в *wav*-формате, поскольку данный формат довольно прост, бесплатен, широко распространен и не использует сжатия. Структура такого файла состоит из заголовка (в нем содержится информация о размере файла, количестве каналов, частоте дискретизации и глубине звучания) и области данных, состоящая из так называемых сэмплов (выборки значений функции от времени, описывающей звук).

Формат заголовка представлен в таблице 2 [10, 11, 12, 13].

Таблица 2. Формат заголовка wav-файла

Порядок байт	Сдвиг	Поле	Длина поля	Описание	Секция
прямой	0	*ChunkID*	4	'*R*', '*I*', '*F*', '*F*' (*ASCII*) 0x52494646; началом *RIFF*-цепочки	секция типа *RIFF*
обратный	4	*ChunkSize*	4	*file size* - 8. 36 + *subchunk2Size*, или более точно: 4 + (8 + *Subchunk1Size*) + (8 + *Subchunk2Size*)	
прямой	8	*Format*	4	'*W*', '*A*', '*V*', '*E*' (*ASCII*) 0x57415645	
прямой	12	*Subchunk1ID*	4	'*f*', '*m*', '*t*', ' ' (*ASCII*) 0x666d7420; в блоке *fmt* определен формат звуковых данных	Секция формата – «fmt»
обратный	16	*Subchunk1Size*	4	оставшийся размер подцепочки, начиная с этой позиции	
обратный	20	*AudioFormat* (*Compression code*)	2	код формата сжатия; для *PCM* = 1 (линейное квантование); значения, отличающиеся от 1, обозначают некоторый формат сжатия	
обратный	22	*NumChannels*	2	количество каналов (1 – моно, 2 – стерео и т.д.)	
обратный	24	*SampleRate*	4	частота дискретизации (кГц)	

обратный	28	*ByteRate*	4	количество байт, передаваемых за секунду воспроизведения (*SampleRate* * *NumChannels* * *BitsPerSample* / 8)	
обратный	32	*BlockAlign*	2	количество байт для одного сэмпла, включая все каналы (*NumChannels* * *BitsPerSample* / 8)	
обратный	34	*BitsPerSample*	2	количество бит в сэмпле; так называемая «глубина» или точность звучания. 8 бит, 16 бит и т.д.	
прямой	36	*Subchunk2ID*	4	'*d*', '*a*', '*t*', '*a*' (*ASCII*) 0x64617461	Секция данных – «data»
обратный	40	*Subchunk2Size*	4	количество байт в области данных (*NumSamples* * *NumChannels* * *BitsPerSample* / 8)	
обратный	44	*data*		аудиоданные	

Для обработки звуковых сигналов была создана программа на языке *C++*, которая способна загружать *wav*-файл и добавлять в него случайный водяной знак, а также сравнивать два сигнала – эталонный и проверяемый – делая вывод о наличии во втором водяного знака. Сам же цифровой водяной знак можно представить в разных форматах. В нашем случае это будет вектор случайных бит. Длина этого вектора будет зависеть от продолжительности аудиосигнала и размера стегоконтейнера.

При загрузке файла программа сначала считывает заголовок, оставшаяся часть файла записывается в буфер и разбивается на целое количество блоков, каждый из которых состоит из N каналов, где

$$N = 2^n, \quad n = 2,3,4,\ldots.$$

Необходимость именно такой длины блока обуславливается особенностями быстрого косинусного преобразования.

Из чего же будут состоять блоки? Каждый сэмпл в файле состоит из *NumChannels* каналов. Каждый канал в свою очередь определяется *BlockAlign* / *NumChannels* байтами. Поскольку в общем случае количество каналов в сэмпле неизвестно, составим блоки из всех байт только первого канала. Обозначим количество блоков через C.

Остальные байты сэмпла не будут использоваться в программе. Их значения будут просто перенесены в формируемый сигнал с ЦВЗ.

Последним считывается остаток файла, длина которого меньше N. В дальнейшем эта часть не будет фигурировать в вычислениях.

Произведя загрузку файла программа готова добавлять в него ЦВЗ. Разложив сигнал на спектр определим, какие частоты будут являться стегоконтейнером. Для звукового сигнала такое разложение будет естественным, поскольку звук фактически является набором волн различной частоты и амплитуды.

Чтобы обеспечить устойчивость (робастность) ЦВЗ, выбирать такие частоты нужно очень тщательно. Один из вариантов выбора – область наивысших (младших) частот, входящих в сигнал. Они в наименьшей мере определяют его вид, поэтому их изменение не вызовет слышимых помех. Выберем h таких частот. В таком случае фактическим контейнером водяного знака будет являться множество байт, соответствующих всему первому каналу. Это обуславливается тем, что даже незначительное изменение одной из частот разложения меняет весь сигнал.

Итак, мы имеем C стегоконтейнеров, состоящих из N значений выбранного канала. Теперь можно выполнить C прямых преобразований с каждым из этих блоков. Результатом работы преобразований будут C векторов длины N, описывающих коэффициенты разложения на спектр частот.

Первые коэффициенты полученных векторов соответствуют наиболее значимым (несущим) частотам, а последние — младшим частотам. Соответственно наша цель – последние h частот. Практика показывает, что уже после 5й частоты вклад в вид сигнала каждой последующей несущественен. Основываясь на это можно взять $h = N - 5$ младших частот.

Получив спектры разложений и выделив в них младшие частоты, нужно добавить наш ЦВЗ в полученный контейнер. Сделать это можно разными способами, например ослабив громкие частоты и усилив тихие. Для этого вводятся два дополнительных параметра T и α, где первый определяет порог амплитуды, а второй – величину изменения. Таким образом

$$\bar{f}_{ij} = \begin{cases} \alpha f_{ij} & f_{ij} < T \\ \dfrac{f_{ij}}{\alpha} & f_{ij} \geq T \end{cases} \qquad i = 0,1,...,C-1; \quad j = N-h-1, N-h,..., N-1.$$

Введем правило, определяющее способ добавления битов знака в частотную область:

$$\bar{f}_{ij} = \begin{cases} f_{ij} & w_k = 0 \\ \begin{cases} \alpha f_{ij} & f_{ij} < T \\ \dfrac{f_{ij}}{\alpha} & f_{ij} \geq T \end{cases} & w_k = 1 \end{cases} \qquad \begin{matrix} i = 0,1,...,C-1; \quad j = N-h, N-h+1,..., N-1; \\ k = ih + j + h - N; \end{matrix}$$

где w — ЦВЗ и

$$w_k = (0,1,...,m), \quad m = Ch.$$

Применим это правило к отобранным частотам спектров и выполним C обратных преобразований. Из полученных новых блоков, неизмененных каналов, старого заголовка и неизмененного остатка соберем новый *wav*-файл. Также отдельно сохраним файл, содержащий битовую последовательность.

Примеры результатов добавления ЦВЗ

Рассмотрим несколько примеров работы программы в режиме добавления в ЦВЗ (рис. 4-7). Для этого построим спектрограммы сигналов до и после их изменений. Горизонтальная ось на графике – время, вертикальная – частота. Цвет точки определяет громкость данной частоты в данный момент времени. Спектрограммы построены с помощью программы *Spek* [15].

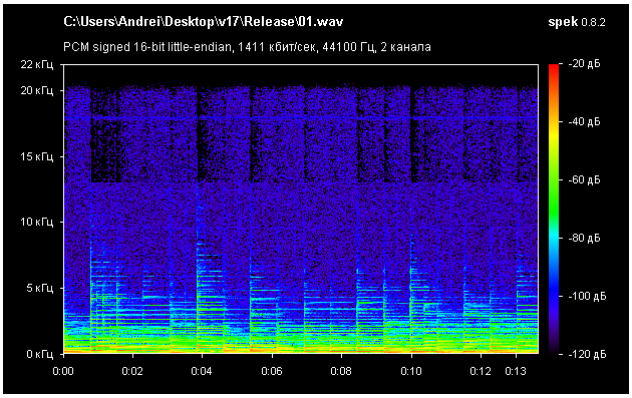

Рис. 4. Спектрограмма оригинального сигнала №1

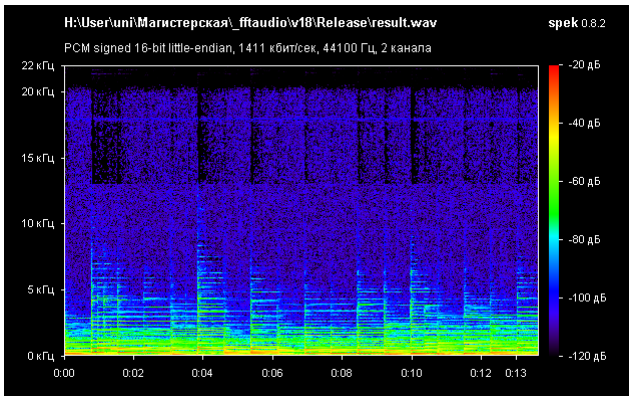

Рис. 5. Спектрограмма сигнала №1 с ЦВЗ (n = 6, T = 100, α = 1.0005, h = 6)

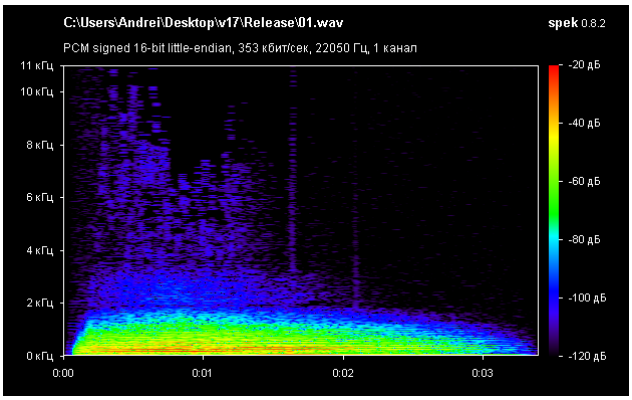

Рис. 6. Спектрограмма оригинального сигнала №2

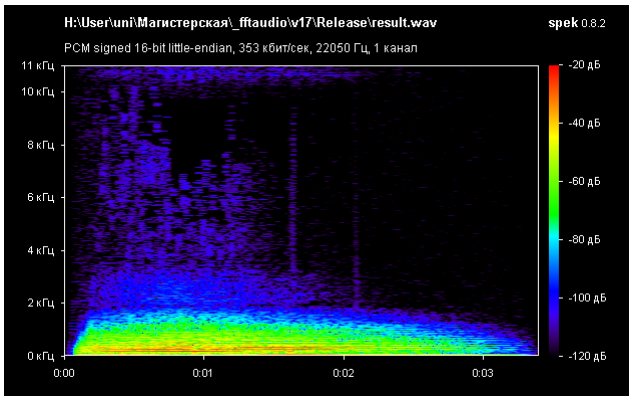

Рис. 7. Спектрограмма сигнала №2 с ЦВЗ ($n = 6$, $T = 100$, $\alpha = 1.0005$, $h = 6$)

Как видно из иллюстраций, при выбранных значениях параметров встраивания ЦВЗ на спектрограмме могут появляться заметные отличия. Впрочем, на слух эти изменения заметить нельзя.

Выбор частот для стегоконтейнера

При выборе частот, которые будут изменены, предпочтение было отдано более высоким. Однако результат их изменения можно обнаружить на спектрограмме. Поэтому такие частоты являются не самым лучшим стегоконтейнером. Рассмотрим вариант с добавлением знака в средние частоты. Если аккуратно добавлять ЦВЗ именно в них, то можно добиться неслышимых изменений и усложнить задачу атакующему, стремящемуся избавиться от ЦВЗ. В случае с контейнером из высоких частот ЦВЗ можно легко убрать, просто удалив эти частоты из сигнала. Для средних частот удаление приведет к потере значимой части сигнала. Дополнительное усиление стойкости (робастности) ЦВЗ – случайный выбор частот из среднего диапазона. Поскольку атакующий не знает какие именно частоты содержат ЦВЗ, он будет вынужден избавиться от всех, что приведет к искажению сигнала.

Какой же диапазон частот можно считать средним? В общем случае это зависит от конкретного сигнала. Будем считать частотой, находящейся в среднем диапазоне, такую частоту, которая близка к максимальной частоте, входящей в сигнал. Степень близости определяется шириной спектра. Чем он шире, тем шире средний диапазон. На спектрограмме этому диапазону соответствует участок перехода в зону черного цвета.

Немаловажный фактор при выборе частот – чувствительность человеческого уха.

Слышимую часть диапазона звуков разделяют на низкочастотные звуки – до 500 герц, среднечастотные – 500-10000 герц и высокочастотные – свыше 10000 герц. Такое разделение очень важно, так как ухо человека неодинаково чувствительно к разным звукам. Наиболее чувствительно ухо к сравнительно узкому диапазону среднечастотных звуков от 1000 до 5000 герц. К более низко- и высокочастотным звукам чувствительность резко падает. Это приводит к тому, что человек способен услышать в среднечастотном диапазоне звуки с энергией около 0 децибел и не слышать низкочастотные звуки в 20–40–60 децибел. Т.е. звуки с одной и той же энергией в среднечастотном диапазоне могут восприниматься как громкие, а в низкочастотном – как тихие или быть вовсе не слышны [16].

На рис. 8 представлены кривые равной громкости чистых тонов, называемые кривыми Флетчера-Мэнсона (*Fletcher-Munson curves*). Они показывают, что с уменьшением силы давления звука для обеспечения равной громкости требуется усиление по громкости на высоких и низких частотах.

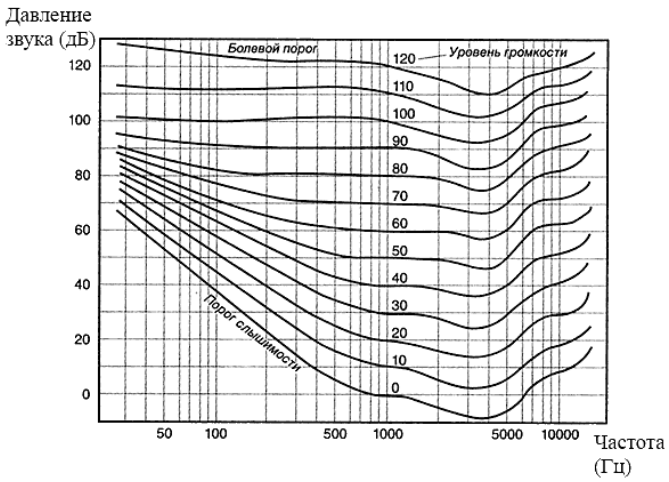

Рис. 8. Кривые Флетчера-Мэнсона

Такая особенность звука сформирована природой не случайно. Звуки, необходимые для его существования – речь, звуки природы – находятся в основном в среднечастотном диапазоне.

На рис. 9 показаны зоны, в которые попадают привычные нам звуки [17].

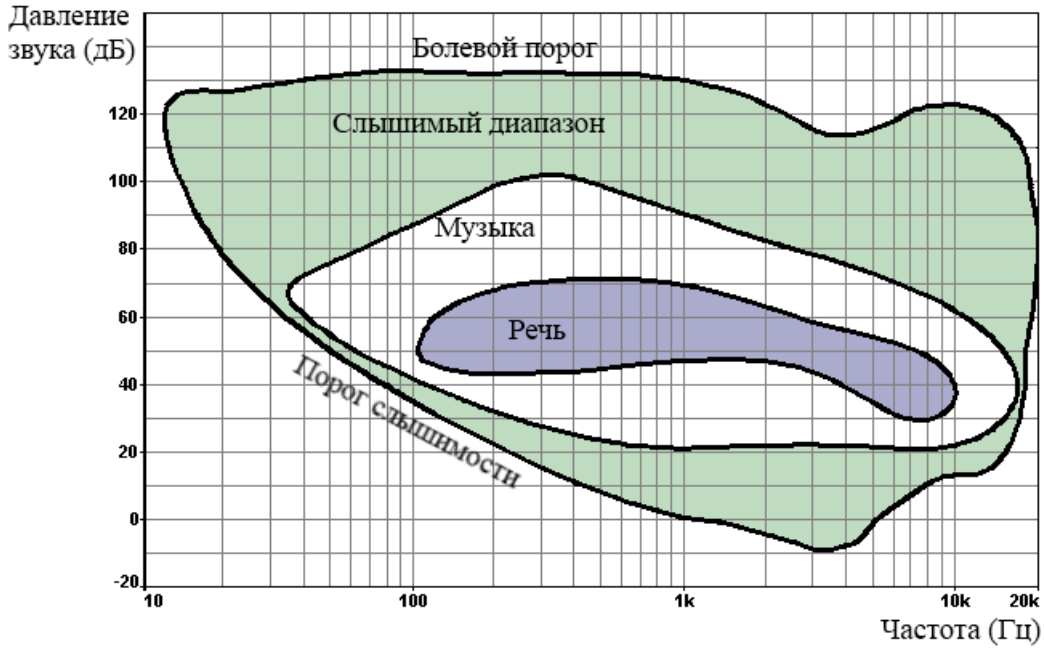

Рис. 9. Зоны слышимости

Таким образом, наилучшим стегоконтейнером будут частоты, попадающие в область пересечения плохо слышимых зон со средним диапазоном сигнала.

Чтобы получить величины частот, на которые раскладывает сигнал косинусное преобразование, в герцах нужно знать частоту дискретизации. Для этого воспользуемся теоремой Котельникова. Она утверждает, что для того чтобы однозначно и без потерь восстановить сигнал по точкам, максимальная частота в его спектре не должна превышать половину частоты дискретизации, т.е.

$$f_{\max} < \frac{f_{SR}}{2}.$$

Зная, что частоты в спектре распределены равномерно получим выражение для *n*-й частоты:

$$f[n] = \frac{f_{SR}/2}{N} n.$$

Тогда индекс коэффициента в векторе косинусного преобразования, определяющего амплитуду ближайшей от указанной частоты *f*, будет выражаться так:

$$n = round\left(\frac{2Nf}{f_{SR}}\right).$$

Так мы можем найти нужные нам (соответствующие амплитудам плохо слышимых частот) коэффициенты преобразования. Пусть теперь n_l и n_h $(0 < n_l < n_h < N-1)$ – индексы амплитуд частот $f[n_l]$ и $f[n_h]$, таких, что частоты между $f[n_l]$ и $f[n_h]$ воспринимаются слабо. Определим множество таких частот:

$$F_1 = \{f[k], \quad k = n_l, n_l + 1, \ldots, n_h\}.$$

Теперь найдем множество частот из среднего диапазона сигнала

$$F_2 = \{f[k], \quad k = k_0 - \delta, k_0 - \delta + 1, \ldots, k_0, \ldots, k_0 + \delta - 1, k_0 + \delta\},$$

где k_0 – индекс частоты, находящейся в середине этого диапазона, а δ – его ширина.

Тогда стегоконтейнером будет следующее множество частот:

$$F = F_1 \bigcap F_2.$$

Теперь остается только выбрать несколько случайных частот из этого множества и добавить в них биты ЦВЗ. Стоит отметить, что таких множеств может оказаться несколько.

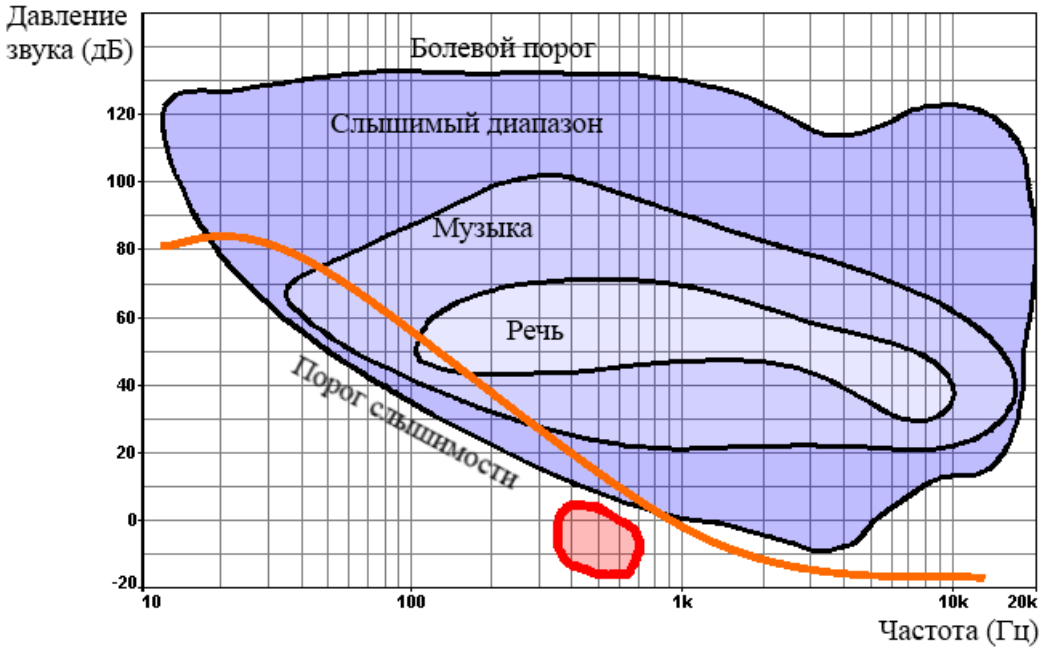

Рис. 10. Область оптимального встраивания ЦВЗ

На рис. 10 показан пример выбора множества *F*. Черной линией показаны амплитуды частот сигнала в определенный момент времени, а оранжевая область – оптимальная зона встраивания знака.

Описанный механизм на данный момент существует только в виде теории и не реализован в программе. На данный момент в программе можно вручную задать количество изменяемых частот и индекс первой такой частоты.

Сравнение двух сигналов

Во втором режиме работы программа сравнивает два сигнала и находит степень их похожести друг на друга. Для работы необходимы три следующих файла:

- эталонный сигнал;
- проверяемый сигнал;
- битовая последовательность.

Поскольку ЦВЗ располагается в частотной области, то на первом шаге после загрузки сигналов и разбиения их на блоки, выполняются прямые преобразования над этими блоками. Получив спектры сигналов, программа выделяет из проверяемого сигнала ЦВЗ. Для каждого очередного бита ЦВЗ, вычисляется исходное значение измененной частоты по формуле обратной формуле (10). Имея две частоты и заданную точность e можно выполнить их сравнение. Пусть w – реальный ЦВЗ, w' – выявляемый ЦВЗ, f_k – одна из младших частот спектра эталонного сигнала и $\bar{f}'_k$ — частота проверяемого сигнала после «удаления» бита ЦВЗ, тогда:

$$w'_k = \begin{cases} w_k & \left|f_k - \bar{f}'_k\right| < e \\ 1 - w_k & \left|f_k - \bar{f}'_k\right| \geq e \end{cases}.$$

Вычислив w', программа сравнивает его с эталонным ЦВЗ. Сравнение может идти с помощью одного из двух способов: подсчет процента совпадений в битах знаков или корреляционным методом.

Корреляционный метод сравнения

Коэффициент корреляции показывает статистическую взаимосвязь двух или нескольких случайных величин.

Коэффициент корреляции R для двух случайных величин X и Y, определённых на одном вероятностном пространстве задаётся формулой:

$$R_{X,Y} = \frac{\operatorname{cov}(X,Y)}{\sqrt{D[X]} \cdot \sqrt{D[Y]}},$$

где $\operatorname{cov}(X,Y)$ – ковариация, D – дисперсия.

Более развернутый вариант предыдущей формулы:

$$R_{X,Y} = \frac{M[XY] - MX \cdot MY}{\sqrt{\left(M\left[X^2\right] - (MX)^2\right)} \cdot \sqrt{\left(M\left[Y^2\right] - (MY)^2\right)}},$$

где

$$X = (x_1, \ldots, x_m),$$

$$M(X) = \frac{1}{m}\sum_{i=1}^{m} x_i,$$

$$M(XY) = \frac{1}{m}\sum_{i=1}^{m} x_i y_i.$$

Величина $R_{X,Y}$ имеет область значений $[-1;1]$. Чем она меньше по модулю, тем сильнее различия между X и Y.

Заключение

В статье описаны теоретические основы работы с цифровыми водяными знаками, базовые аспекты теории цифрового анализа сигналов и рассмотрены вопросы практического применения аналитических методов, таких как преобразование Фурье. Подробно описан один из алгоритмов вычисления быстрого преобразования Фурье, быстрого косинусное преобразование и выполнен анализ этих методов, направленный на оценивание скорости их работы.

В тексте также содержится описание программы, в которой реализован метод добавления ЦВЗ в аудиосигнал и метод проверки его наличия. В основу этих методов был заложен итеративный алгоритм БКП, который был выбран среди множества других аналогичных алгоритмов на этапе анализа. С помощью данной программы были выполнены тесты по добавлению и проверке наличия ЦВЗ с различными сигналами. В результате чего выдвинута теория по оптимальному выбору стегоконтейнера. Эта теория учитывает особенности человеческого восприятия звука и потенциальные варианты атак, нацеленных на удаление ЦВЗ. Также определены оптимальные значения параметров встраивания знака.

Данное направление исследований подразумевает дальнейшее развитие. Так, например, открытым остается вопрос об использовании различных модификаций быстрого преобразования Фурье таких как преобразование в целых числах. Этот метод потенциально мог бы позволить достигнуть прироста вычислительной скорости и избавить от погрешностей при округлениях. Интерес также представляет проверка работоспособности теории по определению оптимального стегоконтейнера в аудиосигнале и анализ устойчивости сигнала.

Список литературы

1. Грибунин В., Оков И., И. Туринцев. Цифровая стеганография. – М.: Солон-Пресс, 2002.
2. Fourier transform. [Электронный ресурс]. URL: http://en.wikipedia.org/wiki/Fourier_transform.
3. Discrete cosine transform. [Электронный ресурс]. URL: http://en.wikipedia.org/wiki/Discrete_cosine_transform.
4. Chung-Bin Wu, Jiun-Lung Wang, Bin-Da Liu, Jar-Ferr Yang. Efficient Recursive Structures for DCT/IDCT // Department of Electrical Engineering National Cheng Kung University, Tainan, Taiwan, Republic of China. [Электронный ресурс]. URL: http://www.ee.nchu.edu.tw/Pic/Writings/2425_WCE99.PDF.
5. Chan Yuk-Pee, Chau Lap-Pui, Siu Wan-chi. Efficient implementation of discrete cosine transform using recursive filter structure // IEEE transactions on circuits and systems for video technology. – 1994. – Vol. 4. – No. 6. – Pp. 550-552.
6. Ruye Wang. Definition of DCT. Fast DCT algorithm. [Электронный ресурс]. URL: http://fourier.eng.hmc.edu/e161/lectures/dct/dct.html.
7. Hassanieh H., Indyk P., Katabi D., Price E. Nearly Optimal Sparse Fourier Transform // STOC'12 Proceedings of the 44th symposium on Theory of Computing. ACM New York, NY, USA. – 2012. – Pp. 563-578.
8. Xuancheng Shao, Steven G. Johnson. Type-II/III DCT/DST algorithms with reduced number of arithmetic operations // Signal Processing. – 2008. – Vol. 88. – Issue 6. – Pp. 1553-1564.
9. Кормен Т., Лейзерсон Ч., Ривест Р. Алгоритмы: построение и анализ / Пер. с англ., под ред. А. Шеня. – М.: МЦНМО: Бином. Лаборатория знаний, 2004.
10. Wilson S. WAVE PCM Soundfile format. [Электронный ресурс]. URL: https://ccrma.stanford.edu/courses/422/projects/WaveFormat/.
11. Wave File Format. [Электронный ресурс]. URL: http://www.sonicspot.com/guide/wavefiles.html.

12. Wave File Format – формат звукового файла WAV. [Электронный ресурс]. URL: http://microsin.ru/content/view/1197/44/.
13. Структура WAV файла. Полный список значений *audioFormat*. [Электронный ресурс]. URL: http://audiocoding.ru/wav_formats.txt.
14. Быстрое преобразование Фурье за $O(NlogN)$. Применение к умножению двух полиномов или длинных чисел. [Электронный ресурс]. URL: http://e-maxx.ru/algo/fft_multiply.
15. Spek – Acoustic Spectrum Analyser. [Электронный ресурс]. URL: http://spek.cc/.
16. Кратко о звуке и особенностях уха человека. [Электронный ресурс]. URL: http://www.fesmu.ru/www2/PolTxt/U0005/auscul.cor2/vvedenie/Sluch/svuk.htm.
17. Db. Frequency. Ohm Law. [Электронный ресурс]. URL: http://lenardaudio.com/education/03_db.html.