Введение в цифровую обработку сигналов

(математические основы)

Алексей Лукин, 2002

Лаборатория компьютерной графики и мультимедиа, МГУ

Содержание

Линейные системы

Определения

Сигнал – зависимость одной величины от другой (функция). Например, зависимость давления воздуха в точке от времени можно рассматривать как звуковой сигнал. Зависимость напряжения в проводнике от времени тоже может представлять звуковой сигнал. Зависимость яркости точки на плоскости от ее координат можно рассматривать как черно-белое изображение.

Будем пока для определенности рассматривать одномерные сигналы, зависящие от времени, и обозначать их $x(t)$. Почти весь материал допускает обобщение и на многомерный случай.

Система – это некоторое преобразование сигнала. Система переводит *входной сигнал x(t)* в *выходной сигнал y(t)*. Будем это обозначать так: $x(t) \longrightarrow y(t)$.

Обычно все рассматриваемые системы *инвариантны к сдвигу*, т.е. если $x(t) \longrightarrow y(t)$, то $x(t+T) \longrightarrow y(t+T)$. Это означает, что форма выходного сигнала зависит только от входного сигнала, а не зависит от времени начала подачи входного сигнала. Далее мы будем рассматривать только такие системы.

Очень большое количество реальных систем можно считать инвариантными к сдвигу. Например, микрофон, переводящий сигнал «плотность воздуха» в сигнал «напряжение в проводе», удовлетворяет этому свойству, если пренебречь изменением свойств микрофона со временем.

Линейная система – это система, в которой выполняется следующее свойство линейности: если $x_1(t) \longrightarrow y_1(t)$ и $x_2(t) \longrightarrow y_2(t)$, то $\alpha \cdot x_1(t) + \beta \cdot x_2(t) \longrightarrow \alpha \cdot y_1(t) + \beta \cdot y_2(t)$. Здесь операции над сигналами следует понимать как операции над функциями от аргумента *t*.

Большое количество реальных систем по преобразованию сигналов можно считать линейными. Например, микрофон является линейной системой (с достаточной степенью точности), так как если в него будут говорить одновременно 2 человека с разной громкостью, то электрический сигнал на выходе будет взвешенной суммой сигналов (от каждого человека в отдельности) на входе, а коэффициенты будут означать громкость разговора первого и второго человека.

Далее мы будем рассматривать линейные инвариантные к сдвигу системы, называя их просто линейными.

Свойства линейных систем:

1. Постоянный (константный) сигнал переводится любой линейной системой в постоянный сигнал.

2. При прохождении через линейную систему синусоида остается синусоидой. Могут измениться лишь ее амплитуда и фаза (сдвиг во времени).

Второе свойство особенно важно, т.к. оно указывает на важнейший метод анализа линейных систем с помощью разложения входных и выходных сигналов на синусоиды (Фурье-анализ).

Что означает «прохождение синусоиды через линейную систему»? Это значит, что синусоида подается на вход системы бесконечно долго, т.е. от $t = -\infty$ до $t = +\infty$. Если же синусоиду начали подавать лишь в некоторый конкретный момент времени (а до этого подавалось что-то другое, например, - 0), то после начала подачи синусоиды на вход мы можем получить синусоиду на выходе не сразу. Выходной сигнал постепенно начнет приобретать синусоидальную форму. Скорость «стремления к верной синусоиде» на выходе зависит от конкретной линейной системы.

Упражнения

1. Система такова, что $x(t) \longrightarrow y(t)$, где $x(t)=\sin(t)+\sin(2t)$, а $y(t)=\cos(t)+\sin(3t)$. Является ли система линейной?

2. На вход неизвестной линейной системы подается сигнал $x(t)=2\sin(t)-\cos(3t)$. Какого вида сигналы можно ожидать на выходе?

3. Доказать, что суперпозиция двух линейных систем (выход первой подключен к входу второй) является линейной системой.

Ответы

1. Нет, т.к. на выходе появилась синусоида $\sin(3t)$, хотя на входе колебания с такой частотой не было. Это противоречит второму свойству. Кстати, исчезновение синусоиды $\sin(2t)$ не противоречит свойствам линейной системы, т.к. амплитуда входных синусоид может изменяться до нуля.

2. По свойствам линейной системы, на выходе может появиться сигнал вида $A \cdot \sin(t + \varphi) + B \cdot \sin(3t + \phi)$, где A, B, φ и ϕ – любые действительные числа.

3. Пусть $x_1(t) \xrightarrow{1} y_1(t) \xrightarrow{2} z_1(t)$ и $x_2(t) \xrightarrow{1} y_2(t) \xrightarrow{2} z_2(t)$. Тогда из линейности первой системы следует $\alpha \cdot x_1(t) + \beta \cdot x_2(t) \xrightarrow{1} \alpha \cdot y_1(t) + \beta \cdot y_2(t)$, а из линейности второй системы следует $\alpha \cdot y_1(t) + \beta \cdot y_2(t) \xrightarrow{2} \alpha \cdot z_1(t) + \beta \cdot z_2(t)$. Следовательно, $\alpha \cdot x_1(t) + \beta \cdot x_2(t) \xrightarrow{1} \xrightarrow{2} \alpha \cdot z_1(t) + \beta \cdot z_2(t)$, что и требовалось доказать.

Дискретные и непрерывные сигналы

Теорема Котельникова

Большинство реальных сигналов (например, звуковых) являются непрерывными функциями (если пренебречь квантовыми эффектами). Для обработки на компьютере требуется перевести сигналы в цифровую форму. Один из способов сделать это – равномерно по времени измерить значения сигнала на определенном промежутке времени и ввести полученные значения амплитуд в компьютер. Если делать измерения достаточно часто, то по полученному *дискретному сигналу* можно будет достаточно точно восстановить вид исходного непрерывного сигнала.

Процесс замера величины сигнала через равные промежутки времени называется равномерной (по времени) *дискретизацией*. Многие устройства для ввода данных в компьютер осуществляют дискретизацию. Например, звуковая карта дискретизирует сигнал с микрофона, сканер дискретизирует сигнал, поступающий с фотоэлемента. В результате дискретизации непрерывный (*аналоговый*) сигнал переводится в последовательность чисел. Устройство, выполняющее этот процесс, называется *аналогово-цифровым преобразователем* (АЦП, analogue-to-digital converter, *ADC*). Частота, с которой АЦП производит замеры аналогового сигнала и выдает его цифровые значения, называется *частотой дискретизации*.

Встает вопрос: при каких условиях на исходный сигнал и на частоту дискретизации можно с необходимой степенью точности восстановить исходный сигнал по его цифровым значениям? Ответ на этот вопрос дает важная теорема Котельникова. Однако чтобы ее понять, необходимо познакомиться с понятием спектра непрерывного сигнала.

Как известно из анализа, любая непрерывная функция может быть разложена в интеграл Фурье. Смысл этого разложения состоит в том, что функция представляется в виде суммы «ряда» синусоид с различными амплитудами. Коэффициенты (амплитуды) при синусоидах называются *спектром* функции. У относительно гладких функций спектр быстро убывает (с ростом частоты коэффициенты быстро стремятся к нулю). Для относительно «ломаных» функций спектр убывает медленно, т.к. для представления разрывов и «изломов» функции нужны синусоиды с большими частотами.

Говорят, что сигнал имеет *ограниченный спектр*, если выше определенной частоты все коэффициенты спектра равны нулю. В этом случае говорят, что спектр сигнала лежит ниже частоты F (ограничен частотой F), где F – частота, выше которой спектр равен нулю.

Теорема Котельникова (Найквиста, Шеннона): если сигнал таков, что его спектр ограничен частотой F, то после дискретизации сигнала с частотой не менее $2F$ можно восстановить исходный непрерывный сигнал по полученному цифровому сигналу абсолютно точно. Для этого нужно проинтерполировать цифровой сигнал «между отсчетами» специального вида функциями (sinc-функциями).

На практике эта теорема имеет огромное значение. Например, известно, что большинство звуковых сигналов можно с некоторой степенью точности считать сигналами с ограниченным спектром. Их спектр, в основном, лежит ниже 20 кГц. Это значит, что при дискретизации с частотой не менее 40 кГц мы можем потом более-менее точно восстановить исходный аналоговый звуковой сигнал по его цифровым отсчетам. Абсолютной точности достичь не удастся, так как в природе не бывает сигналов с идеально ограниченным спектром.

Устройство, которое интерполирует дискретный сигнал до непрерывного, называется *цифро-аналоговым преобразователем* (ЦАП, digital-to-analogue converter, *DAC*). Эти устройства применяются, например, в проигрывателях компакт-дисков для восстановления звука по цифровому звуковому сигналу, записанному на компакт-диск. Частота дискретизации звукового сигнала при записи на компакт-диск составляет 44100 Гц. Таким образом, говорят, что ЦАП на CD-плеере работает на частоте 44100 Гц.

Наложение спектров (алиасинг)

Что произойдет, если попытаться оцифровать сигнал с недостаточной для него частотой дискретизации (или если спектр сигнала не ограничен)? В этом случае по полученной цифровой выборке нельзя будет верно восстановить исходный сигнал. Восстановленный сигнал будет выглядеть таким образом, как если бы частоты, лежащие выше половины частоты дискретизации, отразились от половины частоты дискретизации, перешли в нижнюю часть спектра и наложились на частоты, уже присутствующие в нижней части спектра. Этот эффект называется *наложением спектров* или *алиасингом* (*aliasing*).

Предположим, что мы попытались оцифровать музыку, спектр которой ограничен частотой 20 кГц, но при записи какой-то электроприбор (например, дисплей) сгенерировал сильную помеху с ультразвуковой частотой 39 кГц, которая проникла в аналоговый звуковой сигнал. Мы производим оцифровку с частотой 44.1 кГц. При этом мы предполагаем, что звук, лежащий ниже частоты $\frac{44.1 кГц}{2} = 22.05 кГц$ будет записан правильно (по теореме Котельникова). Но так как помеха лежит выше частоты 22.05 кГц, то возникнет алиасинг, и помеха «отразится» в нижнюю часть спектра, на частоту около 5 кГц. Если мы теперь попробуем пропустить полученный цифровой сигнал через ЦАП и прослушать результат, то мы услышим на фоне музыки помеху на частоте 5 кГц. Помеха «переместилась» из неслышимой ультразвуковой области в слышимую область.

Таким образом, мы видим, что алиасинг – нежелательное явление при дискретизации сигнала. Например, при оцифровке изображения алиасинг может привести к дефектам в изображении, таким как «блочные», «пикселизованные» границы или муар.

Как избежать алиасинга? Первый способ – использовать более высокую частоту дискретизации, чтобы весь спектр записываемого сигнала уместился ниже половины частоты дискретизации. Второй способ – искусственно ограничить спектр сигнала перед оцифровкой.

Существуют устройства, называемые фильтрами, которые позволяют изменять спектр сигнала. Например, *фильтры низких частот* (*НЧ-фильтры*, *low-pass filters*) пропускают без изменения все частоты ниже заданной, и удаляют из сигнала все частоты выше заданной. Эта граничная частота называется *частотой среза* (*cutoff frequency*) фильтра. Одно из важных применений НЧ-фильтров заключается в искусственном ограничении спектра сигнала перед оцифровкой. В этом случае фильтры называются *анти-алиасинговыми*, т.к. они предотвращают возникновение алиасинга при оцифровке сигнала. Частота среза анти-алиасинговых фильтров устанавливается равной половине частоты дискретизации.

Рассмотрим, что произойдет, если в примере с записью музыки и помехи применить анти-алиасинговый фильтр перед оцифровкой сигнала. Так как частота дискретизации составляет 44.1 кГц, то частота среза фильтра устанавливается на 22 кГц. Таким образом, фильтр будет пропускать без изменения все сигналы, спектр которых лежит ниже 22 кГц (музыку), и подавлять все сигналы, со спектром выше 22 кГц (в том числе – и помеху). После применения фильтра из сигнала исчезнет помеха, и спектр полученного сигнала будет лежать ниже 22 кГц. Когда этот сигнал будет подан на АЦП, алиасинга не возникнет, и по полученной цифровой записи можно будет правильно воссоздать исходную музыку (без помехи).

В реальные АЦП почти всегда встраивается анти-алиасинговый фильтр. Обычно эффект от искусственного ограничения спектра вполне приемлем, в то время как алиасинг – недопустим. Однако не всегда искусственное ограничение спектра так благотворно влияет на записываемый сигнал. Например, при оцифровке музыки на низкой частоте дискретизации 11 кГц приходится отфильтровывать из спектра музыки все частоты выше 5.5 кГц. В результате этого музыка теряет в качестве (хотя обычно такие потери лучше, чем алиасинг). При оцифровке изображений необходимо аккуратно проектировать анти-алиасинговый фильтр, чтобы изменение спектра изображения не повлекло видимых артефактов (таких как пульсации вблизи резких границ).

Упражнения

1. Известно, что для получения разборчиво звучащей человеческой речи достаточно оцифровывать ее с частотой 8 кГц. Какой диапазон частот может быть правильно передан такой цифровой записью? Что необходимо предпринять при оцифровке для правильной передачи этого диапазона?

2. При проектировании АЦП с частотой дискретизации 44 кГц был ошибочно реализован анти-алиасинговый фильтр. Его частота среза была установлена на 24 кГц. К каким эффектам может привести такой АЦП? Какая область частот в записи может быть испорчена? Отразится ли это на качестве звучания звукозаписи?

3. Что будет, если частоту среза анти-алиасингового фильтра установить ниже половины частоты дискретизации?

Ответы

1. В соответствии с теоремой Котельникова, при частоте дискретизации 8 кГц можно правильно передать диапазон частот от 0 до 4 кГц. Однако так как человеческая речь содержит частоты и выше 4 кГц (рис. 7), то для предотвращения алиасинга необходимо перед оцифровкой пропустить сигнал через анти-алиасинговый фильтр с частотой среза 4 кГц.

2. Если спектр записываемого сигнала не будет лежать ниже 22 кГц, возникнет алиасинг. Т.к. фильтр удалит все частоты выше 24 кГц, то в сигнале, подаваемом на оцифровку, могут содержаться нежелательные частоты от 22 кГц до 24 кГц. В результате алиасинга, эти частоты «отразятся» от половины частоты дискретизации (от 22 кГц) в нижнюю часть спектра и наложатся на частоты от 20 кГц до 22 кГц. Таким образом, область частот исходного сигнала от 20 кГц до 22 кГц может быть испорчена. Если это была звукозапись, то ее качество почти не ухудшится, т.к. человеческое ухо все равно практически не слышит частоты выше 20 кГц.

3. Алиасинга не возникнет, т.к. спектр будет ограничен нужным образом. Однако для данной частоты дискретизации возможный передаваемый частотный диапазон будет использоваться не полностью.

Импульсная характеристика

Начиная с этого момента, мы будем рассматривать *дискретные линейные системы*, то есть системы, работающие с дискретными сигналами. На вход такой системы подается последовательность чисел $x[n]$ (*дискретный сигнал*), и на выходе получается последовательность чисел $y[n]$ (рис. 1). Свойства линейности для дискретных систем формулируются почти дословно так же, как и для непрерывных линейных систем.

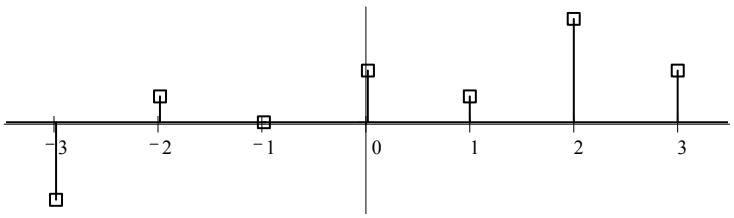


Рис. 1. *Так мы будем изображать дискретные сигналы: по оси абсцисс отложены дискретные моменты времени, по оси ординат – амплитуды сигнала в эти моменты. Отметим, что соединить дискретные отсчеты сигнала на графике прямыми линиями было бы не совсем верно, т.к. при восстановлении непрерывного сигнала по дискретному используются более сложные интерполирующие функции.*

Теперь изучим, каким образом линейная система может преобразовывать входной сигнал в выходной. Для этого рассмотрим реакцию системы на цифровую дельта-функцию (другое название – функция Кронекера).

Дельта-функция (цифровая) – это сигнал вида $\delta[n]=\begin{cases}1, n=0\\0, n\neq 0\end{cases}$, т.е. короткий единичный импульс (рис. 2).

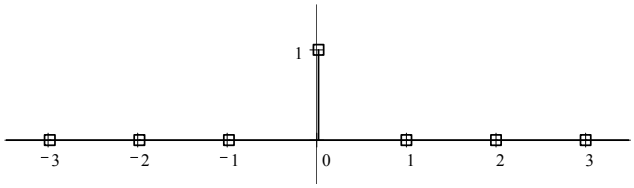


***Рис. 2**. Цифровая дельта-функция.*

Очевидно, что любой дискретный сигнал можно разложить в сумму таких функций, сдвинутых во времени (рис. 3). Например, бесконечный сигнал $x[n]$ можно представить в виде $x[n]=\sum_{i=-\infty}^{+\infty} x[i]\cdot\delta[n-i]$. Здесь дельта-функции – это «базисные функции», а $x[i]$ – это их коэффициенты в линейной комбинации. Если в этой формуле зафиксировать любое n, то мы получим тождество $x[n]=x[n]*1$, т.к. все остальные члены суммы обратятся в 0, ибо дельта-функция отлична от нуля только в нуле.

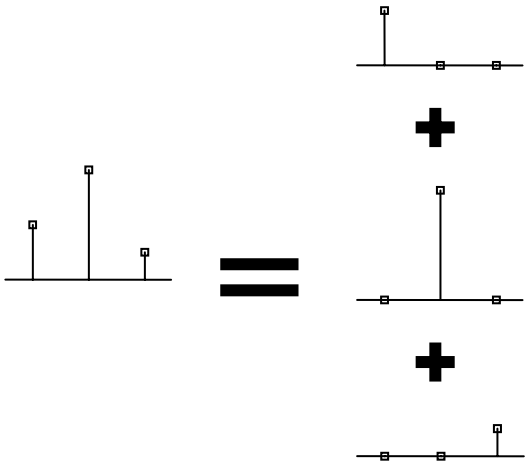


***Рис. 3**. Представление произвольного сигнала в виде линейной комбинации сдвинутых во времени дельта-функций.*

Теперь исследуем отклик (выходной сигнал) линейной системы на цифровую дельта-функцию. Для этого подадим дельта-функцию в систему и измерим выходной сигнал. Пусть выходной сигнал равен $h[n]$, т.е. $\delta[n]\longrightarrow h[n]$ (рис. 4).

Оказывается, зная $h[n]$ (отклик системы на дельта-функцию), можно вычислить отклик системы на любой входной сигнал. Действительно, так как любой входной сигнал является линейной комбинацией сдвинутых во времени дельта-функций, то выходной сигнал будет той же самой линейной комбинацией сдвинутых во времени функций $h[n]$. Это следует из линейности системы и инвариантности к сдвигу по времени (рис. 5).

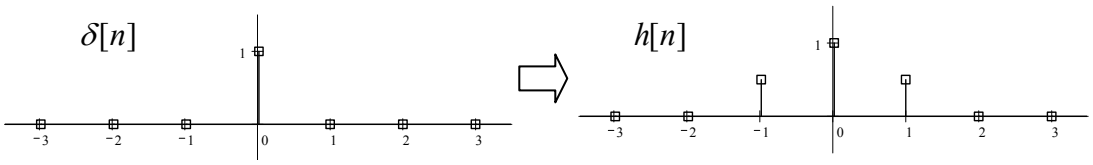


***Рис. 4**. Отклик системы на цифровую дельта-функцию.*

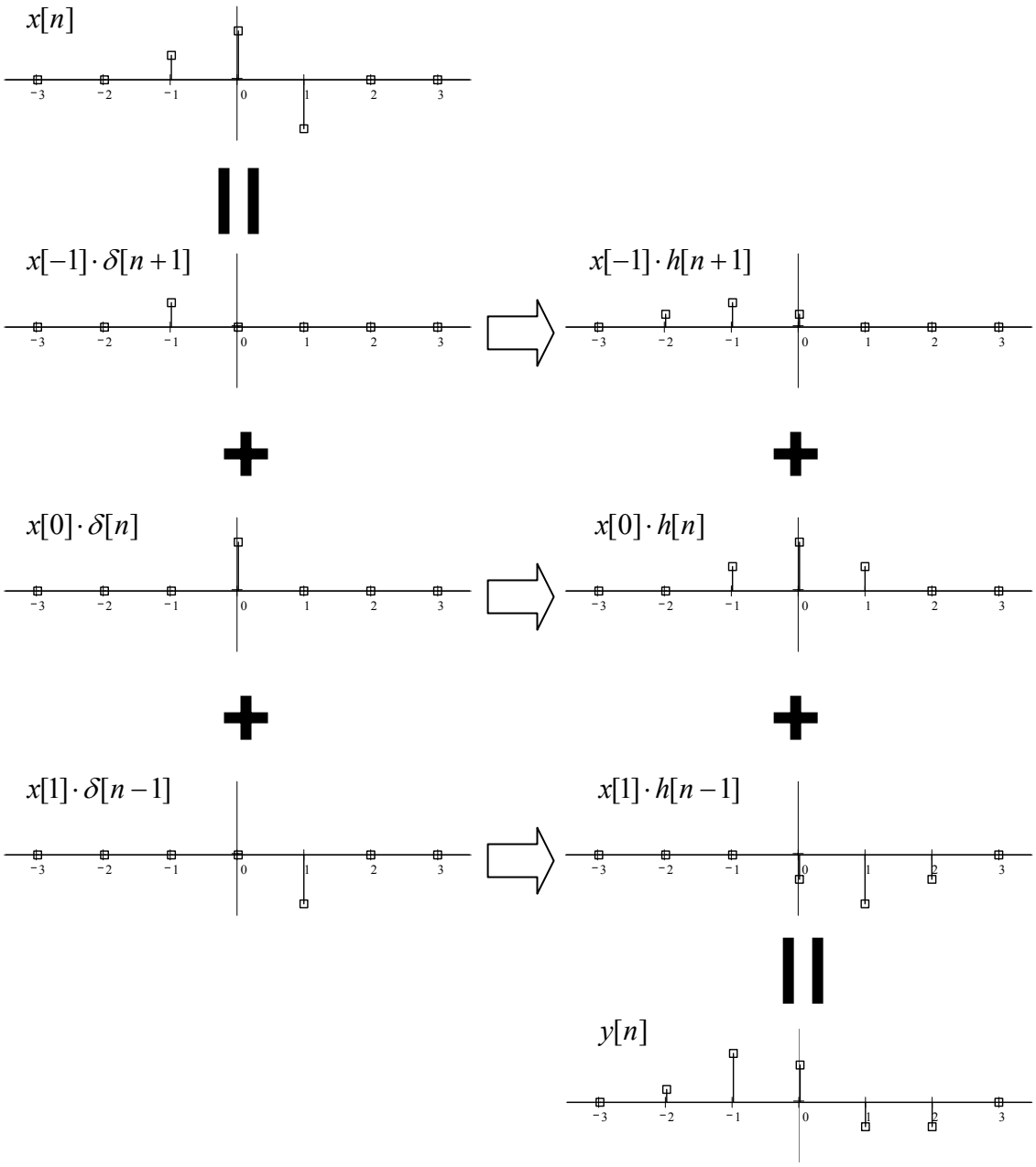


***Рис. 5**. Вычисление выходного сигнала линейной системы по входному сигналу и импульсной характеристике системы h[n].*

Результирующая формула для вычисления выходного сигнала $y[n]$ по входному сигналу $x[n]$ такова: $y[n] = \sum_{k=-\infty}^{+\infty} x[n-k] \cdot h[k]$. Сигнал $h[n]$ называется *импульсной характеристикой* (*impulse response*) системы, т.к. он является откликом системы на единичный импульс (дельта-функцию).

Еще раз рассмотрим алгоритм вычисления отклика линейной системы на произвольный сигнал. Сделаем это на примере конкретной линейной системы по обработке изображения. Дискретное изображение – это двумерный сигнал $x[i,j]$, обозначающий яркость в каждой дискретной точке (i,j) на плоскости. Дельта-функция в двумерном случае – это единичная светлая точка с координатами (0,0) на черном фоне. Пусть наша линейная система такова, что ее импульсная характеристика равна $h[i,j]$, такой что $h[i,j]=const$ на всех точках внутри круга с центром в точке (0,0) и диаметром 3, и равна нулю вне этого круга. Притом, интеграл от $h[i,j]$ по всей плоскости равен 1 (из этого условия выбираем константу *const*).

Рассмотрим действие такой системы на изображение, состоящее из одной точки на черном фоне (но теперь точка имеет произвольные координаты). Заметим, что на изображение $\delta[i-m, j-n]$ (дельта-функция, сдвинутая в точку (m,n)) любая линейная система отвечает изображением $h[i\text{-}m, j\text{-}n]$ в силу инвариантности к сдвигу. Таким образом, на единичные точки в любом месте система отвечает кругами диаметра 3 с центром в положении этих единичных точек. То есть точка как бы размывается в круг. Поэтому в компьютерной графике импульсную характеристику линейной системы называют *PSF – point spread function*, т.е. функция «размытия» точки.

Как было замечено ранее, любой одномерный сигнал можно представить в виде линейной комбинации сдвинутых дельта-функций. Очевидно, аналогично можно представить и изображение в виде линейной комбинации изображений, состоящих каждое из одной точки. Так как система линейная, то после обработки суммы этих изображений (то есть целого изображения) получится изображение, являющееся суммой отдельных кругов, получившихся от каждой точки. Другими словами, каждая точка изображения размоется до круга, а потом все круги наложатся. Таким образом, данная линейная система осуществляет *размытие* изображения.

Отметим, что существует множество других линейных систем (с другими импульсными характеристиками), которые выполняют другие задачи. Например, есть линейные системы для придания изображениям резкости, для выделения краев, для придания эффекта «тиснения». Все эти линейные системы называются *фильтрами*.

Упражнения

1. Найти импульсную характеристику системы $x[n] \longrightarrow x[n]$ (тождественное отображение).

2. Найти импульсную характеристику «усилителя»: $x[n] \longrightarrow 2 \cdot x[n]$.

3. Что делает с сигналом система с импульсной характеристикой $h[n] = \delta[n-1]$?

4. Что делает система с импульсной характеристикой $h[n] = \delta[n] + \frac{1}{2}\delta[n-2]$?

Ответы

1. Импульсная характеристика равна дельта-функции: $h[n] = \delta[n]$. Это легко проверить непосредственно.

2. $h[n] = 2 \cdot \delta[n]$.

3. Такая система называется единичной задержкой. Она сдвигает входной сигнал на один отсчет во времени: $y[n]=x[n-1]$.

4. Такая система добавляет к сигналу однократное «эхо» (задержанную копию сигнала). Задержка эха составляет 2 отсчета, а амплитуда эха в 2 раза меньше амплитуды сигнала.

Свертка

Существует несколько способов вычисления отклика линейной системы на произвольное изображение. Один из них указан в предыдущем параграфе. Каждая точка сигнала превращается в функцию *h* (сдвинутую в нужную позицию и умноженную на величину данной точки сигнала), а потом все эти функции складываются.

Другой способ выполнения того же самого заключается в том, что мы вычисляем значение каждой точки в результирующем сигнале как взвешенную сумму некоторого множества соседних точек исходного сигнала. Коэффициенты этой суммы совпадают с импульсной характеристикой линейной системы, перевернутой относительно точки 0. Например, в рассмотренной ранее системе размытия изображения каждую точку полученного сигнала можно вычислить как среднее арифметическое из точек исходного, сигнала, попадающих в соответствующий круг (диаметра 3 с центром в искомой точке). Отсюда и берется формула для одномерного случая:

$$y[n] = \sum_{k=-\infty}^{+\infty} x[n-k] \cdot h[k] \quad \text{(формула свертки)}$$

Рассмотренная операция получения результирующего сигнала по исходному называется *сверткой* (*convolution*). Итак, любая линейная система осуществляет свертку входного сигнала со своей импульсной характеристикой. Это записывается так: $y[n] = x[n] * h[n]$. Функция $h[n]$ называется *ядром свертки* (*kernel*) или импульсной характеристикой линейной системы.

Обычно все сигналы, обрабатываемые на компьютере, имеют конечную продолжительность (т.е. отличны от нуля лишь на конечном отрезке). Рассмотрим, что происходит с сигналом конечной продолжительности, когда его *сворачивают* с конечным ядром свертки. Пусть сигнал $x[n]$ отличен от нуля только на

отрезке от 0 до N-1 включительно («имеет длину N»). Пусть ядро свертки $h[n]$ отлично от нуля на отрезке от $-m_1$ до m_2 включительно, состоящем из M точек ($M = m_1 + m_2 + 1$). Тогда при подстановке этих сигналов в формулу свертки, мы получим сигнал $y[n]$, который отличен от нуля на отрезке от $-m_1$ до $N-1+m_2$ включительно. Таким образом, длина результирующего сигнала равна N+M-1, т.е. сумме длин исходного сигнала и ядра свертки минус один.

Итак, операция свертки расширяет сигнал на M-1 точку, где M – длина ядра свертки.

Свойства свертки:

1. $x[n] * y[n] = y[n] * x[n]$ (т.е. можно переставлять местами исходный сигнал и ядро свертки – это свойство редко используется на практике).
2. $(x[n] * y[n]) * z[n] = x[n] * (y[n] * z[n])$ (т.е. вместо того, чтобы проводить свертку по очереди в разных системах, можно получить систему с ядром $(y[n] * z[n])$, которая является суперпозицией систем $y[n]$ и $z[n]$).
3. $x[n] * y[n] + x[n] * z[n] = x[n] * (y[n] + z[n])$

Упражнения

1. Сигнал $x[n]$, отличный от нуля на отрезке $[A, B]$, сворачивается с сигналом $h[n]$, отличным от нуля на отрезке $[C, D]$. Найти отрезок, на котором может быть отличен от нуля результирующий сигнал.
2. Посчитать, сколько умножений нужно произвести для вычисления свертки сигнала длины N с ядром длины M.

Ответы

1. Самый левый отсчет сигнала $x[n]$ с координатой A перейдет в результате свертки в ядро свертки в позициях $[A+C, A+D]$. Самый правый отсчет сигнала $x[n]$ с координатой B перейдет в результате свертки в ядро свертки в позициях $[B+C, B+D]$. Таким образом, весь результирующий сигнал займет отрезок $[A+C, B+D]$. Длина сигнала равна $(B-A)+(D-A)+1$, что согласуется с формулой «N+M-1».
2. Для вычисления свертки первым рассмотренным способом нужно каждую из N точек исходного сигнала перевести в M точек импульсной характеристики и сложить соответствующие точки импульсной характеристики. Для каждой точки исходного сигнала нужно умножить на нее вектор из M точек импульсной характеристики (M умножений). Итак, всего для вычисления свертки требуется $M*N$ умножений.

Корреляция

Часто возникает задача обнаружения одного сигнала в другом. Например, может быть известно, что некоторое внешнее событие генерирует в датчике сигнал определенной формы. Однако события могут приходить почти одновре-

менно, а сигналы от них – перекрываться. Кроме того, на выходе датчика может присутствовать шум, затрудняющий нахождение нужных сигналов. Для надежного обнаружения таких сигналов применяется метод *корреляции* (*correlation*).

Пусть датчик генерирует сигнал $x[n]$, и мы хотим обнаружить в нем последовательность $g[n]$ некоторой конечной длины. Для поиска этого сигнала вычисляются скалярные произведения сигналов $x[n]$ и $g[n-k]$ для различных k. То есть мы как бы пытаемся «приложить» искомый сигнал во всех возможных положениях к сигналу с датчика и найти их «степень похожести» (скалярное произведение) для каждого положения. Таким образом, на выходе мы получаем сигнал $y[k]$, показывающий, насколько сигнал с датчика $x[n]$ в позиции k похож на искомый сигнал $g[n]$. Формула вычисления корреляции такова: $y[k] = \sum_{i=-\infty}^{+\infty} g[i] \cdot x[i-k]$. Здесь суммирование можно проводить в конечных пределах, т.е. только для тех i, для которых искомый сигнал $g[i]$ отличен от нуля. Бесконечные пределы суммирования, как и в формуле свертки, записываются для общности (чтобы можно было проводить корреляцию с сигналом любой длины).

Смысл получающегося сигнала $y[n]$ в том, что его величины для каждого n показывают, насколько входной сигнал в позиции n похож на искомый сигнал. Если во входном сигнале присутствует только шум, то и значения корреляции будут шумом небольшой амплитуды. Но как только в шуме входного сигнала появится форма, похожая на искомый сигнал, так значение корреляции в этой точке станет высоким.

Перепишем формулу корреляции в общепринятом виде:

$$y[n] = \sum_{k=-\infty}^{+\infty} x[n+k] \cdot g[k] \qquad \text{(формула корреляции)}$$

Как видно, формула вычисления корреляции очень похожа не формулу вычисления свертки. Действительно, если в формуле корреляции сделать замену переменной суммирования, то мы придем к эквивалентной формуле корреляции $y[n] = \sum_{k=-\infty}^{+\infty} x[n-k] \cdot g[-k]$. Эта формула совпадает с формулой свертки, если положить ядро свертки $h[k]=g[-k]$. Таким образом, корреляцию можно вычислять как свертку, положив в качестве ядра свертки искомый сигнал, развернутый относительно нулевой точки.

Иногда корреляцию называют *кросс-корреляцией* или *перекрестной корреляцией* (*cross-correlation*). Термин «*автокорреляция*» (*autocorrelation*) применяется, когда находится корреляция сигнала с самим собой. Смысл этой операции в том, чтобы найти наиболее вероятные периоды повторения формы исходного сигнала.

Дискретное преобразование Фурье

Вычисление ДПФ

Многие сигналы удобно анализировать, раскладывая их на синусоиды (гармоники). Тому есть несколько причин. Например, подобным образом работает человеческое ухо. Оно раскладывает звук на отдельные колебания различных частот. Кроме того, можно показать, что синусоиды являются «собственными функциями» линейных систем (т.к. они проходят через линейные системы, не изменяя формы, а могут изменять лишь фазу и амплитуду). Еще одна причина в том, что теорема Котельникова формулируется в терминах спектра сигнала.

Преобразование Фурье (*Fourier transform*)– это разложение функций на синусоиды (далее косинусные функции мы тоже называем синусоидами, т.к. они отличаются от «настоящих» синусоид только фазой). Существует несколько видов преобразования Фурье.

1. Непериодический непрерывный сигнал можно разложить в интеграл Фурье.
2. Периодический непрерывный сигнал можно разложить в бесконечный ряд Фурье.
3. Непериодический дискретный сигнал можно разложить в интеграл Фурье.
4. Периодический дискретный сигнал можно разложить в конечный ряд Фурье.

Компьютер способен работать только с ограниченным объемом данных, следовательно, реально он способен вычислять только последний вид преобразования Фурье. Рассмотрим его подробнее.

Пусть дискретный сигнал $x[n]$ имеет период N точек. В этом случае его можно представить в виде конечного ряда (т.е. линейной комбинации) дискретных синусоид:

$$x[n] = \sum_{k=0}^{N/2} C_k \cos\frac{2\pi k(n+\varphi_k)}{N} \qquad \text{(ряд Фурье)}$$

Эквивалентная запись (каждый косинус раскладываем на синус и косинус, но теперь – без фазы):

$$x[n] = \sum_{k=0}^{N/2} A_k \cos\frac{2\pi kn}{N} + \sum_{k=0}^{N/2} B_k \sin\frac{2\pi kn}{N} \qquad \text{(ряд Фурье)}$$

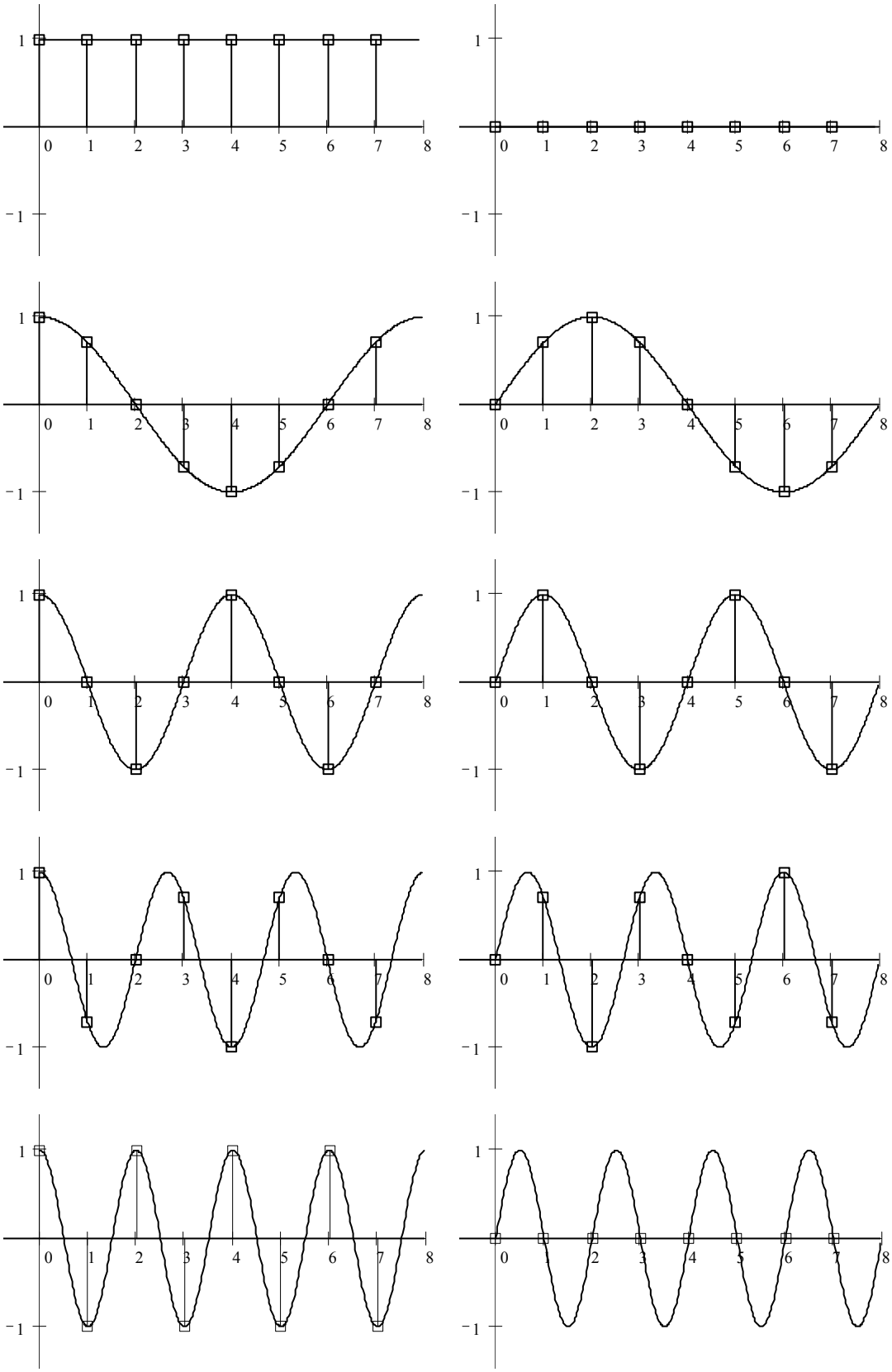


***Рис. 6**. Базисные функции ряда Фурье для 8-точеченого дискретного сигнала. Слева – косинусы, справа – синусы. Частоты увеличиваются сверху вниз.*

Базисные синусоиды имеют кратные частоты. Первый член ряда (k=0) – это константа, называемая *постоянной составляющей* (*DC offset*) сигнала. Самая первая синусоида (k=1) имеет такую частоту, что ее период совпадает с периодом самого исходного сигнала. Самая высокочастотная составляющая ($k=N/2$) имеет такую частоту, что ее период равен двум отсчетам. Коэффициенты A_k и B_k называются *спектром* сигнала (*spectrum*). Они показывают амплитуды синусоид, из которых состоит сигнал. Шаг по частоте между двумя соседними синусоидами из разложения Фурье называется *частотным разрешением* спектра.

На рис. 6 показаны синусоиды, по которым происходит разложение дискретного сигнала из 8 точек. Каждая из синусоид состоит из 8 точек, то есть является обычным дискретным сигналом. Непрерывные синусоиды показаны на рисунке для наглядности.

Как мы увидим далее, для каждого сигнала можно однозначно определить коэффициенты A_k и B_k. Зная эти коэффициенты, можно однозначно восстановить исходный сигнал, вычислив сумму ряда Фурье в каждой точке. Разложение сигнала на синусоиды (т.е. получение коэффициентов) называется *прямым преобразованием Фурье*. Обратный процесс – синтез сигнала по синусоидам – называется *обратным преобразованием Фурье* (*inverse Fourier transform*).

Алгоритм обратного преобразования Фурье очевиден (он содержится в формуле ряда Фурье; для проведения синтеза нужно просто подставить в нее коэффициенты). Рассмотрим алгоритм прямого преобразования Фурье, т.е. нахождения коэффициентов A_k и B_k.

Система функций $\left\{\sin\frac{2\pi kn}{N}, \cos\frac{2\pi kn}{N}\right\}, k = 0,..., \frac{N}{2}$ от аргумента n является ортогональным базисом в пространстве периодических дискретных сигналов с периодом N. Это значит, что для разложения по ней любого элемента пространства (сигнала) нужно посчитать скалярные произведения этого элемента со всеми функциями системы, и полученные коэффициенты нормировать. Тогда для исходного сигнала будет справедлива формула разложения по базису с коэффициентами A_k и B_k.

Итак, коэффициенты A_k и B_k вычисляются как скалярные произведения (в непрерывном случае – интегралы от произведения функций, в дискретном случае – суммы от произведения дискретных сигналов):

$$A_k = \frac{2}{N}\sum_{i=0}^{N-1} x[i]\cos\frac{2\pi ki}{N}, \text{ при } k = 1,..., \frac{N}{2} - 1$$

$$A_k = \frac{1}{N}\sum_{i=0}^{N-1} x[i]\cos\frac{2\pi ki}{N}, \text{ при } k = 0, \frac{N}{2}$$

$$B_k = \frac{2}{N}\sum_{i=0}^{N-1} x[i]\sin\frac{2\pi ki}{N}, \text{ при } k = 0,..., \frac{N}{2}$$

Возникает вопрос: почему в исходном сигнале N чисел, а описывается он с помощью N+2 коэффициентов? Вопрос разрешается следующим образом: коэффициенты B_0 и $B_{N/2}$ всегда равны нулю (т.к. соответствующие им «базисные» сигналы тождественно равны нулю в дискретных точках), и их можно отбросить при вычислении обратного и прямого преобразований Фурье.

Итак, мы выяснили, что спектральное представление сигнала полностью эквивалентно самому сигналу. Между ними можно перемещаться, используя прямое и обратное преобразования Фурье. Алгоритм вычисления этих преобразований содержится в приведенных формулах.

Вычисление преобразований Фурье требует очень большого числа умножений (около N^2) и вычислений синусов. Существует способ выполнить эти преобразования значительно быстрее: примерно за $N \cdot \log_2 N$ операций умножения. Этот способ называется *быстрым преобразованием Фурье* (БПФ, *FFT*, *fast Fourier transform*). Он основан на том, что среди множителей (синусов) есть много повторяющихся значений (в силу периодичности синуса). Алгоритм БПФ группирует слагаемые с одинаковыми множителями, значительно сокращая число умножений. В результате быстродействие БПФ может в сотни раз превосходить быстродействие стандартного алгоритма (в зависимости от N). При этом следует подчеркнуть, что алгоритм БПФ является точным. Он даже точнее стандартного, т.к. сокращая число операций, он приводит к меньшим ошибкам округления.

Однако у большинства алгоритмов БПФ есть особенность: они способны работать лишь тогда, когда длина анализируемого сигнала N является степенью двойки. Обычно это не представляет большой проблемы, так как анализируемый сигнал всегда можно дополнить нулями до необходимого размера. Число N называется *размером* или *длиной БПФ* (*FFT size*).

Комплексное ДПФ

До сих пор мы рассматривали ДПФ от действительных сигналов. Обобщим теперь ДПФ на случай комплексных сигналов. Пусть $x[n]$, n=0,…,N-1 – исходный комплексный сигнал, состоящий из N комплексных чисел. Обозначим $X[k]$, k=0,…N-1 – его комплексный спектр, также состоящий из N комплексных чисел. Тогда справедливы следующие формулы прямого и обратного преобразований Фурье (здесь $j = \sqrt{-1}$):

$$X[k] = \sum_{n=0}^{N-1} x[n] \cdot e^{-jnk(2\pi/N)}$$

$$x[n] = \frac{1}{N} \sum_{k=0}^{N-1} X[k] \cdot e^{jnk(2\pi/N)}$$

Если по этим формулам разложить в спектр действительный сигнал, то первые N/2+1 комплексных коэффициентов спектра будут совпадать со спектром «обычного» действительного ДПФ, представленным в «комплексном» виде, а остальные коэффициенты будут их симметричным отражением относительно

половины частоты дискретизации. Для косинусных коэффициентов отражение четное, а для синусных – нечетное.

Двумерное ДПФ

Для изображений, представляющих собой двумерный сигнал, спектром является также двумерный сигнал. Базисные функции преобразования Фурье имеют вид $h^{\pm}_{k_1,k_2}(n_1,n_2) = \sin\left(\frac{2\pi k_1 n_1}{N_1} \pm \frac{2\pi k_2 n_2}{N_2}\right)$, причем фазы также могут быть различны. На изображении каждая из этих базисных функций представляют собой волну определенной частоты, определенной ориентации и определенной фазы.

Здесь N_1xN_2 – размер исходного сигнала, он же – размер спектра. k_1 и k_2 – это номера базисных функций (номера коэффициентов двумерного ДПФ, при которых эти функции находятся). Поскольку размер спектра равен размеру исходного сигнала, то $k_1 = 0,\ldots,N_1\text{-}1$; $k_2 = 0,\ldots,N_2\text{-}1$.

n_1 и n_2 – переменные-аргументы базисных функций. Поскольку область определения базисных функций совпадает с областью определения сигнала, то $n_1 = 0,\ldots,N_1\text{-}1$; $n_2 = 0,\ldots,N_2\text{-}1$.

Двумерное ДПФ (в комплексной форме) определяется следующими формулами (здесь $x[n_1,n_2]$ - исходный сигнал, а $X[k_1,k_2]$ – его спектр):

$$X[k_1,k_2] = \sum_{n_1=0}^{N_1-1}\sum_{n_2=0}^{N_2-1} x[n_1,n_2]\cdot e^{-jn_1k_1(2\pi/N_1)}e^{-jn_2k_2(2\pi/N_2)}$$

$$x[n_1,n_2] = \frac{1}{N_1N_2}\sum_{k_1=0}^{N_1-1}\sum_{k_2=0}^{N_2-1} X[k_1,k_2]\cdot e^{jn_1k_1(2\pi/N_1)}e^{jn_2k_2(2\pi/N_2)}$$

Непосредственное вычисление двумерного ДПФ по приведенным формулам требует огромных вычислительных затрат. Однако можно доказать, что двумерное ДПФ обладает свойством сепарабельности, т.е. его можно вычислить последовательно по двум измерениям. Для вычисления двумерного ДПФ достаточно вычислить одномерные комплексные ДПФ всех строк изображения, а затем вычислить в результирующем «изображении» одномерные комплексные ДПФ всех столбцов. При этом результаты всех одномерных комплексных ДПФ нужно записывать на место исходных данных для этих ДПФ. Например, при вычислении одномерного ДПФ первой строки изображения нужно результат ДПФ записать в первую строку этого изображения (он имеет тот же размер). Для этого нужно каждый «пиксель» хранить в виде комплексного числа.

Таким образом, эффективный алгоритм вычисления ДПФ изображения заключается в вычислении одномерных БПФ сначала от всех строк, а потом – от всех столбцов изображения.

Упражнения

1. Частота дискретизации сигнала равна 44100 Гц. Размер БПФ равен 4096. Какова длина анализируемого блока в секундах? По каким частотам (в герцах) будет разложен сигнал?

2. Какое частотное разрешение спектра мы получим в предыдущем примере? Какой размер БПФ нужно использовать, чтобы получить частотное разрешение около 4 Гц?

Ответы

1. Продолжительность сигнала из 4096 точек при данной частоте дискретизации составляет 0.0929 секунды. Сигнал будет разложен на постоянную составляющую (0 Гц) и частоты, кратные частоте первой синусоиды разложения. Период первой синусоиды равен 4096 точкам, что по времени составляет примерно 0.0929 секунды. Значит, частота первой синусоиды будет 10.77 Гц. Частота второй будет 21.53 Гц. И так далее. Частота последней синусоиды, с номером 2048 будет равна половине частоты дискретизации, 22050 Гц.

2. Частотное разрешение равно 10.77 Гц. Очевидно, что при фиксированной частоте дискретизации частотное разрешение пропорционально размеру БПФ. Чтобы получить разрешение 4 Гц, нужно увеличить размер БПФ примерно в 2.5 – 3 раза. Так как размер БПФ может быть равен только степеням двойки, то можно принять размер БПФ за 16384 и получить разрешение 2.7 Гц.

Применения ДПФ

Спектральный анализ

Часто ДПФ применяется для наблюдения и анализа спектра сигнала. При этом часто наиболее интересными являются лишь амплитуды C_k отдельных гармоник, а не их фазы. В этом случае спектр обычно отображается в виде графика зависимости амплитуды от частоты (рис. 8). Часто шкала частот градуируется в децибелах. *Децибелы* измеряют не сами амплитуды, а их отношения. Например, разница на 20 дБ означает различие амплитуд в 10 раз, разница на 40 дБ означает отношение амплитуд в 100 раз. Различию амплитуд в 2 раза отвечает разница примерно на 6 дБ. Формула для вычисления разницы в децибелах такова: $D = 20 \cdot \lg \frac{a_1}{a_2}$. Здесь a_1 и a_2 – сравниваемые амплитуды.

Шкала частот также часто градуируется в логарифмическом масштабе.

Перед вычислением спектра сигнала нужно выбрать отрезок сигнала, на котором будет вычисляться спектр. Длина отрезка должна быть степенью двойки (для работы БПФ). Иначе сигнал надо дополнить нулями до нужной длины. После этого к выбранному участку сигнала применяют БПФ. Коэффициенты амплитуд считают по формуле $C_k = \sqrt{A_k^{\,2} + B_k^{\,2}}$.

При вычислении спектра указанным образом возможен следующий нежелательный эффект. При разложении функции в ряд Фурье мы полагаем, что функция периодическая, с периодом, равным размеру БПФ. Вычисляется спектр именно такой функции (а не той, из которой мы извлекли кусок). При

этом на границах периодов такая функция наверняка будет иметь разрывы (ведь исходная функция не была периодической). А разрывы в функции сильно отражаются на ее спектре, искажая его.

Для устранения этого эффекта применяются так называемые *взвешивающие окна*. Они плавно сводят на нет функцию вблизи краев анализируемого участка. Весовые окна имеют форму, похожую на гауссиан. Выбранный для анализа участок сигнала домножается на весовое окно, которое устраняет разрывы функции при «зацикливании» данного участка сигнала. Виртуальное «зацикливание» происходит при ДПФ, так как алгоритм ДПФ полагает, что функция периодическая. Существует множество весовых окон, названных в честь их создателей. Все они имеют похожую форму и в значительной степени устраняют рассмотренные искажения спектра. Мы приведем формулы двух хороших окон: Хэмминга (*Hamming window*) и Блэкмана (*Blackman window*) (рис. 7):

$$w_{Hamm}[i] = 0.54 - 0.46\cos\frac{2\pi i}{N}$$

$$w_{Blackm}[i] = 0.42 - 0.5\cos\frac{2\pi i}{N} + 0.08\cos\frac{4\pi i}{N}$$

Здесь окно применяется к сигналу с индексами от 0 до N. Окно Хэмминга наиболее часто используется. Окно Блэкмана обладает более сильным действием по устранению рассмотренных искажений, однако имеет свои недостатки.

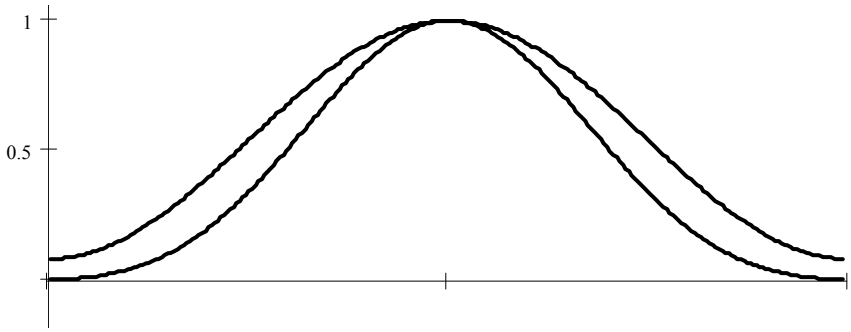


***Рис.** 7. Взвешивающие окна Хэмминга (верхнее) и Блэкмана (нижнее).*

Важное свойство спектрального анализа заключается в том, что не существует одного, единственно правильного спектра какого-либо сигнала. Спектр можно вычислять с применением различных размеров БПФ и различных весовых окон. Для каждого конкретного приложения предпочтительно использовать свои способы. От выбора размера БПФ зависит разрешение спектра по частоте и по времени. Если выбрать длинный участок сигнала для разложения в спектр, то мы получим хорошее разрешение по частоте, но плохое по времени (т.к. спектр будет отражать усредненное поведение сигнала на всем участке взятия БПФ). Если для разложения в спектр выбрать короткий участок сигнала, то мы получим более точную локализацию по времени, но плохое разрешение по частоте (т.к. в преобразовании Фурье будет слишком мало базисных частот). В этом заключается фундаментальный **принцип соотношения неопределенностей** при вычислении спектра: невозможно одновременно получить хорошее

разрешение спектра и по частоте, и по времени: эти разрешения обратно пропорциональны.

Еще одно важное свойство спектрального анализа заключается в том, что при разложении в спектр мы находим не те синусоидальные составляющие, из которых состоял исходный сигнал, а лишь находим, с какими амплитудами нужно взять определенные кратные частоты, чтобы получить исходный сигнал. Другими словами, разложение проводится не по «частотам исходного сигнала», а по «базисным частотам алгоритма БПФ». Однако обычно (особенно при использовании весовых окон) этого почти не заметно по графику спектра, то есть график спектра достаточно адекватно отображает именно частоты исходного сигнала.

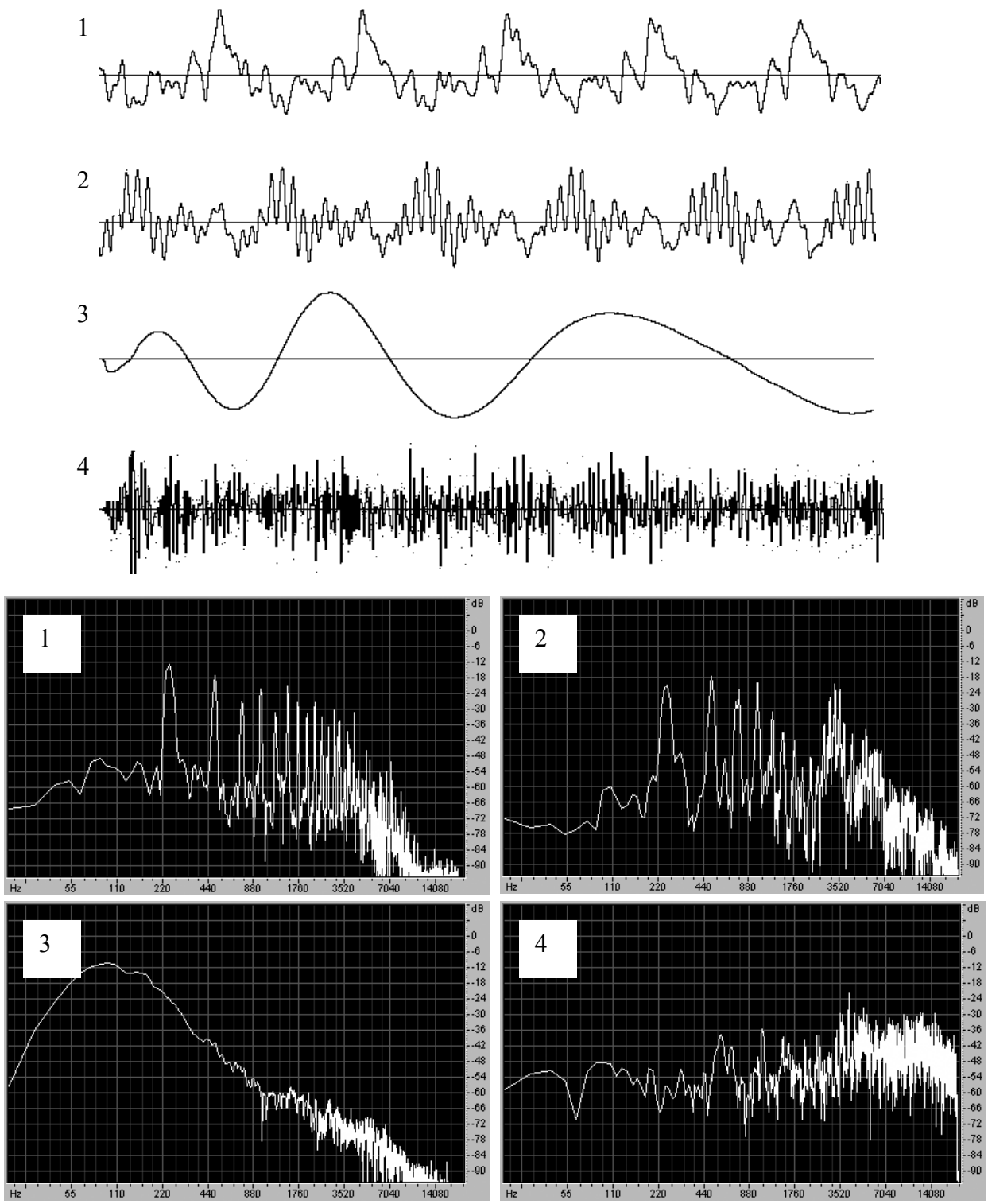


Рис. 8. *Фрагменты различных сигналов (около 800 точек) и спектры более длинных отрезков этих сигналов (4096 точек). Сверху вниз: нота на фортепиано, голос (пение), барабан (бочка), тарелка (открытый хэт).*

Быстрая свертка и корреляция. Теорема свертки.

Свертка – один из важнейших процессов в цифровой обработке сигналов. Поэтому важно уметь эффективно ее вычислять. Прямое вычисление свертки требует *N*M* умножений, где *N* – длина исходного сигнала, а *M* – длина ядра свертки. Часто длина ядра свертки достигает нескольких тысяч точек, и число умножений становится огромным.

Однако существует алгоритм, позволяющий вычислить свертку значительно быстрее. Этот алгоритм основан на следующей важной теореме, которую мы приводим в нестрогой формулировке:

Теорема свертки: **свертка во временной области эквивалентна умножению в частотной области; умножение во временной области эквивалентно свертке в частотной области**.

Это значит, что для выполнения свертки двух сигналов можно перевести их в частотную область, **умножить** их спектры и перевести их обратно во временную область. Такая операция выглядит громоздко. Однако с появлением алгоритмов БПФ, позволяющих быстро вычислять преобразования Фурье, вычисление свертки через частотную область стало широко использоваться. При значительных длинах ядра свертки такой подход позволяет в сотни раз сократить время вычисления свертки.

Кратко опишем алгоритм быстрого однократного вычисления свертки. Сначала исходный сигнал длины *N* и ядро свертки длины *M* дополняются нулями до длины *L* (*L* – степень двойки), причем так, что $L \geq N + M - 1$. Затем вычисляются ДПФ этих двух сигналов. Затем спектры сигналов необходимо перемножить как состоящие из комплексных чисел, т.е. образовать новый спектр из коэффициентов A_k^{new} и B_k^{new}, получающихся по формуле: $A_k^{new} + iB_k^{new} = (A_k^{old1} + iB_k^{old1}) \cdot (A_k^{old2} + iB_k^{old2})$, где $k = 0,..., \frac{L}{2}$, $i = \sqrt{-1}$. Затем из полученного спектра с помощью обратного ДПФ вычисляется сигнал, состоящий из *L* точек. Этот сигнал и содержит результат свертки из *N*+*M*-1 точек, дополненный нулями до *L* точек.

Часто возникает потребность вычислить свертку очень длинного сигнала, не помещающегося в памяти компьютера, с относительно коротким ядром свертки. В таких случаях применяется так называемая ***секционная свертка***. Суть ее состоит в том, что длинный сигнал разбивается на более короткие части и каждая из этих частей сворачивается с ядром отдельно. Затем полученные части объединяются для получения окончательного результата. Для объединения полученных частей достаточно их разместить друг за другом с наложением (перекрытием) в *M*-1 точку, где *M* – длина ядра свертки. В местах перекрытия необходимо произвести суммирование. Обычно для ускорения вычислений размер секции входного сигнала выбирается одного порядка с длиной свертки.

Отметим, что поскольку корреляцию можно вычислять с помощью свертки, то рассмотренный алгоритм подходит и для быстрого вычисления корреляции.

Фильтрация

Эффект от умножения спектров сигналов при свертке называется ***фильтрацией***. Когда спектры умножаются как комплексные числа, происходит умножение амплитуд гармоник C_i исходного сигнала и ядра свертки. Таким образом, мы получаем возможность изменять спектр сигнала. Это очень полезная операция. Например, в звукозаписи изменение спектра сигнала позволяет очищать запись от шумов, компенсировать искажения сигнала различными устройствами звукозаписи, менять тембры инструментов, акцентировать внимание слушателя на отдельных партиях. В обработке изображений фильтрация позволяет применять к изображению разные эффекты: размытие, подчеркивание границ, тиснение и многие другие. В других областях фильтрация часто служит для разделения различных сигналов, смешанных в один, очищения сигнала от шумов. Также фильтрация является составным компонентом многих других, более сложных процессов.

Ядро свертки при фильтрации часто называют ***фильтром***. Часто фильтром называется также все устройство, которое осуществляет процесс фильтрации. *Длина (размер) фильтра* – это длина ядра свертки.

В общем случае, фильтр меняет в спектре сигнала и амплитуды гармоник, и их фазы. Однако фильтры можно проектировать так, чтобы они не меняли фазу сигнала. Такие фильтры называются *фильтрами с линейной фазой*. Это означает, что если они и меняют фазу сигнала, то делают это так, что все гармоники сигнала сдвигаются по времени на одну и ту же величину. Таким образом, фильтры с линейной фазой не искажают фазу сигнала, а лишь сдвигают весь сигнал во времени. Ядро свертки такого фильтра строго симметрично относительно своей центральной точки (хотя бывают типы фильтров с линейной фазой, где ядро антисимметрично).

Основное свойство любого фильтра – это его ***частотная*** (*frequency response*) и ***фазовая характеристики***. Они показывают, какое влияние фильтр оказывает на амплитуду и фазу различных гармоник обрабатываемого сигнала. Если фильтр имеет линейную фазу, то рассматривается только частотная характеристика фильтра. Обычно частотная характеристика изображается в виде графика зависимости амплитуды от частоты (в децибелах). Например, если фильтр пропускает все сигналы в полосе 0…10 кГц без изменения, а все сигналы в полосе выше 10 кГц подавляет в 2 раза (на 6 дБ), то частотная характеристика будет выглядеть так: $A(f) = \begin{cases} 0дБ, f < 10кГц \\ -6дБ, f > 10кГц \end{cases}$ (рис. 9). Частотная характеристика в 0 дБ показывает, что данные частоты фильтр пропускает без изменения. Те частоты, амплитуда которых ослабляется фильтром в 2 раза, должны иметь амплитуду на 6 дБ меньше. Поэтому их амплитуда составляет -6 дБ. Если бы фильтр усиливал какие-то частоты, то его частотная характеристика была бы на этих частотах положительная (в децибелах).

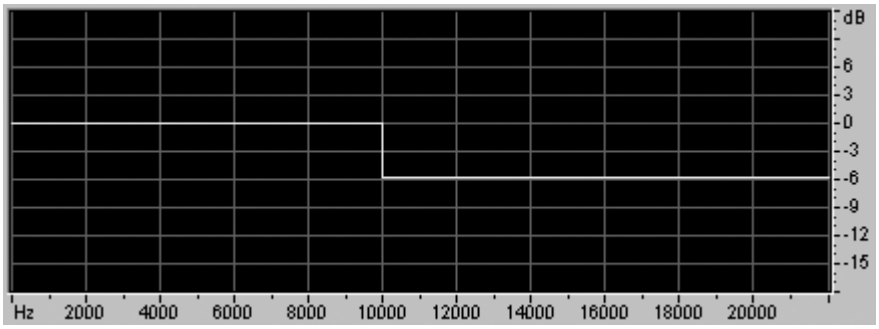


***Рис. 9**. Пример частотной характеристики фильтра.*

В зависимости от общего вида частотной характеристики можно выделить следующие распространенные типы фильтров: НЧ-фильтры (*low-pass filters*), *ВЧ-фильтры* (*high-pass filters*), *полосовые фильтры*, которые пропускают (*band-pass filters*) или подавляют (*band-reject filters*) сигнал только в определенной частотной полосе. Существуют и другие типы фильтров с более сложными частотными характеристиками.

Обычно в задачах фильтрации сигнала для фильтра задается требуемая частотная характеристика. Целью является построить фильтр, отвечающий заданным требованиям, и провести фильтрацию. Часто бывает невозможно построить в точности заданный фильтр. Тогда строится фильтр, близкий по характеристикам к заданному.

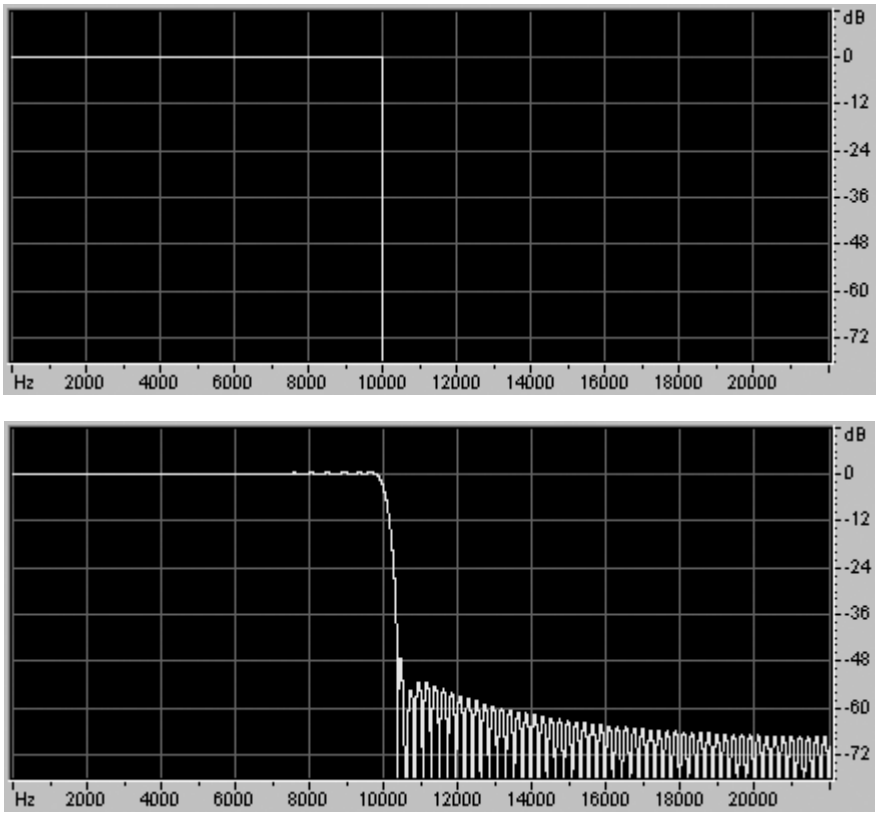


***Рис. 10**. Частотные характеристики идеального (сверху) и одного из реальных НЧ-фильтров.*

Например, невозможно построить *идеальный фильтр низких частот*, то есть такой, который пропускает без изменения все сигналы ниже определенной частоты (в *полосе пропускания, pass band*) и полностью подавляет все сигналы выше этой частоты (в *полосе подавления* или *непропускания, stop band*). Реальные фильтры низких частот обладают более плавной частотной характеристикой (рис. 10). Они не являются идеальными. Это значит, что они слегка изменяют сигнал в полосе пропускания. А в полосе подавления они не совершенно подавляют сигнал, а делают это, скажем, в 1000 раз. Их частотная характеристика обычно выглядит примерно так: до частоты среза она достаточно ровная (0 дБ почти без отклонений), на частоте среза она начинает плавно спадать. Крутизна спада и значения после спада определяются конкретным фильтром и требованиями к нему.

Например, требования к реальному фильтру могут быть такими: в полосе пропускания от 0 до 9500 Гц он должен иметь частотную характеристику 0 дБ с максимально возможным отклонением ±0.5 дБ. От 9500 Гц до 10500 Гц (в *переходной полосе*) частотная характеристика должна спадать. Выше 10500 Гц (в полосе подавления) фильтр должен подавлять все частоты не менее чем на 50 дБ. Это значит, что частотная характеристика фильтра в полосе подавления должна лежать ниже -50 дБ. Такой фильтр вполне возможно спроектировать (рис. 10).

Часто к фильтрам предъявляются более сложные требования. Например, фильтр может иметь несколько частотных полос пропускания и непропускания. Причем для полос пропускания могут быть заданы разные коэффициенты усиления, а для полос непропускания – разные коэффициенты подавления. Иногда требуемая частотная характеристика фильтра задается вообще произвольной кривой.

Существует множество способов построения фильтров с заданной частотной характеристикой. Мы кратко рассмотрим один из них. Это **проектирование фильтров с линейной фазой с помощью взвешивающих окон**. Этот способ является универсальным, т.к. позволяет получить фильтр с **любой заданной частотной характеристикой**. В то же время он достаточно прост и широко применяется.

Пусть частотная характеристика требуемого фильтра задана. Первый шаг при построении фильтра – это определение его размера. Размер фильтра определяется приблизительно из следующих соображений. Если частотная характеристика плавная, без сильных изломов или разрывов, то размер фильтра можно взять небольшой, иначе – большой. Если требуется точно следовать частотной характеристике, то размер фильтра должен быть большой, если допускаются отклонения от частотной характеристики, то размер фильтра можно уменьшить. Обычно сначала стоит попробовать построить фильтр размера порядка 100 – 1000 и оценить его соответствие заданной частотной характеристике. Если соответствие достаточное, то можно попытаться построить более короткий фильтр. Если же фильтр плохо отвечает частотной характеристике, нужно построить более длинный фильтр. Итак, пусть мы определили, что будем строить фильтр длины N. Желательно, чтобы N было нечетным числом.

Идея метода состоит в том, чтобы получить требуемое ядро свертки как обратное преобразование Фурье от требуемой частотной характеристики. Размер об-

ратного преобразования выбирается не меньшим, чем длина фильтра, и равным степени двойки. Пусть мы выбрали размер БПФ L. Для выполнения обратного преобразования Фурье необходимо иметь спектр сигнала на $\frac{L}{2}+1$ частотах, равномерно распределенных между 0 и половиной частоты дискретизации. Для этого сделаем соответствующую выборку из заданной частотной характеристики. Если частотная характеристика задана в децибелах, ее значения нужно перевести в «обыкновенные» значения амплитуд: 0 дБ – в 1.0, -6 дБ – в 0.5 и т.д. В полосе непропускания можно положить амплитуды строго равными нулю.

Теперь, когда значения частотной характеристики на заданных частотах выбраны, можно перевести эту частотную характеристику во временную область. Для выполнения обратного ДПФ нужно кроме амплитуд гармоник задать их фазы. Так как мы хотим получить фильтр с линейной фазой, то полагаем все фазы равными нулю. После этого переводим спектр из полярного представления («амплитуда-фаза») в прямоугольное ($A_k + iB_k$). Так как все фазы нулевые, то все точки спектра представляют собой «действительные числа». То есть в массив A_k без изменения загружаются амплитуды, а в массив B_k – нули. После этого выполняется обратное преобразование Фурье, в результате которого мы получаем сигнал $x[n]$, n=0,…,L-1, имеющий заданный спектр.

Однако полученный сигнал не очень хорошо подходит в качестве ядра свертки. Чтобы получить из него подходящее ядро свертки, нужно выполнить 2 шага. Первый шаг заключается в том, чтобы выбрать другой период сигнала, полученного в качестве ядра свертки. Когда мы проделали обратное ДПФ, мы получили один период бесконечного периодического сигнала. Теперь наша задача «развернуть» этот период в периодический сигнал и выбрать другие границы периода. Границы должны быть такими, чтобы центр нового отрезка совпадал с $x[0]$, и длина нового отрезка была равна выбранной длине фильтра N. Сигнал с этого нового отрезка и принимается за ядро фильтра. Если все было сделано правильно, ядро получается симметричное относительно точки $x[0]$. На практике для «разворачивания» ядра достаточно скопировать массив x[0,…,L-1] в массив x[-L,…-1] и выбрать из этого массива отрезок $x[-\frac{N-1}{2}],...,x[\frac{N-1}{2}]$.

Последний шаг при построении фильтра заключается в том, чтобы применить к ядру взвешивающее окно. Смысл этой операции в том, чтобы улучшить свойства фильтра. Выбор взвешивающего окна определяется необходимой степенью подавления в полосах непропускания фильтра. Разные весовые окна поразному влияют на этот параметр. Свойства и формулы весовых окон можно найти в книгах по цифровой обработке сигналов. Обычно хорошим выбором является окно Хэмминга. При необходимости еще большего подавления применяется окно Блэкмана.

После применения окна, можно оценить реальную частотную характеристику полученного фильтра. Для этого применяется метод спектрального анализа. Размер БПФ выбирается, как правило, примерно в 4 раза больше размера фильтра, фильтр дополняется до нужного размера нулями, и производится

ДПФ (весовые окна применять не нужно). По графику спектра можно определить степень соответствия фильтра требованиям к частотной характеристике. Если соответствие имеется, то проектирования фильтра закончено (хотя можно попытаться построить фильтр меньшего размера). Если соответствие недостаточно, то нужно попытаться построить фильтр большего размера.

После получения готового ядра фильтра, его можно использовать для проведения операции свертки (фильтрации). Пример действия фильтра на спектр сигнала можно увидеть на рис. 11. Заметим, что эффект от фильтрации, заключающийся в перемножении спектров, при логарифмическом масштабе амплитуд соответствует сложению амплитуд в децибелах.

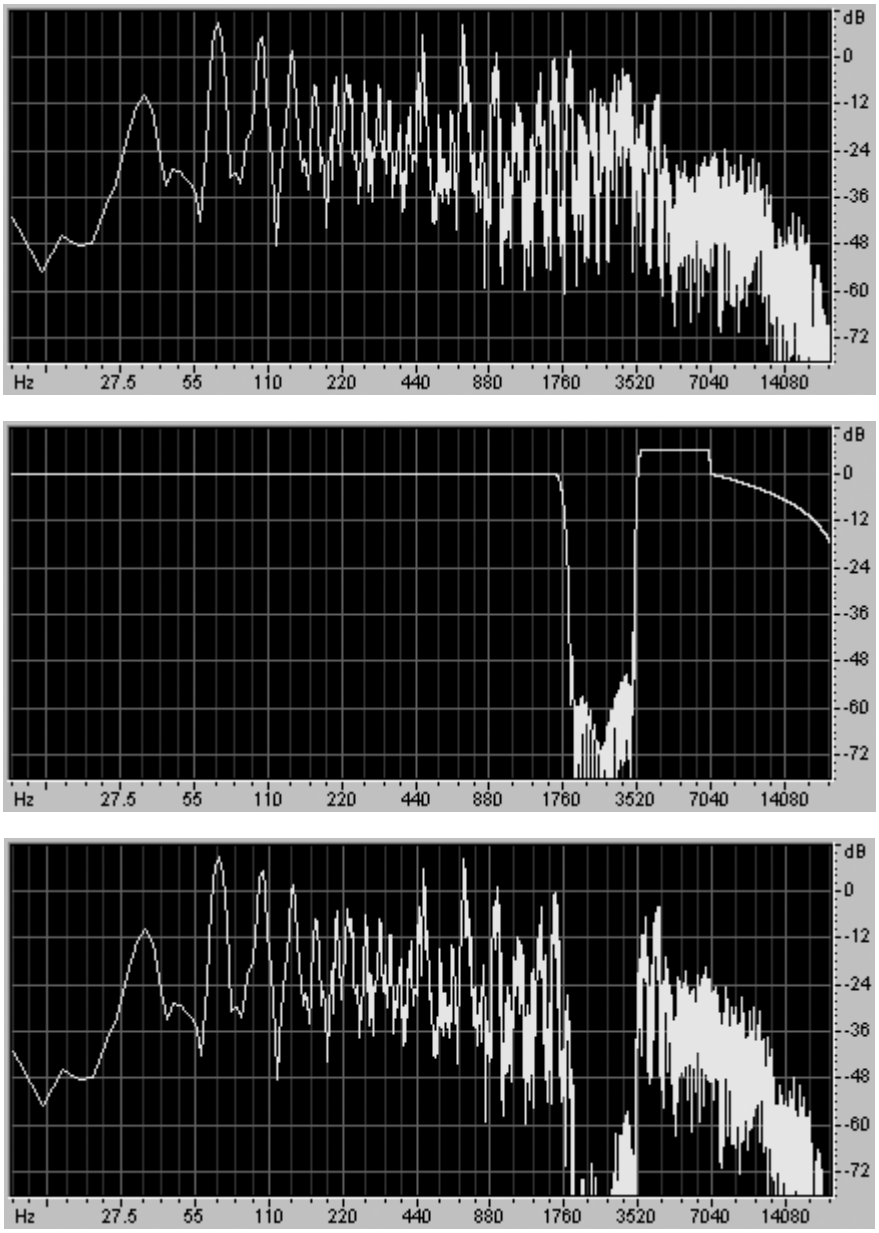


***Рис. 11**. Фильтрация звукового сигнала. Сверху вниз: спектр исходного сигнала, частотная характеристика фильтра, спектр сигнала после фильтрации.*

Деконволюция

Часто возникает необходимость **восстановить сигнал после искажения** его какой-либо линейной системой. Для такого восстановления необходимо хотя бы приблизительно знать частотную характеристику искажающей системы. Например, после записи звука недостаточно качественным микрофоном можно попытаться исправить записанный звук. Для этого нужно знать частотную характеристику микрофона (то есть то, как он меняет амплитуды гармоник в зависимости от их частоты). Часто она указывается в паспорте микрофона. Так как микрофон мы считаем линейной системой, то он осуществляет свертку исходного сигнала с некоторым ядром. Восстановление (хотя бы приближенное) исходного сигнала по свернутому называется *деконволюцией* (*deconvolution*).

Восстановление исходного сигнала во временной области проделать очень сложно. Зато частотная область позволяет выполнить деконволюцию вполне естественно. Достаточно учесть, что свертка в частотной области соответствует перемножению спектров сигналов. Причем спектр ядра свертки микрофона известен (это его частотная характеристика). Значит, восстановить исходный сигнал можно поделив спектр свернутого сигнала на спектр ядра свертки.

Итак, для выполнения деконволюции нужно построить фильтр (назовем его восстанавливающим), с частотной характеристикой, обратной искажающему, и выполнить свертку искаженного сигнала с этим фильтром.

Отметим, что построить такой фильтр не всегда возможно и не всегда целесообразно. Если искажающий фильтр какие-то частоты полностью подавляет, то, очевидно, восстановить их не удастся. Кроме того, гармоники, которые были подавлены достаточно сильно, могут оказаться ниже уровня шума обрабатывающей системы. В этом случае попытка их восстановления приведет к значительному усилению шума на данных частотах.

Кроме того, пока мы не затрагивали вопрос фазовых искажений, а полагали, что искажающая система обладает линейной фазовой характеристикой, и строили восстанавливающий фильтр тоже с линейной фазой. Но если искажающая система вносит в сигнал и фазовые искажения, то для их исправления приходится строить фильтр, который их компенсирует.

Однако не всегда требуется избавляться и от фазовых искажений. Например, при восстановлении сигнала с микрофона это обычно необязательно, так как наше ухо нечувствительно к небольшим фазовым искажениям звука.

Таким образом, мы видим, что далеко не всегда удается полностью восстановить исходный сигнал по свернутому. Но приблизиться к исходному сигналу в несколько раз вполне возможно.

Упражнения

При выполнении упражнений рекомендуется пользоваться готовыми функциями вычисления прямого и обратного БПФ. Сигнал желательно загружать из WAV-файла (см. приложение).

1. Реализовать нахождение и отображение спектра заданного участка сигнала. Ввести возможность выбора длины сигнала, размера БПФ, вида взвешивающего окна.
2. Реализовать быструю свертку двух сигналов через частотную область.
3. Реализовать секционную свертку двух сигналов через частотную область.
4. Реализовать алгоритм проектирования фильтра по заданной частотной характеристике. Спроектировать НЧ-фильтр с произвольными параметрами.

Указания

1. Вот приблизительный алгоритм действий:
   1. Ввести сигнал из файла в массив чисел с плавающей запятой.
   2. Домножить сигнал на весовое окно.
   3. Дополнить сигнал нулями до размера БПФ.
   4. Вычислить БПФ.
   5. По коэффициентам A_k и B_k вычислить коэффициенты C_k. Перевести коэффициенты C_k из значений амплитуд в децибелы.
   6. Отобразить график спектра, промаркировав ось абсцисс частотами, а ось ординат – децибелами.
2. Вот приблизительный алгоритм действий:
   1. Ввести сигнал из файла в массив чисел с плавающей запятой. Загрузить или сгенерировать ядро свертки.
   2. Найти оптимальный размер БПФ, чтобы результат свертки поместился в сигнале-приемнике.
   3. Дополнить нулями оба сигнала до размера БПФ.
   4. Вычислить БПФ от обоих сигналов.
   5. Перемножить спектры сигналов как последовательности комплексных чисел.
   6. Вычислить обратное БПФ от полученного спектра.
   7. Сохранить полученный результирующий сигнал и приблизительно оценить правильность свертки (на простых сигналах и ядрах).
3. Вот приблизительный алгоритм действий:
   1. Загрузить или сгенерировать ядро свертки.
   2. Выбрать размер секции сигнала (одного порядка с размером свертки).

3. Найти оптимальный размер БПФ, чтобы результат каждой свертки поместился в сигнале-приемнике.

4. Вычислить БПФ от ядра свертки.

5. Загрузить очередную секцию исходного сигнала.

6. Вычислить ее БПФ.

7. Перемножить спектры сигнала и ядра свертки.

8. Вычислить обратное БПФ.

9. Наложить со сложением результат этого шага на предыдущий сигнал. Перекрытие составляет длину ядра свертки минус один.

10. Записать в файл очередную часть готового сигнала (учтя, что хвост полученного сигнала еще не готов; на него должен быть наложен следующий сегмент).

11. Если свернуты еще не все сегменты исходного сигнала, то перейти к шагу 5.

4. Вот приблизительный алгоритм действий:

1. Выбрать размер БПФ.

2. Задать требуемую частотную характеристику фильтра на частотах, необходимых для проведения БПФ. Обнулить фазовую характеристику.

3. Вычислить обратное БПФ от частотной характеристики.

4. Из полученного сигнала выбрать отрезок с центром в точке $x[0]$ и длиной, равной проектируемой длине фильтра. Для этого сигнал надо сначала продолжить периодически.

5. Домножить выбранный отрезок сигнала на весовое окно.

6. Фильтр готов. Рекомендуется проверить его работоспособность на тестовом сигнале и провести спектральный анализ ядра фильтра, исходного и полученного после фильтрации сигналов.

Применения цифровой обработки сигналов

Рассмотрим несколько применений цифровой обработки сигналов преимущественно в компьютерной графике и обработке изображений.

Передискретизация

Ресамплинг (*передискретизация, resampling*) – это изменение частоты дискретизации цифрового сигнала. Применительно к цифровым изображениям ресамплинг означает изменение размеров изображения. Существует множество различных алгоритмов ресамплинга изображений. Например, для увеличения изображения в 2 раза можно просто продублировать каждую из его строк и каждый из его столбцов (а для уменьшения – выкинуть). Такой метод называется методом ближайшего соседа (*nearest neighbor*). Можно промежуточные столбцы и строки получить линейной интерполяцией значений соседних столбцов и строк. Такой метод называется билинейной интерполяцией (*bilinear interpolation*). Можно каждую точку нового изображения получить как взвешенную сумму большего числа точек исходного изображения (бикубическая и другие виды интерполяции).

Наиболее качественный ресамплинг получается при использовании алгоритмов, учитывающих необходимость работы не только с временной, но и с частотной областью изображения. Сейчас мы рассмотрим алгоритм ресамплинга, который основан на идее максимального сохранения частотной информации изображения. Алгоритм построен по принципу **интерполяция / фильтрация / прореживание** (*interpolation / filtering / decimation*).

Работу алгоритма будем рассматривать на одномерных сигналах, так как двумерное изображение можно сначала растянуть или сжать по горизонтали (по строкам) а потом – по вертикали (по столбцам). Таким образом, ресамплинг двумерного изображения сводится к ресамплингу одномерного сигнала.

Пусть нам нужно «растянуть» одномерный сигнал от длины n точек до длины m точек, т.е. в $\frac{m}{n}$ раз. Для выполнения этой операции необходимо выполнить 3 шага. Первый шаг – интерполяция нулями, увеличивающая длину сигнала в m раз. Нужно умножить все отсчеты исходного сигнала на m, а потом после каждого отсчета сигнала нужно вставить m-1 нулевое значение. При этом спектр сигнала изменяется следующим образом. Та часть спектра, которая изначально содержалась в цифровом сигнале, остается без изменения (именно этого мы добиваемся). Но выше старой половины частоты дискретизации возникают помехи (отраженные копии спектра), от которых необходимо избавиться с помощью фильтрации.

Второй шаг – это отфильтровывание этих помех с помощью НЧ-фильтра.

Теперь мы получили сигнал, который в m раз длиннее исходного, но сохранил его частотную информацию и не приобрел посторонней частотной информации (ее мы отфильтровали). Если бы нашей задачей было удлинение сигнала в m раз, то на этом шаге можно было бы остановиться. Но наша задача требует теперь уменьшить длину сигнала в n раз. Для этого нужно выполнить 2 шага. Первый шаг – это анти-алиасинговая фильтрация. Так как частота дискретизации уменьшается в n раз, то из спектра сигнала, согласно теореме Котельникова, удастся сохранить только его низкочастотную часть. Все частоты выше половины будущей частоты дискретизации нужно удалить с помощью анти-алиасингового фильтра с частотой среза равной $\frac{1}{n}$ от текущей половины частоты дискретизации. Второй шаг – это прореживание полученного сигнала в n раз. Для этого достаточно выбрать из сигнала каждый n-й отсчет, а остальные – отбросить. Этот алгоритм очень схож с работой АЦП, который тоже сначала отфильтровывает ненужные частоты из сигнала, а потом замеряет значения сигнала через равные промежутки времени, отбрасывая значения в остальные моменты времени.

Заметим, что две НЧ-фильтрации, применяемые в этом алгоритме друг за другом, можно (и нужно) заменить одной. Для этого частоту среза единого НЧ-фильтра нужно выбрать равной минимуму из частот среза двух отдельных НЧ-фильтров.

Еще одно существенное улучшение алгоритма – это поиск общих делителей у чисел m и n. Например, очевидно, что для того, чтобы сигнал из 300 точек сжать до 200 точек, достаточно положить в алгоритме m=2 и n=3.

Заметим, что приведенный алгоритм требует очень большого объема вычислений, т.к. промежуточный размер одномерного сигнала при ресамплинге может быть порядка сотен тысяч. Существует способ существенно повысить быстродействие алгоритма и сократить расход памяти. Этот способ называется *многофазной фильтрацией* (*polyphase filtering*). Он основан на том, что в длинном промежуточном сигнале совсем необязательно вычислять все точки. Ведь большая часть из них все равно будет отброшена при прореживании. Многофазная фильтрация позволяет непосредственно выразить отсчеты результирующего сигнала через отсчеты исходного сигнала и анти-алиасингового фильтра.

Отметим, что здесь мы не рассматриваем такие детали алгоритма, как коррекция границ изображения, выбор фазы сигнала при интерполяции и прореживании и построение хорошего анти-алиасингового фильтра. Отметим только, что для ресамплинга изображений требуется уделить особое внимание как частотной, так и пространственной характеристике фильтра. Если оптимизировать фильтр только в частотной области, то это приведет к большим пульсациям в ядре фильтра. А при ресамплинге изображений пульсации в ядре фильтра приводят к пульсациям яркости вблизи резких перепадов яркости в изображении (***эффект Гиббса****, Gibbs phenomenon*), как в последнем изображении на рис. 12.

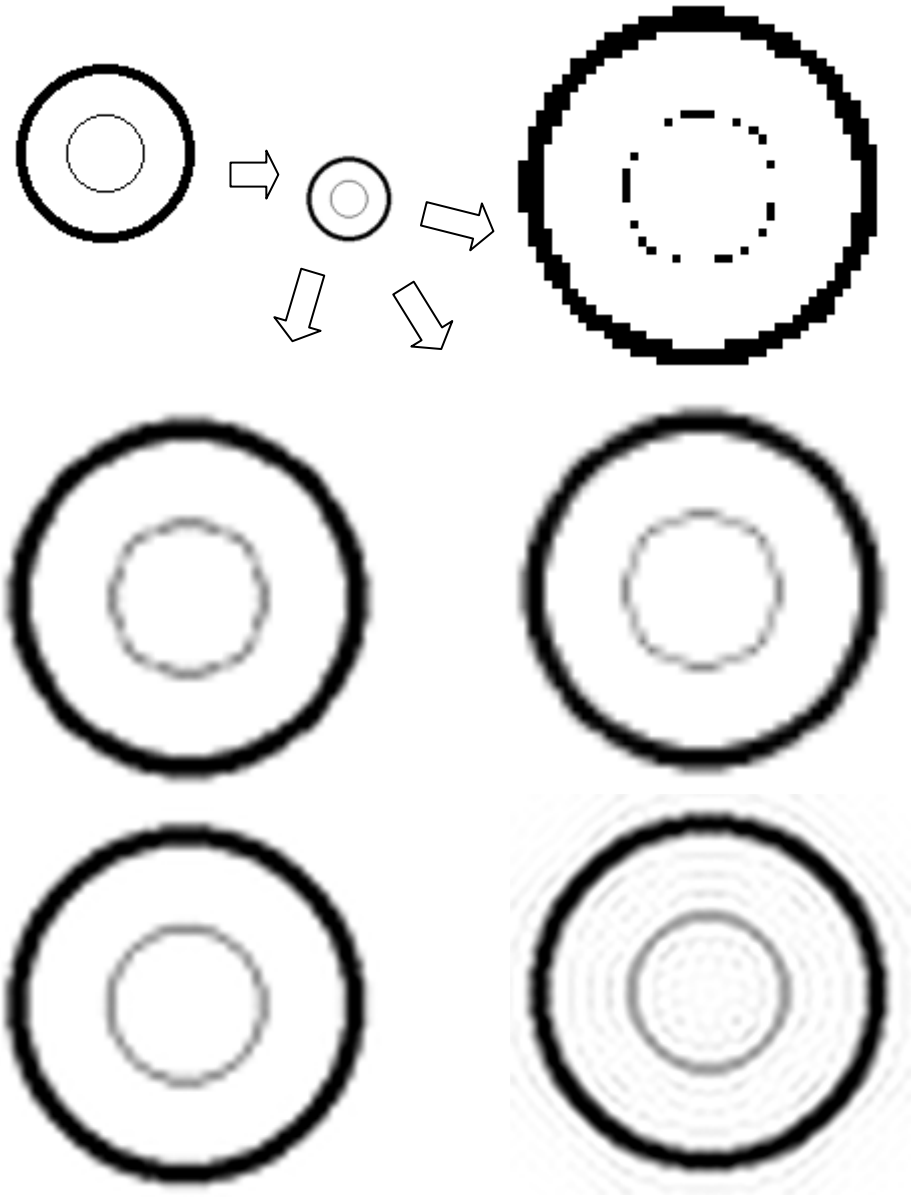


***Рис. 12**. Результат уменьшения и последующего увеличения изображения различными алгоритмами (слева направо, сверху вниз): исходное изображение, уменьшенное изображение (хотя изображение уменьшалось соответствующими алгортимами, здесь приведен только один вариант), метод ближайшего соседа, билинейная интерполяция, бикубическая интерполяция, рассмотренный метод, рассмотренный метод (с фильтром, плохо подходящим для ресамплинга изображений).*

Анти-алиасинг изображений

Избежать алиасинга при генерации изображений – важная задача компьютерной графики. Алиасинг в изображениях приводит к зубчатости краев фигур, муару, плохой читаемости текста и графиков. Одним из основных способов предотвращения алиасинга является так называемый *суперсамплинг* (*supersampling*). Этот прием заключается в генерации изображения с большим разрешением и ресамплингу этого изображения до нужного размера.

Рассмотрим пример. Пусть нам нужно сгенерировать трехмерное изображение шахматной доски с разрешением 200x150 пикселей. Если сделать это непосредственно (например, трассировкой лучей через каждую точку экрана), то результат может быть существенно искажен алиасингом (рис. 13). Применим метод суперсамплинга. Сгенерируем нужное нам изображение с четырехкратным размером 800x600 пикселей, а затем уменьшим его до размера 200x150. Заметим, что качество получаемого таким образом изображения существенно лучше и зависит от качества алгоритма ресамплинга и от степени суперсамплинга (во сколько раз большее изображение мы сгенерировали). Желательно применять алгоритм ресамплинга, обеспечивающий хороший анти-алиасинг.

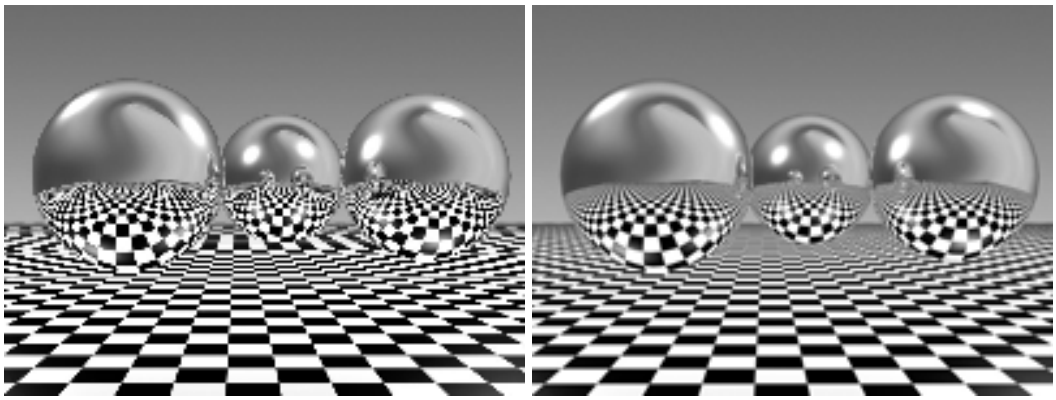


***Рис. 13**. Изображение, сгенерированное без анти-алиасигна и с анти-алиасингом.*

Рассмотренный алгоритм широко применяется в компьютерной графике, несмотря на большие вычислительные затраты. Выбирая степень суперсамплинга, можно варьировать производительность алгоритма. Очевидно, что сложность алгоритма квадратично зависит от степени суперсамплинга. Обычно используются степени суперсамплинга от двух до четырех.

Псевдотонирование изображений

Псевдотонирование (*half-toning*) – это создание иллюзии полноцветности изображения с помощью небольшого реального числа цветов. Пример псевдотонирования – фотографии в газетах, где любые оттенки серого передаются с помощью чередования мелких черных и белых точек.

Мы рассмотрим вариант псевдотонирования для черно-белых изображений. Нашей задачей будет представить изображение с оттенками серого в виде монохромного (двухцветного) изображения.

Пусть мы имеем изображение в оттенках серого, интенсивность точек которого может принимать произвольные значения от 0 до 1. Рассмотрим некоторые алгоритмы приведения такого изображения к монохромному, яркость точек которого может принимать 2 значения: 0 или 1.

Первый самый простой алгоритм – это усечение (порог). Все пиксели с яркостью больше 0.5 получают яркость 1, все остальные – яркость 0. Такой алгоритм обычно дает наихудшие результаты (рис. 15).

Более качественные алгоритмы стремятся так распределить черные и белые пиксели в полученном изображении, чтобы на каждом участке изображения концентрация белых пикселей была пропорциональна яркости этого участка в исходном изображении.

Один из таких методов – *упорядоченное псевдотонирование*. В этом методе исходное изображение разбивается на небольшие блоки одинакового размера (например, 3x3). Затем в каждом блоке находится средняя яркость изображения. В соответствии с этой средней яркостью выбирается количество белых пикселей в соответствующем блоке получаемого монохромного изображения. Обычно эти белые пиксели упорядочиваются в соответствии с некоторым регулярным шаблоном (рис. 15).

Существуют другие алгоритмы достижения нужной концентрации белых пикселей в получаемом монохромном изображении. Например, существует класс алгоритмов, которые достигают этого в 2 стадии. Сначала к изображению добавляется случайный шум необходимой амплитуды, а затем применяется порог. Такие алгоритмы называют *диттерингом* (***dithering***).

Шум представляет собой некий достаточно случайный сигнал, не зависящий от изображения. Например, белый шум – это просто последовательность случайных чисел с математическим ожиданием 0. Спектр такого шума приблизительно равен константе на всех частотах (в пределах половины частоты дискретизации). Последовательные отсчеты такого шума не коррелируют между собой.

Существуют другие виды шума. Например, у розового шума энергия обратно пропорциональна частоте (в определенном рассматриваемом диапазоне частот). Другими словами, амплитуда его гармоник падает на 3 дБ при удвоении частоты. У голубого шума энергия наоборот растет с частотой. Существуют и другие виды шума, однако определения для них могут быть различны в разных областях.

Будем называть ошибкой квантования изображение, равное разности исходного и псевдотонированного изображений.

При псевдотонировании изображений стремятся добиться того, чтобы спектр изображения-ошибки по возможности не содержал низкочастотных и среднечастотных компонент. В этом случае ошибка будет менее заметна человеческому глазу. Например, при диттеринге розовым шумом спектр ошибки тоже близок к светло-розовому, и результирующее изображение выглядит значительно искаженным (рис. 15). При диттеринге белым шумом спектр ошибки белый. Поэтому результирующее изображение выглядит лучше. При диттеринге с диффузией ошибки спектр ошибки получается близок к голубому шуму, т.е. содержит мало низкочастотных компонент. В результате получается приятное глазу изображение.

Нетрудно видеть, что просто диттеринг голубым шумом не приводит к желаемому результату, т.к. ошибка квантования при этом имеет спектр, содержащий значительное количество низкочастотных и среднечастотных компонент. Для избавления от них нужно применить рекурсивный фильтр. Этот метод псевдотонирования называется ***диффузией ошибки*** (*error diffusion*). Его идея в том, что ошибка квантования, возникшая при квантовании данного пикселя, распространяется с обратным знаком на соседние пиксели и таким образом как бы компенсируется.

Рис. 14. *Исходное изображение для тестирования алгоритмов псевдотонирования.*

Рис. 15. *Различные методы прсевдотонирования изображений (слева направо, сверху вниз): порог, упорядоченное псевдотонирование, диттеринг белым шумом, диттеринг светло-розовым шумом, диттеринг голубым шумом, диттеринг с диффузией ошибки.*

Выравнивание освещенности изображений

Часто некоторые участки на изображении бывают слишком темными, чтобы на них можно было что-то разглядеть. Если прибавить яркости ко всему изображению, то изначально светлые участки могут оказаться совсем засвеченными. Чтобы улучшить вид изображения в таких случаях, применяется метод выравнивания освещенности.

Этот метод не является линейным, т.е. не реализуется линейной системой. Действительно, рассмотрим модель типичную освещенности для фотографии. Фотографируемый пейзаж обычно освещен по-разному в разных точках. Причем обычно освещенность меняется в пространстве достаточно медленно. Мы хотим, чтобы все детали на фотографии были освещены более однородно, но при этом оставались достаточно контрастными друг относительно друга. А на реальной фотографии получается произведение той картинки, которую мы хотим видеть и карты освещенности. Там где освещенность близка к нулю, все предметы и детали тоже близки к нулю, то есть практически невидимы.

Поскольку освещенность меняется в пространстве достаточно медленно, то можно считать ее низкочастотным сигналом. Само же изображение можно считать в среднем более высокочастотным сигналом. Если бы в процессе фотографии эти сигналы складывались, то их можно было бы разделить с помощью обычного фильтра. Например, применив ВЧ-фильтр, мы бы «избавились от перепадов освещенности» (НЧ-сигнала), а оставили «само изображение». Но поскольку эти сигналы не складываются, а перемножаются, то избавиться от неравномерностей освещенности простой фильтрацией не удастся.

Для решения таких задач применяется *гомоморфная обработка*. Основной метод гомоморфной обработки заключается в сведении нелинейной задачи к линейной с помощью каких-либо преобразований. Например, в нашем случае можно свести задачу разделения перемноженных сигналов к задаче разделения сложенных сигналов. Для этого нужно взять логарифм от произведения изображений. Логарифм от произведения равен сумме логарифмов сомножителей. Если учесть, что логарифм от НЧ-сигнала остается НЧ-сигналом, а логарифм от ВЧ-сигнала остается ВЧ-сигналом, то мы свели задачу разделения произведения сигналов к задаче разделения суммы НЧ- и ВЧ-сигналов. Очевидно, эту задачу можно решить с помощью ВЧ-фильтра, который удалит из суммы сигналов низкие частоты. После этого останется только взять от полученного сигнала экспоненту, чтобы вернуть его к исходному масштабу амплитуд.

ВЧ-фильтр можно реализовать следующим образом. Сначала к изображению применяется операция размытия (НЧ-фильтр), а потом из исходного изображения вычитается размытое. Наилучший радиус размытия зависит от конкретного изображения. Можно начать эксперименты с радиуса порядка десяти пикселей.

Обычно для размытия изображения применяется двумерный гауссовский фильтр, имеющий вид $h[x,y] = A \cdot e^{-\frac{x^2+y^2}{2\sigma^2}}$. Здесь A – нормирующая константа (выбирается так, чтобы сумма всех коэффициентов фильтра была равна единице), σ – «ширина» фильтра, регулирующая степень размытия.

Непосредственное вычисление двумерной свертки с таким ядром потребует огромных вычислений даже при сравнительно небольшом размере ядра. Однако приведенное гауссово ядро обладает свойством ***сеперабельности***. Это означает, что эквивалентного эффекта можно достичь, отфильтровав сначала все строки изображения одномерным гауссианом, а затем отфильтровав все столбцы полученного изображения таким же одномерным гауссианом.

Полученный от выравнивания освещенности эффект может оказаться слишком сильным (темные области станут по яркости такими же, как и светлые). Чтобы уменьшить эффект, можно просто смешать обработанное изображение с исходным в определенной пропорции.

Рис. 16. *Гомоморфная обработка изображений. Выравнивание освещенности.*

Другие применения

Улучшение изображений и художественные эффекты

Для улучшения изображений и создания различных художественных эффектов часто применяется фильтрация. Например, для придания изображению резкости можно воспользоваться фильтром, который усиливает сигнал на высоких частотах. Существуют фильтры для выделения или нахождения границ в изображении, размытия, направленного смазывания изображений, создания различных эффектов, таких как акварель, тиснение.

Поиск фрагментов в изображениях

Для поиска фрагментов в изображениях применяется двумерная корреляция. Сигналом для поиска является изображение, а искомым сигналом – искомый фрагмент изображения. Эффективное вычисление корреляции стало возможным благодаря двумерному БПФ.

Компрессия изображений

Методы цифровой обработки сигналов позволяют достаточно эффективно сжимать изображения в частотной области. Например, алгоритм JPEG действует следующим образом (упрощенно). Изображение разбивается на фрагменты размером 8x8 пикселей, и каждый фрагмент переводится в частотную область. После этого в каждом фрагменте те высокочастотные составляющие, амплитуда которых мала, выкидываются, а все остальные – кодируются. Ясно, что для тех областей изображения, где яркость изменяется не очень быстро (а таких большинство), высокочастотных компонент почти нет. Таким образом удается выкинуть из спектра существенную часть не очень важной информации. В JPG-файле кодируются оставшиеся «существенные» амплитуды.

В алгоритме JPEG применяется модификация ДПФ: дискретное косинусное преобразование (ДКП). ДКП от двумерного сигнала можно вычислить, отразив четным образом сигнал относительно нулевой точки и вычислив двумерное ДПФ полученного сигнала с двукратными размерами. В полученном спектре будут содержаться только «косинусные» коэффициенты.

Восстановление изображений

При съемке движущегося объекта неподвижной камерой полученное изображение получается смазанным. Если знать параметры движения объекта, то можно построить ядро свертки, которое камера «применила» к снимаемому сигналу. Затем с помощью метода деконволюции можно в значительной степени устранить эффект размытия.

Иногда при съемке камера может вносить в изображение интерференцию – периодический муар, накладываемый на изображение. Часто оказывается, что спектр этой интерференции состоит из одной – двух гармоник. В этом случае ее можно эффективно удалить с помощью фильтра, который подавляет заданные частоты (*notch filter*).

Вейвлеты и банки фильтров

Какова цель разложения сигналов в ряд Фурье? Преобразование сигнала в его Фурье-спектр – это переход в другую модель представления информации, закодированной в сигнале. Часто эта другая модель оказывается более простой для понимания природы сигнала. Например, многие звуковые сигналы состоят из сумм колебаний, близких к синусоидальным. Поэтому после преобразования Фурье они выглядят, как несколько пиков в спектре, хотя их форма волны до преобразования Фурье могла быть очень сложной для анализа.

Более того, сама природа нашего слуха такова, что ухо раскладывает поступающие в него звуки на отдельные частоты. Звуковая волна, поступающая в ухо, преобразуется в колебания базилярной мембраны. Базилярная мембрана имеет разную жесткость по своей длине, и соответственно – разные собственные частоты колебаний разных участков. Когда на мембрану поступает сложное колебание, оно возбуждает колебания тех участков мембраны, которым соответствуют отдельные гармонические составляющие сложного входного колебания. Таким образом, разложение звука на синусоиды (преобразование Фурье) близко по своей природе к механизму функционирования нашего уха.

Рассмотрим, некоторые «недостатки» и «несоответствия» преобразования Фурье.

1. Преобразование Фурье раскладывает сигнал по кратным частотам. Если мы раскладываем звук, то базисные частоты получаются, например, такие: 10 Гц, 20 Гц, …, 300 Гц, 310 Гц, …, 4000 Гц, 4010 Гц, … Это не очень хорошо согласуется с нашим восприятием высоты звука. Наше ухо чувствительно не к абсолютным изменениям высоты (на сколько-то герц), а к относительным (на сколько-то процентов). Поэтому при анализе звука с помощью преобразования Фурье часто оказывается, что частотное разрешение спектра недостаточно на низких частотах и избыточно на высоких. Казалось бы, что в этом случае можно повысить частотное разрешение спектра (увеличив размер FFT), но при этом мы будем анализировать более длинный по времени отрезок сигнала, и полученный спектр будет отражать усредненные свойства исходного сигнала в течение всего отрезка FFT. А такое усреднение по времени не всегда желательно.

2. Базисные функции преобразование Фурье имеют одну и ту же протяженность, как для высоких частот, так и для низких. Это не всегда удобно, например, при сжатии изображений. Вспомним, как работает алгоритм JPEG. Он разбивает изображение на блоки 8x8 пикселей и выполняет а каждом блоке преобразование Фурье (точнее, его разновидность, ДКП). После этого некоторые полученные амплитуды обнуляются (откидываются при кодировании). Чаще всего отбрасываются верхние частоты, т.к. они обычно содержат меньше энергии (их амплитуды меньше). При восстановлении производится обратный процесс: обратное преобразование Фурье. При этом обнуленные амплитуды высоких частот вызывают эффект Гиббса: пульсации яркости декодированного изображения вблизи резких границ в изображении. Визуально такой эффект очень нежелателен. Хотелось бы сделать так, чтобы эти пульсации не расползались на весь блок 8x8, а были локализованы в пространстве. Другими словами, хотелось бы, чтобы базисные функ-

ции, соответствующие высоким частотам в преобразовании Фурье, были короче (лучшая пространственная локализация), чем для низких частот.

Чтобы преодолеть эти недостатки, можно пользоваться, например, таким приемом. Для анализа высоких частот использовать FFT с более коротким окном (лучшая локализация в пространстве), а для анализа низких частот – с более длинным окном (лучшее разрешение по частоте). Для анализа это хорошо, но вот для обработки, сжатия и синтеза – не очень. Для решения таких задач существует вейвлетное преобразование, к рассмотрению которого мы и переходим.

Сначала рассмотрим простой пример – частный случай вейвлетного преобразования. Пусть у нас имеется одномерный сигнал четной длины x[n]. Выполним следующее преобразование. Свернем сигнал x[n] с сигналом $h_1[n]$, где сигнал $h_1[n]$ состоит из двух единичных отсчетов ($h_1[0] = 1$, $h_1[1] = 1$). Эта операция эквивалентна вычислению сигнала $y_1[n]$, который состоит из сумм соседних элементов x[n]. Так если x[n] = {2, 2, 3, 4, 6, 1, 6, 6}, то получается $y_1[n]$ = {2, 4, 5, 7, 10, 7, 7, 12, 6} (не забываем, что свертка может расширять сигнал с концов). Здесь индексы $y_1[n]$ идут от 0 до 8 (см. уравнение свертки). Сигнал $y_1[n]$ – это некоторая сглаженная копия сигнала x[n], т.к. фильтр $h_1[n]$ можно рассматривать как примитивный НЧ-фильтр.

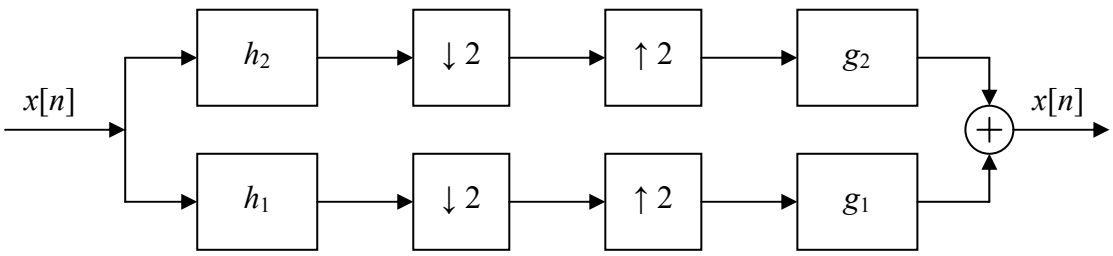
Теперь рассмотрим сигнал $y_2[n]$, получающийся сверткой x[n] с ядром $h_2[n]$, где $h_2[0] = 1$, $h_2[1] = -1$. Очевидно, такой «разностный» сигнал будет равен $y_2[n]$ = {2, 0, 1, 1, 2, -5, 5, 0, -6}. Легко видеть, что исходный сигнал x[n] можно вычислить как полусумму сигналов $y_1[n]$ и $y_2[n]$. Действительно, это следует из свойств свертки и из выбора функций $h_1[n]$ и $h_2[n]$:

$$y_1[n] + y_2[n] = x[n] * h_1[n] + x[n] * h_2[n] = x[n] * \left(h_1[n] + h_2[n]\right) = x[n] * \left(2\delta_0[n]\right) = 2 \cdot x[n]$$

Таким образом, можно считать сигнал $y_1[n]$ неким грубым низкочастотным приближением x[n], а сигнал $y_2[n]$ – «уточняющим» сигналом, содержащим те детали, которые были выброшены их x[n] при НЧ-фильтрации.

Теперь совершим прореживание сигналов $y_1[n]$ и $y_2[n]$ в 2 раза. Получим сигналы $z_1[n]$ = {4, 7, 7, 12}, $z_2[n]$ = {0, 1, -5, 0}. Оказывается, что исходный сигнал x[n] можно восстановить не только из $y_1[n]$ и $y_2[n]$, но и из их прореженных вариантов $z_1[n]$ и $z_2[n]$. Для этого нужно совершить обратные операции. Сначала проводим интерполяцию нулями сигналов $z_1[n]$ и $z_2[n]$. Получаем $u_1[n]$ = {0, 4, 0, 7, 0, 7, 0, 12}, $u_2[n]$ = {0, 0, 0, 1, 0, -5, 0, 0}. После этого сворачиваем полученные сигналы с фильтрами $g_1[n] = h_1[-n]$ и $g_2[n] = h_2[-n]$ соответственно и суммируем результирующие сигналы. Получаем {4, 4, 7, 7, 7, 7, 12, 12} + {0, 0, -1, 1, 5, -5, 0, 0} = 2·{2, 2, 3, 4, 6, 1, 6, 6}, т.е. в точности исходный сигнал x[n].

Подытожим схему этого преобразования (см. рис. 17).



В чем польза такого разложения сигнала x[n] на $z_1[n]$ и $z_2[n]$?

(to be continued)

Алфавитный указатель