ГОСУДАРСТВЕННЫЙ КОМИТЕТ СВЯЗИ, ИНФОРМАТИЗАЦИИ И ТЕЛЕКОММУНИКАЦИОННЫХ ТЕХНОЛОГИЙ РЕСПУБЛИКИ УЗБЕКИСТАН

ТАШКЕНТСКИЙ УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

На правах рукописи

УДК__

МИРЗАЕВ ХАБИБУЛЛА ХАМИДУЛЛАЕВИЧ

Разработка алгоритмов и программ согласования частоты дискретизации для различных носителей звуковой информации

5А330202 – Информационные и мультимедийные технологии

Диссертация
на соискание академической степени
магистра

Научный руководитель:
д.т.н. проф. Мусаев М.М.

Ташкент – 2013

Содержание

ВВЕДЕНИЕ

Развитие информационно-коммуникационных технологий (ИКТ) является одним из основных факторов благосостояния и экономического роста страны. Сегодня ИКТ становится одним из основных приоритетов государственной политике Узбекистана.

В постановлении президента Республики Узбекистана «о мерах по дальнейшему внедрения и развитию информационно - коммуникационных технологии» (собрание законодательства Республики Узбекистана, 2012 г., № 13, ст. 139) одной из основных задач дальнейшего внедрения и развитие информационно – коммуникационных технологий, в частности, является программа программа мер по коренному и качественному улучшению функционирование национальной информационной-поисковой системы, увеличению количество ее ползователей.

Алгоритмы цифровой обработки сигналов, составляющие основу прикладных программ сигнальных процессоров, повсеместно применяются в компьютерных сетях и телекоммуникационных системах, используются в мультимедийных технологиях, цифровой аппаратуре аудио и видеообработки, применяются в медицине, в технологических процессах, в системах контроля и управления объектами или процессами.

Растущая потребность в современных цифровых системах обработки данных с более чем одной частотой дискретизации привела к развитию новой подобласти цифровой обработки сигналов, известной как обработка данных при нескольких скоростях (multirate processing). При такой обработке данных используются две основные операции: децимация и интерполяция, позволяющие эффективно чередовать скорости передачи данных. Децимация уменьшает частоту дискретизации, эффективно сжимая данные и оставляя только необходимую информацию. Интерполяция, наоборот, увеличивает частоту дискретизации [13].

Процедура преобразования частоты дискретизации представляет собой изменение частоты дискретизации сигнала после того, как сигнал был оцифрован. Такое преобразование частоты дискретизации находит множество применений, оно используется для минимизации объема вычислений путем уменьшения потока данных, когда полоса частот, занимаемая сигналом, уменьшается вследствие низкочастотной фильтрации. При обработке медицинских и спутниковых изображений преобразование частоты дискретизации необходимо для улучшения изображений, изменения масштаба, вращения изображений. Преобразование частоты дискретизации используется также для уменьшения сложности некоторых узкополосных цифровых фильтров[9].

Актуальность данной работы. Преобразование частоты дискретизации неизбежно при обработке сигналов в реальном масштабе времени, когда два процессора, работающие на разных частотах, должны обмениваться цифровыми сигналами. Изменения частоты дискретизации необходимо и при обработке цифровой аудио сигнала, когда надо согласовать частоту дискретизации со стандарта Audio CD 44,1 кГц на стандарт DVD 44,0 кГц.

Объект и предмет исследования. Объектом исследования являются носители звуковой информации, компьютеры средства обработки. Предметом исследования являются аудиосигналы мультимедийных систем.

Цель и задачи исследования

Целью работы является разработка алгоритмов и программ изменения частоты дискретизации аудиосигналов, с использованием алгебраических моделей аппроксимации.

Для достижения поставленной цели в диссертационной работе решаются следующие задачи:

- исследование существующих методов цифровой обработки сигналов и способов изменения частоты дискретизации;

- разработка алгоритмов вычисления коэффициентов алгебраических полиномов с переводом цифрового сигнала в спектр.
- разработка алгоритма изменения частоты дискретизации аудиосигналов с использованием полиномиального представления.
- создание прикладных программ изменения частоты дискретизации аудиосигналов для применения в сигнальных процессоров.

Методы исследования. Теоретическую основу проведенных исследований составляют методы линейной алгебры спектрального анализа в Фурье-базисах, численные методы аппроксимации.

Научная новизна проведенных исследований заключается в том, что:

- в результате исследования разработан алгоритм перевода сигнала в область полиномиального представления и нахождения коэффициентов полинома в базисной системе Адамара;
- разработаны скоростные алгоритмы и прикладные программные средства для изменения частоты дискретизации аудиосигналов, в реальном масштабе времени.

Практическая ценность. Разработанные прикладные программы изменения частоты дискретизации могут быть широко использованы в современной радиоаппаратуре, звуковых системах компьютеров, телекоммуникационных системах.

Структура и объём магистерской диссертационной работы

Магистерская диссертация состоит из введения, трёх глав и заключения, изложена на ___ страницах машинописного текста, содержит ___ таблицы и ___ рисунков.

Глава I. ИЗУЧЕНИЕ СОВРЕМЕННЫХ МЕТОДОВ ЦИФРОВОЙ ОБРАБОТКИ СИГНАЛОВ

1. Методы изменения частоты дискретизации

Простой, но слишком наивный подход к изменению частоты дискретизации цифрового сигнала — это вернуть его к аналоговому виду, а затем повторно оцифровать его с иной частотой[13].

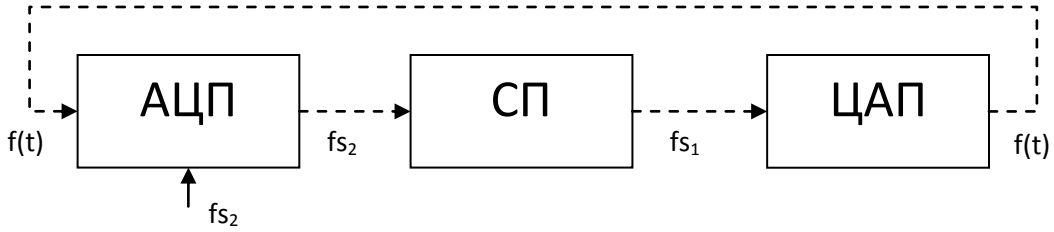


Рис. 1.1. Процесс изменения частоты дискретизации классической методом.

В этом методе дискретный сигнал с частотой дискретизацией fs_1 передаётся в блок ЦАП (цифро-аналоговый преобразователь), и в этом блоке дискретный сигнал переводится в аналоговую форму f(t). После этого аналоговый сигнал f(t) передается в блок АЦП и там сигнал дискретизируется с нужной частотой дискретизацией fs_2.

При таком подходе изменения частоты дискретизации, теряется много времени за счет того что сигнал передается с блоков ЦАП и АЦП. Процессу преобразования цифровой-аналоговый-цифровой присущи такие ошибки, как ошибки квантования и наложения, которые значительно искажают сигнал[13].

Существует и другие методы изменения частоты дискретизации, которые позволяют обрабатывать сигнал в цифровом виде.

При передискретизации отсчёты сигнала, соответствующие одной частоте дискретизации, вычисляются по имеющимся отсчётам этого же сигнала, соответствующим другой частоте дискретизации (при этом предполагается, что обе частоты дискретизации соответствуют условиям

теоремы Котельникова). Идеальная передискретизация эквивалентна восстановлению непрерывного сигнала по его отсчётам с последующей дискретизацией его на новой частоте[9, 13, 17,].

Точное вычисление значения исходного непрерывного сигнала в определённой точке производится следующим образом:

$$x(t) = \sum_i x(t_i) \frac{\sin \frac{\omega_d}{2}(t - t_i)}{\frac{\omega_d}{2}(t - t_i)},$$

где $x(t_i)$ — *i*-й отсчёт сигнала, t_i — момент времени, соответствующий этому отсчёту, $\omega_d = 2\pi f_d$ — циклическая частота дискретизации, $x(t)$ — интерполированное значение сигнала в момент времени t.

Функция $\frac{\sin \frac{\omega_d}{2}(t - t_i)}{\frac{\omega_d}{2}(t - t_i)}$ не является финитной, поэтому для вычисления значения сигнала в определённый момент времени с помощью вышеприведённого выражения необходимо обработать бесконечное число его отсчётов (как в прошлом, так и в будущем), что нереализуемо на практике. В реальной жизни интерполяция осуществляется с помощью других фильтров, при этом выражение для неё принимает следующий вид:

$$x(t) = \sum_i x(t_i) h(t - t_i),$$

где *h(t)* — импульсная характеристика соответствующего восстанавливающего фильтра. Вид этого фильтра выбирается в зависимости от задачи[7].

Прямое вычисление новых отсчётов сигнала по вышеприведённым формулам требует значительных вычислительных ресурсов и нежелательно для приложений реального времени. Существуют важные частные случаи передискретизации, для которых вычисление новых отсчётов производится проще:

– децимация с целым коэффициентом (уменьшение частоты дискретизации в целое число раз);
– интерполяция с целым коэффициентом (увеличение частоты дискретизации в целое число раз);
– изменение частоты дискретизации в рациональное (M/N) число раз (этот случай можно рассматривать как комбинацию двух предыдущих)[7, 16].

Децимация с целым коэффициентом.

На рис. 1.2. представлена блок-схема процесса децимации сигнала *x(n)* с целым шагом *M.* На ней изображены цифровой фильтр защиты от наложения спектров *h(k)* и схема сжатия (компрессор) частоты дискретизации, обозначенная стрелкой, направленной вниз, и коэффициентом децимации *M.* Компрессор частоты дискретизации снижает частоту дискретизации с *Fs* до *Fs/M.* Для предотвращения наложения при более низкой частоте выборки используется цифровой фильтр, предварительно ограничивающий полосу входного сигнала до *FS/2M.* Таким образом, сигнал *x(n)* вначале ограничивается по полосе. Снижение частоты дискретизации достигается за счет того, что из каждых *M* выборок фильтруемого сигнала *w(n)* отбрасывается *M-1* выборка. Вход и выход процесса децимации связаны следующим соотношением:

$$y(m) = w(mM) = \sum_{k=-\infty}^{\infty} h(k)x(mM - k),$$

где

$$w(n) = \sum_{k=-\infty}^{\infty} h(k)x(n - k).$$

На рис. 1.3 описанный процесс иллюстрируется для простого случая *M*=3, где из каждых трех выборок *x(n)* отбрасываются две. Отметим, что, по сути, децимация — это операция сжатия данных[9,12]

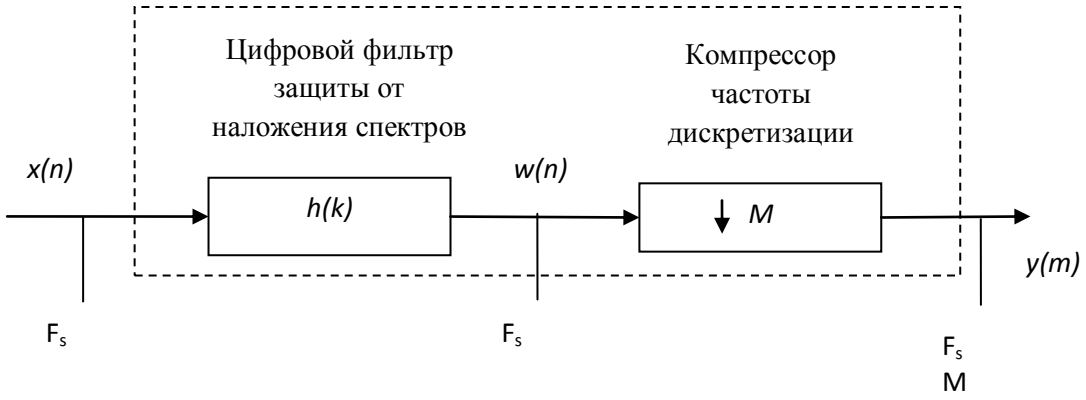


Рис. 1.2. Блок схема децимации с шагом *M*

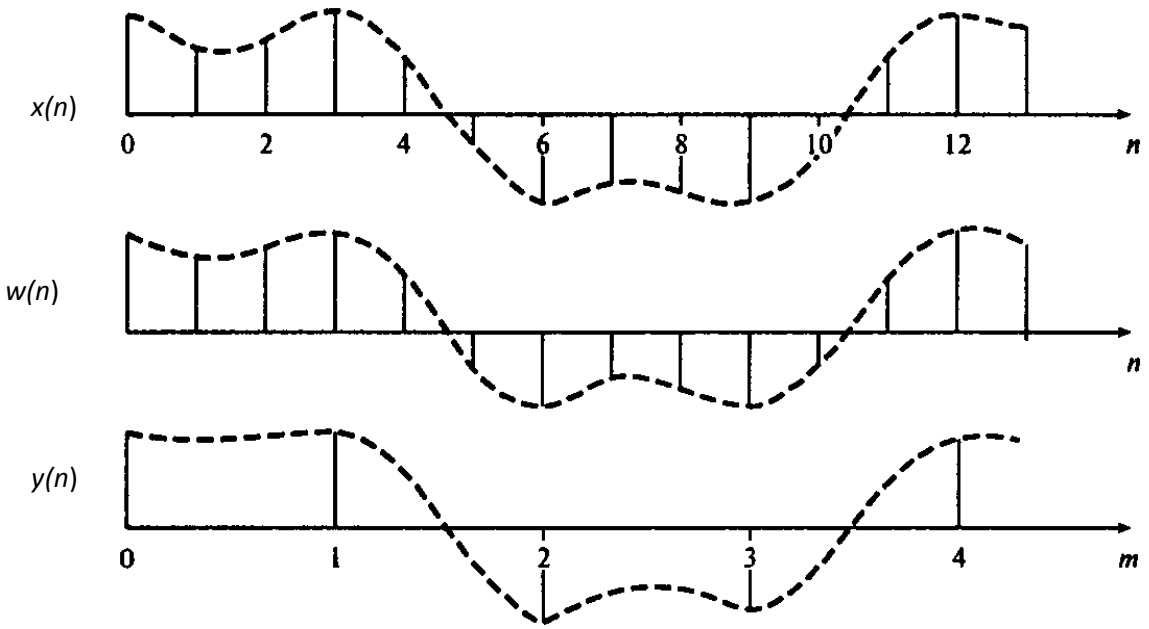


Рис. 1.3. Иллюстрация децимации с шагом *M=3* во временной области.

Спектральное представление процесса децимации приведено на рис. 1.4. где предполагается, что на вход подается широкополосный сигнал *x(n).* На рис. 1.4. *б* пунктиром изображены зеркальные компоненты сигнала, которые привели бы к наложению, если бы входной сигнал *x(n)* не был ограничен по полосе перед децимацией[9, 12, 13].

Интерполяция с целым коэффициентом

По многим пунктам интерполяция — это цифровой эквивалент процесса цифроаналогового преобразования, когда из цифровых выборок, поданных на цифроаналоговый преобразователь, с помощью

интерполяции восстанавливается аналоговый сигнал. При цифровой интерполяции, впрочем, процесс порождает специфические значения[13].

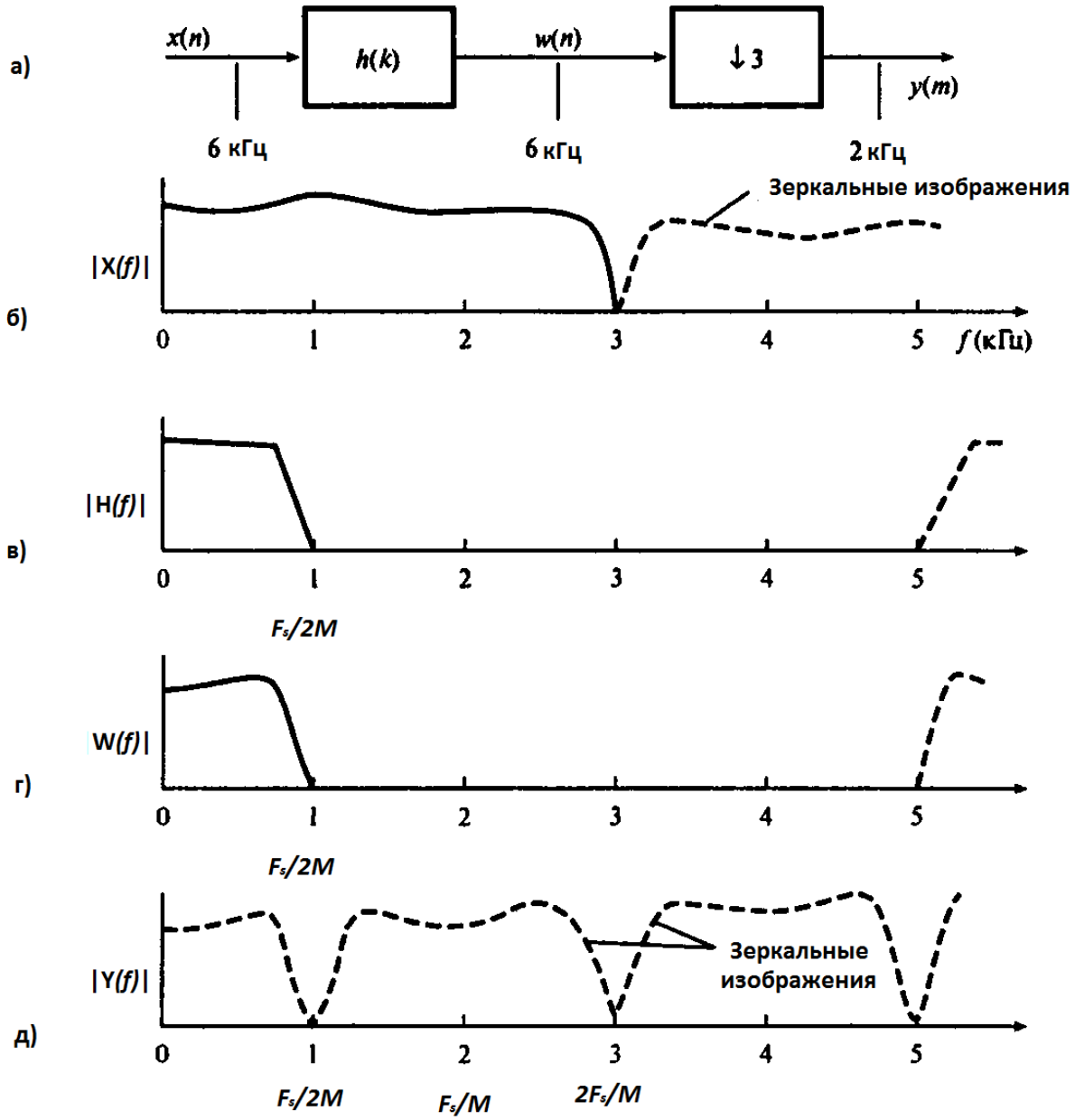


Рис. 1.4. Спектральная интерпретация децимации сигнала с частоты 6 до частоты 2 кГц.

Для данного сигнала *x(n)* с частотой дискретизации *Fs* процесс интерполяции увеличивает частоту дискретизации в *L* раз, т.е. до LF_s. Схема интерполятора приведена на рис. 1.5. Он состоит из экспандера частоты дискретизации, обозначенного стрелкой, направленной вверх, и

коэффициентом интерполяции *L*, который показывает, во сколько раз увеличивается частота дискретизации[9, 12].

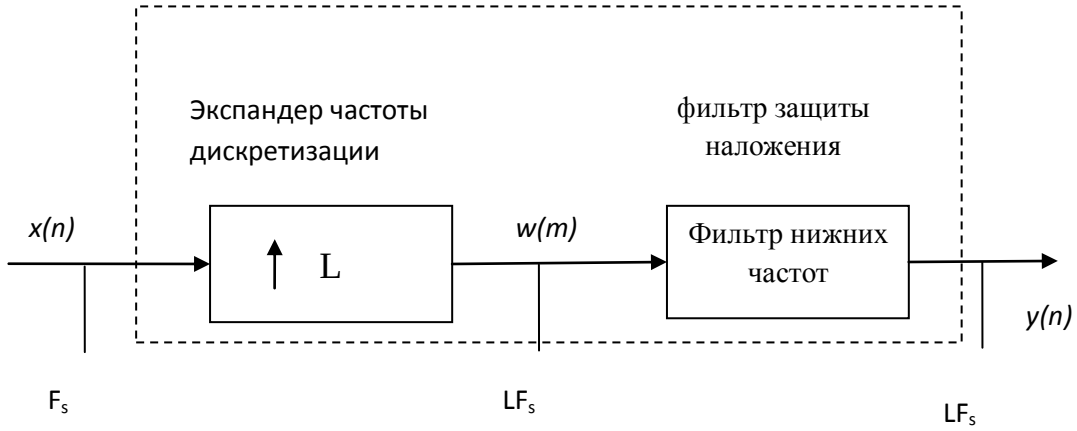


Рис. 1.5. Блок схема интерполяции с шагом *L=3*

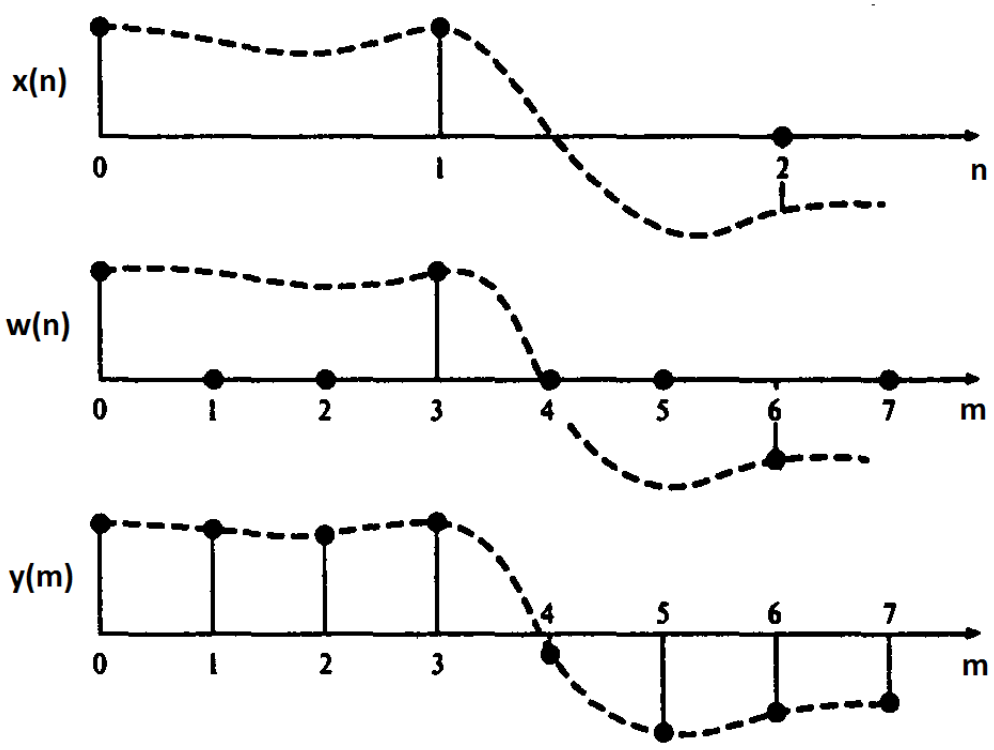


Рис. 1.6. Иллюстрация интерполяции с шагом L=*3* во временной области.

Для каждой выборки сигнала *x(n)* экспандер вводит *(L—1)* нулевую выборку, в результате чего формируется новый сигнал *w(m)* с частотой

дискретизации LF_S. Далее этот сигнал пропускается через фильтр нижних частот для удаления зеркальных частот, введенных при увеличении частоты, и получается сигнал *y(m)*.

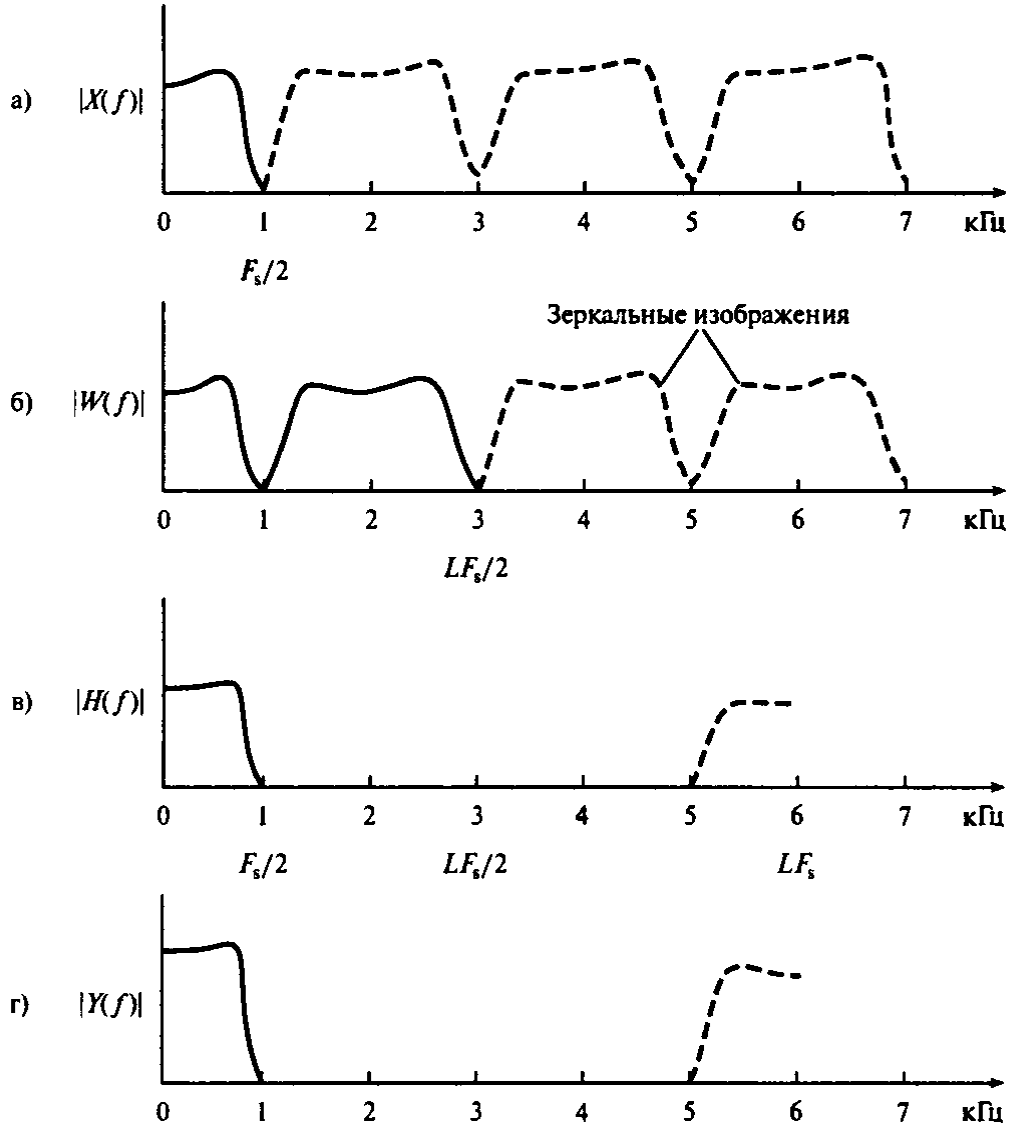


Рис. 1.7. Спектральная интерпретация интерполяции сигнала с 2 до 6 кГц

Введение (*L*-1) нулей приводит к распространению энергии каждой выборки сигнала по *L* выходным выборкам, т.е. каждая выборка ослабляется в *L* раз. Для компенсации этого эффекта можно, например, умножить каждую выборку *y(n)* на *L*. Процесс интерполяции характеризуется следующей связью входа и выхода

$$y(m) = \sum_{k=-\infty}^{\infty} h(k)w(m-k),$$

где

$$w(m) = \begin{cases} x(m/L), & m = 0,\pm L,\pm 2L,.... \\ 0 \end{cases}$$

Процесс интерполяции во временной области иллюстрируется на рис.1.6. для простого случая L = 3. Обратите внимание на то, что каждая выборка *x(n)* порождает три выходные выборки (две нулевые выборки вводит экспандер)[9, 13].

Интерпретация данного процесса в частотной области приведена на рис.1.7. Функции *X(f), W(f) и Y(f)*) представляют собой частотные характеристики сигналов *x(n), w(m)* и *y(m)* соответственно. *H(f)* — это амплитудная характеристика фильтра подавления зеркальных частот. Данный фильтр необходим для удаления зеркальных компонентов, обозначенных в *W(f)* пунктиром. На данном этапе стоит отметить, что процессы децимации и интерполяции дуальны, т.е. взаимно обратны. Данное свойство дуальности означает, что интерполятор можно легко получить из эквивалентного дециматора наоборот.

Преобразование частоты дискретизации с нецелым шагом

В некоторых ситуациях часто бывает нужно изменить частоту дискретизации в нецелое число раз. Пример — цифровое аудио, где может требоваться передача данных с одного запоминающего устройства на другое, причем системы поддерживают разные частоты дискретизации. В частности, это передача данных с системы воспроизведения компакт-дисков (44,1 кГц) на цифровую аудиопленку (digital audio tape — DAT) (48 кГц). Для выполнения требуемого преобразования частоту дискретизации компакт-диска следует увеличить в 48/44,1 раза[13].

На практике подобные нецелые множители представляются рациональным числом, т.е. отношением двух целых чисел, скажем L и M,

таких что L/M максимально близко к желаемому множителю. Необходимое же преобразование частоты дискретизации производится в два этапа: 1) интерполяция данных с шагом L и 2) децимация с шагом M (рис. 1.8, а). Необходимо, чтобы процесс интерполяции предшествовал децимации, поскольку в противном случае при децимации исчезнут некоторые необходимые частотные компоненты. В приведенном выше примере “с компакт-диска на цифровую аудиопленку” преобразования частоты в 48/44,1 раза можно достичь следующим образом: интерполировать с шагом L — 160, а затем провести децимацию с шагом M — 147, т.е. вначале скорость данных компакт-диска увеличивается в L = 160 раз до 7056 кГц, а затем уменьшается в M = 147 раз до 48 кГц.

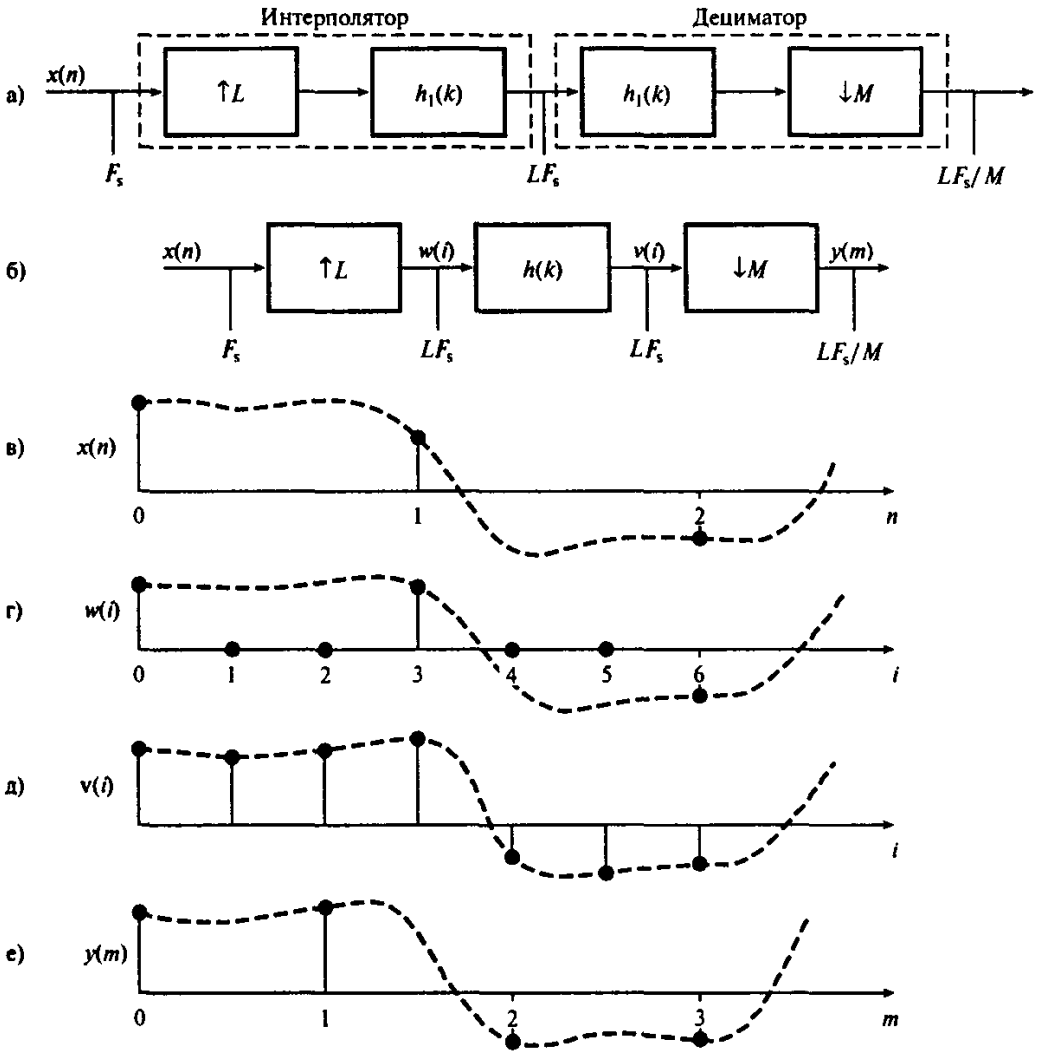


Рис. 1.8. Иллюстрация интерполяции с рациональным шагом *(*L = 3, M = 2)

Поскольку два фильтра нижних частот на рис. 1.8, а h1(k) и h2(k) соединены каскадно и имеют общую частоту дискретизации, их можно объединить, и тогда получится один обобщенный конвертер частоты дискретизации, изображенный на рис. 1.8, б. Если $M > L$, операция, производимая конвертером, называется децимацией с нецелым шагом, а если $M < L$ — интерполяцией. Если $M — 1$, действие обобщенной системы сводится к простой интерполяции с целым шагом, описанной ранее, а при $L = 1$ оно сводится к децимации с целым шагом[9, 13].

На рис. 1.8, в иллюстрируется интерполяция с шагом 3/2. Частота дискретизации вначале увеличивается в 3 раза (к каждой выборке *x(n)* добавляются две нулевые выборки), а затем сигнал пропускается через фильтр нижних частот, результат — *v(i).*

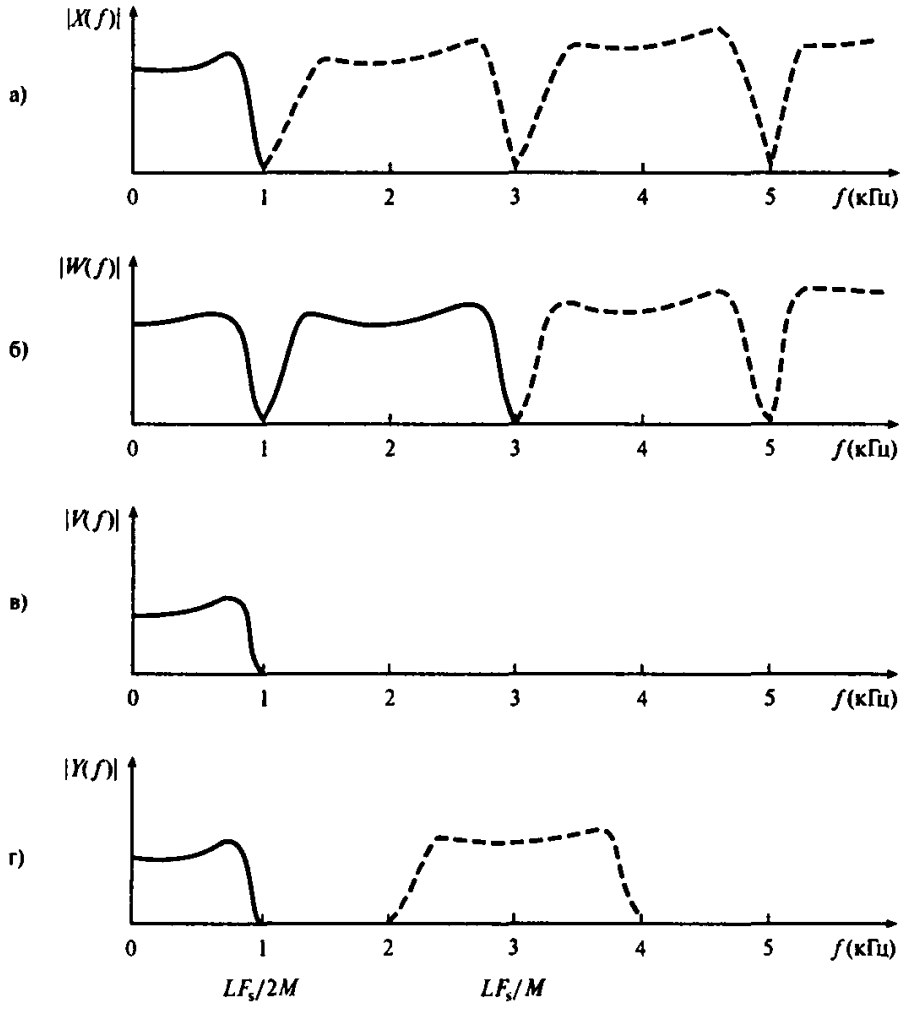


Рис. 1.9. Спектральная интерпретация увеличения частоты дискретизации сигнала с 2 кГц в 3/2 раз

Далее фильтрованные данные сокращаются в два раза, т.е. из каждых двух выборок *v(i)* остается одна. Иллюстрацией данного процесса в частотной области служит рис. 1.9. Входной сигнал *x(n)* с частотой дискретизации 2 кГц вначале ускоряется в 3 раза до 6 кГц, фильтруется с целью устранения зеркальных частот, которые иначе вызвали бы наложение, а затем замедляется в 2 раза до частоты 3 кГц[16].

Недостатком данного метода является необходимость фильтрации сигнала на повышенной в M раз частоте дискретизации, что требует значительных вычислительных ресурсов. При этом соответствующая частота может во много раз превосходить как исходную, так и окончательную частоту передискретизации, особенно если M и N — близкие большие числа. Так, например, при передискретизации звукового сигнала с 44100 Гц до 48000 Гц этим методом необходимо увеличить частоту дискретизации в 160 раз до 7056000 Гц и затем уменьшить её в 147 раз до 48000 Гц. Таким образом, в данном примере вычисления приходится производить на частоте дискретизации более 7 МГц[17].

2. Анализ классических методов обработки сигналов во временной области

Методы цифровой обработки сигналов в настоящее время интенсивно развиваются. Это прежде всего обусловлено прогрессом в области цифровой микросхемотехника, благодаря которому появилось реальная возможность изготовления сложной цифровой аппаратуры. Первые образцы таких устройств, уже освоены промышленностью, вызвали повышенный интерес разработчиков к открывающимся возможностям и привлекли новый приверженцев этого направления исследований к изучению современных методов и алгоритмов цифровой обработки сигналов. Это задача, однако, оказалось довольно трудной, так

как за последнее десятилетие разработано большое количество алгоритмов разной эффективности и разного назначения[4, 6].

Различают следующие методы обработки сигналов: обработка сигналов во временной области (динамическая форма представления сигналов) и спектральные методы цифровой обработки сигналов (спектральное представление сигналов, Фурье анализ).

В этой главе приведен анализ методов цифровой обработки сигналов, как во временной, так и представления сигналов в спектральной области. К классическим численным методам обработки и аппроксимации сигналов во временной области можно отнести полиномиальные методы представления сигналов и функций: интерполяционные полином Ньютона, интерполяционные полином Лагранжа, которые рассчитаны для обработки гладких функций. Для обработки сигналов с шумом, наилучшим решением является метод наименьших квадратов

Приближение функций интерполяционными полиномами

Интерполяция, интерполирование — в вычислительной математике способ нахождения промежуточных значений величины по имеющемуся дискретному набору известных значений[5, 7].

Многим из тех, кто сталкивается с научными и инженерными расчётами часто приходится оперировать наборами значений, полученных экспериментальным путём или методом случайной выборки. Как правило, на основании этих наборов требуется построить функцию, на которую могли бы с высокой точностью попадать другие получаемые значения. Такая задача называется аппроксимацией. Интерполяцией называют такую разновидность аппроксимации, при которой кривая построенной функции проходит точно через имеющиеся точки данных.

Существует также близкая к интерполяции задача, которая заключается в аппроксимации какой-либо сложной функции другой, более простой функцией. Если некоторая функция слишком сложна для производительных вычислений, можно попытаться вычислить её значение

в нескольких точках, а по ним построить, то есть интерполировать, более простую функцию. Разумеется, использование упрощенной функции не позволяет получить такие же точные результаты, какие давала бы первоначальная функция. Но в некоторых классах задач достигнутый выигрыш в простоте и скорости вычислений может перевесить получаемую погрешность в результатах[6, 8].

В вычислительной математике существенную роль играет интерполяция функций, т.е. построение по заданной функции другой (как правило, более простой), значения которой совпадают со значениями заданной функции в некотором числе точек. Причем интерполяция имеет как практическое, так и теоретическое значение. На практике часто возникает задача о восстановлении непрерывной функции по ее табличным значениям, например полученным в ходе некоторого эксперимента. Для вычисления многих функций, оказывается, эффективно приблизить их полиномами или дробно-рациональными функциями. Теория интерполирования используется при построении и исследовании квадратурных формул для численного интегрирования, для получения методов решения дифференциальных и интегральных уравнений[4].

Задача интерполяции функции, интерполяционные полиномы.

Пусть на отрезке $[a,b]$ задана функция $f(x)$. Задача интерполяции (или интерполирования) состоит в построении функции $g(x)$, совпадающей с заданной $f(x)$ в некотором наборе точек $\{x_0,x_1,\ldots,x_n\}$ из отрезка $[a,b]$ (эти точки называются узлами интерполяции), т.е. должны выполняться условия:

$$g(x_k)=y_k,\ k=0,1,\ldots,n,$$

где y_k - известные значения функции $f(x)$ в точках x_k. Функция $g(x)$ называется интерполянтом функции $f(x)$.

Пример интерполяции с четырьмя узлами приведен на следующем 1.1.рисунке, из которого видно, что узлы интерполяции не обязательно должны располагаться равномерно на отрезке $[a,b]$[6, 19].

Если $f(x)$ табличная функция, скажем полученная из эксперимента, т.е. известны только ее значения y_k в точках x_k, то, вообще говоря, о качестве полученного приближения судить трудно. Однако, если значения $f(x)$ могут быть вычислены в любой точке отрезка $[a,b]$, то в этом случае можно исследовать качество получающегося приближения, например найдя максимальное уклонение функции $g(x)$ от функции $f(x)$. На качество приближения сильное влияние оказывает количество и расположение узлов, а также гладкость функции $f(x)$.

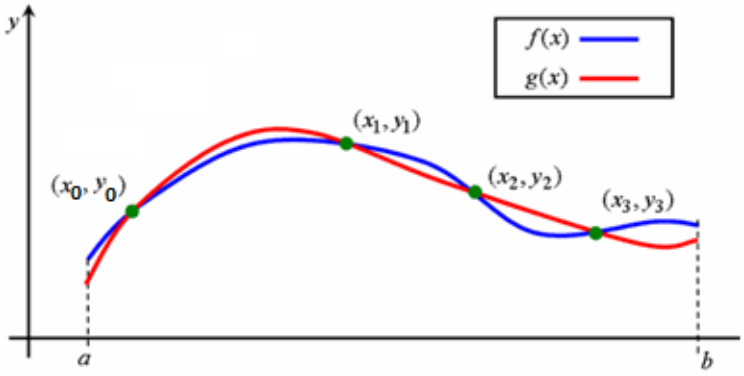


Рис.1.10. Иллюстрация интерполяции с четырьмя узлами.

Мы рассмотрим только линейную интерполяцию, т.е. такую, при которой функция $g(x)$ разыскивается в виде линейной комбинации некоторых функций

$$g(x) = \sum_{k=0}^{n} a_k \varphi_k(x),$$

где для $k=0,1,...,n$: $\varphi_k(x)$ - заданные функции, a a_k - искомые коэффициенты[4, 19].

Ясно, что из постановки задачи интерполяции (т.е. из совпадения значений интерполянта $g(x)$ и интерполируемой функции $f(x)$ в точках x_k) следует, что коэффициенты a_k определяются из решения следующей системы линейных алгебраических уравнений:

$$\sum_{k=0}^{n} a_k \varphi_k(x_j) = y_j, j = 0,1,\dots,n$$

или в развернутой форме

$$\begin{cases} a_0\varphi_0(x_0) + a_1\varphi_1(x_0) + \dots + a_n\varphi_n(x_0) = y_0 \\ a_0\varphi_0(x_1) + a_1\varphi_1(x_1) + \dots + a_n\varphi_n(x_1) = y_1 \\ \vdots \\ a_0\varphi_0(x_n) + a_1\varphi_1(x_n) + \dots + a_n\varphi_n(x_n) = y_n \end{cases}$$

Совершенно ясно, почему число коэффициентов a_k должно совпадать с числом узлов интерполяции x_k. Это нужно для того, чтобы матрица системы была квадратной (т.е. число неизвестных совпадало бы с числом условий, из которых находятся эти неизвестные). Кроме того, для однозначной разрешимости данной системы (при произвольной правой части) необходимо и достаточно, чтобы ее определитель был отличен от нуля, т.е.

$$\begin{vmatrix} \varphi_0(x_0) & \varphi_1(x_0) & \cdots & \varphi_n(x_0) \\ \varphi_0(x_1) & \varphi_1(x_1) & \cdots & \varphi_n(x_1) \\ \vdots & \vdots & \ddots & \vdots \\ \varphi_0(x_n) & \varphi_1(x_n) & \cdots & \varphi_n(x_n) \end{vmatrix} \neq 0$$

Очень часто в качестве системы функций $\varphi_k(x)$ выбирают полиномы, например степени x, именно:

$$\varphi_0(x) = 1, \quad \varphi_1(x) = x, \ \varphi_2(x) = x^2, \dots \varphi_n(x) = x^n$$

Тогда соответствующий интерполянт называют интерполяционным полиномом. Существование и единственность интерполяционного полинома гарантируется, если все узлы интерполяции x_k различны.

$$\begin{vmatrix} x_0^0 & x_0^1 & \cdots & x_0^n \\ x_1^0 & x_1^1 & \cdots & x_1^n \\ \vdots & \vdots & \ddots & \vdots \\ x_n^0 & x_n^1 & \cdots & x_n^n \end{vmatrix}$$

$$\prod_{\substack{j=0 \\ i>j}}^{n} (x_i - x_j)$$

Действительно, определитель системы линейных алгебраических уравнений для нахождения коэффициентов a_k является определителем Вандермонда, который, как известно, равен выражению 1.8 и, следовательно, отличен от нуля в случае, когда все узлы x_k различны. Поскольку матрица системы невырождена, то решение системы существует и единственно. Итак, интерполяционный полином существует и единственный[3, 4, 19].

Можно рассуждать и по-другому. Предположим, что есть два интерполяционных полинома g_k и h_k степени n такие, что для произвольного набора значений выполняются равенства $g(x_k) = h(x_k) = y_k$ для всех $k=0,1,...,n$, т.е. для n-ой точки. Тогда их разность $h_k - g_k$ является полиномом степени не выше n, но обращается в ноль в n-ой точке. По известной теореме алгебры у полинома степени n не может быть больше чем n корней, следовательно, $h_k - g_k \equiv 0$ и $h_k \equiv g_k$. Единственность установлена. А так как полином единственный, то у соответствующей системы линейных алгебраических уравнений есть только одно решение (для произвольной правой части). Из результатов линейной алгебры известно, что у системы может быть либо бесконечное число решений при некоторых правых частях, либо единственное, для произвольной правой части. Последнее как раз и выполняется[4, 6].

Итак, число узлов интерполяционного полинома всегда должно быть на единицу больше его степени. Это понятно также из следующих простых соображений: через две точки проходит единственная прямая, через три - единственная парабола и т.д. Полином может получиться и степени меньшей, чем , например, если три точки лежат на одной прямой, то через них проходит единственный полином первой степени, однако это не нарушает наших рассуждений (просто коэффициент при старшей степени равен нулю). Однако, существует бесконечно много парабол, проходящих через две точки[19].

Казалось бы, при практической реализации интерполяционного процесса коэффициенты интерполяционного полинома a_k можно найти, непосредственно решая систему линейных алгебраических уравнений 1.9 каким-нибудь численным методом. Однако, у такого подхода есть существенный недостаток. Число обусловленности матрицы этой системы быстро растет с ростом числа узлов интерполяции, что может привести к большим ошибкам при решении системы с ней.

$$\begin{cases} a_0 x_0^0 + a_1 x_0^1 + \ldots + a_n x_0^n = y_0 \\ a_0 x_1^0 + a_1 x_1^1 + \ldots + a_n x_1^n = y_1 \\ \vdots \\ a_0 x_n^0 + a_1 x_n^1 + \ldots + a_n x_n^n = y_n \end{cases}$$

Важно, что какие бы подходы для построения интерполяционного полинома не применялись, они всегда должны привести к одинаковому результату (если все вычисления проводятся точно, а не на компьютере), поскольку интерполяционный полином степени существует и единственный при различных -ом узлах интерполяции. Другое дело, что разные способы построения интерполяционного полинома могут обладать разными вычислительными свойствами. Рассмотрим сначала интерполяционный полином в форме Лагранжа. Далее мы будем использовать обозначение для интерполяционного полинома в зависимости от способа его построения[6].

Интерполяционный полином Лагранжа

Итак, мы ищем полином $L_n(x)$ степени не выше n, значения которого совпадают со значениями y_k заданной функции $f(x)$ в узлах x_k, где $k=0,1,\ldots,n$ и все узлы различны.

Одним из способов записи интерполяционного полинома является форма Лагранжа. Предположим, что для $k=0,1,\ldots,n$ функции $\Phi_n(x)$ являются полиномами степени n, которые обладают следующим свойством

$$\Phi_k(x_j) = \begin{cases} 1, & k = j \\ 0, & k \neq j \end{cases}$$

$$L_n(x) = \sum_{k=0}^{n} y_k \Phi_k(x)$$

Тогда полином 1.11 будет как раз тем, который нам и нужен, поскольку это полином степени не выше n и $L_n(x_k) = y_k$ для всех $k=0,1,\ldots,n$.

Функции $\Phi_n(x)$ строятся легко. Действительно, функция 1.12 является полиномом степени n, который обращается в ноль для всех x_j не равных x_k. В точке x_k она принимает значение

$$(x - x_0) \cdot (x - x_1) \cdot \ldots \cdot (x - x_{n-1}) \cdot (x - x_n)$$

$$(x_k - x_0) \cdot (x_k - x_1) \cdot \ldots \cdot (x_k - x_{n-1}) \cdot (x_k - x_n)$$

Тогда

$$\Phi_k(x) = \frac{(x - x_0) \cdot (x - x_1) \cdot \ldots \cdot (x - x_{n-1}) \cdot (x - x_n)}{(x_k - x_0) \cdot (x_k - x_1) \cdot \ldots \cdot (x_k - x_{n-1}) \cdot (x_k - x_n)}$$

или, что то же самое:

$$\Phi_k(x) = \prod_{\substack{j=0 \\ j \neq k}}^{n} \frac{x - x_j}{x_k - x_j}$$

Итак, интерполяционный полином в форме Лагранжа имеет вид

$$L_n(x) = \sum_{k=0}^{n} y_k \prod_{\substack{j=0 \\ j \neq k}}^{n} \frac{x - x_j}{x_k - x_j}$$

или в развернутой форме:

$$L_n(x) = \sum_{k=0}^{n} y_k \frac{(x - x_0) \cdot (x - x_1) \cdot \ldots \cdot (x - x_{n-1}) \cdot (x - x_n)}{(x_k - x_0) \cdot (x_k - x_1) \cdot \ldots \cdot (x_k - x_{n-1}) \cdot (x_k - x_n)}$$

В качестве примера рассмотрим известны следующие значение заданный функции :

n	0	1	2	3
x	0	0,25	0,5	0,75
y	0,82	0,6801	0,6262	0,6817

Необходимо найти интерполяционный полином по формуле Лагранжа.

В результате приведенных выше математических преобразований получим выражение для полинома Лагранжа:

$$L_3(x) = 0{,}2496x^3 + 0{,}4968x^2 - 0{,}6996x + 0{,}82$$

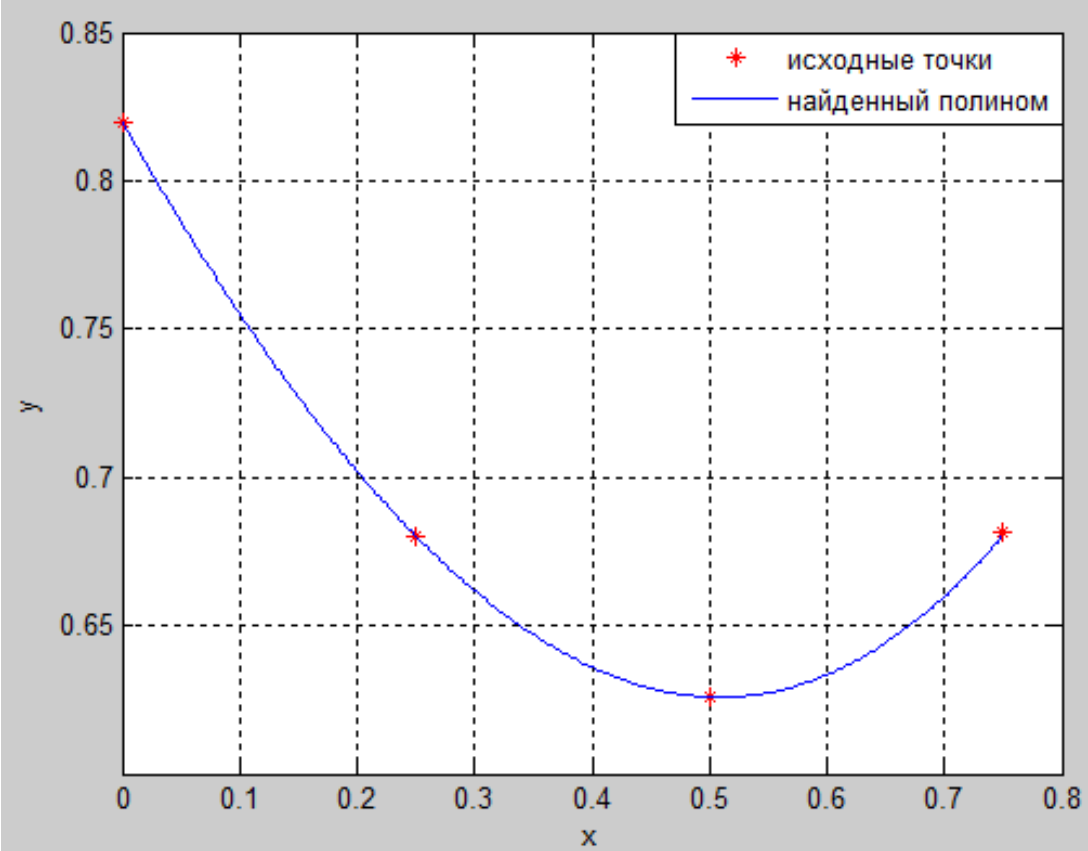


Рис. 1.11. График найденного полинома

Образование интерполяционных многочленов Лагранжа связано с большой вычислительной работой. Также велика вычислительная работа при получении значения $L_n(x)$ для какого-то фиксированного значения x. Если даже мы имеем интерполяционный многочлен Лагранжа, построенный по значением x_0, x_1, … , x_n , то это мало помогает нам при построении интерполяционного многочлена Лагранжа в других точках измерения заданной функции[7].

Интерполяционный полином Ньютона

Форма Лагранжа записи интерполяционного полинома, не является единственно возможной. Кроме этого, она имеет определенные

неудобства. Так предположим, задан набора точек (x_k, y_k) $k = 0,1, \ldots, n$ и по нему построен интерполяционный полином в форме Лагранжа 1.17

Тогда очевидно, что при добавлении новой точки к исходному набору точек придется заново конструировать интерполяционный полином. Этого недостатка лишен интерполяционный полином Ньютона.

Важно понимать, что каким бы способом не строился интерполяционный полином по заданному набору точек, результат всегда получается один и тот же (с точностью до ошибок, возникающих при округлении вещественных чисел), поскольку через заданные точки проходит ровно один полином n -ой степени[7, 19].

Для вывода интерполяционного полинома в форме Ньютона понадобится понятие о разделенных разностях.

Пусть нам заданы различные точки $x_0, x_1, \ldots, x_n$ и функция $f(x)$. Первой разделенной разностью функции $f(x)$ в точках x_i, x_{i+1} называется величина

$$[x_i, x_{i+1}]f = \frac{f(x_{i+1}) - f(x_i)}{x_{i+1} - x_i}$$

Разделенные разности более высоких порядков определяются рекуррентно. Например разделенная разность второго порядка в точках x_0, x_1, x_2 определяется через две разделенные разности первого порядка $[x_0, x_1]f$ и $[x_1, x_2]f$ следующим образом:

$$[x_0, x_1, x_2]f = \frac{[x_1, x_2]f - [x_0, x_1]f}{x_2 - x_0}$$

Разделенная разность второго порядка в точках x_0, x_1, x_2 определяется так:

$$[x_1, x_2, x_3]f = \frac{[x_2, x_3]f - [x_1, x_2]f}{x_3 - x_1}$$

В общем случае разделенная разность s-го порядка определяется через две разделенные разности $(s-1)$-го порядков

$$[x_i, x_{i+1}, \quad \dots x_{i+s}]f = \frac{[x_{i+1}, x_{i+2}, \dots x_{i+s}]f - [x_i, x_{i+1}, \dots x_{i+s-1}]f}{x_{i+s} - x_i}$$

Разделенные разности удобно представить в виде треугольной таблицы

$$\begin{array}{llllll}
f(x_0) & & & & & \\
 & [x_0, x_1]f & & & & \\
f(x_1) & & [x_0, x_1, x_2]f & & & \\
 & [x_1, x_2]f & \vdots & & & \\
f(x_2) & \vdots & [x_1, x_2, x_3]f & & & \\
\vdots & \vdots & \vdots & \ddots & [x_0, x_1, \dots, x_{n-1}]f & \\
\vdots & \vdots & \vdots & & & [x_0, x_1, \dots, x_n]f \\
\vdots & \vdots & \vdots & \ddots & [x_1, x_2, \dots, x_n]f & \\
f(x_{n-1}) & \vdots & [x_{n-2}, x_{n-1}, x_n]f & & & \\
 & [x_{n-1}, x_n]f & & & & \\
f(x_n) & & & & &
\end{array}$$

Разделенные разности обладают рядом важных свойств. Одно из них — симметрия, т.е. разделенная разность $[x_0, x_1, \dots, x_m]$ не меняет своего значения при любой перестановке $x_0, x_1, \dots, x_m$. Это свойство следует, например, из представления разделенной разности в виде.

$$[x_0, x_1, \dots, x_m]f = \sum_{j=0}^{m} \frac{f(x_j)}{\prod_{\substack{i=0 \\ i \neq j}}^{m} (x_j - x_i)}$$

Получим выражение для интерполяционного полинома Ньютона для интерполяции функции $f(x)$ по набору точек $x_0, x_1, \dots, x_n$, воспользовавшись формой Лагранжа.

Интерполяционный полином $L_n(x)$ для набора точек $x_0, x_1, \dots, x_n$ может быть записан в виде $Ln(x) = L_0(x) + (L_1(x) - L_0(x)) + \dots + L_{n-1}(x) - L_n(x))$ (очевидно, что в правой части все слагаемые, кроме $L_n(x)$ сокращаются). Здесь — $L_n(x)$ интерполяционный полином, построенный для $f(x)$ по набору точек $x_0, x_1, \dots, x_k$. Он записывается в форме Лагранжа следующим образом:

$$L_k(x) = \sum_{i=0}^{k} f(x_i) \prod_{\substack{j=0 \\ j \neq i}}^{k} \frac{x - x_j}{x_i - x_j}$$

Ясно также, что $L_0(x) = f(x_1)$ т.к. интерполяционный полином нулевой степени, построенный по одной точке, есть константа, значение которой равно $f(x_1)$[6,8].

Наша цель — установить следующие равенства:

$$L_{k+1}(x) - L_k(x) = [x_0, x_1, \ldots, x_{k+1}]f \cdot w_{k+1}(x)$$

Где $w_{k+1}(x) = \prod_{i=0}^{k}(x - x_i)$

Т.е., надо доказать, что

$$L_1(x) - L_0(x) = [x_0, x_1]f \cdot w_1(x), \quad w_1 = x - x_0$$

$$L_2(x) - L_1(x) = [x_0, x_1, x_2]f \cdot w_2(x), \quad w_2 = (x - x_0) \cdot (x - x_1)$$

$$\ldots$$

$$L_n(x) - L_{n-1}(x) = [x_0, x_1, \ldots, x_n]f \cdot w_n(x), \quad w_n = (x - x_0) \cdot (x - x_1) \cdot \ldots \cdot (x - x_{n-1})$$

Тогда интерполяционный полином $L_n(x)$ приобретает следующую форму

$$L_n(x) = f(x_0) + [x_1, x_2]f \cdot (x - x_1) + [x_1, x_2, x_3]f \cdot (x - x_1)(x - x_2) + \ldots + [x_1, x_2, \ldots, x_{n+1}]f \cdot (x - x_1)(x - x_2)\ldots(x - x_n),$$

которая называется формой Ньютона интерполяционного полинома или интерполяционным полиномом Ньютона с разделенными разностями.

Преимущества записи интерполяционного полинома Ньютона очевидны. Если в точке x вычислено значение интерполяционного полинома $L_n(x)$, построенного по набору точек $x_1, x_2, \ldots, x_{n+1}$, а затем к этому набору добавилась еще одна точка x_{n+2}, то для получения значения нового интерполяционного полинома $L_{n+1}(x)$ (степень которого на единицу выше) в точке достаточно добавить к значению $L_n(x)$ слагаемое

$$[x_1, x_2, \ldots, x_{n+1}, x_{n+2}]f \cdot (x - x_1)(x - x_2)\ldots(x - x_{n+1}).$$

Интерполяционные формулы для равноотстоящих узлов

Если узлы интерполяции отстоят друг от друга на одинаковое расстояние, т.е. $x_2 - x_1 = x_3 - x_2 = \ldots = x_{n+1} - x_n = h$ то для интерполяционных полиномов в форме Лагранжа и Ньютона можно получить специальные формулы, упрощающие вычисления

Рассмотрим, например интерполяционный полином в форме Лагранжа

$$L_n(x) = \sum_{k=1}^{n+1} y_k \Phi_k(x),$$

где

$$\Phi_k(x) = \frac{(x-x_1)\cdot(x-x_2)\cdot\ldots\cdot(x-x_{k-1})\cdot(x-x_{k+1})\cdot\ldots\cdot(x-x_n)\cdot(x-x_{n+1})}{(x_k-x_1)\cdot(x_k-x_2)\cdot\ldots\cdot(x_k-x_{k-1})\cdot(x_k-x_{k+1})\cdot\ldots\cdot(x_k-x_n)\cdot(x_k-x_{n+1})}.$$

Введем обозначение $\frac{x-x_1}{h} = t$.

Тогда выражение для $\Phi_k(x(t))$принимает вид

$$\Phi_k(x(t)) = \frac{th(th-h)\cdot\ldots\cdot(th-(k-2)h)\cdot(th-kh)\cdot\ldots\cdot(th-(n-1)h)\cdot(th-nh)}{(k-1)h\cdot(k-2)h\cdot\ldots\cdot(1)h\cdot(-1)h\cdot\ldots\cdot(-(n-k))h\cdot(-(n-k+1))h}$$

Проводя сокращения в числителе и знаменателе и предполагая, что $t \neq k$ (если $t = k$ то соответствующий x(t) является одним из узлов интерполяции и значение интерполяционного по) получим, что

$$\Phi_k(x(t)) = \frac{t(t-1)\cdot\ldots\cdot(t-n)}{(t-k)}\frac{(-1)^{n-k+1}}{(k-1)!(n-k+1)!} = (-1)^n\frac{t(t-1)\cdot\ldots\cdot(t-n)}{n!}(-1)^{k-1}C_n^{k-1}\frac{1}{t-k}.$$

Тогда

$$L_n(x(t)) = (-1)^n\frac{t(t-1)\cdot\ldots\cdot(t-n)}{n!}\sum_{k=1}^{n+1} y_k(-1)^{k-1}C_n^{k-1}\frac{1}{t-k}.$$

В полученной записи интерполяционного полинома коэффициент $\frac{t(t-1)\cdot\ldots\cdot(t-n)}{n!}$ не зависит ни от интерполируемой функции, ни от набора узлов. Такие коэффициенты можно вычислить один раз перед вычислением интерполяционного полинома и записать в массив, так же как и значения C_n^{k-1}. [4, 7]

В качестве примера рассмотрим известны следующие значение заданный функции :

n	0	1	2	3
x	0	0,25	0,5	0,75
y	0,82	0,6801	0,6262	0,6817

Необходимо найти интерполяционный полином Ньютона.

В результате приведенных выше математических преобразований получим выражение для полинома Ньютона:

$$L_3(x) = 0{,}2496x^3 + 0{,}5008x^2 - 0{,}7004x + 0{,}82$$

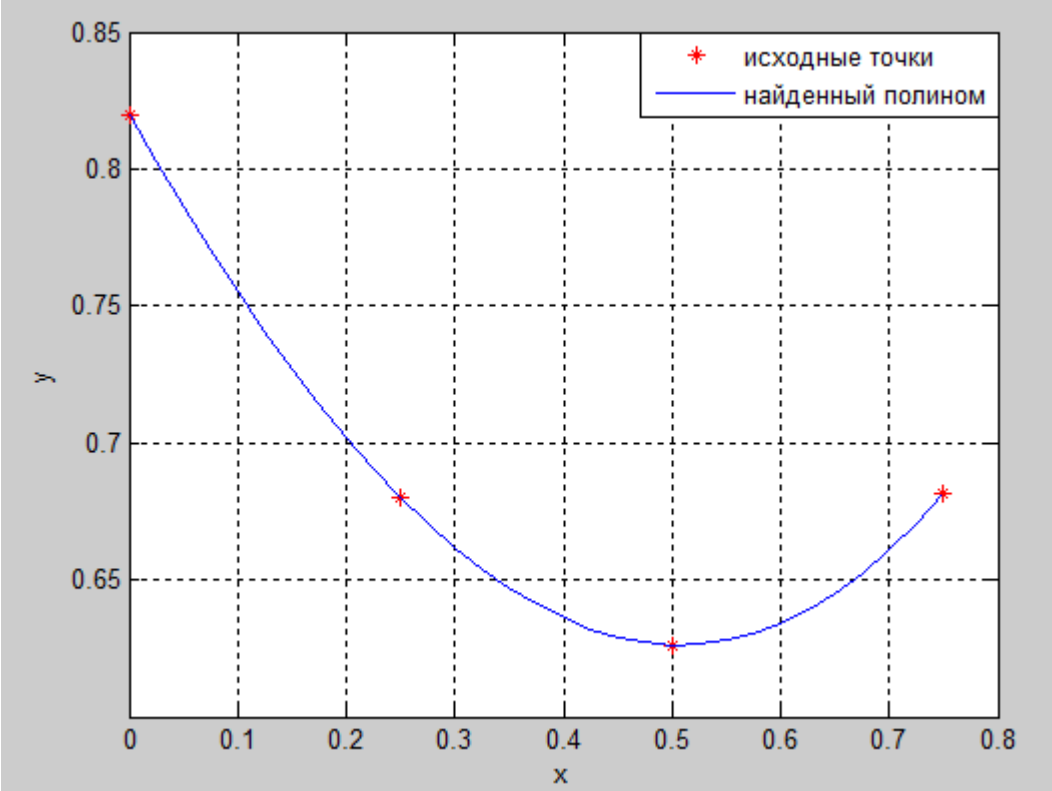


Рис. 1.12. График найденной функции полином Ньютона

Задача интерполяции имеет единственное решение, поэтому формулы Лагранжа и Ньютона для данных значений x_i и y_i тождественны и отличаются лишь по группировкой членов. На практике формула Ньютона более удобна. Особенность ее заключаются в том, что в случае добавления новых узлов интерполяции в формуле Лагранжа надо пересчитывать

заново все коэффициенты, а в формуле Ньютона добавляется только новые слагаемые, а старые остаются без изменения.

Метод наименьших квадратов

Метод наименьших квадратов (МНК) — один из базовых методов регрессионного анализа для оценки неизвестных параметров регрессионных моделей по выборочным данным. Метод основан на минимизации суммы квадратов остатков регрессии.

Необходимо отметить, что собственно методом наименьших квадратов можно назвать метод решения задачи в любой области, если решение заключается или удовлетворяет некоторому критерию минимизации суммы квадратов некоторых функций от искомых переменных. Поэтому метод наименьших квадратов может применяться также для приближённого представления (аппроксимации) заданной функции другими (более простыми) функциями, при нахождении совокупности величин, удовлетворяющих уравнениям или ограничениям, количество которых превышает количество этих величин. Эти утверждения справедливы и для сигналов, обрабатываемых на ограниченном интервале

Если набор экспериментальных данных получен со значительной погрешностью, то не имеет смысла использовать интерполяцию Лагранжа полиномами и сплайнами для обработки результатов. В этом случае необходимо провести аппроксимирующую кривую, которая не проходит через экспериментальные точки, но в то же время отражает исследуемую зависимость и сглаживает "выбросы", возможные за счет погрешности эксперимента[6, 8].

Обозначим узлы исходной таблицы данных через x_i, где $0 <= i <= n$ - номер узла. Считаем известными значения экспериментальных данных в узловых точках $f(x_i) = f_i$. Введем непрерывную функцию $\varphi(x)$ для аппроксимации дискретной зависимости $f(x_i)$. В узлах функции $\varphi(x)$ и $f(x)$ будут отличаться на величину $e_i = \varphi(x_i)-f(x_i)$. Отклонения ei могут

принимать положительные и отрицательные значения. Чтобы не учитывать знаки, возведем каждое отклонение в квадрат и просуммируем квадраты отклонений по всем узлам:

$$Q = \sum_{i=0}^{n} \varepsilon_i^2 = \sum_{i=0}^{n} [\varphi(xi) - f(xi)]^2$$

Метод построения аппроксимирующей функции *φ(x)* из условия минимума величины Q называется методом наименьших квадратов (МНК).

Наиболее распространен способ выбора функции *φ(x)* в виде линейной комбинации:

$$\varphi(x) = c_0\,\varphi_0(x) + c_1\,\varphi_1(x) + \ldots + c_m\,\varphi_0(x)$$

$\varphi_0(x)$, $\varphi_1(x), \ldots,$ $\varphi_m(x)$ - базисные функции; $m <= n$; c_0, $c_1, \ldots, c_m$ - коэффициенты, определяемые при минимизации величины Q.

Математически условия минимума суммы квадратов отклонений Q запишем, приравнивая нулю частные производные от Q по коэффициентам c_k, $0<=k<=n$:

$$\begin{cases} \frac{\partial Q}{\partial c_0} = 2\sum_{i=0}^{n}[\,c_0\varphi_0(x_i) + c_1\varphi_2(x_i) + \cdots + c_m\varphi_m(x_i) - f_i]\,\varphi_0\,(x_i) = 0, \\ \frac{\partial Q}{\partial c_1} = 2\sum_{i=0}^{n}[\,c_0\varphi_0(x_i) + c_1\varphi_2(x_i) + \cdots + c_m\varphi_m(x_i) - f_i]\,\varphi_1\,(x_i) = 0, \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ \frac{\partial Q}{\partial c_m} = 2\sum_{i=0}^{n}[\,c_0\varphi_0(x_i) + c_1\varphi_2(x_i) + \cdots + c_m\varphi_m(x_i) - f_i]\,\varphi_m\,(x_i) = 0. \end{cases}$$

Из системы линейных алгебраических уравнений (3) определяются все коэффициенты c_k. Система (3) называется системой нормальных уравнений. Матрица этой системы имеет следующий вид:

$$\begin{vmatrix} (\varphi_0, \varphi_0) & (\varphi_0, \varphi_0) & \cdots & (\varphi_0, \varphi_m) \\ (\varphi_0, \varphi_1) & (\varphi_0, \varphi_1) & \cdots & (\varphi_1, \varphi_m) \\ \vdots & \vdots & \ddots & \vdots \\ (\varphi_0, \varphi_m) & (\varphi_0, \varphi_m) & \cdots & (\varphi_m, \varphi_m) \end{vmatrix}$$

и называется матрицей Грама. Элементы матрицы Грама являются скалярными произведениями базисных функций[5,8]

$$(\varphi_j, \varphi_k) = \sum_{i=0}^{n} \varphi_j\,(x_i), \varphi_k(x_i)$$

Расширенная матрица системы уравнений получится добавлением справа к матрице Грама столбца свободных членов

$$\begin{vmatrix} \varphi_0 , f \\ \varphi_1 , f \\ \vdots \\ \varphi_m , f \end{vmatrix}$$

где скалярные произведения, являющиеся элементами столбца, определяются аналогично :

$$(\varphi_k, f) = \sum_{i=0}^{n} \varphi_k (x_i), f_i)$$

Отметим основные свойства матрицы Грама, полезные при программной реализации алгоритмов МНК:

1) матрица симметрична, т.е. $a_{i,j} = a_{j,i,}$ что позволяет сократить объем вычислений при заполнении матрицы;

2) матрица является положительно определенной, следовательно, при решении системы нормальных уравнений методом исключения Гаусса можно отказаться от процедуры выбора главного элемента;

3) определитель матрицы будет отличен от нуля, если в качестве базиса выбраны линейно независимые функции $\varphi_k(x)$, при этом система имеет единственное решение.

При обработке экспериментальных данных, определенных с погрешностью e в каждой узловой точке, обычно начинают с аппроксимации функцией $\varphi(x)$, представимой одной-двумя базисными функциями. После определения коэффициентов ck вычисляют величину *Q* по формуле (1). Если получится, что корень*(Q)* > *e* , то необходимо расширить базис добавлением новых функций $\varphi_k(x)$. Расширение базиса необходимо осуществлять до тех пор, пока не выполнится условие корень*(Q)~=e* [5, 8].

Выбор конкретных базисных функций зависит от свойств аппроксимируемой функции *f(x),* таких, как периодичность, экспоненциальный или логарифмический характер, свойства симметрии, наличие асимптотики и т.д. На практике чаще всего используется решение, показанное далее.

Выберем базисные функции $\varphi_k(x)$ в виде последовательности степеней аргумента x, которые линейно независимы,

$$\varphi_0(x) = x^0 = 1, \varphi_1(x) = x^1 = x, \dots, \varphi_m(x) = x^m$$

В этом случае мы будем аппроксимировать экспериментальную зависимость полиномом. Степень полинома m выбираем $m<<n$ (при лагранжевой интерполяции $m=n$). Аппроксимирующая кривая в МНК не проходит через значения исходной функции в узлах, но проведена из условия наименьшего суммарного квадратичного отклонения. Экспериментальные данные "сглаживаются" с помощью функции $\varphi(x)$. Если же выбрать $m=n$, то на основании единственности интерполяционного полинома получим функцию $\varphi(x)$, совпадающую с каноническим интерполяционным полиномом степени n, аппроксимирующая кривая пройдет через все экспериментальные точки и величина Q будет равна нулю. Последнее обстоятельство используется для отладки и тестирования программ, реализующих алгоритмы МНК [6, 7] .

Запишем расширенную матрицу системы нормальных уравнений для базиса (8):

$$\begin{vmatrix} n+1 & \sum_{i=0}^{n} x_i & \sum_{i=0}^{n} x_i^2 & \cdots & \sum_{i=0}^{n} x_i^m & \sum_{i=0}^{n} f_i \\ \sum_{i=0}^{n} x_i & \sum_{i=0}^{n} x_i^2 & \sum_{i=0}^{n} x_i^3 & \cdots & \sum_{i=0}^{n} x_i^{m+1} & \sum_{i=0}^{n} x_i f_i \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ \sum_{i=0}^{n} x_i^m & \sum_{i=0}^{n} x_i^{m+1} & \sum_{i=0}^{n} x_i^{m+2} & \cdots & \sum_{i=0}^{n} x_i^{2m} & \sum_{i=0}^{n} x_i^m f_i \end{vmatrix}$$

Нетрудно видеть, что для формирования расширенной матрицы достаточно вычислить только элементы первой строки и двух последних столбцов, остальные элементы не являются "оригинальными" и заполняются с помощью циклического присваивания.

Для решения систем уравнений с матрицей Грама разработаны методы сингулярного разложения. Если же $m <= 4..5$, то такие системы можно решать и более простым методом исключения Гаусса.

В качестве примера рассмотрим следующие значение заданный функции:

n	0	1	2	3	4	5	6	7
x	0	0,125	0,25	0,375	0,5	0,625	0,75	0,875
y	0,82	0,7408	0,6801	0,6410	0,6262	0,6388	0,6817	0,7578

Требуется найти аппроксимирующую структуры кубического полинома по методу наименьших квадратов.

Запишем системы уравнений для кубической аппроксимации:

$$\begin{cases} A_0(n+1) + A_1\sum_{i=0}^{n} x_i + A_2\sum_{i=0}^{n} x_i^2 + A_3\sum_{i=0}^{n} x_i^3 = \sum_{i=0}^{n} y_i \\ A_0\sum_{i=0}^{n} x_i + A_1\sum_{i=0}^{n} x_i^2 + A_2\sum_{i=0}^{n} x_i^3 + A_3\sum_{i=0}^{n} x_i^4 = \sum_{i=0}^{n} x_i y_i \\ A_0\sum_{i=0}^{n} x_i^2 + A_1\sum_{i=0}^{n} x_i^3 + A_2\sum_{i=0}^{n} x_i^4 + A_3\sum_{i=0}^{n} x_i^5 = \sum_{i=0}^{n} x_i^2 y_i \\ A_0\sum_{i=0}^{n} x_i^3 + A_1\sum_{i=0}^{n} x_i^4 + A_2\sum_{i=0}^{n} x_i^5 + A_{13}\sum_{i=0}^{n} x_i^6 = \sum_{i=0}^{n} x_i^3\, y_i \end{cases} \quad (1.5)$$

Составим вспомогательную таблицу

Таблица 1.1

n	x	y	x^2	x^3	x^4	x^5	x^6
0	0,0000	0,8200	0,0000	0,0000	0,0000	0,0000	0,0000
1	0,1250	0,7408	0,0156	0,0020	0,0002	0,0000	0,0000
2	0,2500	0,6802	0,0625	0,0156	0,0039	0,0010	0,0002
3	0,3750	0,6410	0,1406	0,0527	0,0198	0,0074	0,0028
4	0,5000	0,6263	0,2500	0,1250	0,0625	0,0313	0,0156
5	0,6250	0,6388	0,3906	0,2441	0,1526	0,0954	0,0596
6	0,7500	0,6817	0,5625	0,4219	0,3164	0,2373	0,1780
7	0,8750	0,7578	0,7656	0,6699	0,5862	0,5129	0,4488
∑	3,5000	5,5866	2,1875	1,5313	1,1416	0,8853	0,7050

Подставим данные из таблицы 1.1 в систему 1.5, и получим систему уравнений 1.6, решив систему уравнений 1.6, получим выражение с

помощью которого можно найти коэффициенты аппроксимирующую структуры.

$$\begin{cases} A_0 8 + A_1 3.5 + A_2 2.1875 + A_3 1.5313 = y_0 + y_1 + y_2 + y_3 + y_4 + y_5 + y_6 + y_7 \\ A_0 3.5 + A2.1875_1 + A_2 1.5313 + A_3 1.1416 = \frac{1}{8}(y_1 + 2y_2 + 3y_3 + 4y_4 + 5y_5 + 6y_6 + 7y_7) \\ A_0 2.1875 + A_1 1.5313 + A_2 1.1416 + A_3 0.8853 = \frac{1}{64}(y_1 + 4y_2 + 9y_3 + 16y_4 + 25y_5 + 36y_6 + 49y_7) \\ A_0 1.5313 + A_1 1.1416 + A_2 0.8853 + A_3 0.70050 = \frac{1}{512}(y_1 + 8y_2 + 27y_3 + 36y_4 + 125y_5 + 216y_6 + 343y_7) \end{cases} \quad (1.6)$$

$$\begin{aligned} A0 &= -\frac{108383}{2384356}y4 + \frac{578031}{2384356}y1 - \frac{144499}{2384356}y2 - \frac{867029}{7153068}y3 + \\ &+ \frac{6394387}{7153068}y0 + \frac{144499}{2384356}y5 + \frac{650263}{7153068}y6 - \frac{144499}{2384356}y7; \\ A1 &= \frac{1192528}{12517869}y4 + \frac{24637762}{12517869}y1 + \frac{19207040}{4172623}y2 + \frac{13511234}{4172623}y3 - \\ &- \frac{4106352}{596089}y0 - \frac{10861794}{4172623}y5 - \frac{10994336}{4172623}y6 + \frac{3973810}{1788267}y7; \\ A2 &= \frac{39882704}{12517869}y4 - \frac{101441728}{12517869}y1 - \frac{54333584}{4172623}y2 - \frac{87569824}{12517869}y3 + \\ &+ \frac{26010800}{1788267}y0 + \frac{44796160}{4172623}y5 + \frac{110979152}{12517869}y6 - \frac{16473376}{1788267}y7; \\ A3 &= \frac{3853440}{596089}y1 - \frac{5394816}{596089}y0 + \frac{5394816}{596089}y2 + \frac{2312064}{596089}y3 - \\ &- \frac{2312064}{596089}y4 - \frac{5394816}{596089}y5 - \frac{3853440}{596089}y6 + \frac{5394816}{596089}y7. \end{aligned} \quad (1.7)$$

Вычислим коэффициенты аппроксимирующего полинома по выражению (1.7)

A_0 = 0.8211;

A_1 = -0.7204;

A_2 = 0.5604;

A_3 = 0.2048.

$$Y = A0 + A_1 x + A_2 x^2 + A_3 x^3 \quad (1.8)$$

Поставив найденные коэффициенты аппроксимирующего полинома в формулу (1.8) получим уравнение кубической зависимости данной функции:

$$Y = 0{,}8211 - 0{,}7204x + 0{,}5604x^2 + 0{,}2048x^3$$

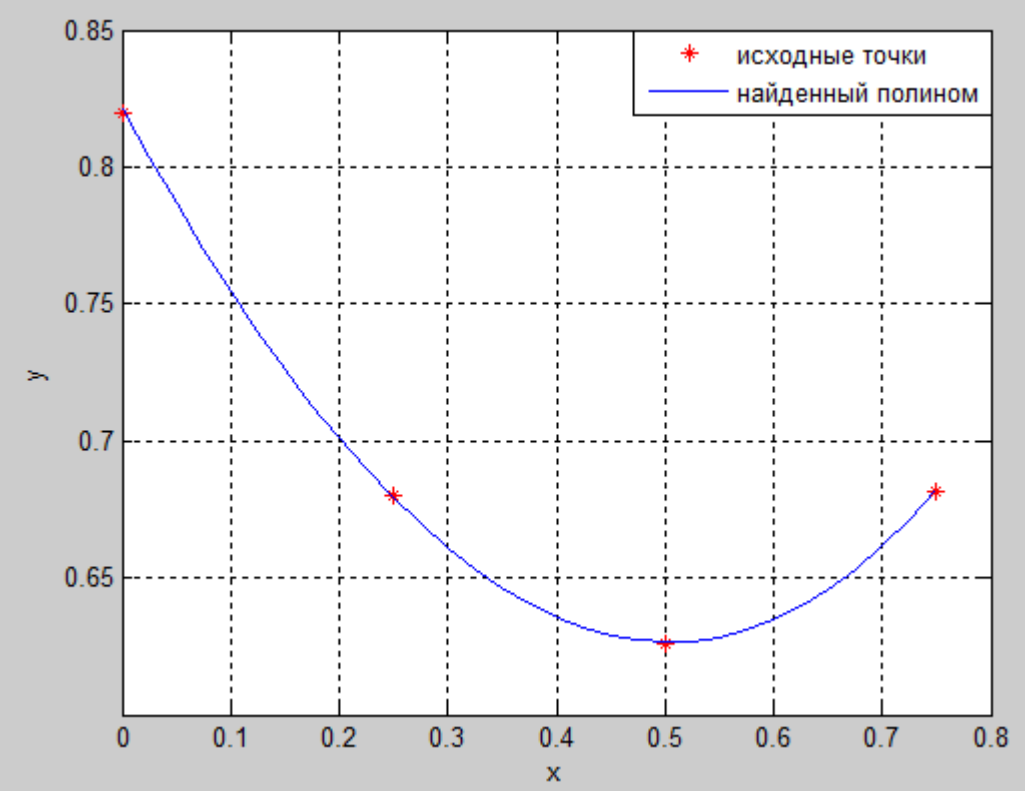


Рис. 1.13. График найденного полинома

Таким образом, из классических методов аппроксимации наиболее близким и применимым в практике цифровой обработке сигналов является метод наименьших квадратов. Он минимизирует среднеквадратичную ошибку аппроксимации, что весьма удобно для обработки сигналов при наличии шумов.

3. Исследование спектральных методов цифровой обработки сигналов

Наряду представлением сигналов во временной области широко применяется представление сигналов в спектральной области.

Спектральный анализ заключается в разложении сигнала на его частотные или спектральные составляющие и оценке или измерении их

спектральных характеристик − амплитуды, фазы, мощности, спектральной плотности мощности.

К задачам, решаемым методами спектрального анализа, относятся, обнаружение, разрешение и оценивание параметров сигналов, сжатие данных, выделение информативных признаков, идентификация объектов (определение частотных, импульсных и других характеристик), распознавание образов (речи, изображений). Для случайных сигналов с помощью спектрального анализа решается общая задача выявления скрытых периодичностей и статистических (корреляционных) связей. Спектральный анализ детерминированных периодических (регулярных) сигналов и сигналов конечной длительности называют также гармоническим анализом [9, 11].

Одним из основных алгоритмов является дискретное преобразование Фурье (ДПФ). ДПФ – одна из двух наиболее распространенных и мощных процедур цифровой обработки сигналов (Другая процедура – цифровая фильтрация). ДПФ позволяет анализировать, преобразовывать и синтезировать сигнал такими способами, которые невозможны при непрерывной (аналоговой) обработке. Сегодня ДПФ используется практически во всех областях инженерной деятельности.

ДПФ – это математическая процедура, используемая для определения гармонического, или частотного, состава дискретного сигнала. На выходе алгоритма ДПФ образуется набор спектральных (весовых) коэффициентов *F(k)*, где *k*- номер коэффициента, соответствующий порядковой частоте гармоники.

Прямое преобразование ДПФ осуществляется над конечным числом отсчетов (*2 n*) входного сигнала, что позволяет преобразовать сигнал из временного представления в спектральное [13, 14].

$$F(k) = \frac{1}{N} \sum x(n)\left(\cos(2\pi nk / N) - j \sin(2\pi nk / N)\right)$$

Обратное ДПФ преобразование позволяет наоборот перевести сигнал из представления в спектральной области в его представление во временной области в том же количестве точек.

$$x(n) = \frac{1}{N}\sum F(k)\left(\cos(2\pi nk / N) + j\sin(2\pi nk / N)\right)$$

Спектральные коэффициенты *F(k)* получаются в результате свертки (попарного перемножения) входных отсчетов сигнала с базисными функциями Фурье – синусами и косинусами.

Так как базисные функции *Sin* и *Cos* равны нулю соответственно в точках 0 и 90 градусов фазовой плоскости, поэтому в разложении присутствуют как вещественная (*Cos*) так и мнимая (*Sin*) составляющие.

Базис Адамара. Разложение сигнала в спектр реализуется аналогично базису Фурье, т.е. попарным перемножением отсчетов сигнала и базисных функций. Формулы прямого и обратного преобразований имеют вид:

$$a_k = \frac{1}{N}\sum x(n)W(n) \qquad (1.9)$$

Где, a_k- Спектральные коэффициенты базиса Адамара, N – количество отсчетов, $W(n)$ - матрица базиса Адамара, $x(n)$ – дискретные отсчеты сигнала.

Формула обратного преобразования:

$$x(n) = \sum a_k W_k \qquad (1.10)$$

Количество спектральных коэффициентов меньше числа отсчетов сигнала из-за эффекта сходимости, многие коэффициенты равны или близки к нулю. В отличие от базиса Фурье здесь нет умножения на значения Sin или Cos, так как базисные функции имеют вид +1 или -1, т.е. отсчеты сигнала умножаются на единицу, фактически выполняется операция присвоения знака.[14, 15]

В качестве примера рассмотрим известны следующие значение заданный функции :

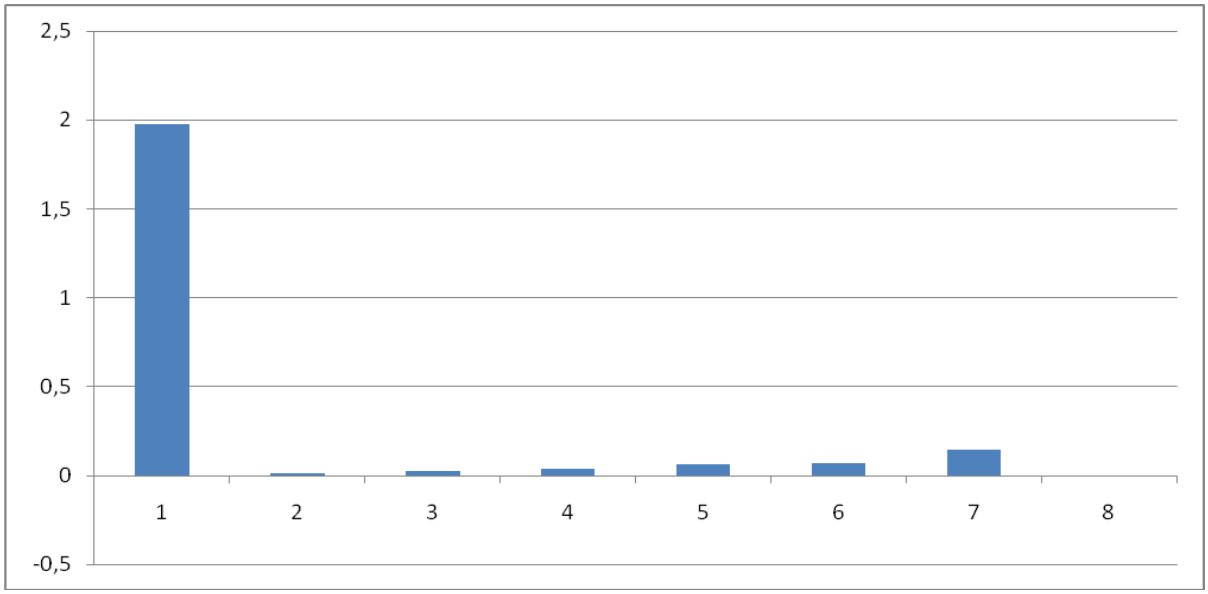
n	0	1	2	3	4	5	6	7
x	0	0,125	0,25	0,375	0,5	0,625	0,75	0,875
y	0,82	0,7408	0,6801	0,6410	0,6262	0,6388	0,6817	0,7578

Требуется найти спектр сигнала на основе базиса функции Адамара.

Чтобы найти спектральные коэффициенты сигнала воспользуемся формулой:

1/8	*	0,82	*	1	1	1	1	1	1	1	1	=	1,975
		0,7408		1	-1	1	-1	1	-1	1	-1		0,0104
		0,6801		1	1	-1	-1	1	1	-1	-1		0,023
		0,641		1	-1	-1	1	1	-1	-1	1		0,0366
		0,6262		1	1	1	1	-1	-1	-1	-1		0,06272
		0,6388		1	-1	1	-1	-1	1	-1	1		0,07318
		0,6817		1	1	-1	-1	-1	-1	1	1		0,14644
		0,7578		1	-1	-1	1	-1	1	1	-1		-0,0083

Построим спектрограмму исходного сигнала:



Сделаем обратное преобразование и найдём исходный сигнал по формуле:

1	1	1	1	1	1	1	1		1,975		0,82
1	-1	1	-1	1	-1	1	-1		0,0104		0,7408
1	1	-1	-1	1	1	-1	-1		0,023		0,6801
1	-1	-1	1	1	-1	-1	1	*	0,0366	=	0,641
1	1	1	1	-1	-1	-1	-1		0,06272		0,6262
1	-1	1	-1	-1	1	-1	1		0,07318		0,6388
1	1	-1	-1	-1	-1	1	1		0,14644		0,6817
1	-1	-1	1	-1	1	1	-1		-0,086		0,7578

Выводы по Главе I

В этой главе было изучены современные и классические методы цифровой обработки сигналов. Проанализированы классические методы цифровой обработки функции и сигналов, такие как, интерполяция полиномами Лагранжа и Ньютона и метод наименьших квадратов, приведены контрольные примеры для каждого метода. Кроме этого в главе приведен анализ спектральных методов цифровой обработки сигналов, основы Фурье анализа, базис функций Адамара, в качестве примера показано переход сигнала от временного представления к спектральному представлению на основе базиса функции Адамара.

Постановка задачи.

Применение классических интерполяционных полиномов в задачах цифровой обработки сигналов ограничивается использованием метода наименьших квадратов, так как он обеспечивает два ключевых требования: сглаживание зашумлённых сигналов и возможность их представления в виде алгебраических полиномов. Оба этих качества необходимы при реализации процедуры согласование частоты дискретизации.

Однако, метод наименьших квадратов обладает существенными недостатками. Во-первых, при переходе к полиномам более высокой степени необходимо пересчитывать всех коэффициентов. Во-вторых, из-за

отсутствие сходимости в аппроксимирующих выражениях участвуют все отсчеты входного сигнала увеличивая сложность вычислений.

В данной работе ставится задача от временного представления сигнала к спектральному, а затем к полиномиальному . При этом основная задача – получить аналитическое представление сигнала или его фрагмента в виде алгебраического полинома чтобы иметь возможность произвольно изменять количество отсчетов сигнала как в сторону увеличения, так и сторону уменьшения. По сравнению с существующим способом изменения частоты дискретизации, когда необходимо сгладить сигнал и повторить процедуру дискретизации, предлагаемый подход отличается простотой и быстродействием.

Глава II. Разработка спектральных методов согласование частоты дискретизации

1 Построение алгоритма полиномиального представление сигналов

Для того чтобы изменить частоту дискретизации сигнала, сначала надо построить математическую модель сигнала из отсчетов. Нахождение математического моделя всего сигнала, процесс очень трудоёмкий и тяжелый. По этой причине сигнал разбивается на фрагменты с фиксированной длиной. Математическую модель фрагмент сигнала можно представить в виде алгебраического полинома. Такой подход был представлен в первой главе на примере метода наименьших квадратов, блок-схема этого процесса представлена на рис. 2.1

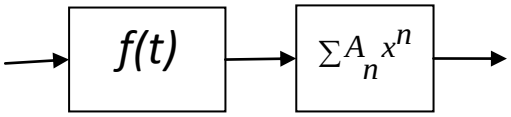


Рис. 2.1. Блок-схема перевода сигнала в полиномиальную форму с помощью метода наименьших квадратов

Как было показано в первой главе при переводе сигнала в полиномиальную форму с помощью метода наименьших квадратов, вычисление полиномиальных коэффициентов становится очень тяжёлой процедурой.

В предлагаемом алгоритме изменения частоты дискретизации сигнала, берётся 2^n отсчетов исходного дискретного сигнала, далее находится математическая модель сигнала для этого фрагмента в виде полиномиального выражения общего вид:

$$f(t) = \sum_{k=0}^{m} A_k t^k \qquad (2.1)$$

где, m- степень полинома

С помощью полученного выражение (2.1) можно восстановить сигнал с нужным количеством отсчетов в соответствующем фрагменте сигнала.

В качестве основы для перевода сигнала из его обычного представления в виде последовательности отсчетов $f(t)$ в спектр сигнала берётся одна из базисных функции преобразования Фурье. При этом блок-схема перевода сигнала в полиномиальную форму будит иметь структуру показанной на рис. 2.2.

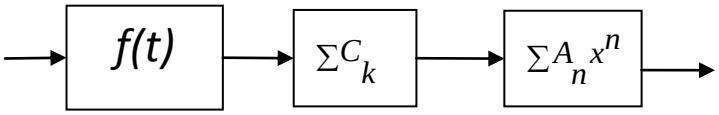


Рис. 2.2. Блок-схема перевода сигнала в полиномиальную форму с помощью спектрального подхода.

Для представления фрагментов звуковых сигналов (при кусочно-полиномиальной аппроксимации) на практике достаточно выбрать m=1,2,3, что представляет выходные сигналы полиномами не выше третьей степени.

2 Выбор базисной системы функции для аппроксимирующего полинома

Сегодня спектральный анализ широко используется в разных областях, таких как радио и телевещание, мобильный связи, при обработке и передачи цифровых сигналов, изображений, а также видео изображений. Разработаны, многие базисной системы Фурье преобразований, таких как дискретное Фурье преобразование, дискретное косинусное преобразование, Вейвлет преобразование, преобразование Уоша-Адамара, преобразование Хаара, для каждого преобразование разработаны быстрые алгоритмы прямого преобразование. От поставленной задачи выберается та или иная система базиса. Существует множество других преобразований. Преобразование Винограда Фурье и алгоритм

первоначального множителя представляют собой оригинальные, но слишком сложные методы повышения скорости вычисления БПФ. Дискретное косинус преобразование используются для ускорения вычисления корреляции и свертки, а также в спектральном анализе, эти методы особенно подходит применяются еще и для сжатия данных при, например, передаче речи или видеосигналов, а также для записи медицинских сигналов, таких как сигналы ЭКГ или ЭЭГ. А также, они используются при распознавании шаблонов. При преобразовании Уолша сигнал раскладывается на прямоугольные импульсы, а не на синусоиды, и оно вычисляется быстрее, чем БПФ. Преобразование Адамара, построенное с помощью перестановки последовательности Уолша, вычисляется еще быстрее. Хотя для некоторых целей преобразования Уолша и Адамара позволяют получить определенные преимущества, они имеют ряд недостатков, которые ограничивают область их применения, преобразование Хаара особенно полезно для определения краев при обработке изображений и в подобных приложениях [13].

Система Фурье хорошо подходит для обработки гармонических сигналов колебательного характера. Для сигналов с другими свойствами – негармонических, пологих, имеющих характер трендов, с нарушениями стационарности, а также графических изображений (графики, фото) применяются дискретные преобразования с другими типами и формами базисных функций. К такому типу относятся базисы Адамара, Хаара, вейвлет - функции и многие другие аналогичные базисные системы. [14]

При выборе той или иной базисной системы учитывается сходство параметров базиса и исходного сигнала. Различают базисы с локальными свойствами (вейвлет-функции и Хаара) и базисы с интегральными свойствами (Адамара и Фурье). Еще одним критерием выбора базисной системы является вычислительная сложность алгоритма преобразования, влияющая на скорость обработки и необходимые аппаратные ресурсы.

В качестве базисной системы выберем систему функций Уолша-Адамара. Преобразование Адамара не требует больших вычислительных ресурсов. В основе лежит понятие матрицы Адамара.[15]

В отличие от базиса Фурье здесь нет умножения на значения Sin или Cos, так как базисные функции имеют вид +1 или -1, т.е. отсчеты сигнала умножаются на единицу, фактически выполняется операция присвоения знака.

Матрица ядра этого преобразование содержит целочисленные коэффициенты из множества {-1; +1}. Очевидно, что при выполнении подобных преобразований существенно сокращается объем вычислений за счет исключения умножения в каждой базовой операции. Матрица ядра преобразования Уолша - Адамара для $N = 2^m$ может быть описана как результат кронекеровского произведения m матриц дискретных экспоненциальных функций (ДЭФ) E_2 размера 2x2 :

$$A_N = A_{2^m} = E_2 \otimes E_2 \otimes E_2 \otimes \ldots \otimes E_2 = \left[E_2\right]^m = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}^m$$

где символ $\otimes$ - операция кронекеровского умножения векторов, в результате чего порождается матрица блочной структуры. Операция кронекеровского умножения двух матриц и состоит в получении блочной матрицы, блоками которой является умноженная на соответствующий элемент правой матрицы левая матрица, т.е.:

$$A_4 = E_2 \otimes E_2 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \otimes \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} = \left|\begin{array}{cc:cc} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ \hdashline 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{array}\right|$$

аналогично можно получить и для N = 8:

$$A_8 = E_2 \otimes E_2 \otimes E_2 = A_4 \otimes E_2 = \left|\begin{array}{cccccccc} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \end{array}\right|$$

Матрица ядра Адамара обладает следующими свойствами:

1) цикличностью $a_{N+k} = a_k$; $a_{N-k} = a_{-k}$
2) мультипликативностью $a_{k+l} = a_k * a_l$
3) симметричностью $A_N = A_N^T$

Разложение сигнала в спектр реализуется аналогично базису Фурье, т.е. попарным перемножением отсчетов сигнала и базисных функций.[16] Формулы прямого и обратного преобразований имеют вид:

$$c_k = \frac{1}{N}\sum f(n)H(n)$$
$$f(n) = \sum c_k H_k \qquad (2.2)$$

где c_k- спектральные коэффициенты, $f(n)$- отсчеты сигнала, $H(n)$ матрица базисной системы..

Количество спектральных коэффициентов меньше числа отсчетов сигнала из-за эффекта сходимости, многие коэффициенты равны или близки к нулю.

Алгоритм прямого преобразования и перевода сигнала от последовательных отсчетов в спектр имеет вид

$$\left|\begin{array}{cccc} H_{0,0} & H_{0,1} & \cdots & H_{0,N-1} \\ H_{1,0} & H_{1,1} & \cdots & H_{1,N-1} \\ \vdots & \vdots & \vdots & \vdots \\ H_{N-1,0} & H_{N-1,1} & \cdots & H_{N-1,N-1} \end{array}\right| \cdot \left|\begin{array}{c} f_0 \\ f_1 \\ \vdots \\ f_{N-1} \end{array}\right| = \left|\begin{array}{c} c_0 \\ c_1 \\ \vdots \\ c_{N-1} \end{array}\right| \qquad (2.3)$$

где, N – число отсчетов сигнала ($N = 2^n$, $n = 1,2,3,\ldots$),

$H_{N-1,N-1}$ - элементы матрицы базисной системы Адамара;

$f_j = f(t_j)$ - отсчеты входного сигнала ($j = 0, 1, 2,\ldots, N\text{-}1$);

c_i - спектральные коэффициенты ($i = 0, 1, 2,\ldots, N\text{-}1$).

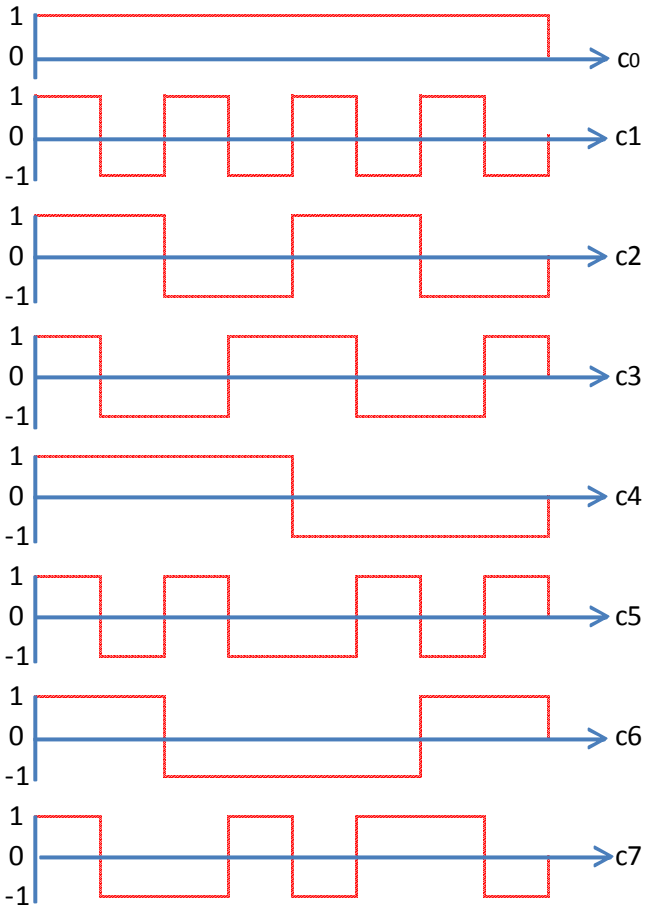


Рис. 2.3. Базисные функции Адамара для $N = 8$

3. Разработка общего алгоритма перевода сигнала в полиномиальную форму

Сформируем равенство, в котором преобразование массива отсчетов сигнала $f(t)$ в спектр по базису H приравнивается разложению алгебраического полинома A_k по тому же базису:

$$\sum_{j=1}^{N-1}\left(\sum_{k=0}^{m} A_k t^k\right) H_{ij} = \sum_{j=0}^{N-1} f(t_j) H_{ij} \qquad (2.4)$$

где i=0, 1, 2,…,N-1; $t \in [0,1]$.

Алгебраический полином второй степени имеет следующий вид:

$$\sum_{k=0}^{2} A_k t^k = A_0 + A_1 t + A_2 t^2 \tag{2.5}$$

Алгебраический полином третьей степени имеет следующий вид:

$$\sum_{k=0}^{3} A_k t^k = A_0 + A_1 t + A_2 t^2 + A_3 t^3 \tag{2.6}$$

Правая часть равенства (2.6) приравняется спектральным коэффициентам базиса Адамара. Выражение (2.6) можно написать следующим виде:

$$\sum_{j=1}^{N-1}\left(\sum_{k=0}^{m} A_k t^k\right) H_{ij} = c_j \tag{2.7}$$

Полученное выражение (2.9) можно представить в виде системы уравнений, связывающей спектральные коэффициенты входного сигнала с коэффициентами алгебраического полинома:

Для второй степени

$$\begin{cases} 8A_0 + 3.5A_1 + 2.1575A_2 = c_0 \\ -0.5A_1 + 0.4375A_2 = c_1 \\ -1.25A_1 + 0.875A_2 = c_2 \\ 0.125A_2 = c_3 \\ -2A_1 + 1.75A_2 = c_4 \\ 0.25A_2 = c_5 \\ 0.5A_2 = c_6 \end{cases} \tag{2.8}$$

Для третьей степени

$$\begin{cases} 8A_0 + 3.5A_1 + 2.1575A_2 + 1.531A_3 = c_0 \\ -0.5A_1 + 0.4375A_2 + 0.4063A_3 = c_1 \\ -1.25A_1 + 0.875A_2 + 0.7891A = c_2 \\ 0.125A_2 + 0.1614A_3 = c_3 \\ -2A_1 + 1.75A_2 + 1.391A_3 = c_4 \\ 0.25A_2 + 0.3281A_3 = c_5 \\ 0.5A_2 + 0.6563A_3 = c_6 \\ -0.09375A_3 = c_7 \end{cases} \tag{2.9}$$

Решая систему уравнений можно получить аналитические выражения, где A_k вычисляется как функция от спектра входного сигнала, при этом полиномиальные аппроксимирующие структуры имеют следующий вид.

Для второй степени:

$$A_0 = \frac{7}{32}(c_6 + c_4) + \frac{1}{8}c_0$$
$$A_1 = -\frac{4}{7}c_6 - \frac{1}{2}c_4 \qquad (2.10)$$
$$A_2 = 2c_6$$

Для третьей степени:

$$A_0 = -\frac{571}{1746}c_7 + \frac{7}{32}(c_6 + c_4) + \frac{1}{8}c_0$$
$$A_1 = \frac{3774}{781}c_7 - \frac{4}{7}c_6 - \frac{1}{2}c_4 \qquad (2.11)$$
$$A_2 = -\frac{13133}{938}c_7 + 2c_6$$
$$A_3 = \frac{32}{3}c_7$$

Получились достаточно простые аналитические связи между спектральными коэффициентами c_i и коэффициентами аппроксимирующего полинома A_k. Отметим, что из 8 возможных спектральных коэффициентов c_i (N=0÷7) в аппроксимирующей структуре присутствуют только четыре: c_0, c_4, c_6, c_7.

Поскольку для нахождение коэффициентов аппроксимирующего полинома достаточно всего четырех спектральных коэффициентов, нет необходимости вычислять остальных спектральных коэффициентов (c_1, c_2, c_3, c_5).

Проверим разработанный алгоритм:

Возьмем 8 отсчетов функции *y=ln(x+1)* в интервале [0; 7/8],

Исходныё отсчеты:

N	1	2	3	4	5	6	7	8
y	0	0.1177830	0.2231435	0.3184537	0.4054651	0.4855078	0.5596158	0.6286086

С помощью разработанного алгоритма найдём коэффициенты аппроксимирующего полинома для второй степени.

a0 = 0.004052

a1 = 0.931393

a2 = -0.253125

Значит аппроксимирующий полином второй степени имеет следующий вид:

$$y`(x)= 0.004052 +0.931393 *x-0.253125*x^2$$

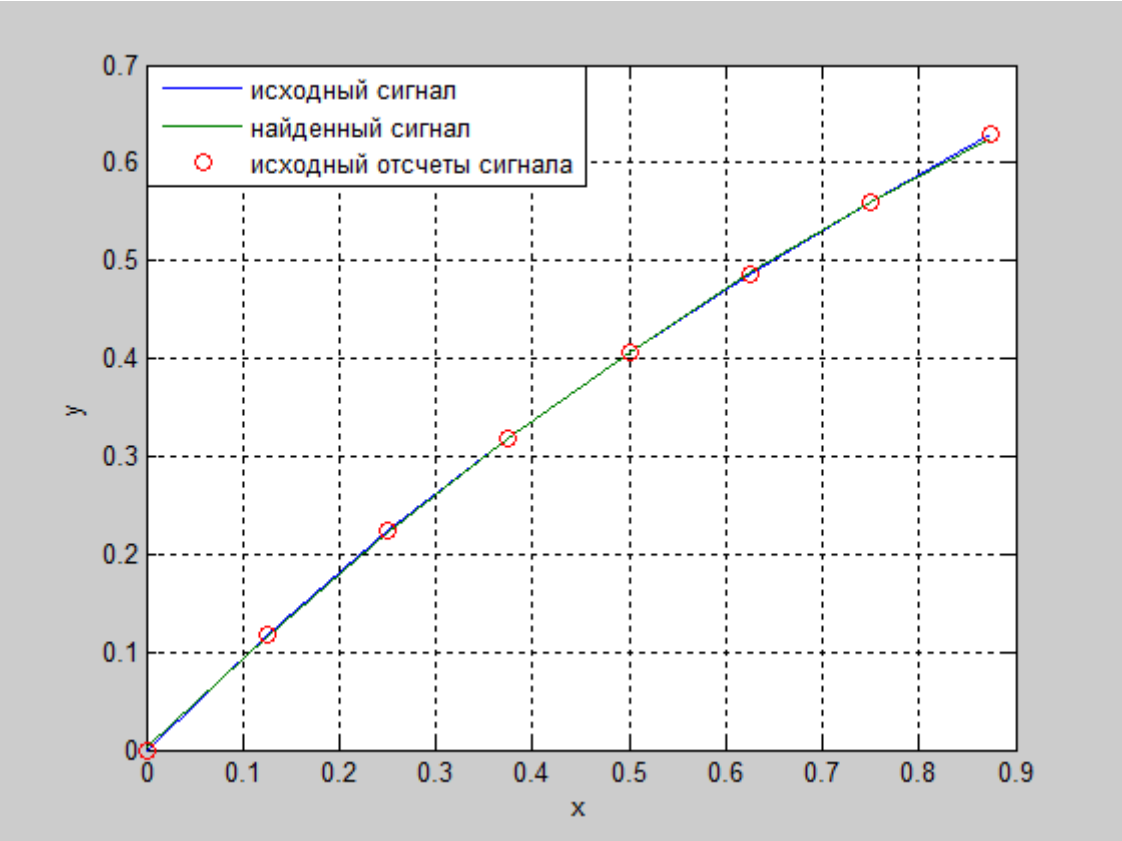


рис.2.4. Иллюстрация аппроксимирующий полином второй степени для функции *y=ln(x+1)*

Возьмем 8 отсчетов функции *y=ln(x+1)* в интервале [0; 7/8],

Исходныё отсчеты:

N	1	2	3	4	5	6	7	8
y	0	0.1177830	0.2231435	0.3184537	0.4054651	0.4855078	0.5596158	0.6286086

С помощью разработанного алгоритма найдём коэффициенты аппроксимирующего полинома для третьей степени.

a0 = 0.000320

a1 = 0.986533

a2 = -0.413048

a3 = 0.121716

Значит аппроксимирующий полином третьей степени имеет следующий вид:

$$y`(x)= 0.000320 +0.986533*x--0.413048 *x^2+0.121716*x^3$$

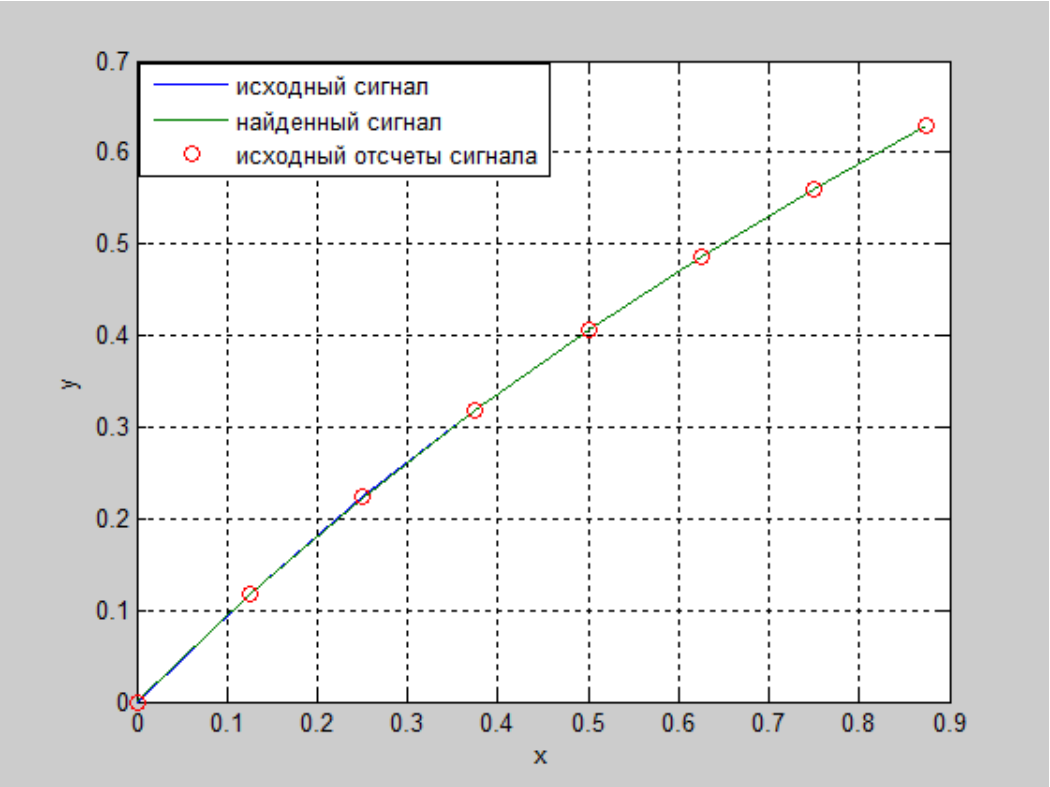


Рис.2.5. Иллюстрация аппроксимирующий полином второй степени для функции *y=ln(x+1)*

В таблицах 2.1 и 2.2.приведены среднеквадратические ошибки аппроксимации функциональных зависимостей и экспериментальных данных

Таблица 2.1

Функциональные зависимости (степень полинома k=2)	Среднеквадратические ошибки	
	МНК (%)	Пол. Мет. (%)
ln(x+1)	0,005795	0,006231
e^x	0,012459	0,013411
*sin(x*π)*	0,056695	0,061
*cos(x*π)*	0,2134	0,2302

Таблица 2.2

Функциональные зависимости (степень полинома k=3)	Среднеквадратические ошибки	
	МНК (%)	Пол. Мет. (%)
ln(x+1)	0,0006738	0,000728
e^x	0,0006778	0,00084
*sin(x*π)*	0,0376258	0,040295
*cos(x*π)*	0,008853	0,013237

Выводы по Главе II

По результатам исследований во второй главе можно сделать следующие выводы:

1. Разработан алгоритм вычисления коэффициентов аппроксимирующих полиномов. Предлагаемый подход базируются на представлении входного сигнала не в виде последовательности отсчетов, а в виде спектров в базисной системе Адамара.
2. Разработанный алгоритм состоит в решении системы алгебраических уравнений, на его базе предложены структуры вычисления коэффициентов полиномов.

3. Получены оценки точности выполнения аппроксимирующих выражений, которые показывают
4. Для разработанного алгоритма реализованы контрольные примеры (экспериментальные функции) для оценки возникающих при обработке погрешностей.

Глава III. Разработка программных средств согласование частоты дискретизации

1. Характеристик современных средств записи и хранения звуковой информации

На основа записи хранение и воспроизведение цифрового сигнала на компьютере лежит звуковая карта.

Звуковая карта (звуковая плата, аудиокарта; англ. sound card) — дополнительное оборудование персонального компьютера, позволяющее обрабатывать звук (выводить на акустические системы и/или записывать). На момент появления звуковые платы представляли собой отдельные карты расширения, устанавливаемые в соответствующий слот. В современных материнских платах представлены в виде интегрированного в материнскую плату аппаратного кодека (согласно спецификации Intel AC’97 или Intel HD Audio).

Для того чтобы понять принцип работы звуковой карты рассмотрим следующую схему.

Рис. 3.1. Упрощенная схема устройства звуковой карты

Звуковой сигнал с микрофона или плеера подается на один из входов звуковой карты. Это аналоговый сигнал. Он поступает на входной микшер, который служит для смешивания сигналов, если их поступает на вход несколько. Затем сигнал с входного микшера поступает на аналого-цифровой преобразователь (АЦП), с помощью которого происходит оцифровка аналогового сигнала, т.е. преобразование его в дискретный двоичный сигнал. Потом цифровые данные поступают в сердце звуковой платы - процессор (DSP - Digital Signal Processor). Этот процессор управляет обменом данными с компьютером через шину PCI материнской платы. Когда центральный процессор компьютера выполняет программу записи звука, то цифровые данные поступают через шину PCI либо прямо на жесткий диск, либо в оперативную память компьютера. Присвоив этим данным имя, мы получим звуковой файл.

При воспроизведении этого звукового файла данные с жесткого диска через шину PCI поступают в сигнальный процессор звуковой платы, который направляет их на цифро-аналоговый преобразователь (ЦАП). Цифро-аналоговый преобразователь преобразует двоичный сигнал в аналоговый. Электрический сигнал, получившийся в результате преобразования, поступает на выходной микшер. Этот микшер идентичен входному и управляется при помощи той же самой программы. Сигнал с выходного микшера поступает на линейный выход звуковой карты и выход на звуковые колонки, подключив к которому колонки или наушники мы слышим звук[17, 18].

Интегрированная аудиоподсистема AC’97

AC’97 (сокращенно от англ. audio codec '97) — это стандарт для аудиокодеков, разработанный подразделением Intel Architecture Labs компании Intel в 1997 г. Этот стандарт используется в основном в системных платах, модемах, звуковых картах и корпусах с аудиорешением передней панели. AC’97 поддерживает частоту дискретизации 96 кГц при использовании 20-разрядного стерео-разрешения и 48 кГц при

использовании 20-разрядного стерео для многоканальной записи и воспроизведения.

AC’97 состоит из встроенного в южный мост чипсета хост-контроллера и расположенного на плате аудиокодека. Хост-контроллер (он же цифровой контроллер, DC’97; англ. digit controller) отвечает за обмен цифровыми данными между системной шиной и аналоговым кодеком. Аналоговый кодек — это небольшой чип (4×4 мм, корпус TSOP, 48 выводов), который осуществляет аналогоцифровое и цифроаналоговое преобразования в режиме программной передачи или по DMA. Состоит из узла, непосредственно выполняющего преобразования — АЦП/ЦАП (аналоговоцифровой преобразователь / цифроаналоговый преобразователь; англ. analog digital converter / digital analog converter, сокр. ADC/DAC). От качества применяемого АЦП/ЦАП во многом зависит качество оцифровки и декодирования цифрового звука.

HD Audio

HD Audio (от англ. high definition audio — звук высокой четкости) является эволюционным продолжением спецификации AC’97, предложенным компанией Intel в 2004 году, обеспечивающим воспроизведение большего количества каналов с более высоким качеством звука, чем при использовании интегрированных аудиокодеков AC’97. Аппаратные средства, основанные на HD Audio, поддерживают 24-разрядное качество звучания (до 192 кГц в стереорежиме, до 96 кГц в многоканальном режимах — до 8 каналов).

Формфактор кодеков и передачи информации между их элементами остался прежним. Изменилось только качество микросхем и подход к обработке звука.

SRC

Преобразователи частоты семплирования (SRC, Sample Rate Convertors) задействуются в тех случаях, когда частота семпла не совпадает с опорной частотой текущего режима (иначе семпл проиграется

с неправильной скоростью и изменит тон). Используется 256 интерполяторов, с качеством выше, чем у типичного полифазного КИХ-фильтра 100-го порядка (FIR, finite impulse response = КИХ, конечная импульсная характеристика). Диапазон сдвига тона от 0 до 8.

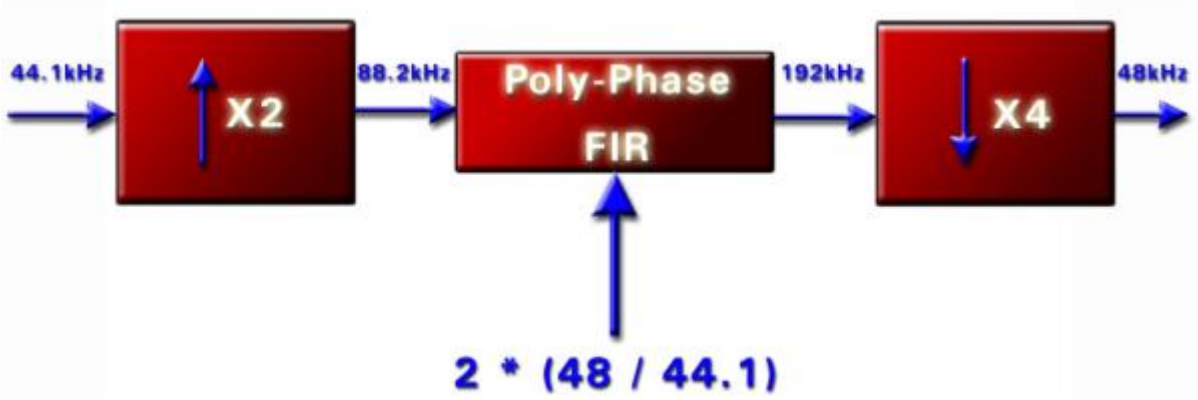


Рис.3.2. Схема работы SRC X-Fi

Картинка иллюстрирует работу SRC X-Fi по конвертации 44.1 кГц в 48 кГц. Сперва отсчеты сигнала удваиваются до 88.2 кГц. Полифазный КИХ фильтр конвертирует сигнал до 192 кГц с коэффициентом, равным удвоенному отношению 48/44.1. В финальной стадии частота уменьшается до 48 кГц. Такая схема более эффективна с точки зрения вычислительных затрат и даёт более качественный результат, так как на последнем этапе не образуется алиазинг из-за некратных частот.

При конвертации стандартного тона 997 Гц из 44.1 кГц в 48 кГц стандартный параметр, отвечающий за мощность шумов и искажений, THD+N равен -136 дБ, дрожание частоты в полосе пропускания ±0.00025 дБ. Проверять эти параметры Creative рекомендует как минимум с помощью измерительной станции AudioPrecision.

Учитывая запас по перегрузке (headroom) в 6 дБ и 32 стадии цикла обработки, THD+N может ухудшиться до –124 дБ, а неравномерность АЧХ возрастёт до ±0,01 дБ. Но параметры даже самых лучших на сегодня ЦАПов по шумам и искажениям уступают таким показателям.

Следует лишь учесть, что параметры качества в частотной области не показывают работу фильтров во временн ом домене. Но, учитывая позиционирование карт в первую очередь на игры и фильмы, такую точность преобразования можно считать великолепной. В прошлых изделиях интермодуляционные искажения от SRC доходили до 0,1% в высокочастотной области, при этом на падение качества жаловались лишь единичные пользователи.

Имеется очень важный момент, который не сильно афишируется производителем — в отличие от звуковых процессоров предыдущего поколения, в X-Fi можно отключить передискретизацию вместе с остальными эффектами DSP! При этом, конечно, теряется уникальная возможность одновременного прослушивания нескольких сигналов разных частот дискретизации, с последующим наложением всяческих эффектов и улучшайзеров. Однако поклонникам максимально качественного саунда подобное нужно меньше всего.

Преобразователи SRC работают также в режиме DMA, позволяя производить преобразования прямо в ОЗУ компьютера без участия ЦПУ. Для исключения коллизий при загруженной шине PCI имеется настраиваемый кэш.

2. Разработка метода изменения частоты дискретизации аудио сигналов

На сегодняшний день широко используются две способы изменения частоты дискретизации, которые приведены в первой главе. Первый способ изменения частоты дискретизации основывается на преобразовании дискретного сигнала в аналоговую форму, затем заново дискретизировать аналоговый сигнал с нужной частотой дискретизаций. При такой подходе, расходывается много времени за счет этого такой подход не удовлетворяет требованием систем реального времени. Второй подход изменения частоты дискретизации на целое число раз. Недостатком данного метода является

необходимость фильтрации сигнала на повышенной в M раз частоте дискретизации, что требует значительных вычислительных ресурсов. При этом соответствующая частота может во много раз превосходить как исходную, так и окончательную частоту передискретизации, особенно если M и N — близкие большие числа. Так, например, при передискретизации звукового сигнала с 44100 Гц до 48000 Гц этим методом необходимо увеличить частоту дискретизации в 160 раз до 7056000 Гц и затем уменьшить её в 147 раз до 48000 Гц. Таким образом, в данном примере вычисления приходится производить на частоте дискретизации более 7 МГц.

В отличие от традиционных методов разрабатываемый метод изменения частоты дискретизации, не требует больших вычислительных ресурсов, задержка сигнала чуть больше одной секунды, метод легко реализуется на аппаратном уровне.

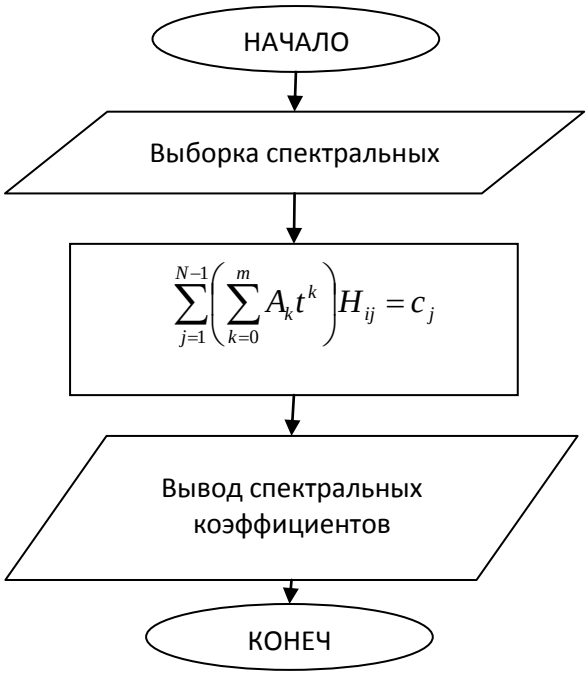


Рис. 3.3. Алгоритм вычисления спектральных коэффициентов входного аудио сигнала.

Во второй главе бил разработан алгоритм вычисления коэффициентов аппроксимирующих полиномов. Предлагаемый подход

базируются на представлении входного сигнала не в виде последовательности отсчетов, а в виде спектров в базисной системе Адамара. Модуль вычисление коэффициентов аппроксимирующего полинома показано на Рис. 3.3

Разработанный алгоритм состоит в решении системы алгебраических уравнений, на его базе предложен структуры вычисления коэффициентов полиномов. Коэффициенты алгебраического полинома вычисляется по формуле 2.11. Алгоритм вычисления алгебраического полинома показано на рисунке 3.4.

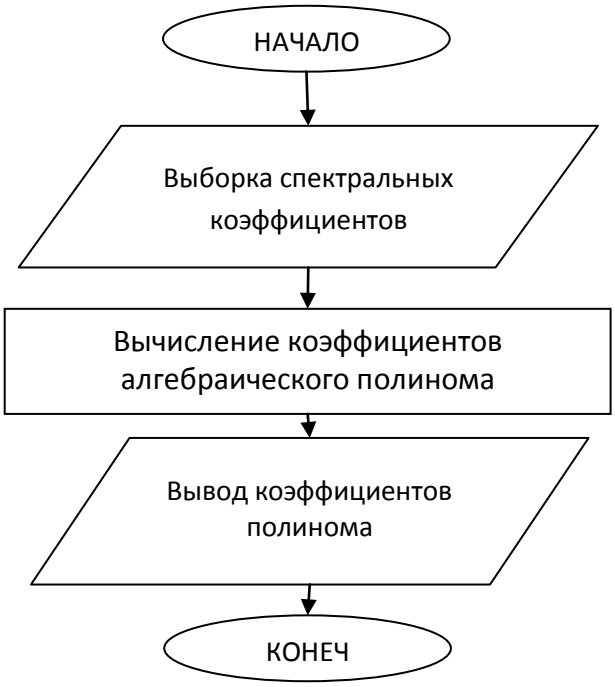


Рис. 3.4. Алгоритм вычисления коэффициентов алгебраического полинома.

Следующий вопрос, которого надо решить, заключается в следующем: разделение аудио сигнала на фиксированные фрагменты. В каждом фрагменте должно быть по 8 отсчетов, так как разработанный алгоритм, перевода сигнала в полиномиальную форму, имеет такое ограничение. Поэтому в цикле берутся каждый 8 отсчетов, вычисляются спектральные коэффициенты по этим отсчетам и затем, вычисляются полиномиальные коэффициенты аппроксимирующего полинома для этого

фрагмента. Все это повториться для следующих восьми точек отсчета. И это процесс будет продолжаться пока не вычисляться полиномиальные коэффициенты аппроксимирующего полинома, для всех фрагментов, число повторений равно *количество_отсчетов_в_сигнале/8*. Это процедура необходимо для того чтобы разделить вес сигнал на фрагменты. В каждом фрагменте будут участвовать восемь отсчетов входного сигнала. Алгоритм реализации этой проблемы представлена на Рис.3.6.

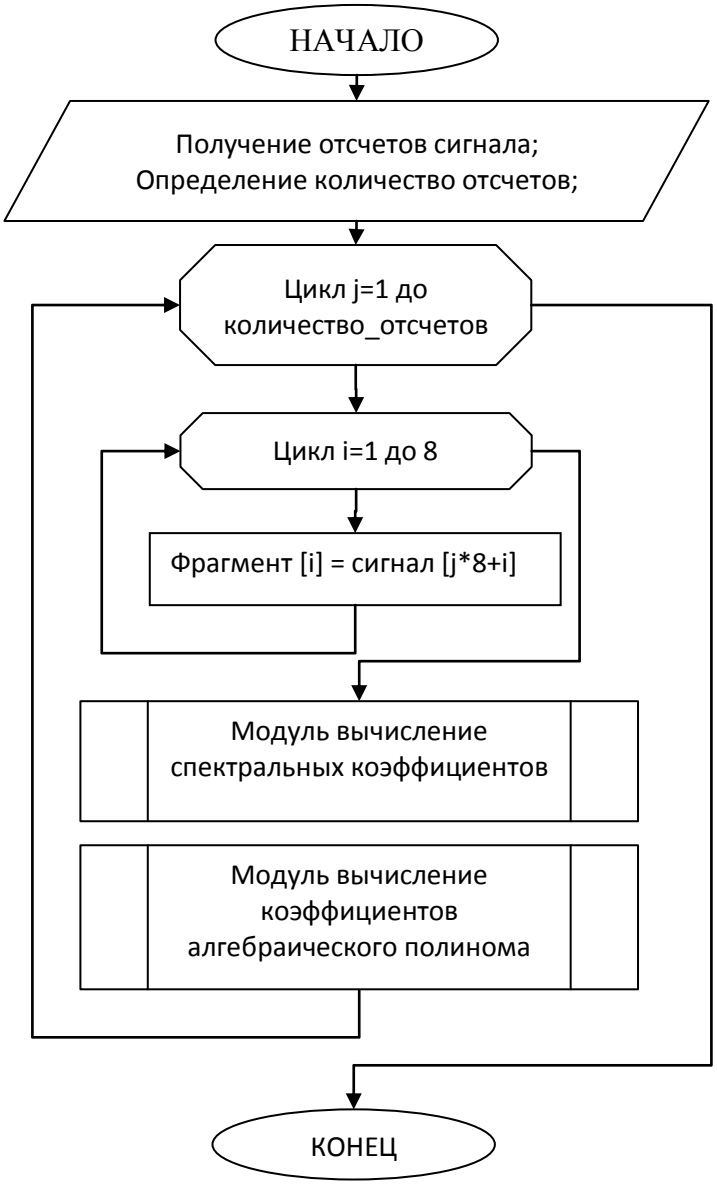


Рис. 3.5. Алгоритм фрагментации аудиосигнала для формулирования аппроксимирующего полинома

Если продолжительность аудио сигнала меньше одной секунды считывается отсчёты аудиосигнал, и записываются в память для дальнейшей обработки, если продолжительность аудиосигнала больше одной секунды, в память записывается отсчеты текущей секунды, и обработка происходить с отсчетами одной секунды, после обработки текучего секунды аудиосигнала, результаты обработки записываются на выходной файл. Такой подход необходим при работе системах реального времени. После записи результатов обработки, выбираются следующие отсчеты в секунде. Этот процесс продолжается до окончание аудиосигнала. Количество точек в одноканальном аудиосигнале вычисляется следующем образе:

количество_отсчетов = продолжительность_аудиосигнала * частота_дискретизации_аудиосигнала

Теперь, посмотрим, что мы имеем. У нас есть алгоритм, который делить аудио сигнал на фрагменты по 8 отсчетов каждого, а также алгоритм нахождения коэффициентов аппроксимирующего полинома третьего степени для этих фрагментов. Это означает, что мы теперь сможем заняться изменение частоты дискретизации аудиосигнала.

Увеличение частоты дискретизации назевается интерполяцией, уменьшение частоты дискретизации назевается децимацией. Чтобы изменить частоту дискретизации аудио сигнала мы находим математическую модель сигнала. С помощью математической модели сигнала можно вычислить значение сигнала в любом точке.

На рис. 3.6 приведён алгоритм изменения частоты дискретизации аудисигнала. Алгоритм изменения частоты дискретизации работает следующим образом: для начала модуль фрагментации аудиосигнала разделяет аудиосигнал с продолжительностью не боле одной секунды на фрагментов из 8 отсчетов, затем это процедура вычисляет спектральный коэффициентов этого фрагмента, после этого вычисляются коэффициенты аппроксимирующего полинома для этого фрагмента.

На следующем шаге вычисляются количество фрагментов, а также, шага дискретизации. Они вычисляются по формуле:

количество_фрагментов = длина_аудио_сигнала/8;

шага дискретизации = (количество_фрагментов -0.125) /
новая_частота_дискретизации;

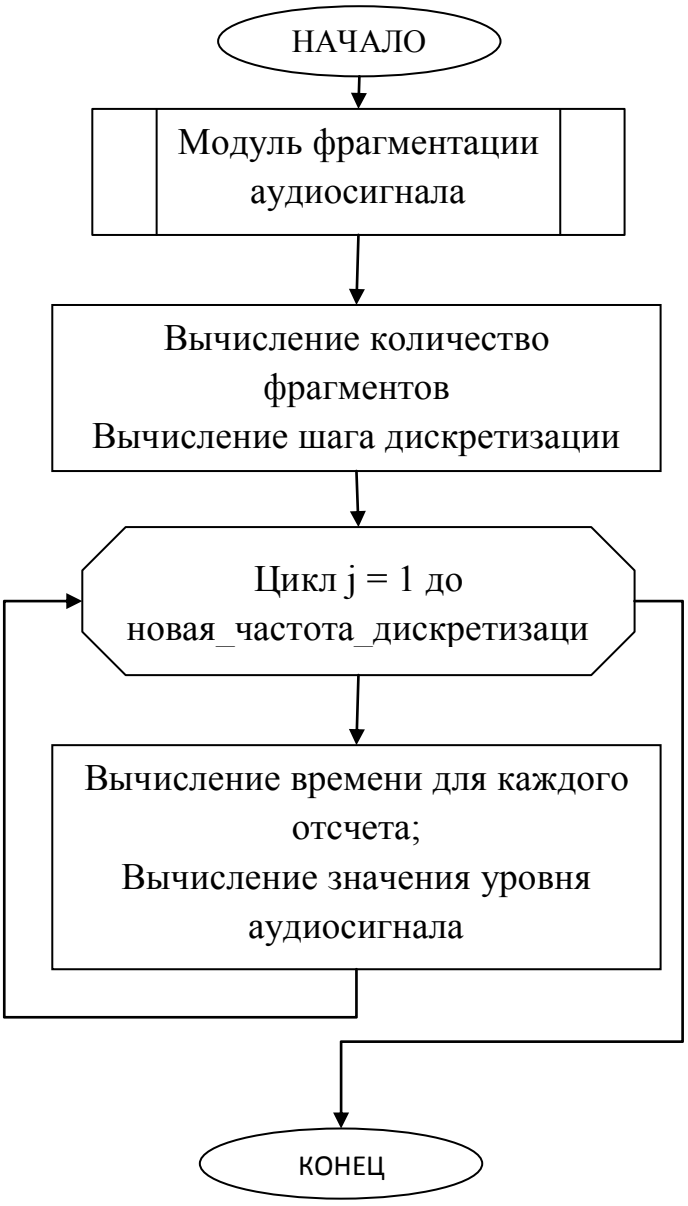


Рис. 3.6. Алгоритм фрагментации аудиосигнала для формулирования аппроксимирующего полинома

И так теперь вычислены все нужные параметры для изменения частоты дискретизации. Открываем цикл до нового частоты дискретизации

здесь, стоит одна проблема, разработанный нам алгоритм рассчитан на вычисления уровня сигнала в интервале от 0 до 1, по этому при вычислении шага берется дробная часть.

3. Разработка программы согласования частоты дискретизации для сигнальных процессоров.

На основе разработанного алгоритма било разработано модельные программы для персональных компьютеров на платформе MatLab и было проверено серия экспериментов, а также была написано программа на языке c++ которая, реализует алгоритм изменения частоты аудиоигналов.

В настоящее время наблюдается непрерывный рост интереса специалистов-разработчиков к использованию в различных устройствах обработки сигналов цифровых сигнальных процессоров. Это обусловлено как удобствами применения, доступностью и широкими возможностями отладочных инструментов самих процессоров. Среда разработки проектов VisualDSP++ позволяет разработчикам разрабатывать и выполнять отладку приложений. Эта среда включает лёгкий в использовании ассемблер, компилятор C/C++ и библиотеку исполняемых функций C/C++, включающую математические функции и функции ПЦОС. Ключевой особенностью средств разработки программного обеспечения является эффективность кода, написанного на языках C/C++

Проанализирован класс процессоров цифровой обработки сигналов и для реализации изменения частоты дискретизации аудиосигналов процессоры фирмы Analog Devices – ADSP Blackfin, отличающиеся развитыми средствами эмуляции и удобством отладки комплекса. Гарвардская архитектура и наличие аппаратно – реализованных специальных команд умножения, параллельного умножения с накоплением дают возможность широкого применения.

Системные требования программы Analog Devices:

- Операционная система - Windows 2000® SP4, Windows XP® SP2 or greater, Windows Vista™ Business edition, Windows Vista™ Enterprise edition or Windows Vista™ Ultimate edition, Windows Seven™ Business edition, Windows Seven ™ Enterprise edition or Windows Seven ™ Ultimate edition.
- Центральный процессор - Intel Pentium® 32-bit, 1 GHz
- Оперативная память - 512 MB RAM
- Свободное пространство на жестком диске - 2 GB

Программа VisualDSP++ имеет очень удобный интерфейс разработки. На рисунке 3.3.1. показано основной вид программы.

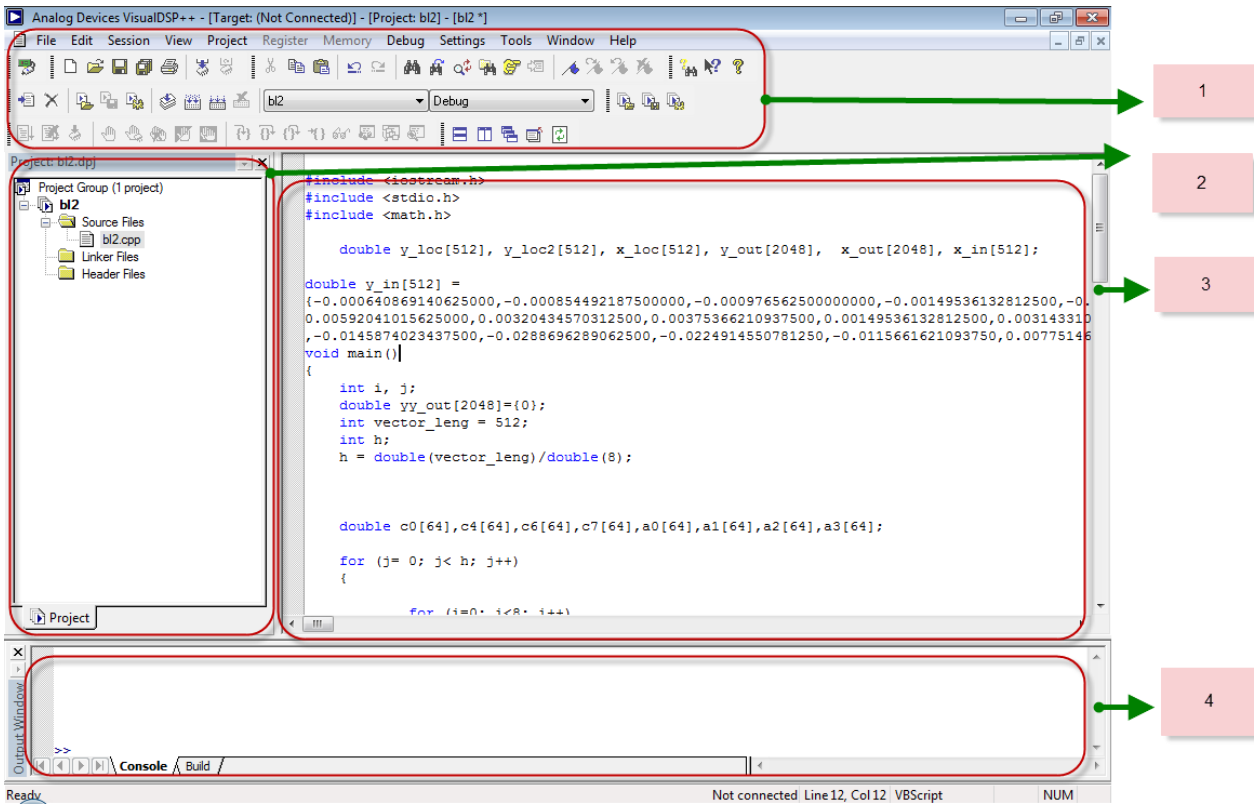


Рис.3.7. Интерфейс программы VisualDSP++.

Здесь 1- панель инструментов и строка меню; 2 - окно управления проектами; 3 – окно для написание кода программы; 4 - окно консоли, в котором отображаются результаты.

В разработанным алгоритме, изменение частоты дискретизации выполняется на произвольном числе точек, то есть можно изменят частоту дискретизации аудиосигнала в дробное число раз. Например, изменить частоту дискретизации аудиосигнала стандарта CD 44,1кГц на DVD 48,0 кГц. Чтобы достичь такого результата, било решено вычислять новые отсчеты сигнала в интервале одной секунды. Такой подход вполне оправдал себя, результаты изменений частоты дискретизации показаны на следующих рисунках. Чтобы посмотреть изменения сигнала нам необходимо построить график аудио сигнала, для этого надо сделать следующие действия, после компиляции программы заходим в меню View→ Debug Windows →Plot→New эти

На рисунке 3.10 показано изменения частоты дискретизации сигнала с 48000 Гц на 44100 Гц. Для простоты в графике показано первые 16 отсчетов исходного сигнала и 15 отсчетов нового сигнала с измененной чистотой дискретизацией. При этом в сигнале с частотой дискретизаций 48000 Гц отсчеты сигнала измеряются на каждой 20,8 мкс, а в сигнале частотой дискретизаций 44100 Гц сигнала измеряются на каждой 22,675 мкс. На рисунке 3.11 показано изменения частоты дискретизации сигнала с 48000 Гц на 44100 Гц. Для простоты в графике показано первые 64 отсчетов исходного сигнала и 36 отсчетов нового сигнала с измененной чистотой дискретизацией.

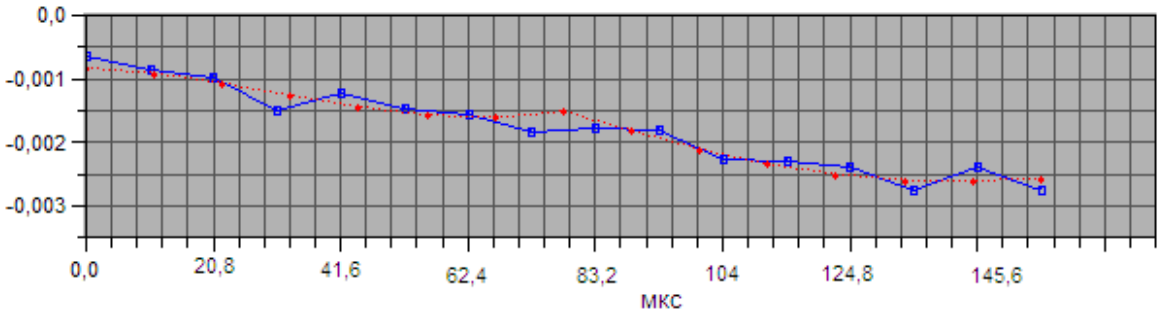


Рис. 3.10. Изменения частоты дискретизации сигнала с частотой дискретизацией 48000 Гц на 44100 Гц в 16 отсчетах.

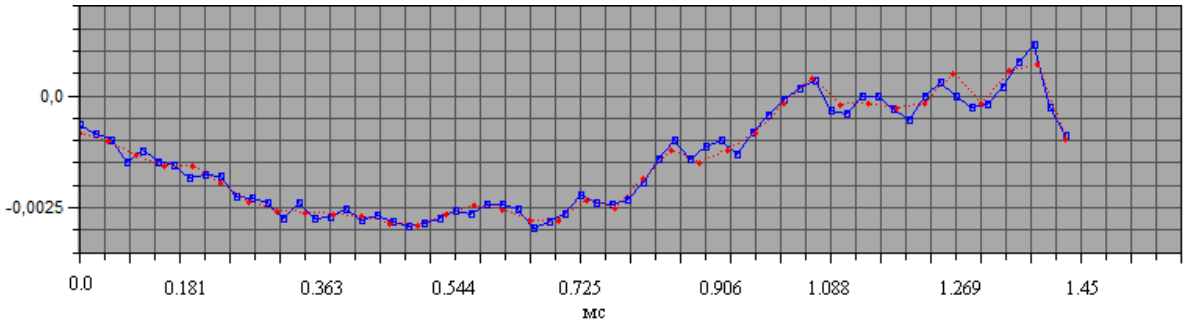


Рис. 3.11. Изменения частоты дискретизации сигнала с частотой дискретизацией 44100 Гц на 24800Гц в 64 отсчетах.

Разработанный алгоритм изменения частоты дискретизации не требует больших вычислительных операции, в отличии от метода изменения частоты дискретизации с целым шагом.

Выводы по Главе III

В этой главе было рассмотрена разработка программных средств согласования частоты дискретизации, в том числе м рассмотрены следующие моменты:

- Характеристики современных средств записи и хранения звуковой информации;
- Разработан метод изменения частоты дискретизации аудио сигналов, который работает с фиксированными фрагментами сигнала.
- На основе разработанного алгоритма, изменения частоты дискретизации аудио сигналов, била разработана программа для сигнальных процессоров компании Analog Device, на платформе VisualDSP++ ;
- Проанализировано эффективность разработанних алгоритмов и программ.

Заключение

Основные результаты работы заключаются в следующем:

1. Исследованы существующие методы цифровой обработки сигналов во временной области и в спектральной области, также выявлены недостатки классических методов аппроксимации сигналов.

2. Сформулирован и обоснован метод перевода сигнала из временной формы в спектральную, а затем в полиномиальную на основе Фурье-базисов.

3. Для выбранных базисных систем поставлены и решены теоретические вопросы поиска коэффициентов полиномов, обладающих свойствами математической корректности, алгоритмической эффективности и простотой реализуемости на сигнальных процессорах.

4. Для реализации разрабатываемых методов и алгоритмов обработки выбраны процессоры семейства Blackfin ADSP BF 525 компании Analog Devices, отличающиеся развитыми средствами эмуляции и удобством отладки комплекса.

5. Получены оценки эффективности разработанных полиномиальных аппроксимирующих структур, проведен анализ их качественных характеристик при обработке сигналов с шумами, выявлены их основные преимущества относительно существующими методами.

Список литературы

1. Каримов И.А.: Постановление Президента Республики Узбекистан «О мерах по дальнейшему внедрению развитию современных информационно-коммуникационных технологий» / Собрание законодательства Республики Узбекистан, 2012 г., № 13, стр. 139. Ташкент, 2012.
2. Закон Республики Узбекистан «Об информатизации»/ Ведомости Олий Мажилиса Республики Узбекистана, 2004 г., № 1-2, ст. 10. Ташкент, 11.12.2003
3. Л. З. Румшинский. Математическая обработка результатов эксперимента. Справочная руководство. Главная редакция физико-математической литературы. – М.: Изд-во Наука, 1971. – 192 с.
4. Осипова, А.П. Практикум по аппроксимации табличных функций различными аналитическими зависимостями с помощью метода наименьших квадратов. – М.: Изд-во МАИ, 1992. – 258 с.
5. Самарский, А.А.Введение в численные методы. – М.: Изд-во Наука,1997. – 352 с.
6. Волков, Е.А. Численные методы. – М.: Изд-во Наука, 1987. – 345 с.
7. И.С. Березин, Н.П. Жидков. Методы вычислений. – М.: Изд-во ФизМатЛит , 1962. – 254с.
8. Дж. Форсайт, М.Мальком, К. Моулер. Машинные методы математических вычислений. – М.: Изд-во "Мир", 1980. – с.
9. Ричард Лайонс Цифровая обработка сигналов: второе издание. — Бином-Пресс, 2006. – 656 с.
10. Л. Рабинер, Б. Гоулд Теория и применение цифровой обработки сигналов. — М.: Мир, — 848 с.
11. Романюк Ю.А. Основы цифровой обработки сигналов. В 3-х ч. Ч.1. Свойства и преобразования дискретных сигналов: Учебное пособие. —М.: МФТИ, 2005.—332 с.

12. А. Б. Сергиенко. Цифровая обработка сигналов. – СПб.: СпецЛит, 2002. – 608 с.
13. Айфичер, Эммануил С. Джервис, Барри У. Цифровая обработка сигналов: практический подход, 2-е издание – М.: Вильямс, 2004. — 992 с.
14. Юкио Сато. Без паники! Цифровая обработка сигналов. – М .: Додэка-XXI, 2010. —176 с.
15. Дахнович, А.А. Дискретные системы и цифровая обработка сигналов: учебное пособие–Тамбов: Изд-во Тамб. гос. техн. ун-та, 2007. – с.
16. А. Оппенгейм, Р. Шафер. Цифровая обработка сигналов. – М .: Техносфера, 2006. – 859 с.

Интернет ресурсы

17. http://ru.wikipedia.org/ (Википедия — свободная энциклопедия)
18. http://intuit.ru/ (Национальном Открытом Университете «ИНТУИТ»)
19. http://matlab.exponenta.ru/ (Сообщество пользователей Matlab и Simulink)
20. http://kit-e.ru/articles/elcomp/2010_1_64.php (Знакомство с семействами процессоров Analog Devices. Введение в VisualDSP++)

ПРИЛОЖЕНИЕ

Листинг кода программы изменения частоты дискретизации на языке программирование C++ на платформе VisualDSP++

diskretlash.cpp

```
// diskretlash.cpp : Defines the entry point for the console application.
#include <iostream.h>
#include <stdio.h>
#include <math.h>
void main()
{
	int i, j;
	double yy_out[512]={0};
	int vector_leng = 15;
	int hh;
	hh = int(double(vector_leng)/double(8));
		double c0[16],c4[16],c6[16],c7[16],a0[16],a1[16],a2[16],a3[16];
	for (j= 0; j< hh+1; j++)
	{
			for (i=0; i<8; i++)
			{
				y_loc[i] = y_in[(j*8+i)];
			}
		// вычисление спектральных коэффициентов
		c0[j] =
y_loc[0]+y_loc[1]+y_loc[2]+y_loc[3]+y_loc[4]+y_loc[5]+y_loc[6]+y_loc[7];
		c4[j] = y_loc[0]+y_loc[1]+y_loc[2]+y_loc[3]-y_loc[4]-y_loc[5]-
y_loc[6]-y_loc[7];
		c6[j] = y_loc[0]+y_loc[1]-y_loc[2]-y_loc[3]-y_loc[4]-
y_loc[5]+y_loc[6]+y_loc[7];
		c7[j] = y_loc[0]-y_loc[1]-y_loc[2]+y_loc[3]-
y_loc[4]+y_loc[5]+y_loc[6]-y_loc[7];
```

```
            cout << c0[j] <<'\n' << c4[j] <<'\n' << c6[j] <<'\n' << c7[j] <<'\n'<<
endl;
                    // вычисление коэффициентов кубического полинома
            a3[j] = -c7[j]/0.09385;
            a2[j] = 2*((c6[j])+c7[j]*7);
            a1[j] = -0.5*(c4[j]+3.5*c6[j]+ 9.6542*c7[j]);
            a0[j] = 0.125*(c0[j]+1.75*(c4[j]+c6[j])+2.6135*c7[j]);
            cout << "a0= "<< a0[j] <<'\n' << "a1= "<< a1[j] <<'\n' << "a2= "<<
a2[j] <<'\n'<< "a3= " << a3[j] <<'\n' << endl;
        }
        for (i= 0; i < ((hh+1)*8); i++)
        {
            x_in[i] = double(i)/double(8);
        }
        int d_n = 14;
        double h = 0;
        int oraliq_soni = int((vector_leng+1)/8);
        double qadam  = (oraliq_soni-0.125)/d_n;
        int nn = 0;
        double xlok=0.0;
        j = 0;
        int k = 0;
        for (i=0;i<d_n+1;i++)
        {
            x_out[i] = h;
            j= int(h);
            xlok = h-j;
            yy_out[i] = a0[j]+a1[j]*xlok+a2[j]*pow(xlok, 2)+
                                                a3[j]*pow(xlok, 3);
```

```
        h = h + qadam;
}
for (j=0;j<h;j++)
{
        ii = 0;
        for (nn = 0; nn < d_n; nn++)
        {

                x_loc[nn] = ii;
                y_loc2[nn] = a0[j]+a1[j]*x_loc[nn]+
                        a2[j]*pow(x_loc[nn], 2)+a3[j]*pow(x_loc[nn], 3);
                cout  << x_loc[nn] << '\t'<< y_loc2[nn] <<endl;
                if (j==0)
                {
                        x_out[nn] = x_loc[nn];
                        yy_out[nn] = y_loc2[nn];
                }
                else
                {
                        x_out[(j*d_n+nn)] = x_loc[nn]+j;
                        yy_out[(j*d_n+nn)] = y_loc2[nn];
                }
                ii = ii + double(1)/double(d_n);

        }
        cout <<endl;

}
for (i=0; i<d_n+1; i++)
{
```

```
                y_out[i] = yy_out[i];
                cout << i+1 <<"  "<< y_out[i] << " \t"<< y_in[i]<<" \t"<<
fabs(y_in[i] - y_out[i]) << endl;
        }
}
```

Листинг кода модельной программы изменения частоты дискретизации на платформе MatLab

wave.m

```
clc
clear
% считывание аудио файла с расширением *.wave
[y, fs, nbits, opts] = wavread('salom.wav'); d_old = fs
n = size(y,1);
% определение длительности файла
duraction = n/d_old
format short;
yy = y(1:n, 1);
% воспроизведение  аудио файла
wavplay(yy, 48000)
for j = 1:fix(n/8)-1
   for i=1:8
      y_loc(i) = yy(((j)*8+i));
   end
   y_loc;
   c0(j) =
y_loc(0+1)+y_loc(1+1)+y_loc(2+1)+y_loc(3+1)+y_loc(4+1)+y_loc(5+1)+y_loc
(6+1)+y_loc(7+1);
        c4(j) = y_loc(0+1)+y_loc(1+1)+y_loc(2+1)+y_loc(3+1)-y_loc(4+1)-
y_loc(5+1)-y_loc(6+1)-y_loc(7+1);
```

```
        c6(j) = y_loc(0+1)+y_loc(1+1)-y_loc(2+1)-y_loc(3+1)-y_loc(4+1)-
y_loc(5+1)+y_loc(6+1)+y_loc(7+1);
        c7(j) = y_loc(0+1)-y_loc(1+1)-y_loc(2+1)+y_loc(3+1)-
y_loc(4+1)+y_loc(5+1)+y_loc(6+1)-y_loc(7+1);



        %высисления коэффитцентов кубического полинома
        a3(j) = -c7(j)/0.09385;
        a2(j) = 2*((c6(j))+c7(j)*7);
        a1(j) = -0.5*(c4(j)+3.5*c6(j)+ 9.6542*c7(j));
        a0(j) = 0.125*(c0(j)+1.75*(c4(j)+c6(j))+2.6135*c7(j));
end
d_n = 44100
oraliq_soni = fix(n/8)-1;
qadam = (oraliq_soni-0.125)/(d_n*duraction);
h=0;
if duraction > 1
   for t =1:fix(duraction)
      for i=1:d_n*duraction
       xout(i) = h;
       j = fix(h);
       xlok = h-j;
       yy_out(t+i) = a0(j+1)+a1(j+1).*xlok+a2(j+1).*xlok.^2 + a3(j+1)*xlok.^3;
       h = h+ qadam;
      end
   end
   ennd = i+t;
else
   for i=1:d_n*duraction
       xout(i) = h;
```

```
        j = fix(h);
        xlok = h-j;
        yy_out(i) = a0(j+1)+a1(j+1).*xlok+a2(j+1).*xlok.^2 + a3(j+1)*xlok.^3;
        h = h+ qadam;
    end
end
figure
plot(yy)
figure
plot(yy_out)
wavplay(yy_out, d_n)
```

Листинг кода модельной программы изменения частоты дискретизации на платформе Microsoft Visual Studio 2010 на языке програмирование C#

Form1.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.IO;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using Alvas.Audio;

namespace AudioResampling
{
    public partial class Form1 : Form
    {
        public Form1()
```

```
{
    InitializeComponent();
}
private void button1_Click(object sender, EventArgs e)
{
    if (openFileDialog1.ShowDialog() == DialogResult.OK)
    {
        textBox1.Text = openFileDialog1.FileName;
        if (File.Exists(textBox1.Text))
        {
            richTextBox1.Text = GetFileInfo(textBox1.Text);
        }
        else
        {
            richTextBox1.Text = "Файл не найден !";
        }
    }
}
private void button2_Click(object sender, EventArgs e)
{
    if (saveFileDialog1.ShowDialog() == DialogResult.OK)
    {
        txtOutput.Text = saveFileDialog1.FileName;
    }
}
private string GetFileInfo(string fileName)
{
    var sb = new StringBuilder();
    WaveReader wr = new WaveReader(File.OpenRead(fileName));
    IntPtr format = wr.ReadFormat();
```

```
            byte[] data = wr.ReadData();
            wr.Close();
            var fd = AudioCompressionManager.GetFormatDetails(format, true);
            WaveFormat wf =
AudioCompressionManager.GetWaveFormat(fd.FormatHandle);

            sb.AppendFormat("Format: [{0}], \nFormatTag: {1}, \nChannels: {2},
\nSamplesPerSec: {3}, \nBitsPerSample: {4}, \nBlockAlign: {5},
\nAvgBytesPerSec: {6}\n\n",
                fd,
                wf.wFormatTag,
                wf.nChannels,
                wf.nSamplesPerSec,
                wf.wBitsPerSample,
                wf.nBlockAlign,
                wf.nAvgBytesPerSec);
            return sb.ToString();
        }
        private void ConvertWav()
        {
            var fileName = textBox1.Text;
            IntPtr formatNew = AudioCompressionManager.GetPcmFormat(2, 16,
Convert.ToInt32(txtSamples.Text));
            string fileNameNew = txtOutput.Text;
            WaveReader wr = new WaveReader(File.OpenRead(fileName));
            IntPtr format = wr.ReadFormat();
            byte[] data = wr.ReadData();
            wr.Close();
            byte[] dataNew = AudioCompressionManager.Convert(format,
formatNew, data, false);
```

```
            WaveWriter ww = new WaveWriter(File.Create(fileNameNew),
                AudioCompressionManager.FormatBytes(formatNew));
            ww.WriteData(dataNew);
            ww.Close();
        }
        private void button3_Click(object sender, EventArgs e)
        {
            if (string.IsNullOrWhiteSpace(txtOutput.Text))
            {
                MessageBox.Show("Пожалуйста, выберите выходной файл !");
            }
            else
            {
                ConvertWav();
            }
        }
        }
}
```

Form1.Designer.cs

```
namespace AudioResampling
{
   partial class Form1
   {
      private System.ComponentModel.IContainer components = null;
disposed; otherwise, false.</param>
      protected override void Dispose(bool disposing)
      {
         if (disposing && (components != null))
         {
            components.Dispose();
```

```
    }
    base.Dispose(disposing);
}
#region Windows Form Designer generated code
private void InitializeComponent()
{
    this.button1 = new System.Windows.Forms.Button();
    this.textBox1 = new System.Windows.Forms.TextBox();
    this.label1 = new System.Windows.Forms.Label();
    this.richTextBox1 = new System.Windows.Forms.RichTextBox();
    this.groupBox1 = new System.Windows.Forms.GroupBox();
    this.label2 = new System.Windows.Forms.Label();
    this.txtOutput = new System.Windows.Forms.TextBox();
    this.button2 = new System.Windows.Forms.Button();
    this.groupBox2 = new System.Windows.Forms.GroupBox();
    this.txtSamples = new System.Windows.Forms.TextBox();
    this.label3 = new System.Windows.Forms.Label();
    this.button3 = new System.Windows.Forms.Button();
    this.openFileDialog1 = new System.Windows.Forms.OpenFileDialog();
    this.saveFileDialog1 = new System.Windows.Forms.SaveFileDialog();
    this.groupBox1.SuspendLayout();
    this.groupBox2.SuspendLayout();
    this.SuspendLayout();
    // button1
    this.button1.Location = new System.Drawing.Point(376, 12);
    this.button1.Margin = new System.Windows.Forms.Padding(2);
    this.button1.Name = "button1";
    this.button1.Size = new System.Drawing.Size(26, 19);
    this.button1.TabIndex = 0;
    this.button1.Text = "...";
```

```
            this.button1.UseVisualStyleBackColor = true;
            this.button1.Click += new System.EventHandler(this.button1_Click);
            // textBox1
            this.textBox1.Location = new System.Drawing.Point(208, 12);
            this.textBox1.Margin = new System.Windows.Forms.Padding(2);
            this.textBox1.Name = "textBox1";
            this.textBox1.Size = new System.Drawing.Size(164, 20);
            this.textBox1.TabIndex = 1;
            // label1
            this.label1.AutoSize = true;
            this.label1.Location = new System.Drawing.Point(12, 15);
            this.label1.Name = "label1";
            this.label1.Size = new System.Drawing.Size(191, 13);
            this.label1.TabIndex = 2;
            this.label1.Text = "Выберите файл с расширением wav";
            // richTextBox1
            this.richTextBox1.Anchor =
((System.Windows.Forms.AnchorStyles)((((System.Windows.Forms.AnchorSty
les.Top | System.Windows.Forms.AnchorStyles.Bottom)
                        | System.Windows.Forms.AnchorStyles.Left)
                        | System.Windows.Forms.AnchorStyles.Right)));
            this.richTextBox1.Location = new System.Drawing.Point(6, 19);
            this.richTextBox1.Name = "richTextBox1";
            this.richTextBox1.Size = new System.Drawing.Size(374, 160);
            this.richTextBox1.TabIndex = 3;
            this.richTextBox1.Text = "";
            // groupBox1
            this.groupBox1.Controls.Add(this.richTextBox1);
            this.groupBox1.Location = new System.Drawing.Point(15, 37);
            this.groupBox1.Name = "groupBox1";
```

```
this.groupBox1.Size = new System.Drawing.Size(386, 185);
this.groupBox1.TabIndex = 4;
this.groupBox1.TabStop = false;
this.groupBox1.Text = "Информация о выбранном файле ";
// label2
this.label2.AutoSize = true;
this.label2.Location = new System.Drawing.Point(6, 47);
this.label2.Name = "label2";
this.label2.Size = new System.Drawing.Size(149, 13);
this.label2.TabIndex = 7;
this.label2.Text = "Название выходного файла";
// txtOutput
this.txtOutput.Location = new System.Drawing.Point(193, 43);
this.txtOutput.Margin = new System.Windows.Forms.Padding(2);
this.txtOutput.Name = "txtOutput";
this.txtOutput.Size = new System.Drawing.Size(158, 20);
this.txtOutput.TabIndex = 6;
// button2
this.button2.Location = new System.Drawing.Point(354, 43);
this.button2.Margin = new System.Windows.Forms.Padding(2);
this.button2.Name = "button2";
this.button2.Size = new System.Drawing.Size(26, 19);
this.button2.TabIndex = 5;
this.button2.Text = "...";
this.button2.UseVisualStyleBackColor = true;
this.button2.Click += new System.EventHandler(this.button2_Click);
// groupBox2
this.groupBox2.Controls.Add(this.txtSamples);
this.groupBox2.Controls.Add(this.label2);
this.groupBox2.Controls.Add(this.label3);
```

```
this.groupBox2.Controls.Add(this.txtOutput);
this.groupBox2.Controls.Add(this.button2);
this.groupBox2.Location = new System.Drawing.Point(15, 228);
this.groupBox2.Name = "groupBox2";
this.groupBox2.Size = new System.Drawing.Size(386, 78);
this.groupBox2.TabIndex = 8;
this.groupBox2.TabStop = false;
this.groupBox2.Text = "Информация о выходном файле";
// txtSamples
this.txtSamples.Location = new System.Drawing.Point(193, 16);
this.txtSamples.Margin = new System.Windows.Forms.Padding(2);
this.txtSamples.Name = "txtSamples";
this.txtSamples.Size = new System.Drawing.Size(187, 20);
this.txtSamples.TabIndex = 9;
// label3
this.label3.AutoSize = true;
this.label3.Location = new System.Drawing.Point(6, 19);
this.label3.Name = "label3";
this.label3.Size = new System.Drawing.Size(154, 13);
this.label3.TabIndex = 0;
this.label3.Text = "Частота дискретизации  (Hz)";
// button3
this.button3.Location = new System.Drawing.Point(15, 311);
this.button3.Margin = new System.Windows.Forms.Padding(2);
this.button3.Name = "button3";
this.button3.Size = new System.Drawing.Size(385, 28);
this.button3.TabIndex = 10;
this.button3.Text = "Изменить частоту дискретизации";
this.button3.UseVisualStyleBackColor = true;
this.button3.Click += new System.EventHandler(this.button3_Click);
```

```
    // openFileDialog1
    this.openFileDialog1.FileName = "openFileDialog1";
    // Form1
    this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);
    this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
    this.ClientSize = new System.Drawing.Size(411, 351);
    this.Controls.Add(this.button3);
    this.Controls.Add(this.groupBox2);
    this.Controls.Add(this.groupBox1);
    this.Controls.Add(this.label1);
    this.Controls.Add(this.textBox1);
    this.Controls.Add(this.button1);
    this.Margin = new System.Windows.Forms.Padding(2);
    this.Name = "Form1";
    this.Text = "Resampling wav file";
    this.groupBox1.ResumeLayout(false);
    this.groupBox2.ResumeLayout(false);
    this.groupBox2.PerformLayout();
    this.ResumeLayout(false);
    this.PerformLayout();
}
#endregion
private System.Windows.Forms.Button button1;
private System.Windows.Forms.TextBox textBox1;
private System.Windows.Forms.Label label1;
private System.Windows.Forms.RichTextBox richTextBox1;
private System.Windows.Forms.GroupBox groupBox1;
private System.Windows.Forms.Label label2;
private System.Windows.Forms.TextBox txtOutput;
private System.Windows.Forms.Button button2;
```

```
        private System.Windows.Forms.GroupBox groupBox2;
        private System.Windows.Forms.TextBox txtSamples;
        private System.Windows.Forms.Label label3;
        private System.Windows.Forms.Button button3;
        private System.Windows.Forms.OpenFileDialog openFileDialog1;
        private System.Windows.Forms.SaveFileDialog saveFileDialog1;
    }
}
```

Program.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Windows.Forms;

namespace AudioResampling
{
    static class Program
    {
        /// <summary>
        /// The main entry point for the application.
        /// </summary>
        [STAThread]
        static void Main()
        {
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
            Application.Run(new Form1());
        }
    }
}
```